ABSTRACT

A three-dimensional model for the simulation of transient groundwater flow is developed. The model is called REGFED for REGional flow using Finite Elements and Difference methods. A review of groundwater flow and contaminant transport concepts and theory reveals that three-dimensional representation of groundwater systems is essential for realistic simulation of flow and transport. From an analysis of currently available groundwater flow models and algorithms, it is apparent that a mixed numerical method consisting of finite-elements and finite differences is a suitable method for solving the groundwater flow equation in three dimensions. An algorithm known as ALALS (ALternate sublayer And Line Sweep) is selected for the basic model algorithm.

Finite elements are applied to areal components, and finite differences are applied to vertical components of flow. The model accomodates both confined and unconfined groundwater flow problems and is also capable of handling the draining and refilling of individual elements or entire layers. Because of the model's efficient algorithm, it can accomodate thousands of nodal unknowns with minimal computer storage and CPU time.

Quasilinear unconfined groundwater flow problems are solved using a Picard iteration scheme. Entire confined layers are skipped in the iteration scheme in order to decrease the CPU time required to solve the problem. The model is validated under a wide assortment of conditions including confined flow, confined flow with partially screened wells, unconfined flow, combined confined/unconfined flow, and flow with drained and refilled layers. A heuristic error analysis shows that model results compare well with validation results. Mass-balance errors for various groundwater flow problems are minimal for most cases.

The convergence speed and stability of the iteration scheme is evaluated for solution of unconfined groundwater flow problems. A benchmark comparison using sample groundwater flow problems was performed with the REGFED model and with the USGS McDonald-Harbaugh model. Example applications further demonstrate the flexibility of the model.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Thanks first to my intrepid advisor, Dr. Cass T. Miller. Without his encouragement and guidance, this research never would have gotten off the ground. He has helped reveal to me the many fascinating facets of the groundwater field.

To my office-mates, thank you for bearing with me. Thanks especially to Joe Pedit for his valuable insight and reminders that this isn't the most important thing in the world.

But most of all, thank you Harriet for your love and support. I couldn't have done it without you.

1 INTRODUCTION

1.1 Background and Motivation

Approximately half the population of the U.S. depends on groundwater for its drinking water supplies. There is growing evidence that this resource, once thought to be contaminant-free, is being contaminated by municipal, industrial, and agricultural wastes. Researchers are thus focusing upon studying the mechanisms responsible for contaminant transport in groundwater systems. To prevent the further deterioration of groundwater quality, researchers are developing methodologies for monitoring, analyzing, and predicting the movement of contaminants in the subsurface. Predictive models of groundwater contaminant transport can provide the information needed for the accurate assessment of health risks resulting from contamination of drinking water supplies, or for the design and evaluation of measures for renovating contaminated groundwater aquifers.

1.1.1 Relationship between Groundwater Flow and Contaminant Transport

One of the most important factors in predicting the movement of contaminants in the subsurface is the analysis of groundwater flow systems. In the past, groundwater flow simulation has been mainly a tool for quantifying yields of groundwater resources. For example, the amount of water available from an aquifer to support a given population, industrial, or agricultural base is a problem that groundwater flow researchers have studied in detail. The increasing urgency of groundwater quality problems has changed the focus of groundwater research by spurring the development of predictive tools in the form of mathematical models designed to simulate the transport of contaminants in groundwater. However, in the mathematical simulation of aquifer contamination, an accurate definition of the flow system still is of vital importance (Frind et al., 1985). Thus, groundwater flow

models can be developed in the context of groundwater contamination problems.

In order to understand the relationship between groundwater flow and contaminant transport, one must examine the equations that govern the hydrodynamics of contaminant transport. Determinstic and stochastic approaches for mathematically describing contaminant transport are possible. This report focuses upon deterministic apporaches for transport and flow, due to the relative difficulty of applying the stochastic approach to practical contaminant transport problems.

The advective-dispersive equation is generally considered to be the equation that governs contaminant transport (Anderson, 1979), although other researchers have proposed different approaches (Gillham et al., 1982; Tompson, 1986). The advective-dispersive equation considers solute flux to be the result of the average bulk movement of the fluid in the direction of groundwater flow (advection) and a Fickian-type mixing in the displacing fluid (dispersion) (Gillham et al., 1984). For saturated flow in heterogeneous porous media, the general form of the advective-dispersive equation is written as

$$\frac{\partial C}{\partial t} = \nabla \cdot (\mathbf{D} \cdot \nabla C) - \vec{v} \cdot \nabla C + \left(\frac{\partial C}{\partial t}\right)_{rxn} + \Gamma(C) \tag{1.1}$$

where

C = solute phase concentration (M/L^3)

t = time (T)

$\vec{v}$ = vector of average groundwater pore velocity (L/T)

$\mathbf{D}$ = hydrodynamic dispersion tensor (L^2/T)

$\nabla\cdot$ = divergence operator

∇ = gradient operator

$\left(\frac{\partial C}{\partial t}\right)_{rxn}$ = reactive term $(M/L^3/T)$

$\Gamma(C)$ = source or sink term $(M/L^3/T)$

Reactive processes such as sorption, chemical reactions, and biological degradation can play important roles in the fate of contaminants and should also be accounted for in any model of non-conservative groundwater contaminant transport. The focus of this report is not on the reactive portion of Equation 1.1, but concentrates on the the hydrodynamics.

The conservative form of Equation 1.1 implies that

$$\left(\frac{\partial C}{\partial t}\right)_{rxn} = 0 \tag{1.2}$$

which reduces Equation 1.1 to

$$\frac{\partial C}{\partial t} = \nabla \cdot (\mathbf{D} \cdot \nabla C) - \vec{v} \cdot \nabla C + \Gamma(C) \tag{1.3}$$

Bear (1972) describes the hydrodynamic dispersion tensor as the sum of two components, which can be represented as

$$D_{ij} = \alpha_T \overline{v} \delta_{ij} + (\alpha_L - \alpha_T)\overline{v}_i \overline{v}_j / \overline{v} + D^* \tag{1.4}$$

where

D_{ij} = i, j term of dispersion tensor (L^2/T)

i, j = components of Cartesian coordinate system

α_T = transverse dispersivity (L)

α_L = longitudinal dispersivity (L)

$\overline{v}$ = average groundwater pore velocity (L/T)

D^* = effective molecular diffusion coefficient (L^2/T)

δ_{ij} = Kronecker delta function (*dimensionless*)

= 1 for $i = j$

$= 0$ for $i \neq j$

The product of dispersivity and flow velocity is known as the mechanical dispersion component. The mechanical mixing is a process introduced by averaging irregular advective displacements taking place within the porous groundwater matrix (Fried and Cornabous, 1971). In active groundwater flow through a granular medium, mechanical dispersion is usually dominant over diffusion, and so the D^* term is often a relatively small component.

By examining Equation 1.2, one can see that groundwater velocity, through the advective term, is a crucial part of the advective- dispersive approach to modeling contaminant transport, for a typical groundwater aquifer. In addition, Equation 1.4, which describes the dispersion tensor, includes velocity-dependent terms.

Various mathematical solutions to the advective-dispersive equation have been proposed. These solutions have been compared to experimental results from laboratory-scale soil columns. The solutions have been shown to provide accurate representations of conservative solute transport, under laboratory conditions (Gillham et al., 1984). Longitudinal dispersivities have been found that range within a couple of orders of magnitudes of each other (10^{-4} to 10^{-2} meters). However, when the solutions of the advective-dispersive equation are applied to field-scale tracer tests, longitudinal dispersivities in the range of 1 to 100 meters have been commonly reported (Gelhar et al., 1985). This variation in dispersivity poses a difficulty in the use of predictive models of solute transport based on the advective-dispersive equation.

The discrepancies between longitudinal dispersivities obtained from laboratory- and field-scale experiments have led some researchers to conclude that dispersivity is a parameter which is scale-dependent (Fried, 1975; Peaudecerf and Sauty, 1978; Sudicky and Cherry, 1979; Pickens and Grisak, 1981). The scale dependency is generally attributed to the effect of heterogeneity of the geological media (Skibitzke et al., 1963; Fried, 1975;

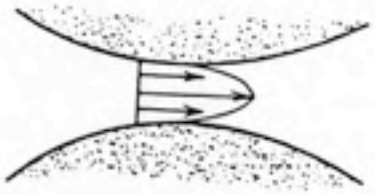
Bear, 1977; Schwartz, 1977; Anderson, 1979).

Heterogeneities can be found in a range of scales in the geological media. Figure 1.1 illustrates various types of heterogeneities that occur and the subsequent effect on velocity distributions. These heterogeneities affect the velocity field, and therefore groundwater flow, at all scales. Figure 1.1 illustrates that heterogeneities can occur from the grain-size scale to the geologic- layering scale. The layering scale could, if necessary, be identified and mapped by careful drilling, sampling or geophysical logging. If the smallest scale of heterogeneities in a deterministic-type media could be identified and accounted for, then the differences in advection or groundwater flow could be accurately simulated. However, these heterogeneities cannot be identified by conventional methods of field testing (Freeze and Cherry, 1979). As long as the smallest heterogeneities cannot be identified, it is important that models of groundwater flow provide accurate simulations, using the best available information from the scales that can be identified.

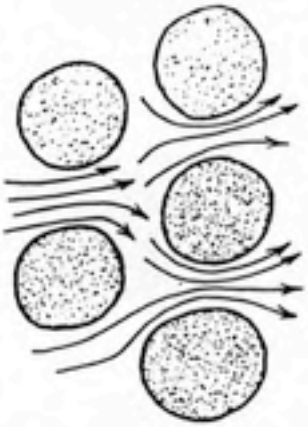
The dispersivity parameter is a result of averaging over scales larger than the smallest scales. The average linear groundwater velocity that is used as input to the advective-dispersive equation reduces the individual velocities in the interstitial flow paths to a single value (Freeze and Cherry, 1979). Averaging of velocities often goes one step further where individual velocities within layers of different hydraulic conductivities are averaged to a single value. The result of this averaging is that the observed dispersity parameters contain the deviations in velocities at the scale over which the flow has been averaged (Anderson, 1984). Deterministic models of the advective-dispersive equation assume that hydrodynamic process occur over measurable scales. If the models included the smallest heterogeneities, theoretically there would be no deviations in velocities over a small scale, and therefore the dependence of prediciting hydrodynamics on dispersivities would be reduced.

1.1.2 Importance of Modeling Flow and Transport in Three Dimensions

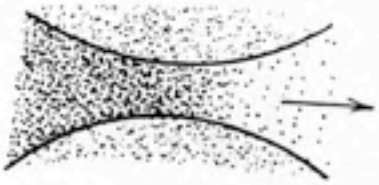
FIGURE 1.1
DISPERSION PHENOMENA



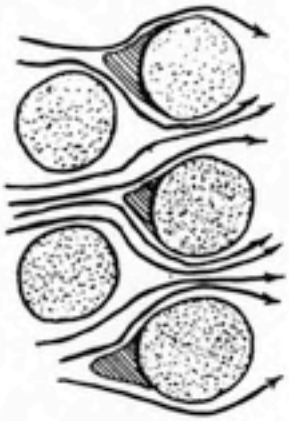
Mixing in individual pores



Mixing of pore channels

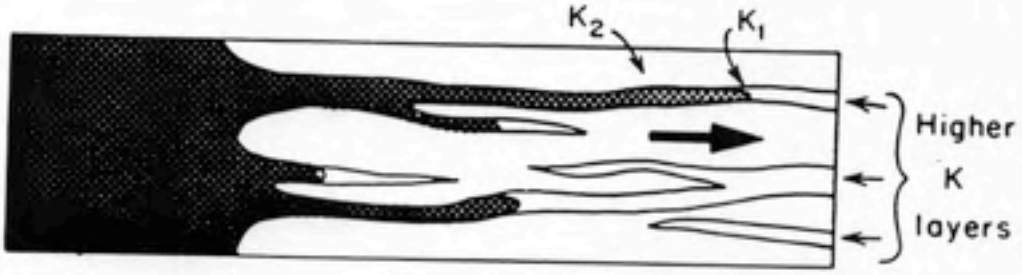


Mixing by molecular diffusion



Immobile Fluid

Mobile/Immobile Exchange



Macrodispersive Spreading

The majority of contaminant transport models have been developed over two dimensions (Burnett and Frind, 1987). In these cases, important aquifer properties such as velocity magnitudes and directions are spatially averaged over the relevant dimensions (usually the vertical dimension). This approach dooms the prediction of contaminant transport to failure, in all but the simplest of groundwater systems. Figure 1.2 provides a schematic illustration of a typical three-dimensional groundwater flow and contaminant transport problem.

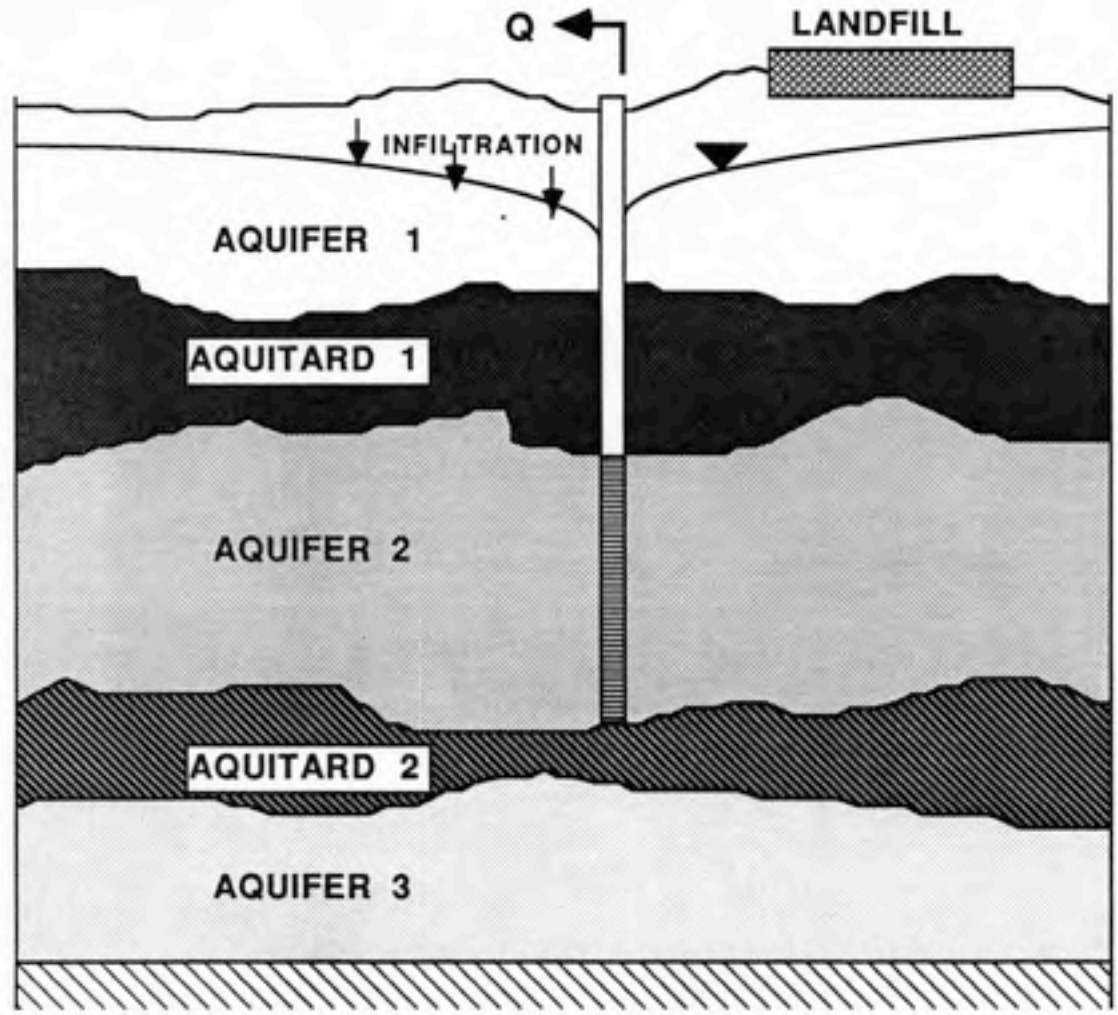
Various researchers have reported that three-dimensional modeling of flow and contaminant transport improves the accuracy of such simulations relative to two- or three-dimensional simulations. Stochastic analyses performed by Freeze (1975) and Gelhar (1976) show that there is considerably less variation about a mean hydraulic head value for three-dimensional flow models than for two- and one-dimensional models. Increases in these variations tend to inflate the value of dispersion and produce poor predictive ability in contaminant transport models.

The scale effects on dispersion that were discussed previously may be an artifact of the dimensionality of the models employed to predict dispersion (Domenico and Robbins, 1984). Results of Domenico and Robbins (1984) indicate that a "scaling-up" of dispersivity will occur when the dimensionality of a model fails to match that of a natural system. Molz et al. (1983) conclude that the vertical distribution of hydraulic conductivity (and the subsequent effect on mixing) is a key parameter that affects overall dispersivity.

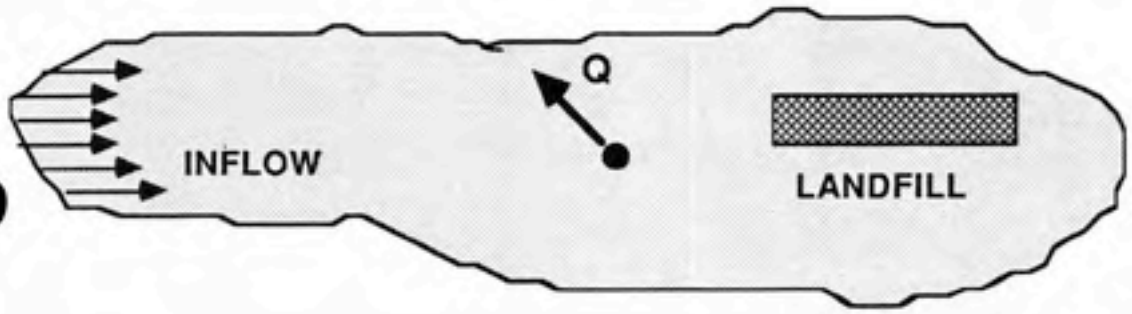
Burnett and Frind (1987) describe variations in hydrodynamic parameters in three dimensions that influence the shape of a contaminant plume. Arnett et al. (1977) report that three-dimensional models of contaminant movement compare better with observed contaminant movement at the Hanford, Washington site, than for two-dimensional models.

Vertically-layered groundwater systems are often found in the field (Huyakorn et al., 1986). Such systems occur commonly in stratigraphic sections, as a result of most depositional processes. Differences in hydraulic conductivities between layers can be several

FIGURE 1.2
TYPICAL 3-D GROUNDWATER FLOW AND TRANSPORT PROBLEM



CROSS-SECTIONAL VIEW



AERIAL VIEW

orders of magnitude (Sudicky, 1986). If the variations in flow caused by the differences are not taken into account, at best only averaged contaminant concentrations can be predicted rather than in the individual layers. The presence of a high conductivity layer may direct the contaminants toward this layer, the effects of which may be ignored in a one- or two-dimensional analysis. Results from Sudicky (1986) and Molz (1986) showing the vertical distribution of hydraulic conductivities are shown in Figure 1.3. These results show that hydraulic conductivity, and thus velocities, can vary more than an order of magnitude in the vertical direction.

The effects of the vertical averaging of groundwater velocity distributions can be shown through some hypothetical simulations. An analytical model of the one dimensional, conservative form of the advective-dispersive equation (Bear, 1979) was applied to two cases: 1) a five-layer aquifer, with each layer having a different groundwater velocity, and a line source of contaminant, as shown in Figure 1.4; and 2) the same aquifer, but with the velocities of the five layers averaged into a single value of velocity. The simulations were performed at three different positions down-gradient from the contaminant source. The value of longitudinal dispersivity for the second case was fitted to the results from the first case at the first down-gradient position. All other parameter values were the same for each case.

The results are shown in Figure 1.5 (note that the time axis has a log scale). These results show that, at the first position (where the dispersivity was fitted), the vertically averaged results resemble the non-vertically averaged reslts. However, as the simulations move farther from the contaminant source, the vertically averaged results no longer resemble the non-vertically averaged results. Thus, the vertical variations cannot be averaged while expecting the simulations to provide accurate results.

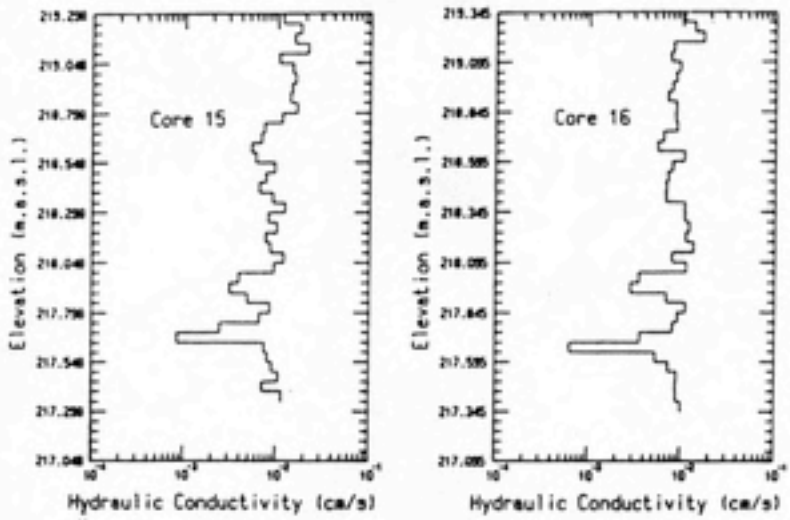
1.1.3 Importance of Modeling Unconfined Flow

Aquifers are generally classified as either confined (artesian) or unconfined (water

FIGURE 1.3

VERTICAL VARIATIONS IN HYDRAULIC CONDUCTIVITY

After Sudicky (1986) and Molz (1986)



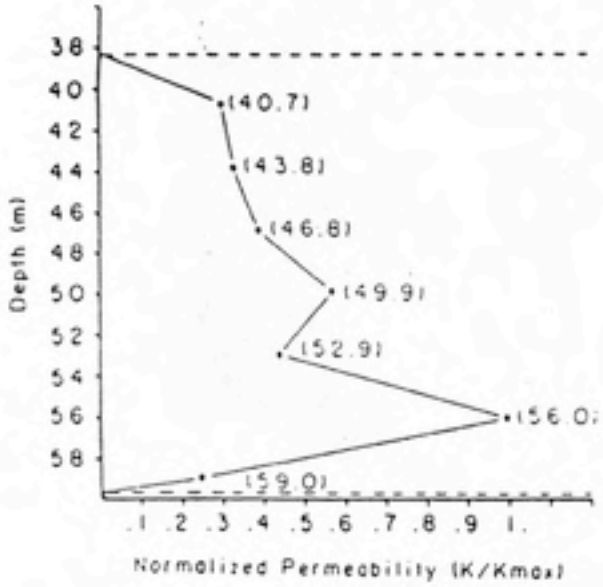
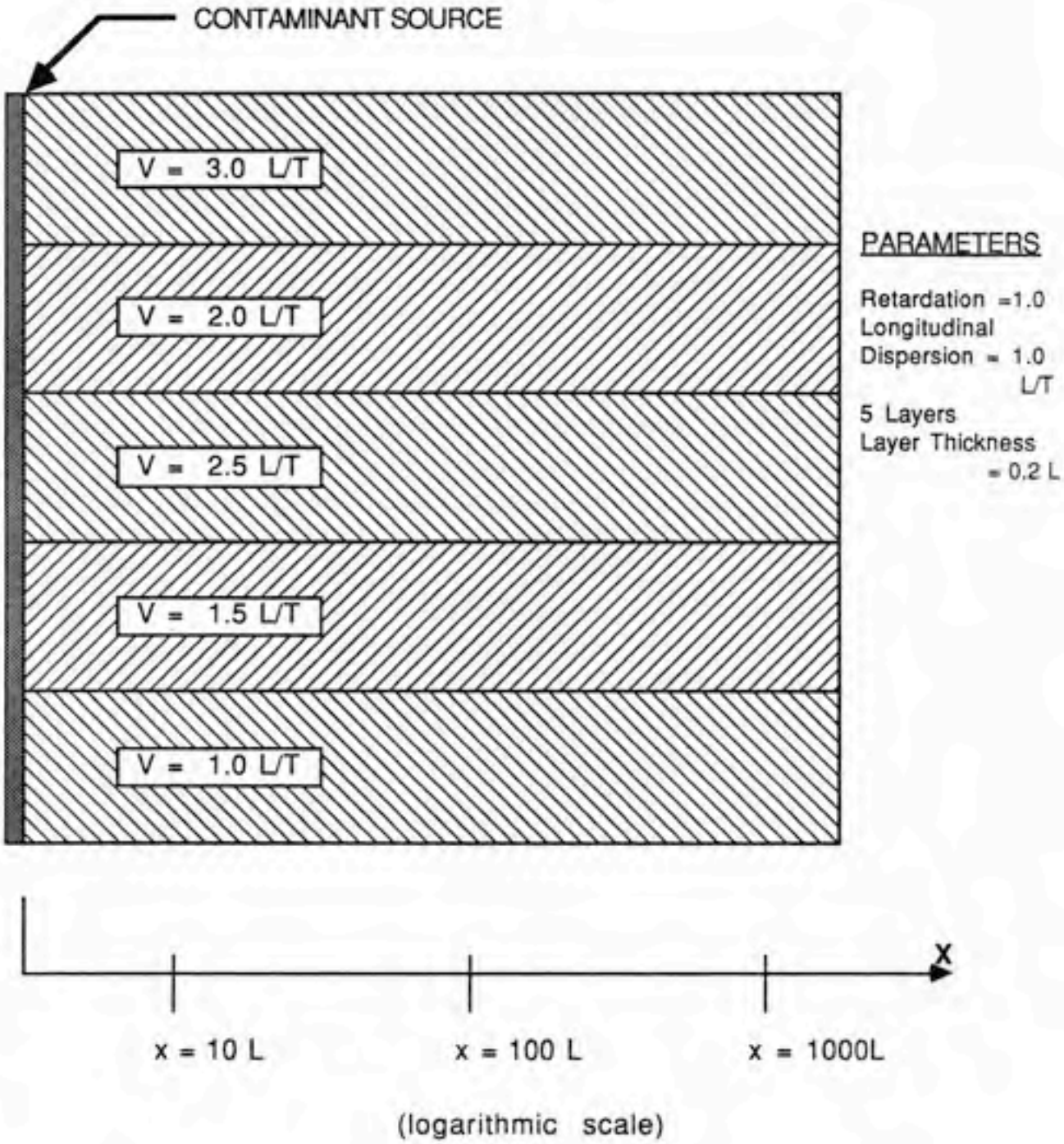


FIGURE 1.4
HYPOTHETICAL LAYERED AQUIFER



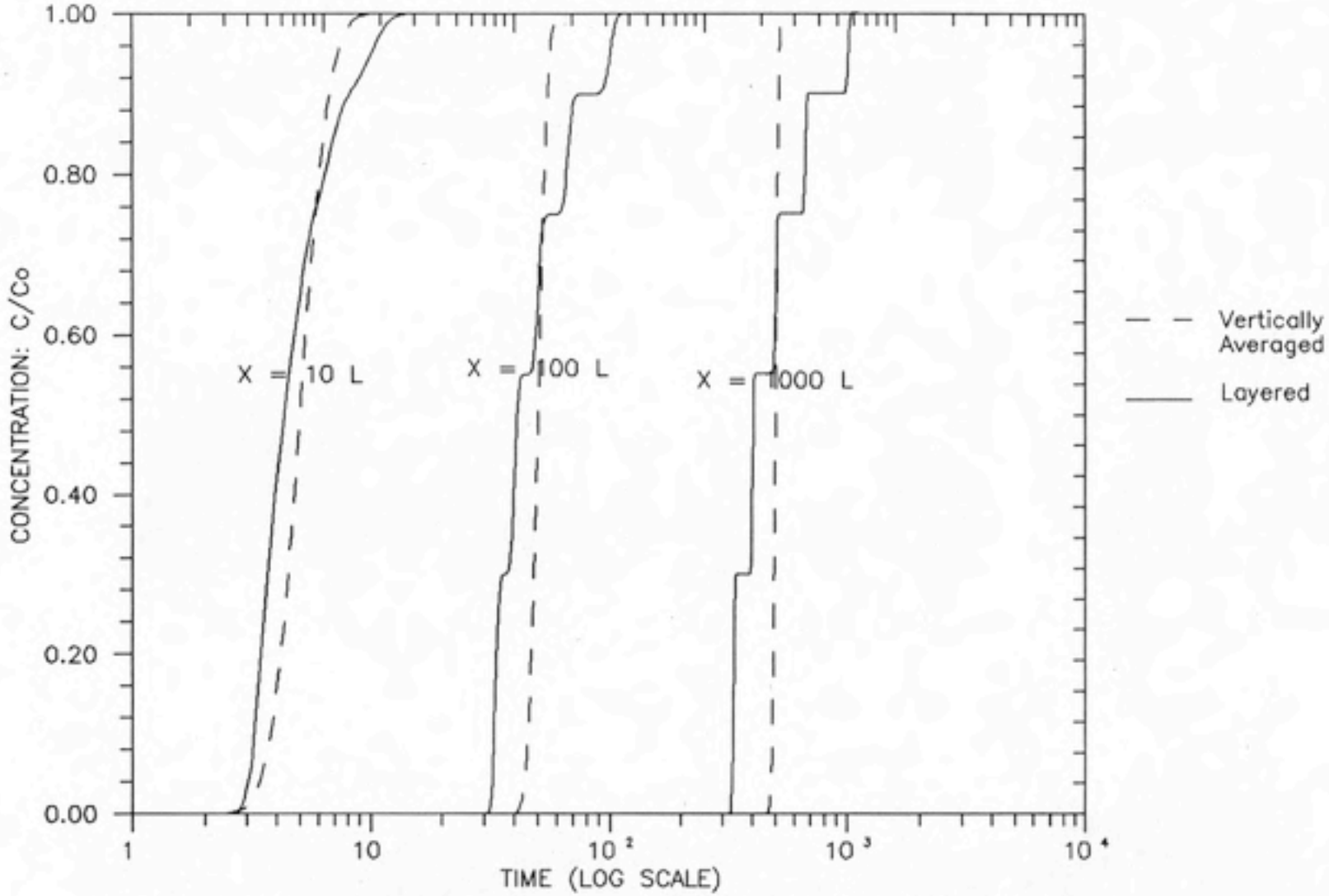


FIGURE 1.5: COMPARISON OF VERTICALLY-AVERAGED AND LAYERED CONTAMINANT TRANSPORT SIMULATIONS

table). Flow in confined aquifers is bounded above and below by impervious layers. Unconfined aquifers are bound below by an impervious layer, but are bound above by the top of the water table. Figure 1.6 shows a schematic representation of the two types of aquifers. Most contamination cases can be found in unconfined aquifers, due to the lack of a protective confining layer and thus an increased vulnerability over confined aquifers. Shallow unconfined aquifers are particularly susceptible to pollution from contaminants when little or no treatment is afforded by the overlying strata (Guvanasen and Volker, 1981).

However, most of the available flow models either do not accomodate unconfined flow at all or do so unreliably. Modeling an unconfined groundwater system as a confined system usually is inaccurate because the flow regimes may differ greatly between the two types of systems. The presence of a free upper boundary in an unconfined aquifer can significantly affect groundwater velocities, especially in shallower aquifers. These differences can translate to poor estimates for the movement of groundwater contaminants, if the wrong system is modeled.

1.3 Research Goals and Objectives

Thus, the goal of this research is to develop a versatile model that accurately and efficiently simulates confined and unconfined groundwater flow in three dimensions.

The objectives to be met with this research are:

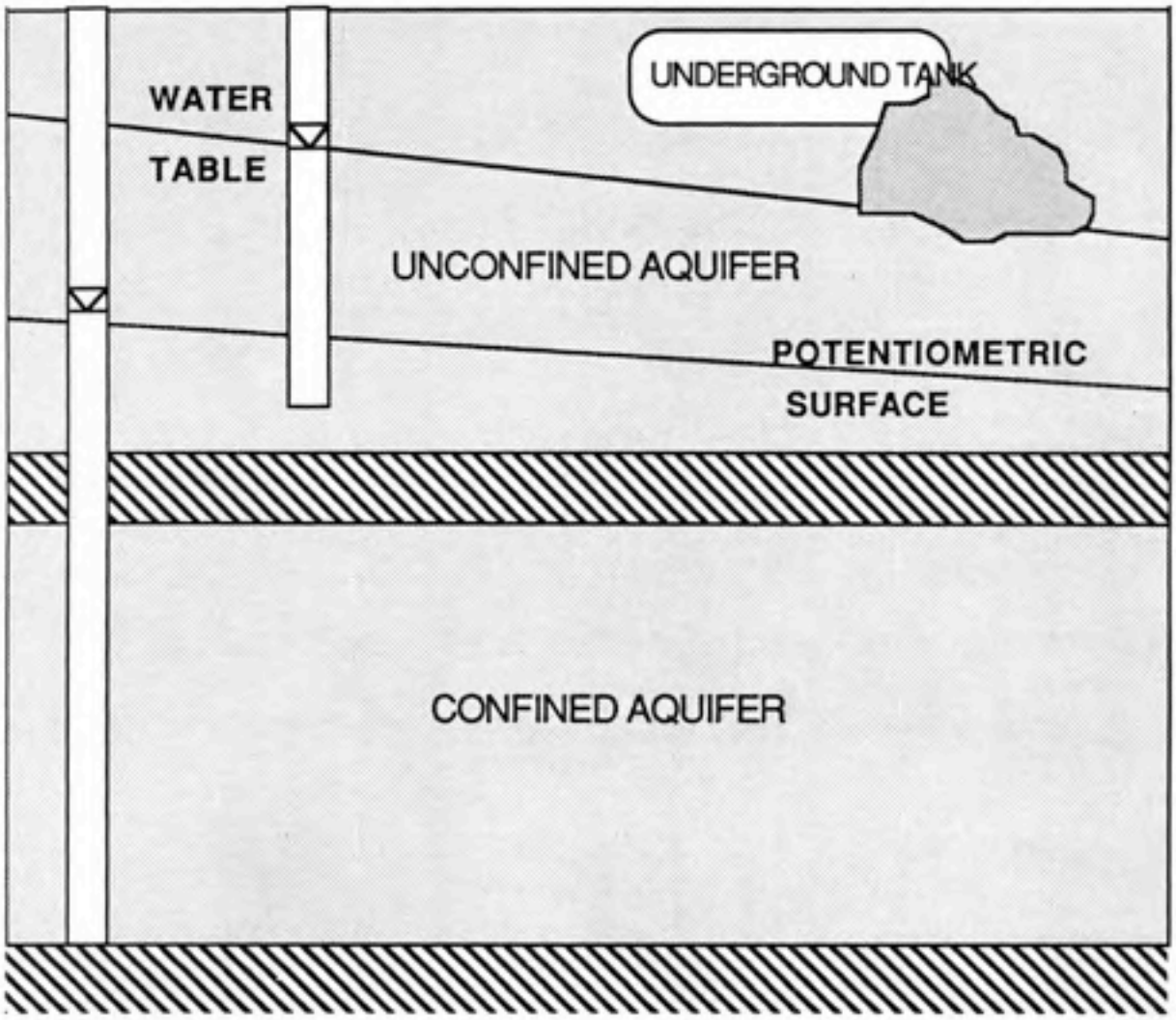
1) To develop a three-dimensional numerical model for simulating confined groundwater flow.

2) To develop a three-dimensional numerical model for simulating unconfined groundwater flow, using the confined flow model for the basic structure so that a combination of confined and unconfined flow can be accomodated in the final model.

3) To test the accuracy of both the confined and unconfined flow portions of the final model.

FIGURE 1.6

CONFINED AND UNCONFINED AQUIFERS



4) To apply the model to some hypothetical groundwater flow situations.

1.3 Methodology

To meet the first objective, an algorithm consisting of a numerical solution of the groundwater flow equation is selected. This algorithm, called the ALALS algorithm (for ALternate sublayer And Line Sweep procedure) has been described in the literature. The complete derivation of this algorithm is performed. Next, a structured and commented computer code that incorporates the ALALS algorithm is developed. Various modifications of the algorithm are also included in the code, such as the ability to simulte steady-state as well as transient problems, and a provision for calculating mass balance errors. The result is a model for simulating confined flow.

The model developed under the first objective is then modified to include unconfined flow. The equation describing unconfined flow is not linear, as it was the confined case. An iterative algorithm (Picard iteration) is utilized to solve the unconfined flow equation. Application of the iterative algorithm results in a significant increase in computaional effort over the confined model. The computational effort is reduced by modifying the iterative algorithm to include unconfined aquifer layers only. Other problems resulting from modeling unconfined flow, such as the draining and refilling of aquifer layers, are incorporated into the model. The resulting model can simulate confined and unconfined flow separately or at the same time. The model is named REGFED for REGional flow using Finite Element and Differences methods.

The third objective involves testing the accuracy of the unconfined and confined portions of the model. In this research, accuracy is evaluated by graphical comparisons of model results with results from analytical solutions. In a few cases where analytical models are not availble for comparison, the ability of the model to balance mass in and out of the groundwater system is analyzed. The sensitivity of the model to various model parameters, such as vertical and horizontal discretization schemes and time step sizes, is

analyzed by graphical comparisons of model results with analytical solution results.

Hypothetical applications are simulated with the model. The example applications include flow within a monitoring well, flow in an aquifer-confining layer system, and flow resulting from a two-well tracer test. In addition, the performance of the model is compared to the most popular three-dimensional public domain groundwater flow model, the McDonald-Harbaugh model. This comparison provides a way to gauge the relative efficiency of the WELFED model. The total computational time required for each model to simulate a sample problem is compared.

2 THEORETICAL BACKGROUND AND LITERATURE REVIEW

The determination of groundwater flow requires the evaluation of either or both of the hydraulic head variable or the velocity variable. Hydraulic head is a measure of fluid potential; it consists of the sum of a pressure head and an elevation head. Most groundater flow models simulate distributions of hydraulic heads. Groundwater velocity is the velocity variable found in the advective-dispersive equation. Velocity is proportional to the negative of the groundwater gradient (Darcy's Law).

Generally, there are two approaches towards simulating groundwater flow velocities: the indirect and direct method. The indirect method– the most popular– consists of simulating hydraulic head distributions and then using Darcy's Law to approximate groundwater velocities. The direct method uses Darcy's Law directly to simulate groundwater velocities. This report focuses on simulating distributions of hydraulic heads.

Before discussing the approaches toward obtaining hydraulic head and velocity, the classical theories of groundwater flow should be reviewed. By examining the theory first, one can understand the necessary steps in each approach.

2.1 Governing Equations for Groundwater Flow

2.1.1 Theory: Darcy's Law

Groundwater flow theory begins with Darcy's Law. Darcy's Law is an empirically derived formula that relates specific discharge to the groundwater gradient. It is usually represented as

$$q = -K\frac{\partial h}{\partial x} \tag{2.1}$$

where

q = specific discharge (L/T)

K = hydraulic conductivity (L/T)

h = hydraulic head (L)

$\frac{\partial h}{\partial x}$ = groundwater gradient (*dimensionless*)

Darcy's law is valid for groundwater flow in any direction in space. However, it should be understood that the specific discharge calculated from Darcy's Law is a macroscopic concept, which is averaged over a portion of the porous medium. The specific discharge is clearly differentiated from the velocities encountered in the actual path of the fluid particle through a porous medium (Bear, 1979).

The average velocity, v, represents the flow that passes through only the portion of the porous medium occupied by voids in the porous matrix. The average velocity is found in the advective and dispersive terms of the advective-dispersive equation. It is obtained by

$$\overline{v} = \frac{q}{n} \tag{2.2}$$

where

$\overline{v}$ = average groundwater pore velocity (L/T)

n = porosity (*dimensionless*)

2.1.2 Theory: Groundwater Flow Equation

The continuity equation for groundwater flow is a partial differential equation that describes the conservation of fluid mass during flow through a porous medium. The groundwater flow equation for saturated flow in confined aquifers is generally represented as

$$\nabla \cdot (\mathbf{K} \cdot \nabla h) + \Gamma(h) = S_s \frac{\partial h}{\partial t} \tag{2.7}$$

where

h = hydraulic head (L)

$\mathbf{K}$ = hydraulic conductivity tensor (L/T)

S_s = specific storage $(1/L)$

$\Gamma(h)$ = source or sink term $(1/T)$

The assumptions implied in this equation are that (1) the flow of water is laminar, (2) the fluid is incompressible and of constant density, (3) the porous medium is rigid, and (4) the unsaturated portion of flow can be negelected. Assumptions (1) through (3) are most commonly applied in groundwater flow analysis. Assumption (4) involves the unsaturated region. This region involves the two- phase flow of air and water and is found directly above the top of the water table (see Figure 1.6). Unsaturated flow is important when considering infiltration of fluids from above the water table. The unsaturated portion of flow is neglected in this report, because the difficulty of modeling unsaturated flow outweighs the practical advantages to be gained.

Equation 2.3 can be simplified further by assuming that the components of the conductivity tensor are aligned with the directions of the gradients of head. This assumption allows for the consideration of only the diagonal components of the conductivity tensor and reduces Equation 2.3 to

$$\frac{\partial}{\partial x}\left(K_x\frac{\partial h}{\partial x}\right)+\frac{\partial}{\partial y}\left(K_y\frac{\partial h}{\partial y}\right)+\frac{\partial}{\partial z}\left(K_z\frac{\partial h}{\partial z}\right)+\Gamma(h)=S_s\frac{\partial h}{\partial t} \tag{2.4}$$

where

K_x, K_y, K_z = components of conductivity in the x, y, and z directions, respectively (L/T)

2.1.3 Theory: Unconfined Flow

When examining flow in unconfined aquifers, the physics governing flow change. These changes are reflected in the groundwater flow equation. The conductivity parameters found in Equation 2.4 are constant for confined flow. However, for unconfined flow, vertical averaging produces the transmissivity parameter, which is a function of the saturated thickness of the aquifer. The storage parameter found in Equation 2.4 also changes for unconfined aquifers, to represent the saturated/unsaturated interaction of the aquifer.

In order to analyze unconfined flow with Equation 2.4, the equation is often vertically averaged (using the Dupuit assumptions of negligible vertical gradients). The averaging produces the new parameters of transmissivity (the vertically averaged hydraulic conductivity) and storativity (the vertically averaged specific storage). Vertical averaging also eliminates the terms that are a function of z. Equation 2.4 can be rewritten as

$$\frac{\partial}{\partial x}\left(T_x\frac{\partial h}{\partial x}\right)+\frac{\partial}{\partial y}\left(T_y\frac{\partial h}{\partial y}\right)+\Gamma'(h)=S_y\frac{\partial h}{\partial t} \tag{2.5}$$

where

T_x, T_y = components of transmissivity in the x, and y directions, respectively, where $T = Kh$ (L^2/T)

S_y = specific yield (*dimensionless*)

$\Gamma'(h)$ = vertically averaged source or sink term (L/T)

The resulting differential equation is more difficult to solve, because it is no longer a linear function of h (hydraulic head).

2.2 Solutions of the Groundwater Flow Equation

Solutions to groundwater flow equations such as 2.4 or 2.5 can be solved for the hydraulic head variable. Analytical and numerical solutions of the flow equations are used in the analysis of groundwater flow. However, analytical solutions usually are not sophisticated enough to handle heterogeneous aquifers of irregular shape that are most often encountered in the field. The analysis and prediction of aquifer performance in such situations is normally carried out by numerical simulation. However, analytical solutions can be used for some types of aquifer evaluations and also serve as convenient benchmarks for evaluating the accuracy of numerical models.

2.2.1 Analytical Flow Models

Simulations of hydraulic head distributions have been performed for at least 50 years. Theis (1935) solved a radial form of the groundwater flow equation to obtain an analytical expression for the change in hydraulic head around a pumped well in a confined aquifer. Many other analytical solutions for various types of flow have been produced since Theis.

In the case of unconfined flow, transient groundwater flow is more difficult to simulate. The analytical (and numerical) solutions available to analyze unconfined flow are consequently fewer than those for confined flow. Analytical solutions proposed to simulate unconfined flow are still under scrutiny by groundwater researchers. The problems arise from the fact that the top boundary (also known as the free surface) of the aquifer moves as hydraulic head changes and that the groundwater flow equation is no longer linear. To simplify the treatment of such problems, researchers have relied on the Dupuit assumptions

(Streltsova, 1973; Bear, 1979). These assumptions basically mean that vertical gradients within the aquifer can be ignored. These assumptions give rise to the Boussinesq equation for unconfined flow:

$$\frac{\partial}{\partial x}\left(K_x \frac{\partial h}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial h}{\partial y}\right) + \Gamma'(h) = S_y \frac{\partial h}{\partial t} \tag{2.6}$$

Freeze and Cherry (1979) identified three approaches to analyze unconfined flow in pumped wells. The first recognizes that the unconfined problem involves a saturated-unstaurated flow system in which changes in hydraulic head are accompanied by changes in the moisture content above the water table. An analytical solution for this case was presented by Kroszynski and Dagan (1975). However, the conclusions from this and other studies (Taylor and Luthin, 1969; Cooley, 1971) is that hydraulic heads are not substantially affected by including the unsaturated flow component.

The second approach is to use the confined aquifer (the Theis equation) defined in terms of specific yield instead of storativity. This method effectively relies on the Dupuit assumptions. Jacob (1950) has shown that this approach is nearly correct as long as drawdowns are small in comparison with saturated thickness. The third approach is based on the concept of a delayed water-table response. Neuman (1972) presents an analytical solution for this approach. After long times or at a long enough distance from the well, the hydraulic head distributuon eventually mimics the Theis solution for unconfined flow.

2.2.2 Numerical Flow Models

Numerical simulations of confined and unconfined groundwater flow are also well established. Various numerical methods are available for solving the groundwater flow equations. These methods include finite differences, finite elements, finite element- finite difference hybrids, and boundary integral equation methods (BIEM).

Finite difference methods have been applied to groundwater flow problems for many

years. This method is relatively easy to apply, as long as the problem domain has boundaries that are relatively regular in shape. Irregular boundaries are simulated inefficiently with finite differences. The accuracy of results obtained from finite difference methods is generally lower than results from finite elements, given the same number of nodes used in the discretization. However, the computational effort required to solve a finite difference problem is usually smaller than for finite elements, given the same level of desired accuracy (Faust and Mercer, 1981).

With finite-elements, problems can be solved using fewer unknowns than for finite differences, given the same degree of accuracy. Finite element methods have the advantage of being able to fit irregular boundaries without additional comptutational effort over simpler boundaries. In addition, finite elements provide values of the dependent variable over the entire problem domain, not just at selected nodal locations as in finite differences. However, the computational cost of three-dimensional finite element applications is prohibitive, due to the large amount of data and operations that must be carried through the computational procedure.

Hybrid finite element-finite difference methods combine the good points from both methods. Irregular boundaries are usually encountered in the horizontal or areal directions, therefore, finite elements are applied in this direction. Vertical changes in parameters such as hydraulic conductivity often occur as changes from one parallel layer to another, which makes for a suitable application of finite differences. By reducing the dimensionality over which finite elements are applied, the computational cost of the applied method is reduced.

The BIEM also has the advantage of providing flexible boundaries. It is also especially suited for unconfined flow problems. However, this method contains some serious drawbacks. First, current theory does not allow for the convenient solution of time dependent problems. Second, model parameters must be constant within the domain– a substantial disdvantage for any method where heterogeneous conditons are encountered.

Numerical simulations are commonly performed in two space dimensions, either with

cross-sectional or areal models. There are at least two two-dimensional aquifer-simulation programs that have been completely documented and widely applied in North America. These programs are the Trescott-Pinder-Larson model (Trescott et al., 1976) and the Illinois Water Survey model (Prickett and Lonnquist, 1971). Both of these models utilize finite difference formulations to produce head distributions.

Numerical methods for simulating two-dimensional unconfined flow have also been proposed. The problem of locating the position of the free surface is usually resolved by approximating the free surface location and then iterating, successively solving the complete flow problem, and relocating the approximate surface. Alternatively, if it is not necessary to determine the position of the free surface, heads are approximated only at fixed nodal positions (the approach taken in this research).

Neuman and Witherspoon (1971) developed what is believed to be the earliest numerical model in which vertical gradients are not assumed to be negligible. Their transient, two-dimensional flow model is based on the finite element method. The free surface boundary is simulated by changing the location of the nodes at the top of the aquifer as the hydraulic head changes.

The Boundary Integral Equation Method (BIEM) has been employed by Liggett (1977) and Lennon et al. (1980) to resolve the free surface problem. The advantage of using the BIEM is that the flow equations at the free surface at the boundary depends only on boundary data and thus the free surface can be located without solving the complete flow problem (Liggett, 1977). However, the disadvantage of the BIEM is that hydrologic paramters are assumed to be constant over the entire domain. In addition, most BIEM theory has been developed for steady-state conditions only. Applications of the BIEM to subsurface hydrology problems are found in Huyakorn and Pinder (1983).

Trescott and Larson (1977) compared the efficiency of various iteration methods for simulating unconfined flow. Using a two- dimensional finite difference model, they found that the Strongly Implicit Procedure (SIP) was superior to Line Succesive Overrelaxtion

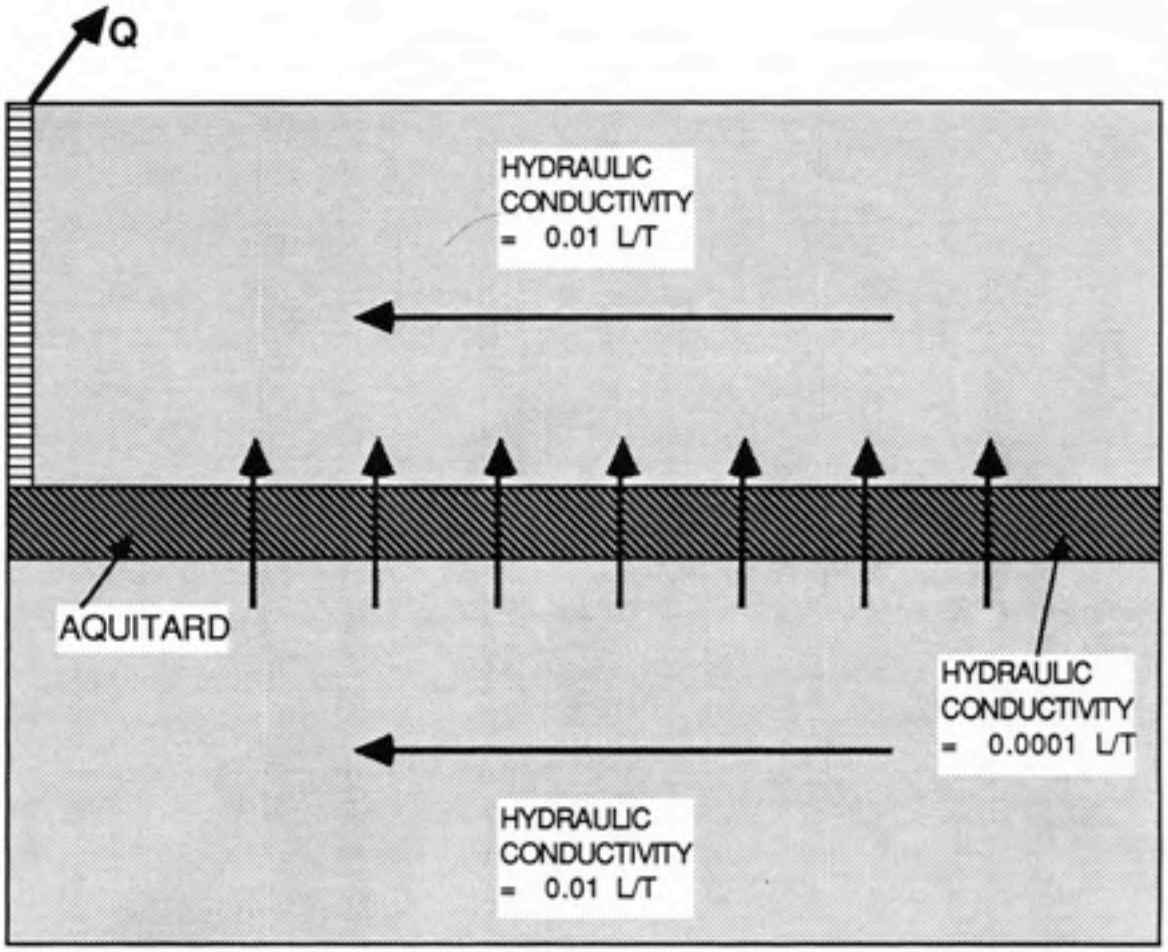
(LSOR) and the Alternating Direction Implicit procedure (ADI) for solving the nonlinear free-surface problem. Huyakorn and Pinder (1983) offer general procedures for solving nonlinear flow problems by iteration. These procedures include the Newton-Raphson and Picard iteration procedures.

Complexity and high computational cost are usually the reasons for avoiding three-dimensional analyses. Modeling transient unconfined flow in three dimensions is especially difficult because the top boundary of the water table must be moveable and because of the quasilinearity of the equations. However, as was discussed in Chapter 1, there are many instances where two-dimensional approaches are not adequate.

Three-dimensional groundwater flow models can be described as either "fully" three-dimensional, in order to distinguish them from "quasi" three-dimensional models. The fully three-dimensional models represent all dimensions of flow equally. The quasi three-dimensional models, however, take advantage of the fact that groundwater systems often consist of several aquifers separated by confining or semi-confining layers. These layers transmit water and interconnect the aquifers to various degrees. The contrast in permeability between the confining layers and the aquifers is usually several orders of magnitude. The system can be simplified by assuming that vertical components of flow within the aquifer are negligible and that the horizontal components of flow in the confining layers are negligible. Figure 2.1 shows how the presence of a semi-confining layer can affect flow in a layered system.

The quasi three-dimensional approach is attractive to many researchers because of the reduced computational costs resulting from the assumptions described above. Bredehoeft and Pinder (1970) used a finite difference scheme in their transient, quasi three-dimensional flow model. Finite elements were employed by Chorley and Frind (1978) in a transient, quasi three-dimensional model. They showed that their model required about 4al. (1982) developed a quasi three-dimensional flow model that also simulated land subsidence. The transient flow and subsidence problems were simulated with finite elements.

FIGURE 2.1
EFFECT OF SEMI-CONFINING LAYER ON GROUNDWATER FLOW



Gambolati et al. (1986) also used finite elements to simulate transient, quasi three- dimensional flow. Several three-dimensional numerical models have been proposed for flow analysis.

Fully three-dimensional flow models also have been developed. Freeze (1971) was perhaps the first researcher to develop a fully three-dimensional model of transient groundwater flow. He used a finite difference formulation to solve the groundwater flow equation. The model included the unsaturated zone and could accomodate both confined and unconfined conditions. Narasimhan and Witherspoon (1976) developed a general model for transient, three-dimensional flow based on the integrated finite difference approach. Confined and unconfined flows were included in the formulation of the model.

Trescott (1976) developed a transient, three-dimensional, finite difference flow model for confined aquifers. Winter (1978) derived a steady-state, three-dimensional, finite difference model to analyze the interaction between lakes and groundwater flow. An integrated model for flow and transport employing finite differences was developed by Reeves and Cranwell (1981). The SWENT model, developed by Intera Environmental Consultants (1983), simulates flow, energy and radionuclide transport. The model utilizes finite differences to solve the various equations. There are other models which couple flow with transport, but they do not have the ability to check flow results independently. (Anderson, 1979).

The USGS McDonald-Harbaugh model (McDonald and Harbaugh, 1984) is another transient three-dimensional flow model that uses finite differences. Of all the the three-dimensional flow models, it is the most fully documented and widely applied (International Ground Water Modeling Center, 1987).

The finite element method is another method for solving the groundwater flow equation. Narasimhan et al. (1978) employed three- dimensional finite elements to model unconfined flow, but assumed that flow was horizontal at or near the free surface, and thus could employ the Boussinesq equation (Equation II-7) at this location. Gupta and

Tanji (1976) used a three-dimensional finite element model in the analysis of flow in Sutter Basin, California. This model is mainly suited for steady-state flow (Frind and Verge, 1978). Huang and Sonnenfeld (1974) used three-dimensional finite elements to analyze the time-dependent drawdown in the vicinity of a well. Frind and Verge (1978) solved the unsaturated-saturated form of the groundwater flow equation. The model employs finite elements to simulate three-dimensional flow.

Gupta et al. (1984) developed the FE3DGW model, using a finite element scheme. This transient flow model was applied to the groundwater basin beneath Long Island, New York. Babu et al. (1982) produced a hybrid finite difference-finite element scheme for analyzing transient, three-dimensional flow. This scheme was employed later by Huyakorn et al. (1986). Gambolati et al. (1986) developed a three-dimensional, transient flow model. This model has the feature of automatically generating the finite element discretization scheme.

2.2.3 Indirect Velocity Estimation

Groundwater velocities can be estimated from simulated head distributions. Having obtained the head field, the velocity field is determined from Darcy's Law (Equation 2.1) by using some type of numerical differentiation. The advantage to the indirect estimation approach is that head distributions can be verified easily in the field. Heads can be measured at the desired spatial locations, in all spatial dimensions, with relatively simple equipment and procedures.

The numerical differentiation can be performed by using either the finite difference or finite element method. A simple example of numerical differentiation by finite differences is as follows.

$$v_x = \frac{1}{n} q_x \approx -\frac{1}{n} K_x \frac{\Delta h}{\Delta x} \tag{2.7}$$

where

Δx = distance between spatial location x_i and x_{i+1} (L)

Δh = change in hydraulic head from x_i to x_{i+1} (L)

n = porosity $(dimensionless)$

The differentiation is followed by averaging of hydraulic conductivities over a single finite element so that a continuous distribution of velocities is obtained. Pinder (1973), Reeves and Duguid (1975), and Segol (1976) simulated head distributions using finite element flow models. These researchers used numerical differention of the head distribution to produce velocities located at the center of each element. Pinder et al. (1981) and Abriola and Pinder (1982) introduced a finite element interpolation method to obtain a head gradient estimation in two and three dimensions. Because the interpolation function applied was linear, this approximation is essentially the same as the numerical differentiation of the previous work.

However, when applying the differentiation approach to heads obtained by conventional finite element methods, there is a resulting discontinuity in the velocity at nodal points and element boundaries (Yeh, 1981). The discontinuity leads to a violation of the conservation of mass around a single element. In areas where there are significant variations in hydraulic conductivity, the resulting error can range from very small to several hundred percent (Yeh, 1981). In addition, applying the approach to aquifers with low hydraulic gradients can result in roundoff errors that produce spurious gradients (Frind et al., 1985).

Because of the problems with the differentiation approach, some researchers have introduced methods of estimating velocities from head distributions that somewhat overcome these inaccuracies. Yeh (1981) applied the finite element method to the velocity field, after obtaining the head field with the same finite element method. The velocity field is then continuous and the mass balance error is reduced (Yeh, 1981). Batu (1984)

proposed creating a "dual" discretization mesh for estimating velocities. In this method a second discretization scheme for estimating velocities is created that is shifted away from the discretization scheme used to estimate heads. This approach somewhat avoids the discontinuity problem and satifies the conservation of mass principles to an acceptable degree (Batu, 1984).

2.2.4 Direct Mehtods of Obtaining Velocity

The direct estimation of groundwater velocities is a relatively new approach. Direct estimation of velocities avoids the mass balance and discontinuity problems described above. However, the results obtained from a direct method are not easily verifiable in the field. Currently, the instrumentation available for measuring velocities in groundwater relies on sending heat pulses out through the water and measuring the time it takes for those pulses to reach a heat sensing device. This type of instrumentation produces an unacceptable degree of error. Tracer tests are unreliable for predicting velocities because of dispersion effects. Examples of this approach are scarce, due to the newness of the approach and the difficulty of field verification.

Segol et al. (1975) presented an approach where finite element theory is used to obtain the head and velocity fields simultaneously, by carrying the derivative terms for velocity through the finite element estimation. Zijl (1984) applied a non-porous media fluid dynamics approach where pressure (or hydraulic head) is eliminated and a set of equations for the vorticity and vector potentials is produced. The vector potentials are applied to Darcy's Law, resulting in a velocity vector field. This method required fewer computer operations and less computer storage to solve a flow problem to the same accuracy as a hydraulic head estimator (Zijl, 1984).

A streamline and equipotential approach was taken by Frind and Matanga (1985). Galerkin finite elements were applied to stream and potential functions. This method is especially suited for aquifers with low gradients (Frind et al., 1985). However, stream

functions can only provide velocities for steady-state conditions. Zijl (1986) applied both the stream function and a direct velocity approach. Derivation of the velocity expressions was performed by vector analysis.

A more general approach to the problem is to employ Hermite finite elements (Van Genuchten et al., 1977). This type of finite element provides continuity at the element nodes for higher-order derivatives, and can provide solutions for groundwater gradients at the nodes. However, the computational effort required to simulate a groundwater flow problem with Hermite finite elements can be prohibitive.

3 DEVELOPMENT OF CONFINED FLOW MODEL

3.1 Overview of Model Algorithm

The development of any model upon which engineering decisions are to be based should be founded on a set of engineering criteria. The first step in developing the confined flow model of this research is to select a basic algorithm for solution of the flow equations. From the discussion in Chapter 1, several of criteria concerning the model algorithm can be formalized. The criteria can be stated as

- The algorithm should provide accurate solutions
- The algorithm should be able to represent the true nature of the physical system, e.g. fully three-dimensional representation

In addition to the above, there are other criteria which should be applied to any algorithm that is to be used in a groundwater flow model:

- The algorithm should utilize state -of-the-art procedures
- The algorithm should be computationally efficient (in terms of speed and storage requirements)
- The algorithm should be flexible, e.g. be able to adapt to irregular boundaries, multiple stresses, etc.

The literature review in Chapter 2 identified the various methods available for solving the three-dimensional groundwater flow equation. These methods included finite differences, finite elements, finite element-finite difference hybrids, and boundary integral equation methods (BIEM).

From the discussion in Chapter 2, it is evident that the hybrid finite element-finite difference method is suitable for solving three-dimensional groundwater flow problems. This hybrid method has been developed into an algorithm by Babu and Pinder (1982),

and later refined by Huyakorn et al. (1986). The algorithm is best known by its acronym, ALALS, for ALternate sublayer And Line Sweep. The ALALS algorithm is designed to solve transient groundwater flow problems in three dimensions.

The algorithm employs a finite element method in the areal plane, and a finite difference method in the vertical dimension. The algorithm is especially suited for multilayer systems because it maintains the inherent flexibility of the finite element discretization in the areal plane, where it is needed most.

The algorithm allows for the uncoupling of the vertical equations while the areal equations are being solved, thus making it computationally more efficient than other fully three-dimensional algorithms. This efficiency has been demonstrated by Huyakorn (1986) a model devloped from the ALALS algorithm is compared to a two-dimensional and a three-dimensional finite element model. The ALALS model required considerably less CPU time to simulate a sample problem than either of the two finite element models.

The derivation of the ALALS algorithm, and a discussion of additional refinements included in the confined flow model are found in the following sections.

3.2 Derivation of Algorithm

Transient groundwater flow in a confined aquifer is described by:

$$\frac{\partial}{\partial x}\left(K_x\frac{\partial h}{\partial x}\right)+\frac{\partial}{\partial y}\left(K_y\frac{\partial h}{\partial y}\right)+\frac{\partial}{\partial z}\left(K_z\frac{\partial h}{\partial z}\right)+\Gamma(h)=S_s\frac{\partial h}{\partial t} \tag{3.1}$$

This equation can be solved by combining the Galerkin finite element method and the finite difference method (Huyakorn et al. 1986). The finite difference and finite element methods are reviewed in Appendix 1. In this case, a three-dimensional aquifer region is divided into a number of layers, and each layer is subdivided into a number of elements, as shown in Figure 3.1. Although triangular-shaped elements are applied here, other shapes

or element types may also be applied by substituting the appropriate basis functions. For some special cases of boundary or other conditions different elements may be more suitable.

The discretization is performed so that each sublayer has the same projected area in the x-y plane. The resulting three-dimensional elements need to have planar vertical sides, but the bases and tops do not need to be parallel to each other. The discretization thus allows for layering that is not necessarily parallel to the x-y plane. By dividing the three-dimensional region into sublayers, finite elements can be applied to the individual sublayers. Thus, finite elements are applied only in the x-y plane.

The first step in the finite element procedure is to approximate (hydraulic head) by a trial function:

$$h(x,y,z,t) \approx \hat{h}(x,y,z,t) = \sum_{\mathrm{n}_{xy}} N_{\mathrm{n}}(x,y) h_{\mathrm{n}}(z,t) \tag{3.2}$$

where

h = hydraulic head (L)

$\hat{h}$ = trial function for hydraulic head (L)

$N_{\mathrm{n}}(x,y)$ = two-dimensional basis function in the x-y plane

h_{n} = nodal parameter dependent on z and time (L)

n_{xy} = number of nodes in the x-y plane of each layer

Applying the Galerkin criterion over the x-y plane, the weighted residual approximation of Equation 3.1 becomes

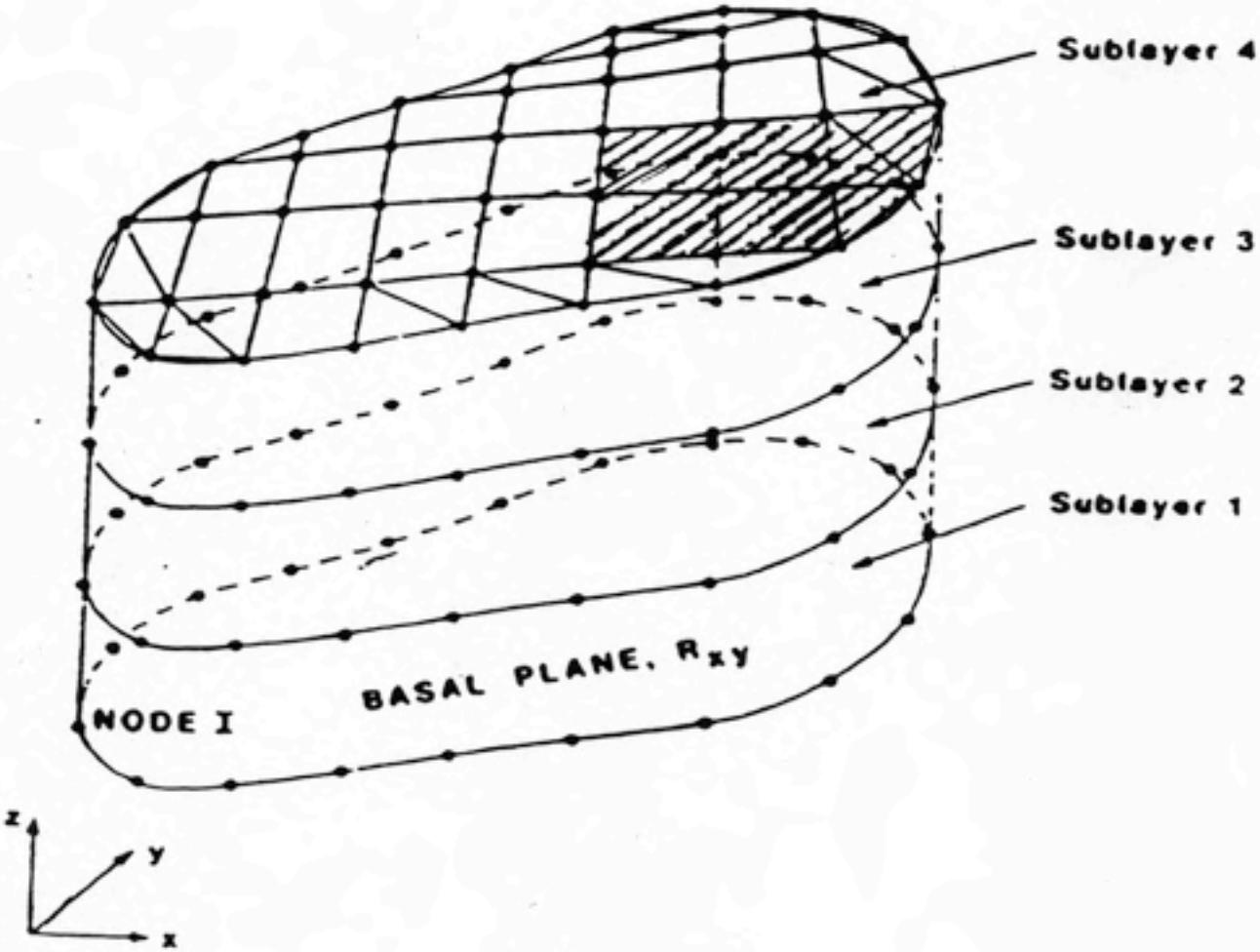
$$\iint_{\mathcal{R}} N_{\mathrm{i}} \left(\frac{\partial}{\partial x} \left[K_x \frac{\partial \hat{h}}{\partial x} \right) + \frac{\partial}{\partial y} \left(K_y \frac{\partial \hat{h}}{\partial y} \right) + \frac{\partial}{\partial z} \left(K_z \frac{\partial \hat{h}}{\partial z} \right) + \Gamma(h) - S_s \frac{\partial \hat{h}}{\partial t} \right] dx\, dy = 0$$

for $\mathrm{i} = 1, \ldots, \mathrm{n}_{xy}$

(3.3)

FIGURE 3.1

THREE-DIMENSIONAL DISCRETIZATION OF AQUIFER DOMAIN



where

N_{i} = two-dimensional basis function in the x-y plane

$\mathcal{R}$ = x-y problem domain

The cross-sectional area $\mathcal{R}$ over which the integration in (3.3) is performed is assumed to remain unchanged in the z- direction. This assumption allows for the use of a single discretization in the x-y plane. Substitution of (3.2) into (3.3) yields

$$\iint_{\mathcal{R}} N_{\mathrm{i}}\left[\frac{\partial}{\partial x}\left[K_x\frac{\partial}{\partial x}\left(\sum_{\mathrm{n}_{xy}} N_{\mathrm{n}}h_{\mathrm{n}}\right)\right]+\frac{\partial}{\partial y}\left[K_y\frac{\partial}{\partial y}\left(\sum_{\mathrm{n}_{xy}} N_{\mathrm{n}}h_{\mathrm{n}}\right]\right)+\frac{\partial}{\partial z}\left(K_z\frac{\partial \hat{h}}{\partial z}\right)\right.$$
$$\left.+\Gamma(h)-S_s\frac{\partial}{\partial t}\left(\sum_{\mathrm{n}_{xy}} N_{\mathrm{n}}h_{\mathrm{n}}\right)\right]\,dx\,dy=0 \tag{3.4}$$
$$\text{for } \mathrm{i}=1,\ldots,\mathrm{n}_{xy}$$

Integration by parts using Green's Theorem reduces the order of the highest derivatives. This operation gives

$$\iint_{\mathcal{R}}\left(K_x\frac{\partial N_{\mathrm{i}}}{\partial x}\frac{\partial \hat{h}}{\partial x}+K_y\frac{\partial N_{\mathrm{i}}}{\partial y}\frac{\partial \hat{h}}{\partial y}\right)\,dx\,dy+\iint_{\mathcal{R}} N_{\mathrm{i}}S_s\frac{\partial \hat{h}}{\partial t}\,dx\,dy$$
$$=\iint_{\mathcal{R}} N_{\mathrm{i}}\frac{\partial}{\partial z}\left(K_z\frac{\partial \hat{h}}{\partial z}\right)\,dx\,dy+\iint_{\mathcal{R}} N_{\mathrm{i}}\Gamma(h)\,dx\,dy+\oint_{\mathcal{B}} N_{\mathrm{i}}\left(K_n\frac{\partial \hat{h}}{\partial n}\right)\,d\mathcal{B} \tag{3.5}$$
$$\text{for } \mathrm{i}=1,\ldots,\mathrm{n}_{xy}$$

where

$\mathcal{B}$ = boundary of the cross-section of $\mathcal{R}$

$\frac{\partial \hat{h}}{\partial n}$ = outward normal derivative on $\mathcal{B}$

K_n = normal component of hydraulic conductivity on $\mathcal{B}$

A finite difference approximation is applied to the z-derivative terms of (3.5), using a central difference, block-centered approach. The block-centered approach refers to the location of the nodes in the finite difference approximation and is illustrated in Figure 3.2. By using a block-centered approach, discontinuities in hydraulic conductivity can be treated by taking a harmonic mean of the conductivity divided by the layer thickness. This approach reduces the z-derivative terms to

$$
\begin{aligned}
\frac{\partial}{\partial z}\left(K_z \frac{\partial \hat{h}}{\partial z}\right) &= \frac{\partial}{\partial z}\left(K_z \frac{\partial}{\partial z} \sum_{\mathrm{n}=1}^{\mathrm{N}} N_{\mathrm{n}} h_{\mathrm{n}}\right) \\
&= \frac{\partial}{\partial z}\left(K_z \sum_{\mathrm{n}=1}^{\mathrm{N}} N_{\mathrm{n}} \frac{\partial h_{\mathrm{n}}}{\partial z}\right) \\
&= \frac{\partial}{\partial z}\left(K_z \sum_{\mathrm{n}=1}^{\mathrm{N}} N_{\mathrm{n}} \frac{h_{\mathrm{n,k+1/2}} - h_{\mathrm{n,k-1/2}}}{\Delta z}\right) \\
&= \sum_{\mathrm{n}=1}^{\mathrm{N}} N_{\mathrm{n}} \left[K_{z+}\left(\frac{h_{\mathrm{n,k+1}} - h_{\mathrm{n,k}}}{\Delta z_{+} \Delta z}\right) - K_{z-}\left(\frac{h_{\mathrm{n,k}} - h_{\mathrm{n,k-1}}}{\Delta z_{-} \Delta z}\right)\right]
\end{aligned}
\tag{3.6}
$$

where

indices $\mathrm{k} + \frac{1}{2}, \mathrm{k} - frac12, \mathrm{k}+1$, and $\mathrm{k}-1$ are as shown in Figure 3.2

Δz terms are as shown in Figure 3.2 (L)

K_{z+} = upper-weighted, harmonic-mean hydraulic conductivity (L/T)

K_{z-} = lower-weighted, harmonic-mean hydraulic conductivity (L/T)

and harmonic mean is defined as $K_z = \frac{d}{\sum_{i=1}^{n} \frac{d_i}{K_i}}$ where

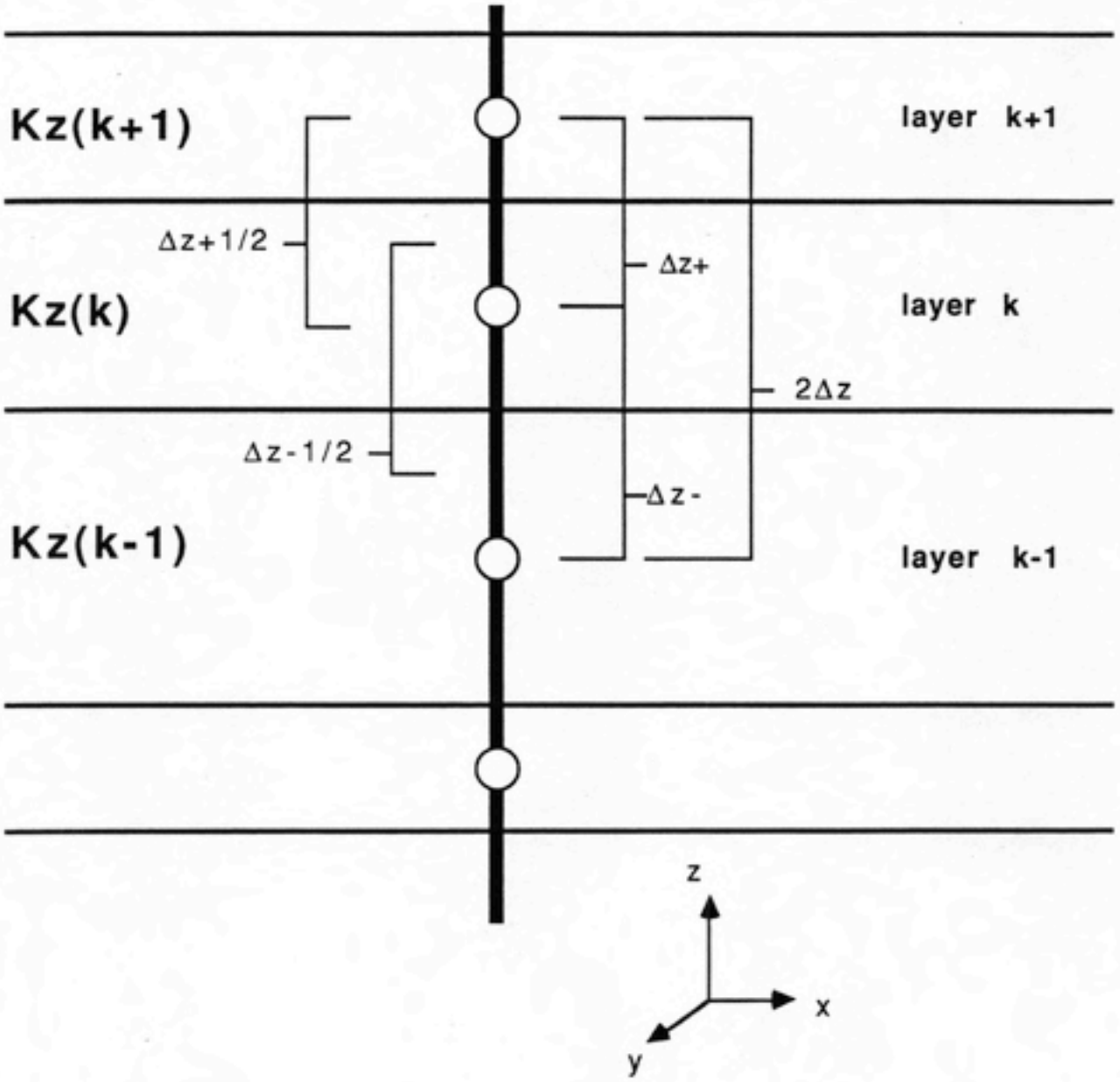
d = total thickness (L)

d_i = thickness of individual layer (L)

K_i = hydraulic conductivity in indivdual layer (L/T)

Equation 3.6 can also be written as

FIGURE 3.2
BLOCK-CENTERED APPROACH



$$
\begin{aligned}
&\sum_{n=1}^{N} N_n \left[K_{z+} \left(\frac{h_{n,k+1} - h_{n,k}}{\Delta z_+ \Delta z} \right) - K_{z-} \left(\frac{h_{n,k} - h_{n,k-1}}{\Delta z_- \Delta z} \right) \right] \\
&= \sum_{n=1}^{N} N_n \left(\frac{K_{z+}}{\Delta z_+ \Delta z} \right) h_{n,k+1} - \sum_{n=1}^{N} N_n \left(\frac{K_{z+}}{\Delta z_+ \Delta z} + \frac{K_{z-}}{\Delta z_- \Delta z} \right) h_{n,k} \\
&\quad + \sum_{n=1}^{N} N_n \left(\frac{K_{z-}}{\Delta z_- \Delta z} \right) h_{n,k-1}
\end{aligned}
\tag{3.7}
$$

The next step is to approximate the temporal derivatives using finite differences. An explicit, forward-difference approximation provides a first-order correct approximation. Applying this approximation yields

$$
\frac{\partial h}{\partial t} \approx \frac{\partial \hat{h}}{\partial t} = \frac{\partial}{\partial t} \sum_{n=1}^{N} N_n h_n(z,t) = \sum_{n=1}^{N} N_n \frac{\partial h_n}{\partial t} = \sum_{n=1}^{N} N_n \frac{h_n^{l+1} - h_n^{l}}{\Delta t} \tag{3.8}
$$

where

l = index for present time step

$l + 1$ = index for next time step

Δt = time increment for time step (T)

The remaining derivative terms can be expressed as follows

$$
\frac{\partial h}{\partial x} \approx \frac{\partial \hat{h}}{\partial x} = \frac{\partial}{\partial x} \sum_{n=1}^{N} N_n h_n(z,t) = \sum_{n=1}^{N} h_n(z,t) \frac{\partial N_n}{\partial x} \tag{3.9}
$$

$$
\frac{\partial h}{\partial y} \approx \frac{\partial \hat{h}}{\partial y} = \frac{\partial}{\partial y} \sum_{n=1}^{N} N_n h_n(z,t) = \sum_{n=1}^{N} h_n(z,t) \frac{\partial N_n}{\partial y} \tag{3.10}
$$

The terms from Equations 3.7 through 3.10 can be substituted into Equation 3.5.

By time-lagging the z-component terms, the original set of n_{xy} by n_z terms are split into n_z subsets of equations, each of which contains n_{xy} equations. Time-lagging the z-component terms implies that these terms are evaluated at the old (l) time step, while the other components are evaluated at the new (l+1) timestep. The resulting equations can be split into two parts, the first representing a prediction of the approximate values of head at the new (l+1) timestep, and the second representing the corrected approximate values of head at the new (l+1) timestep. Splitting the equations in this manner allows for computations in the x-y plane to be separated from computations for the z-direction, thus easing the computational burden.

Instead of explicitly writing all terms of the two equations, they can be represented in matrix form as

$$\begin{aligned}&[KH]_k\{\vec{h}\}_k^{(l+1)^*} + \frac{[ST]_k}{\Delta t}\left(\{\vec{h}\}_k^{(l+1)^*} - \{\vec{h}\}_k^l\right)\\ =&\{\vec{\Gamma}(h)\}_k^{(l+1)^*} + \{\vec{F}(h)\}_k^{(l+1)^*} + (KL\cdot h)_{k-1}^l - [(KU+KL)\cdot h]_k^l + (KU\cdot h)_{k+1}^l\end{aligned} \tag{3.11}$$

and

$$\begin{aligned}&[KH]_k\{\vec{h}\}_k^{(l+1)^*} + \frac{[ST]_k}{\Delta t}\left(\{\vec{h}\}_k^{(l+1)} - \{\vec{h}\}_k^l\right)\\ =&\{\vec{\Gamma}(h)\}_k^{(l+1)^*} + \{\vec{F}(h)\}_k^{(l+1)^*} + (KL\cdot h)_{k-1}^{l+1} - [(KU+KL)\cdot h)_k^{l+1} + (KU\cdot h)_{k+1}^{l+1}\end{aligned} \tag{3.12}$$

where

index * refers to predicted solutions

and

$$[KH]\{\vec{h}\} = \iint_{\mathcal{R}} \left(K_x \frac{\partial N_i}{\partial x}\frac{\partial \hat{h}}{\partial x} + K_y \frac{\partial N_i}{\partial y}\frac{\partial \hat{h}}{\partial y}\right)\, dx\, dy$$

$$\frac{[ST]}{\Delta t}\left(\{\vec{h}\}^{(i+1)} - \{\vec{h}\}^{i}\right) = \iint_{\mathcal{R}} N_i S_s \frac{\partial \hat{h}}{\partial t}\, dx\, dy$$

$$\{\vec{\Gamma}(h)\} = \iint_{\mathcal{R}} N_i \Gamma(h)\, dx\, dy$$

$$\{\vec{F}(h)\} = \oint_{\mathcal{B}} N_i \left(K_n \frac{\partial \hat{h}}{\partial n}\right)\, d\mathcal{B}$$

$$(KL \cdot h)_{k-1} - [(KU + KL) \cdot h]_k + (KU \cdot h)_{k+1} = \iint_{\mathcal{R}} N_i \frac{\partial}{\partial z}\left(K_z \frac{\partial \hat{h}}{\partial z}\right)\, dx\, dy$$

The model considered in this paper utilizes linear triangular elements in the x-y plane, as shown in Figure 3.3. The basis functions for this type of element are as follows.

$$N^e_{n_i} = \frac{1}{2A_e}[(x_{n_j} y_{n_m} - x_{n_m} y_{n_j}) + (y_{n_j} - y_{n_m})x + (x_{n_m} - x_{n_j})y]$$

$$N^e_{n_j} = \frac{1}{2A_e}[(x_{n_m} y_{n_i} - x_{n_i} y_{n_m}) + (y_{n_m} - y_{n_i})x + (x_{n_i} - x_{n_m})y]$$

$$N^e_{n_m} = \frac{1}{2A_e}[(x_{n_i} y_{n_j} - x_{n_j} y_{n_i}) + (y_{n_i} - y_{n_j})x + (x_{n_j} - x_{n_i})y] \tag{3.37}$$

where

$n_i, n_j,$ and n_m = nodal indices on triangular element

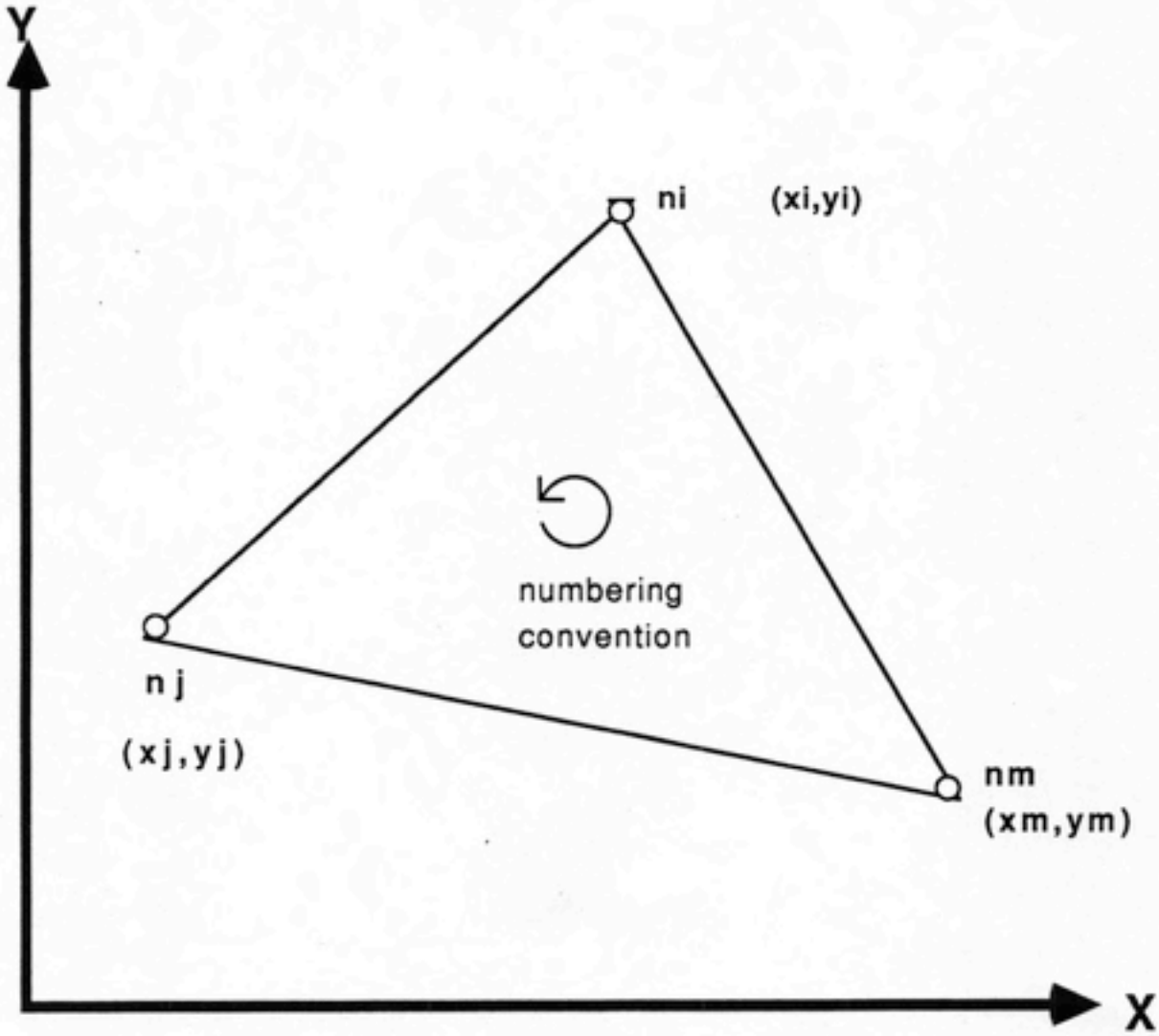
$x_i, x_j, x_m, y_i, y_j,$ and y_m = coordinates of triangle vertices (L)

A_e = area of triangle (L^2)

These basis functions are substituted into Equations 3.11 and 3.12 and subsequently differentiated and integrated.

Equations 3.11 and 3.12 give the matrix form solution of the groundwater flow equation in three dimensions. These equations can be solved for the hydraulic head distribution in a two-step procedure. However, there are further refinements which can simplify the

FIGURE 3.3
LINEAR TRIANGULAR ELEMENT



computational procedure. The first step is to lump-diagonalize the [ST] matrix, which contains the storage terms, and the [KU], [KU+KL], and [KL] matrices, which contain the vertical flow terms. The algorithm used to lump-diagonalize is as follows.

$$a_{ii} = \sum_{j} a_{ij} \ , \quad a_{ij} = 0 \ \text{ for i} \neq \text{j} \tag{3.13}$$

where

$a_{ij} = i, j$ element of $[A]$ matrix

This approach greatly simplifies the computation of the two equations (3.11 and 3.12). However, the lump-diagonalizing procedure implicitly assumes that the values of terms in [A] do not differ signifcantly over the nodes of a single element. The set of equations in (3.11) can be termed the predictor equations.

The first stage of the solution procedure amounts to a layer-by-layer solution of the predictor equations for $h^{(l+1)*}$. After the sublayer-sweeping operation has been completed, the second stage of the algorithm is achieved by solving Equation 3.12 for $h^{(l+1)}$, the corrective version of the flow equation. It is apparent that there are several terms in (3.11) that are identical to those in (3.12). There is no need to solve the entirety of both equations. The repeated terms can be eliminated by taking the difference of (3.11) and (3.12), thus obtaining

$$\begin{aligned} &\frac{[ST]_k}{\Delta t}\left(\{\vec{h}\}_k^{(l+1)} - \{\vec{h}\}_k^{(l+1)^*}\right) \\ =&(KL \cdot h)_{k-1}^{l+1} - [(KU + KL) \cdot h]_k^{l+1} + (KU \cdot h)_{k+1}^{l+1} \\ -&(KL \cdot h)_{k-1}^{l} + [(KU + KL) \cdot h)_k^{l} - (KU \cdot h)_{k+1}^{l} \end{aligned} \tag{3.14}$$

The overall coefficient matrix on the right-hand-side of Equation 3.14 can be made tridiagonal if the matrix [ST] is lump- diagonalized and the matrices [KU], [KU+KL],

and [KL] are lump-diagonalized. A highly efficient tridiagonal solver such as the Thomas Algorithm can be used to solve (3.14). The second stage of the computational procedure thus involves solving n_{xy} subsets of equations, with each subset containing n_z equations with n_z unknowns. The resulting solutions for $h^{(l+1)}$ are the current hydraulic head values at the nodes on a vertical line along the complete thickness of the aquifer domain.

The computational procedure for setting up and solving the predictor and corrector equations is summarized in Figure 3.4. The procedure is repeated for each time step until the maximum number of timesteps (specified by the user) is reached. No iterations within the timestep are necessary for the confined flow case, because direct solution procedures are used.

3.3 Application of Boundary Conditions and Source and Sink Terms

Suitable boundary conditions and source or sink terms can be applied to the ALALS algorithm. The most commonly applied boundary conditions for groundwater flow problems are the Neumann or Dirichlet boundary conditions. Typical boundary conditions and sources or sinks are shown in Figure 3.5.

The Neumann boundary condition can be generalized as

$$\left.\frac{\partial u(x,y,x,t)}{\partial n}\right|_{\mathcal{B}} = g(x,y,z,t) \tag{3.15}$$

where

n = outward normal vector

$\mathcal{B}$ = problem boundary

g = arbitrary boundary function

The specialized Neumann condition of no flow is implicitly applied in the x-y plane

FIGURE 3.4
FLOW DIAGRAM FOR ALALS ALGORITHM

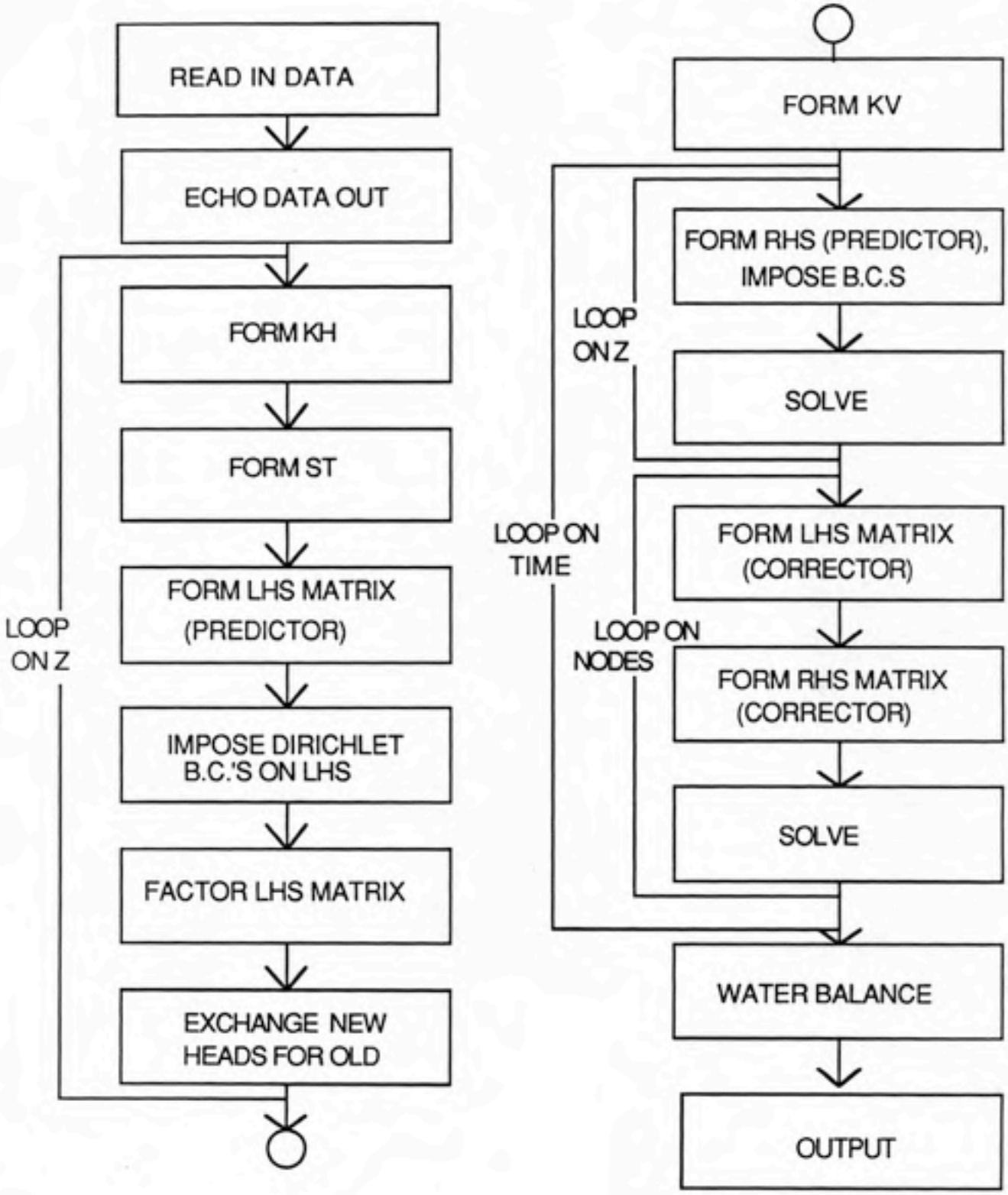
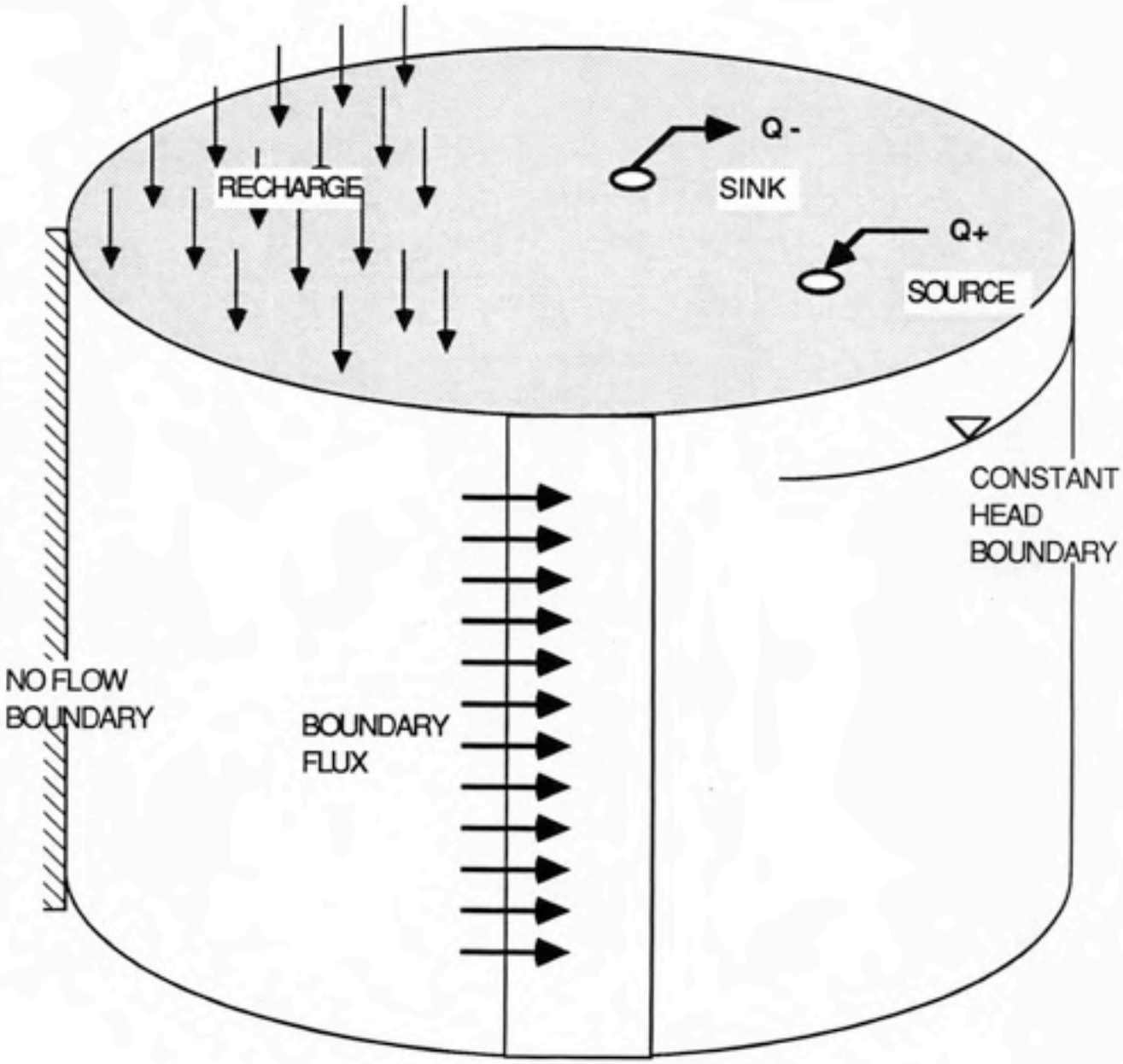


FIGURE 3.5
TYPICAL BOUNDARY CONDITIONS, SOURCES AND SINKS FOR GROUNDWATER FLOW



by simply placing the free edge of an element on the relevant boundary. In the z-direction, no flow conditions are imposed by setting the relevant portion of the the vertical flow component equal to zero (see Equation 3.11). For example, at the top layer of an aquifer system, the condition of no flow onto the top reduces the expression for vertical flow components from Equation 3.11 as follows.

$$\iint_{\mathcal{R}} N_{\mathrm{i}} \frac{\partial}{\partial z}\left(K_z \frac{\partial \hat{h}}{\partial z}\right)\, dx\, dy = (KL \cdot h)_{k-1}^{l} - [(KU + KL) \cdot h)_{k}^{l} \tag{3.16}$$

Fluxes into boundary elements can be applied in all three dimensions by integrating the flux over the relevant element and applying the resultant to the nodes of the element. The hydrologic quantity of recharge is an example of a boundary flux that may be applied in groundwater flow. The integration is analogous to the boundary term found in Equation 3.5. Recharge can be handled as follows

$$\iint_{\mathcal{R}} \Gamma_{\mathrm{r}} N_{\mathrm{i}} \;\; dx \;\; dy = \text{recharge} \tag{3.17}$$

where

Γ_{r} = recharge rate (L/T)

Dirichlet boundary conditions can be generalized as

$$u_B(x, y, z, t) = f(x, y, z, t) \tag{3.18}$$

where

u_B = problem boundary

f = arbitrary boundary function

In groundwater flow, a Dirichlet boundary condition usually implies that a hydraulic head, or set of heads, is constant over specified boundary nodes. The boundary nodes are accomodated by operating on the global coefficient matrix to ensure that the solution of the resultant equations satisfies the constant head conditions. The conditions may be satisfied by forcing the boundary nodes to possess a value of one in the relevant location in the global coefficient matrix, and forcing the remaining terms in that location to be equal to zero. The right hand side of the equation is forced to be equal to f from Equation 3.18.

Source or sink terms are applied in a manner similar to boundary fluxes, except that they are placed at any node throughout the three-dimensional domain. For example, recharge is essentiallly a source term, but it is applied as a boundary flux. Withdrawal or injection wells are examples of point sink or source terms that are quite common in groundwater flow. These terms are applied simply by subtracting or adding the relevant quantity from right or left hand side of the flow equations found in Equation 3.11 and 3.12, or

$$\pm[\Gamma_p(h)] = \iint_{\mathcal{R}} \Gamma_p N_i \ dx \ dy \tag{3.19}$$

where

Γ_p = withdrawal or injection quantity (L^3/T)

3.4 Matrix Solution Methods

The solution of Equation 3.13 requires a solution to the generic matrix problem

$$[A]\{\vec{h}\} = \{\vec{b}\} \tag{3.20}$$

where

$[A]$ = sum of left-hand-side matrices (known)

$\{\vec{h}\}$ = the solution, or hydraulic head vector (unknown)

$\{\vec{b}\}$ = sum of right-hand-side vectors (known)

The Gaussian Elimination algorithm is used to solve this system of equations. After the matrix [A] is formed, it is factored into a upper triangular matrix, which can be saved as long as no changes in transmissivity or storage terms occur after the first time step (as in the confined case). A backward substitution procedure is used after the vector b is formed, in order to solve for x. The fact that [A] is a banded matrix is taken advantage of with the Gaussian Elimination solver, thus reducing computational time and storage requirements. More details on the Gaussian Elimination algorithm can be found in Strang (1986).

Solution of Equation (III-36) is similar, except that in this case, the tridiagonal nature of the matrix [A] allows for the use of a more efficient solution algorithm. The Thomas algorithm, a variation of Gaussian Elimination, is most suitable for this problem. A detailed explanation of the Thomas algorithm can be found in Wang and Anderson (1982).

3.5 Other Model Features

3.5.1 Steady-State Case

Groundwater flow at steady-state is often an important case. Steady state flow can be approximated by the model simply by increasing time until the change in hydraulic heads between previous and current timesteps becomes insignificant. However, this process can

be time-consuming or lead to innacuracies if large timesteps are used. A more appropriate way to model steady-state flow is to set the term containing derivatives with respect to time equivalent to zero, or

$$\frac{[ST]}{\Delta t}(h^{(l+1)} - h^l) = 0 \tag{3.21}$$

Only one timestep is required to solve this problem, but of course with unconfined flow, iterations may be required before convergence is achieved. Convergence to a steady-state solution for the unconfined case may be accelerated by using the square of the hydraulic heads for boundary conditions and initial guesses of heads at interior nodes. Flux terms must be multiplied by a factor of two. The square root of the resultant head distribution provides the correct solution. The modfication of the algorithm for the steady-state case is shown in Figure 3.6.

3.5.2 Water Balance Error

The water balance error is a measure of how well the model can balance the changes in mass, or water in the case of groundwater flow. The concept that underlies the mass or water balance is that mass is conserved throughout the model system. In groundwater flow with a source or sinks term, the conservation of mass can be stated as

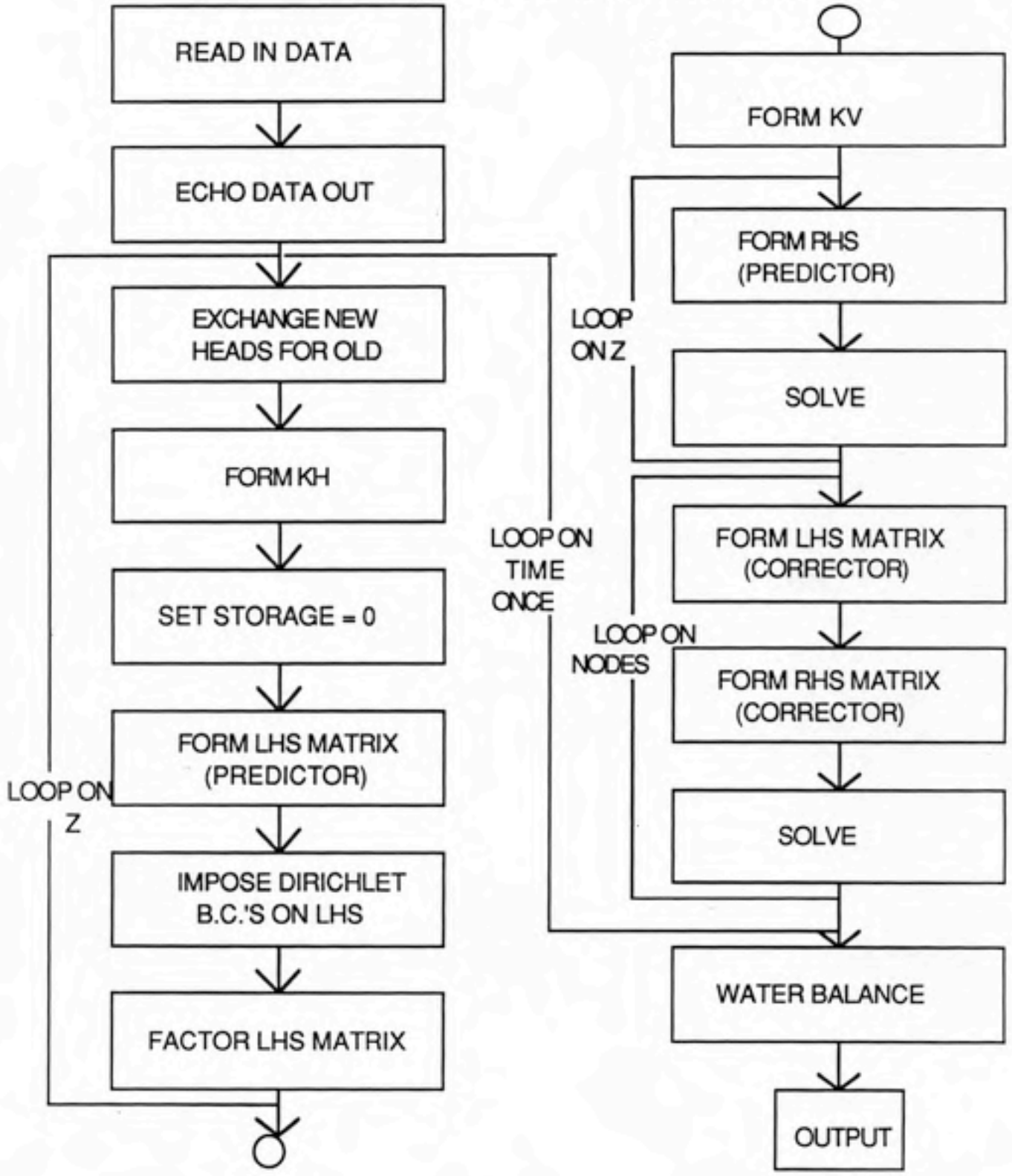
[volume of water in or out from source or sink terms]

=[volume of water released from aquifer storage]

or in mathematical terms (over an individual element)

$$t_T\Big[\sum^{sinks} Q_{sink} - \sum^{sources} Q_{sources}\Big] = t_T S\left(\frac{h_f - h_i}{\Delta t}\right)\iint_e N_i \, dx \, dy \tag{3.22}$$

FIGURE 3.6
STEADY STATE MODIFICATION



where

t_T = total time elapsed over simulation (T)

Thus the water balance error for an individual element is calculated as

$$\varepsilon_{wb} = \frac{t_T\left[\sum^{sinks} Q_{sink} - \sum^{sources} Q_{sources}\right]}{t_T S\left(\frac{h_f - h_i}{\Delta t}\right)\iint\limits_{e} N_i \, dx \, dy} - 1 \tag{3.23}$$

where

ε_{wb} = water balance error (*dimensionless*)

h_i = head at start of simulation (L)

h_f = head at end of simulation (L)

The model calculates the water balance over the entire domain after the final timestep by summing the errors from individual elements.

4 DEVELOPMENT OF UNCONFINED FLOW MODEL

In this chapter, the development of an unconfined flow model is discussed. The unconfined flow model is based on the algorithm and model developed in the previous chapter, which does not accomodate unconfined flow in its current state. The resulting flow model, REGFED (for REGional flow using Finite Element and Difference methods), is capable of simulating confined and unconfined flow.

The accomodation of unconfined flow decreases the efficiency of the model (as compared to confined flow only), due to an increase in the number of operations needed to produce a solution. The efficiency can be improved by refining the unconfined flow algorithm, however. A discussion of this refinement and other needed modifications is found in the following sections.

4.1 Overview of Unconfined Flow Modeling

In order for the model to accomodate unconfined flow, Equations 4.11 and 4.12 must be modified. The horizontal flow components are modified first. In the x-y plane, the parameters of conductivity and storativity must be vertically averaged to simulate unconfined flow. Vertical averaging yields horizontal transmissivities and storage terms consisting of storativity or specific yield (for confined or unconfined flow, respectively). Thus the [KH] and [ST] terms become

$$[KH]'\{\vec{h}\} = \iint_{\mathcal{R}} \left(T_x \frac{\partial N_i}{\partial x} \frac{\partial \hat{h}}{\partial x} + T_y \frac{\partial N_i}{\partial y} \frac{\partial \hat{h}}{\partial y} \right) dx\, dy \tag{4.1}$$

where

T = transmissivity = Kb (L^2/T)

where b = saturated thickness in aquifer layer (L)

and

$$\frac{[ST]'}{\Delta t}\{h^{l+1} - h^l\} = \iint_{\mathcal{R}} N_i S \frac{\partial \hat{h}}{\partial t}\, dx\, dy \tag{4.2}$$

where

S = specific yield (unconfined aquifer) or storativity (confined aquifer) (*dimensionless*)

The vertical flow components also can be vertically averaged. However, the quantity of vertical transmissivity is meaningless in groundwater flow. Instead of producing a vertical transmissivity term, vertical averaging produces a source or sink term for each layer. In this case vertical averaging simply reduces the order of the second-order spatial derivatives. The KL and KU terms thus become

$$KU' = K_{z+} \frac{h_{k+1} - h_k}{\Delta z_+} \frac{A_e}{3}$$

and

$$KL' = K_{z-} \frac{h_k - h_{k+1}}{\Delta z_-} \frac{A_e}{3} \tag{4.3}$$

Solution of these new equations modified for unconfined flow provides values of hydraulic head at fixed locations, and thus the location of the free surface is not known.

The equations that result when Equations 4.1 and 4.2 are substituted into Equations 3.11 and 3.12 are no longer linear, and thus cannot be solved directly for hydraulic heads.

These equations are called quasi-linear. The quasi-linearity is a result of transmissivity being a function of the aquifer saturated thickness. Figure 4.1 illustrates the difference between confined and unconfined flow conditions with respect to saturated thickness. The saturated thickness is essentially equivalent to hydraulic head.

The quasi-linear equations can be represented in matrix form as

$$[A]\{\vec{h}\} = \{\vec{b}\} \tag{4.4}$$

where

$[A]$ = sum of left-hand-side matrices (unknown)

$\{\vec{h}\}$ = the solution, or hydraulic head vector (unknown)

$\{\vec{b}\}$ = sum of right-hand-side vectors (known)

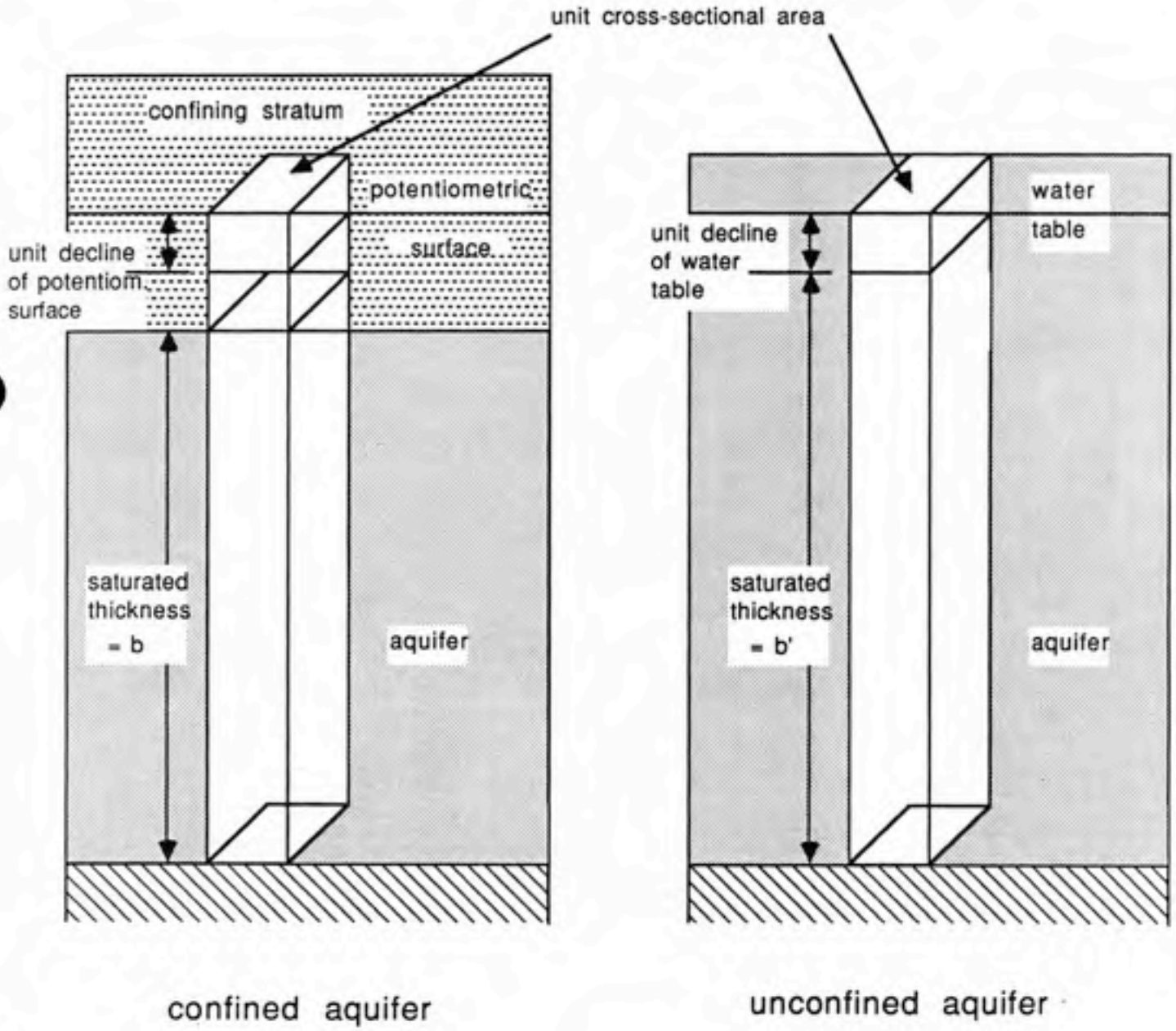
This problem can be solved by iterating over the equations within each timestep. There are a variety of iterative methods available. These methods include Picard iteration and Newton-Raphson iteration schemes. Both schemes require an initial estimate of the solution at the start of a timestep, but the two schemes differ in how the new estimate is produced. The Newton-Raphson scheme requires the additional evaluation of a derivative term at each iteration. The Picard iteration scheme was selected for use in the algorithm because of its ease of application.

4.2 Picard Iteration

Picard iteration is the simplest of the iteration schemes. The general algorithm for Picard iteration can be described as follows. First, consider a set of quasilinear equations:

$$f_I(x_1, x_2, \ldots, x_{N_I}) = 0 \quad \text{for} \quad I = 1, 2, \ldots, N_I \tag{4.5}$$

FIGURE 4.1
SCHEMATIC REPRESENTATION OF SATURATED THICKNESS FOR UNCONFINED AND CONFINED AQUIFERS



where

$(x_1, x_2, \ldots, x_{N_I})$ = unknowns

A set of auxiliary functions $b(x1, x2, ..., xM)$ is constructed next. In the case of groundwater flow, these auxiliary functions would be the right hand side equations from equations 3.11 and 3.12. They can be described as

$$[A]_{IJ}\{\vec{x}\}_J = \{\vec{b}\}_J \tag{4.6}$$

The iteration is started by assuming an initial solution $(x_{11}, x_{12}, ..., x_{1M})$ and this solution is used to evaluate the left hand coefficients and the right hand side of Equation 4.6. Thus, Equation 4.6 becomes a set of linear equations which can be solved for the next set of x_J values. The solution for x_J can be expressed as

$$\{\vec{x}\}_J^{r+1} = [A]_{IJ}^{-1}\{\vec{b}\}_I^r \tag{4.7}$$

where

r = iteration counter

$[A]_{IJ}^{-1}$ = elements of the inverse of matrix $[A]$

Equation 4.7 provides the means for obtaining successive solutions of x_J. It should be noted that the inverse of matrix [A] is not produced by the algorithm; the inverse is shown here to illustrate the nature of the solution. At each iterative cycle, the left hand coefficients and the right hand side equations are updated. The iterations are performed until satisfactory convergence is achieved. The criterion used for checking convergence is given by

$$\frac{\max_J |\{\vec{x}\}_J^{r+1} - \{\vec{x}\}_J^r|}{\max_J |\{\vec{x}\}_J^{r+1}|} \leq \varepsilon_b \tag{4.8}$$

where

ε_b = prescribed residual tolerance

$\max_J$ = maximum over all nodes

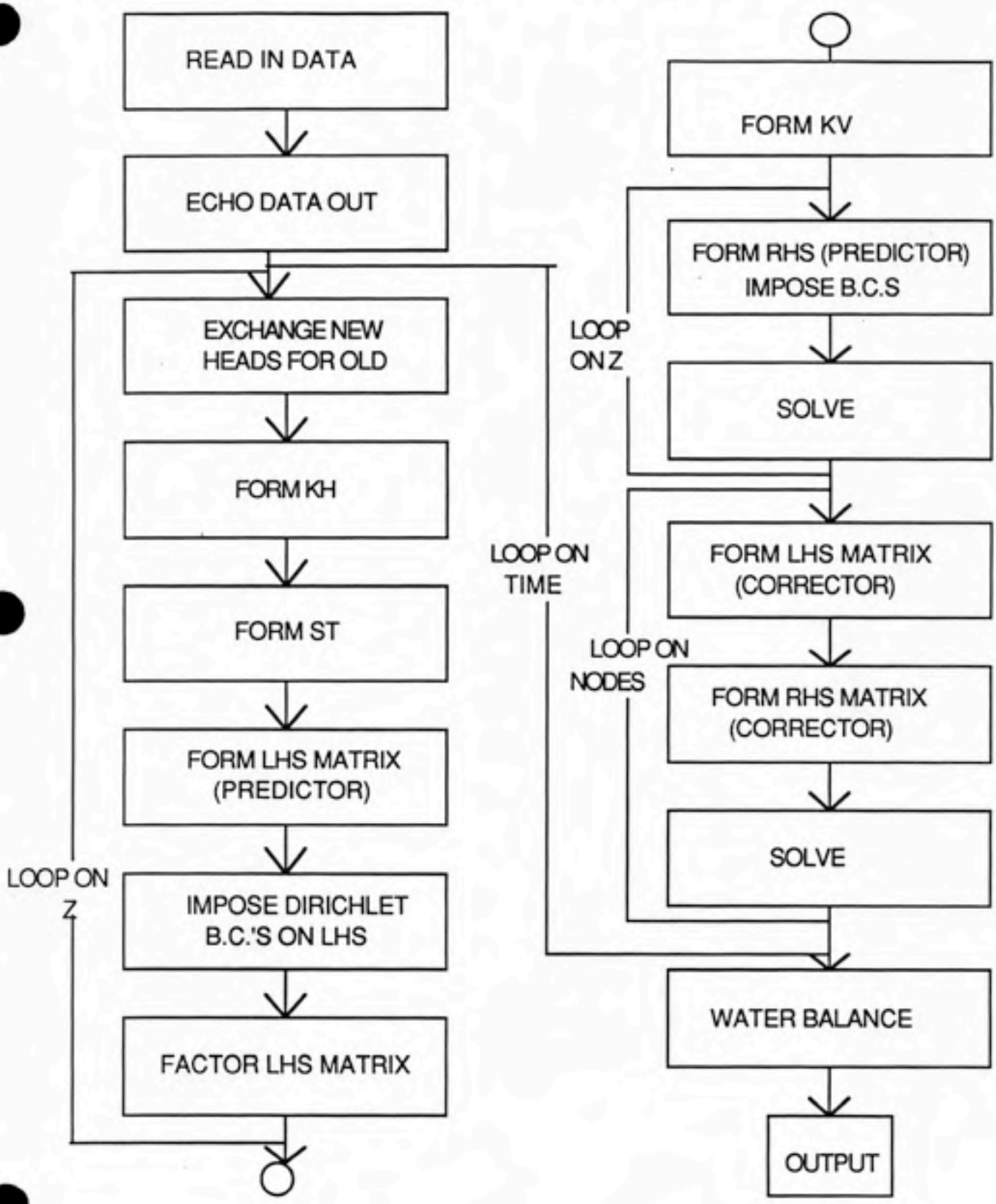
The application of the Picard iteration scheme to the previously developed algorithm is illustrated in Figure 4.2.

4.3 Skipping Confined Layers

The addition of an iterative scheme to the ALALS algorithm increases the number of computational operations that must be performed in order to achieve a solution. The flow diagram in Figure 4.2 indicates that the entire system of equations must be included in reforming or updating of the coefficient matrices [KH] and [ST]. The updating of the matrices includes the factorization of [KH] + [ST], which can involve a great deal of computational operations. However, because the ALALS algorithm uncouples the three-dimensional system of equations into a set of x-y equations, it is not be necessary to update the coefficient matrices for all of the x-y equations. The removal of these unnecessary operations can be performed without affecting the accuracy of the solution.

For each node where unconfined conditions exist, there will be only one layer out of the entire system of layers that has unconfined flow. The layers below the node will be confined, and the layer above the node will be either nonexistent or drained. Thus, it is necessary to update the coefficient matrices only for the layers that have unconfined flow conditions. For example, for a typical layered system shown in Figure 4.3, only the nodes included in the top layer would be included in the update of the coefficient matrices. Skipping the reforming of the coefficient matrices for the confined layers would then decrease the

FIGURE 4.2 FLOW DIAGRAM FOR PICARD ITERATION



number of operations by up to 20

4.4 Draining, Refilling Selected Nodes

As hydraulic head declines as a result of a withdrawal well, it can fall below the top of a layer. If the water table drops below the top of the uppermost layer, flow at the affected nodes changes from confined to unconfined. If the water table continues to drop, and falls below the top of a layer below the uppermost layer, the affected nodes in that layer are effectively drained. This concept is illustrated in Figure 4.5. Conversely, as the water table rises as a result of recharge or an injection well, the drained nodes are refilled. This type of situation is encountered often in water supply pumping operations, contaminant recovery operations, and in natural recharge and discharge of groundwater aquifers.

The draining and refilling of nodes requires modification of the algorithm. If the drained nodes are not accounted for, the algorithm will attempt to calculate the heads at these nodes, resulting in an insolvable system of equations. The equations that include the drained nodes could be removed from the system, but then refilling of the drained nodes would be impossible. Instead of removing the equations, the coefficients of the drained nodes can be operated upon in a way similar to that of imposing Dirichlet boundary conditions. The relevant coefficients on the left hand side of Equations 3.11 and 3.12 are forced to equal one or zero. On the right hand side, the coeffcients are forced to equal the hydraulic heads from the layer immediately below

$$[h_k]_{drained} = h_{k-1} \tag{4.9}$$

This procedure guarantees that the drained nodes will not influence the remaining nodes, because as long as the heads in adjoining layers are equal, then there is no exchange of flow between the adjoining layers. As soon as the head in the layer immediately below

FIGURE 4.3

ILLUSTRATION OF LAYER-SKIPPING FOR UNCONFINED FLOW

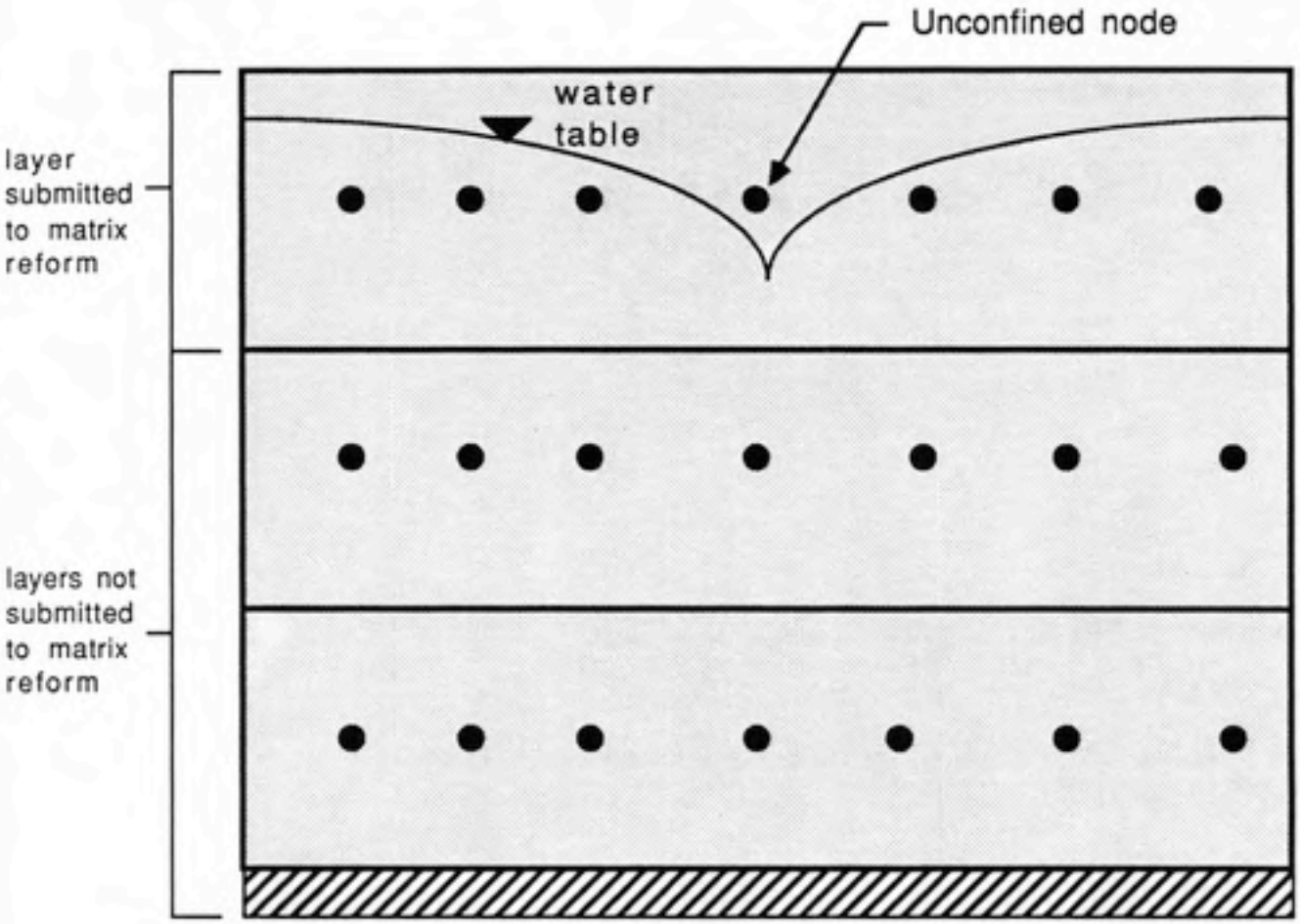


FIGURE 4.4
LAYER-SKIPPING MODIFICATION

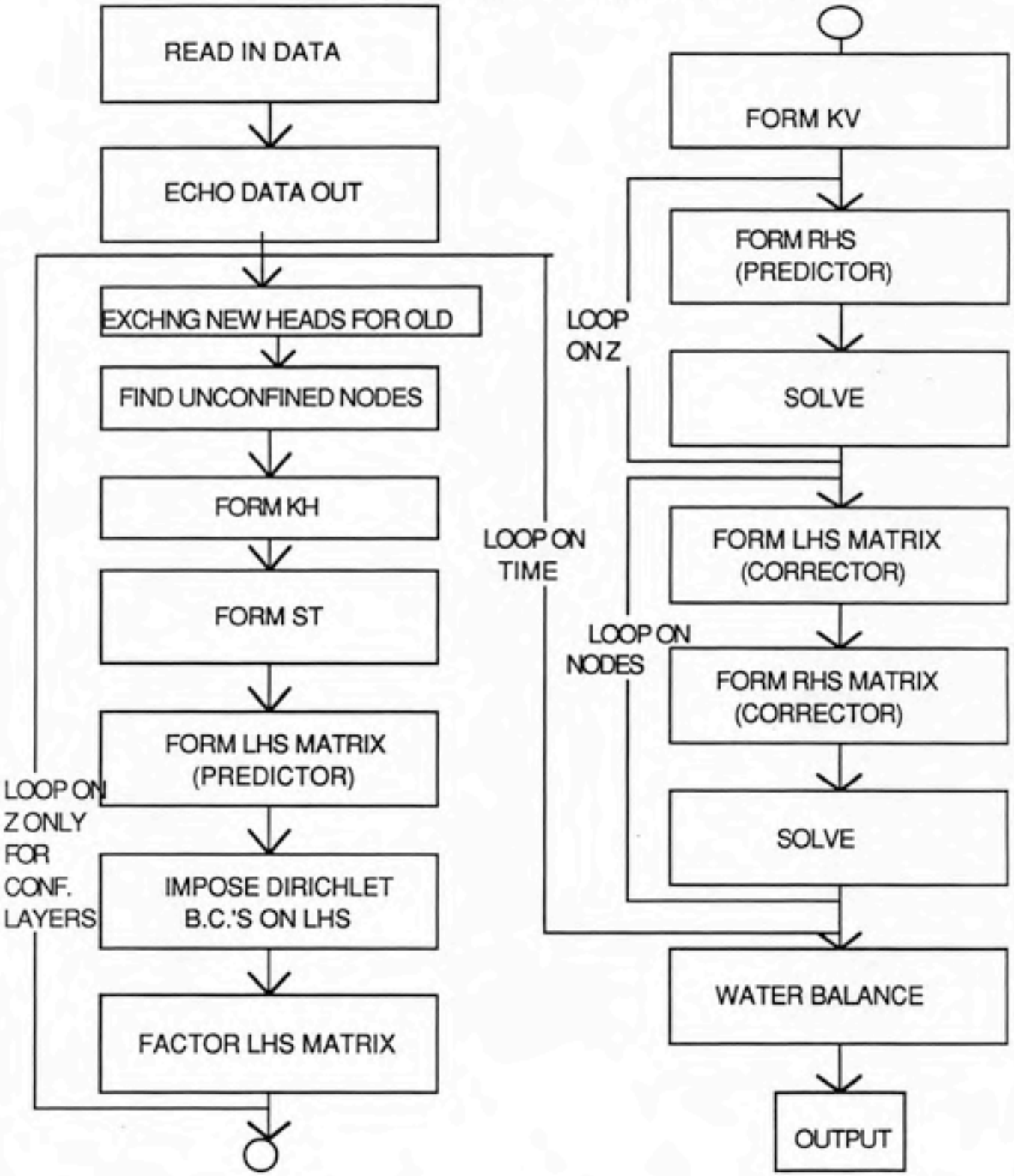
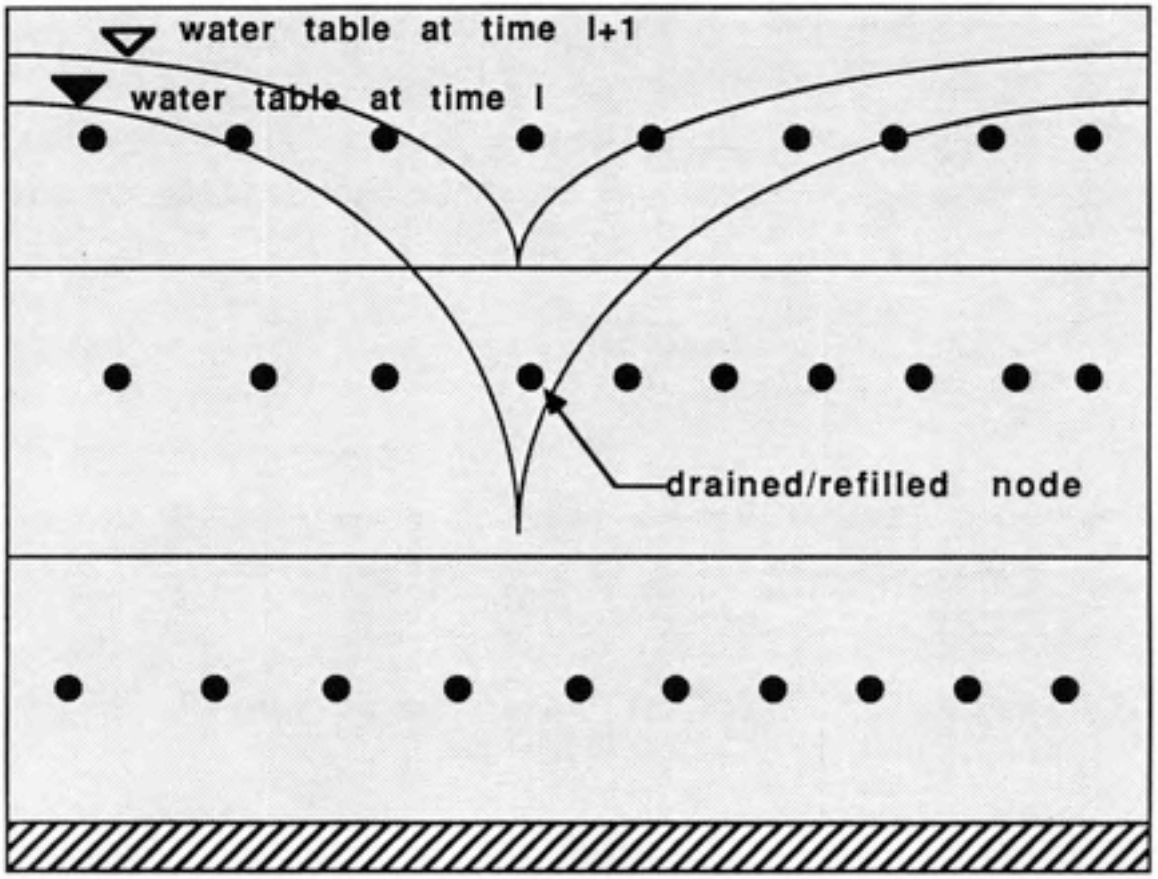


FIGURE 4.5
ILLUSTRATION OF DRAINED/REFILLED NODES



the drained node exceeds the top of the layer immediately below, then the drained node is refilled, and calculated with the normal procedure.

If a source or sink term exists at a drained node, then the location of the term must be adjusted, when the drained node is removed from the system of equations. The adjustment is accomplished by temporarily removing the stress from the drained node and replacing it in the layer immediately below. This adjustment ensures that the same overall magnitude of sources or sinks remains constant, and thus conservation of mass is not violated. When the drained node is refilled, then the source or sink is replaced at the refilled node. The modification to the algorithm for draining and refilling of nodes is shown in Figure 4.6.

4.5 Storage Estimation

The storage terms found in the unconfined or confined version in the model can be written as the amount of water released from aquifer storage

$$Q_{conf} = S\frac{h^{(l+1)} - h^l}{\Delta t}\iint_{\mathcal{R}} N_i \, dx \, dy$$

and (3.55)

$$Q_{unconf} = S_y\frac{h^{(l+1)} - h^l}{\Delta t}\iint_{\mathcal{R}} N_i \, dx \, dy$$

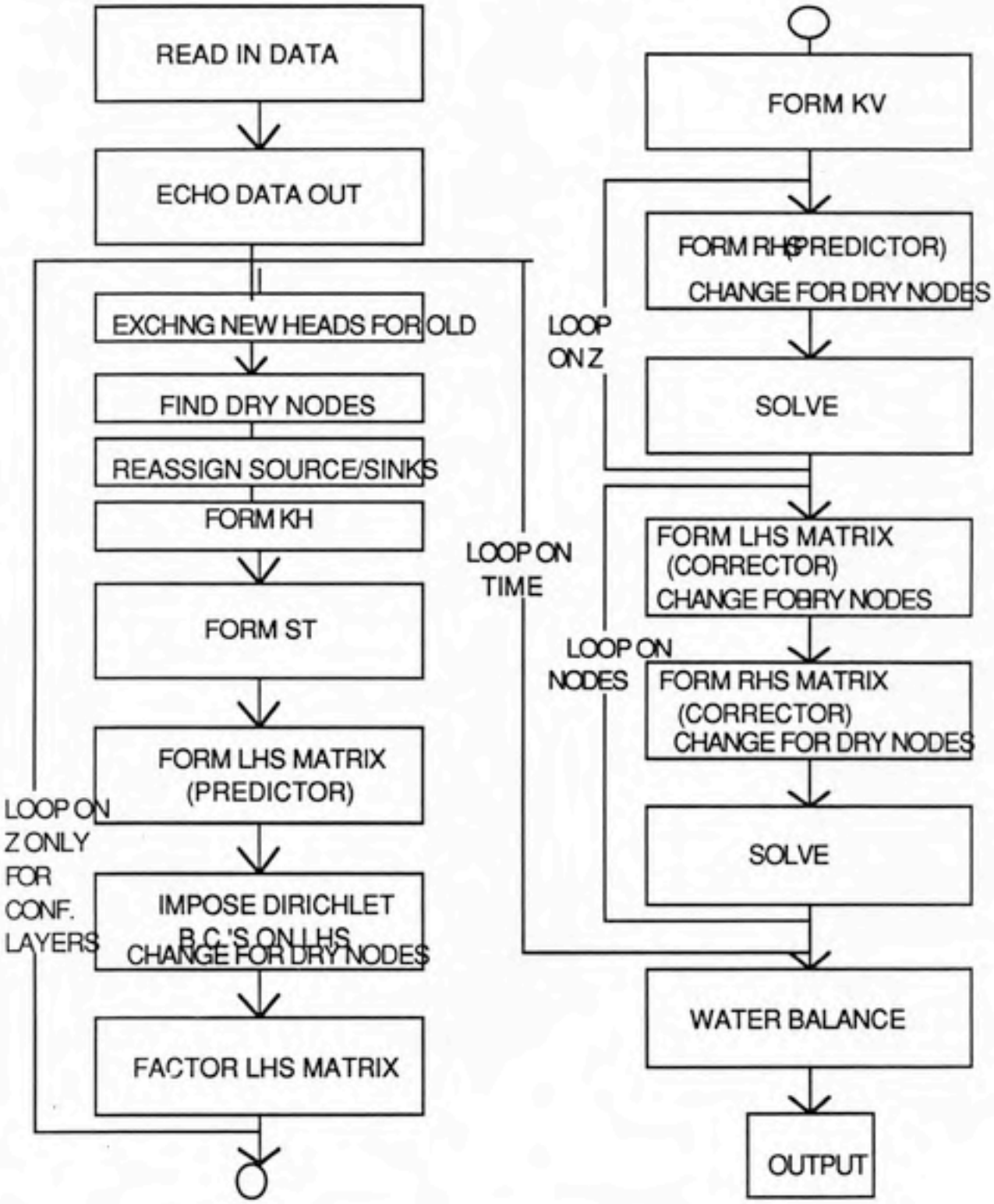
where

Q = rate of change in storage of water in an element (L^3/T)

In the case of a strictly confined or unconfined aquifer, these terms are suitable for describing the rate of change in storage. However, these equations must be expanded to allow for the simulation of a model node that can change from confined to unconfined in one timestep (or the reverse). This situation is illustrated in Figure 4.7. During a time step when a node changes from confined to unconfined, the storage equation becomes

FIGURE 4.6

DRAINED NODE MODIFICATION



$$Q = \frac{S^l(h^l - \text{top}) + S^{(l+1)}(\text{top} - h^{(l+1)})}{\Delta t} \iint_{\mathcal{R}} N_i \, dx \, dy \tag{4.10}$$

where

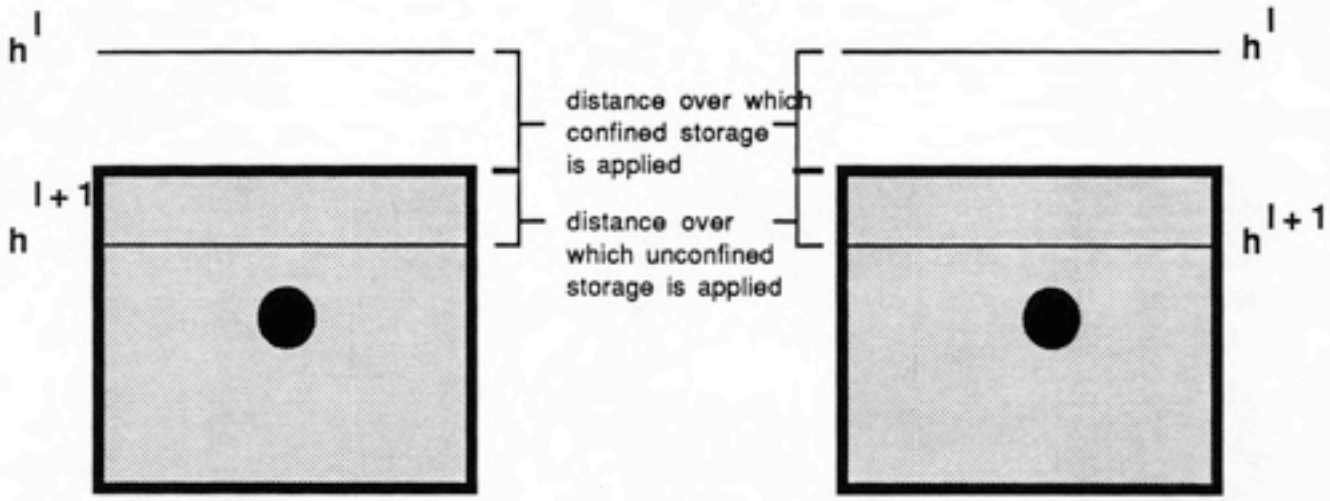
top = elevation of the top of an element layer (L)

S^l = storage factor (specific yield or storativity) in effect at time l (*dimensionless*)

$S^{(l+1)}$ = storage factor (specific yield or storativity) in effect at time $l+1$ (*dimensionless*)

FIGURE 4.7

ILLUSTRATION OF STORAGE TERMS FOR UNCONFINED/CONFINED TRANSITION



5 TESTING OF MODEL ACCURACY AND SENSITIVITY: RESULTS AND DISCUSSION

Model testing is an evaluation of the accuracy of a model. It is one of the most important steps in the development of a model. If a model tests favorably under a range of conditions, then one can be confident that the model will perform well when applied to more realistic situations. Model tests in this research consist of analyzing relatively simple problems and graphically comparing numerical model results with results from analytical solutions. One assumes that the exact analytical solutions used for comparison are a surrogate for real groundwater flow.

Various flow conditions, such as confined and unconfined groundwater flow, can be tested with exact analytical solutions. The ability of the model to handle flow in three dimensions can also be tested. The results of such tests are reported in this chapter. In addition to model validations, the sensitivity of the model to various parameters such as timestep size, convergence criteria, and grid spacing is considered.

Ideally, one would like to test the model against all possible situations that may be encountered when applying the model. But exact analytical solutions of the flow or transport equations are available only for relatively simple applications. Mass balances, however, can be performed under any flow conditions. Mass balances provide a convenient way to check accuracy where analytical solutions are unavailable.

Models can be tested with methods more sophisticated than graphical comparisons. Residual errors between the numerical model results and analytical model results can be calculated and subsequently provide various accuracy criteria. These criteria may include Mean Square Error (MSE) or Sums of Squares of Residuals (SSR). The calculation of these types of criteria is left to further studies; graphical comparisons shall suffice for this research.

All computer runs used in this chapter were performed on an IBM Personal Computer AT.

5.1 Confined Flow

Confined flow validations can be performed with the Theis equation (Theis, 1935). The Theis equation is a solution that governs the transient response of an aquifer to a pumped well. The assumptions for this solution include radially-symmetric flow towards the well, a homogeneous and isotropic aquifer that is infinite in areal extent, and an infinitessimal diameter for the well. A schematic illustration of these conditions is found in Figure 5.1.

The Theis equation is represented as

$$s = \frac{Q}{4\pi T} W(u_c) \tag{5.1}$$

where

s = drawdown = initial head - new head (L)

r = radial distance from well (L)

$W(u_c)$ = well function for nonleaky aquifer (*dimensionless*)

$$= \int_{u}^{\infty} \frac{e^{-w}}{w}\, dw$$

u_c = argument of the well function (*dimensionless*)

$$= \frac{r^2 S}{4Tt}$$

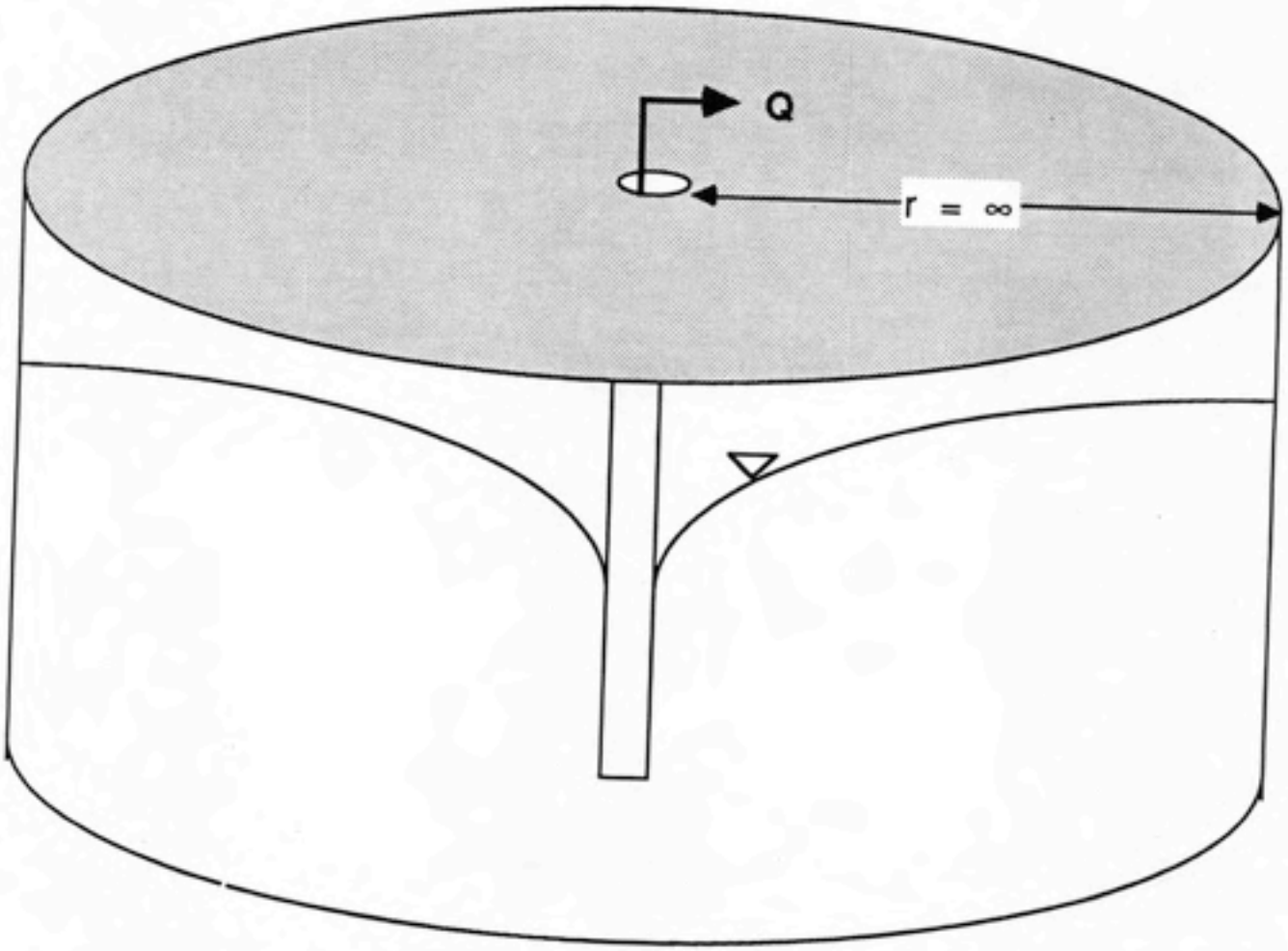
T = transmissivity (L^2/T)

S = storativity (*dimensionless*)

Q = pumping rate (L^3/T)

FIGURE 5.1

SINGLE PUMPED WELL, RADIALLY SYMMETRIC HOMGENEOUS AQUIFER



t = time (T)

The boundary conditions requiring an infinite aquifer radius is simulated with the REGFED model by placing a constant head boundary far enough away from the well so that no drawdown occurs at the boundary. This situation ensures that no flux through the boundary occurs and thus the existence of the boundary has no effect on the hydraulic head distributions.

A comparison of results from the REGFED model and the Theis equation are shown in Figure 5.2. The parameters used are also listed on the figure. Three different sets of hydraulic conductivities were used, in order to test the sensitivity of the model. The sets of parameters are meant to be increasingly difficult. The difficulty in modeling flow near a well increases as the slope of the drawdown curve increases. Figure 5.2 indicates that the model agrees quite well with the Theis equation, under all three sets of parameters.

The validity of the vertical flow components of the model can be tested against an exact analytical solution that is a variation on the Theis equation. The Theis equation contains the assumption that the pumped well fully penetrates the confined aquifer, as shown in Figure 5.3. If this assumption is violated, then the well only partially penetrates the aquifer, and vertical flow components are introduced (see Figure 5.3). An exact analytical solution has been found for transient, confined flow under partially penetrating conditions (Hantush, 1961). This solution is represented as

$$s = \frac{Q}{4\pi T}\left[W(u_c) + \frac{2b}{\pi(l_w - d)}\sum_{n_s}^{\infty}\cos\left(\frac{n_s\pi z}{b}\right)\left[\sin\left(\frac{n_s\pi l_w}{b}\right)\right.\right.$$
$$\left.\left. - \sin\left(\frac{n_s\pi d}{b}\right)\right]W\left(u_c, \frac{n_s\pi r}{b}\right)\right. \tag{5.2}$$

where

$W\left(u_c, \frac{n_s\pi r}{b}\right)$ = well function for leaky aquifer (*dimensionless*)

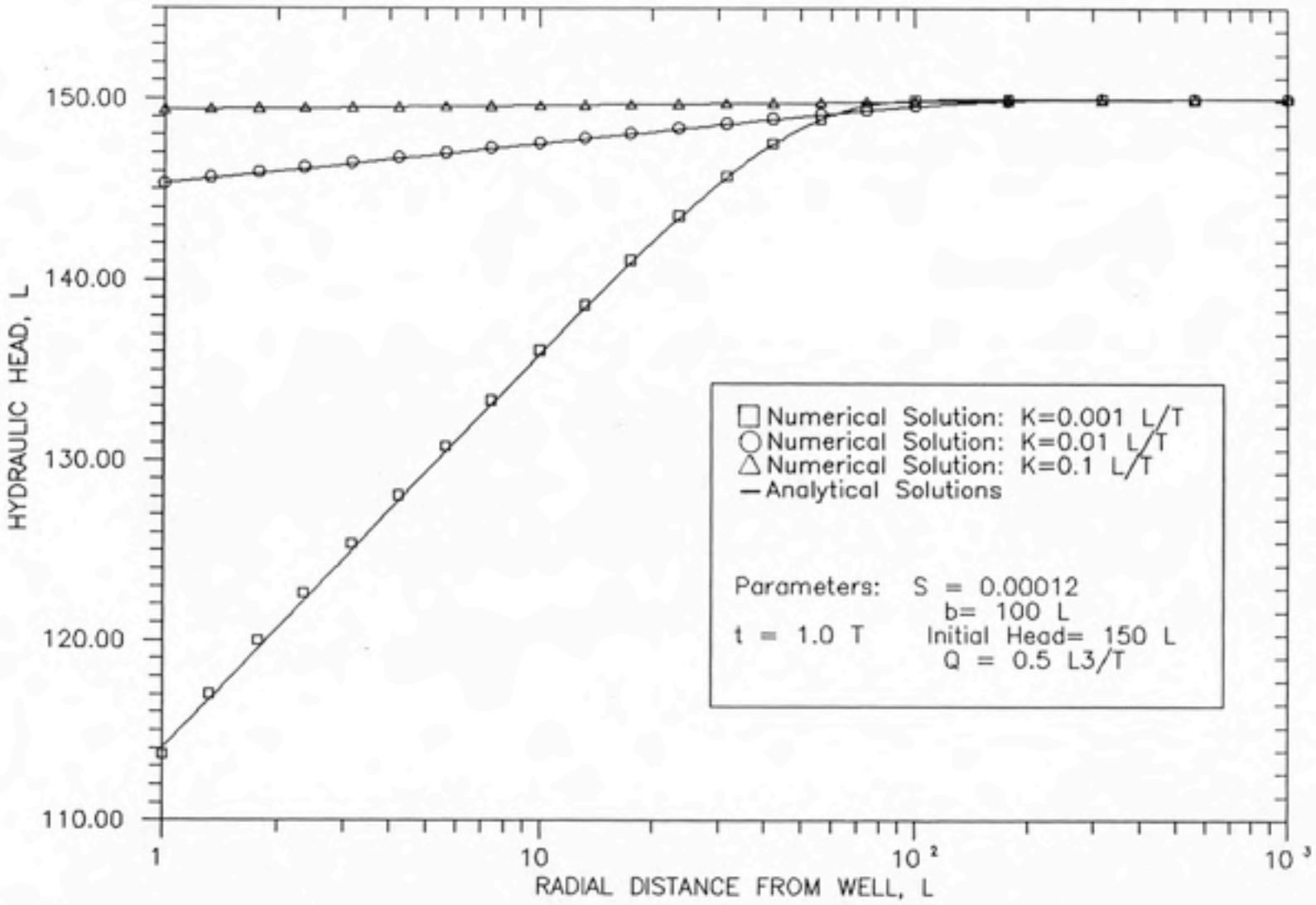


FIGURE 5.2: COMPARISON OF ANALYTICAL AND NUMERICAL SOLUTIONS-- CONFINED FLOW

$$= \int_{u}^{\infty} \frac{1}{v} \exp\left(-v - \frac{r^2 n_s^2 \pi^2}{4b^2 v}\right) \, dv$$

$W(u_c)$ = well function for nonleaky aquifer (*dimensionless*)

z = vertical distance from top of aquifer (L)

n_s = summation index

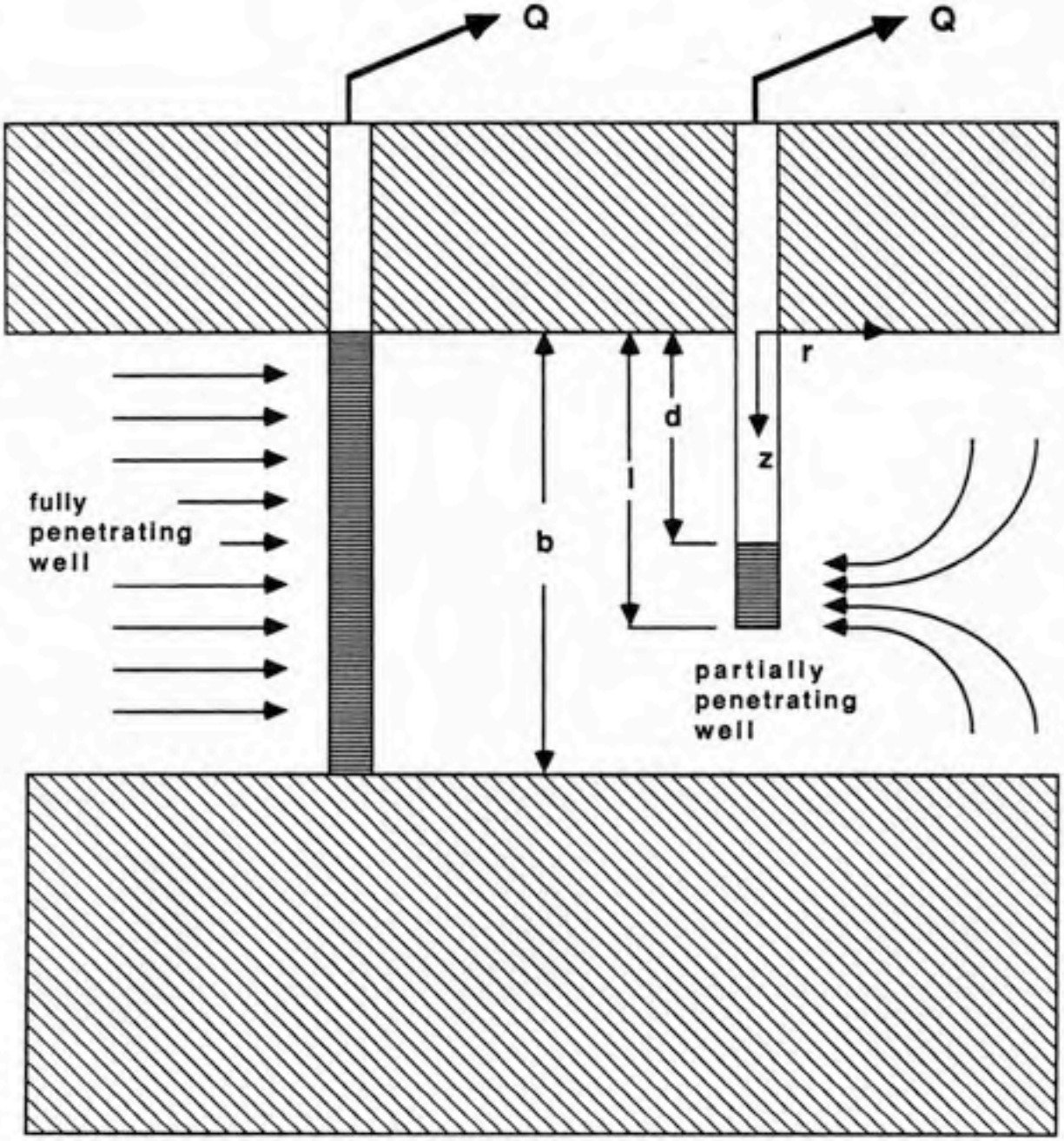
This equation also contains the assumptions of a radially infinite aquifer that is homogeneous and isotropic. A partially penetrating well is especially easy to simulate with the REGFED model. Sink terms are placed only at nodes in the relevant layers; sink terms are excluded from the other layers.

Results from the REGFED model and the analytical solution for partially penetrating are compared in Figure 5.4. The parameters used in the comparison are listed on the figure. The hydraulic heads at three different radial distances from the well were included, in order to test the sensitivity of the model. Figure 5.4 indicates that model accuracy increases as distance from the well increases. The inaccuracy near the well is due to the especially steep vertical gradients in this area. Accuracy could be improved by using smaller timesteps, or by making the vertical discretization finer, especially in the vicinity of the lower end of the well screen. In this case the timestep size was one time unit out of a total of ten time units, and the vertical discretization consisted of ten equally spaced layers.

The mass balance error for this simulation was $1.93 \times 10^{-6}\%$. This error is only slightly larger than those found for fully penetrating conditions, which were on the order of $10^{-7}\%$, indicating that, although the model does not always agree with the analytical solution, it still behaves well with respect to mass balance. The low mass balance errors are due to the fact that changes in vertical flow components do not affect the mass balance over the groundwater system.

FIGURE 5.3

PARTIALLY PENETRATING WELL



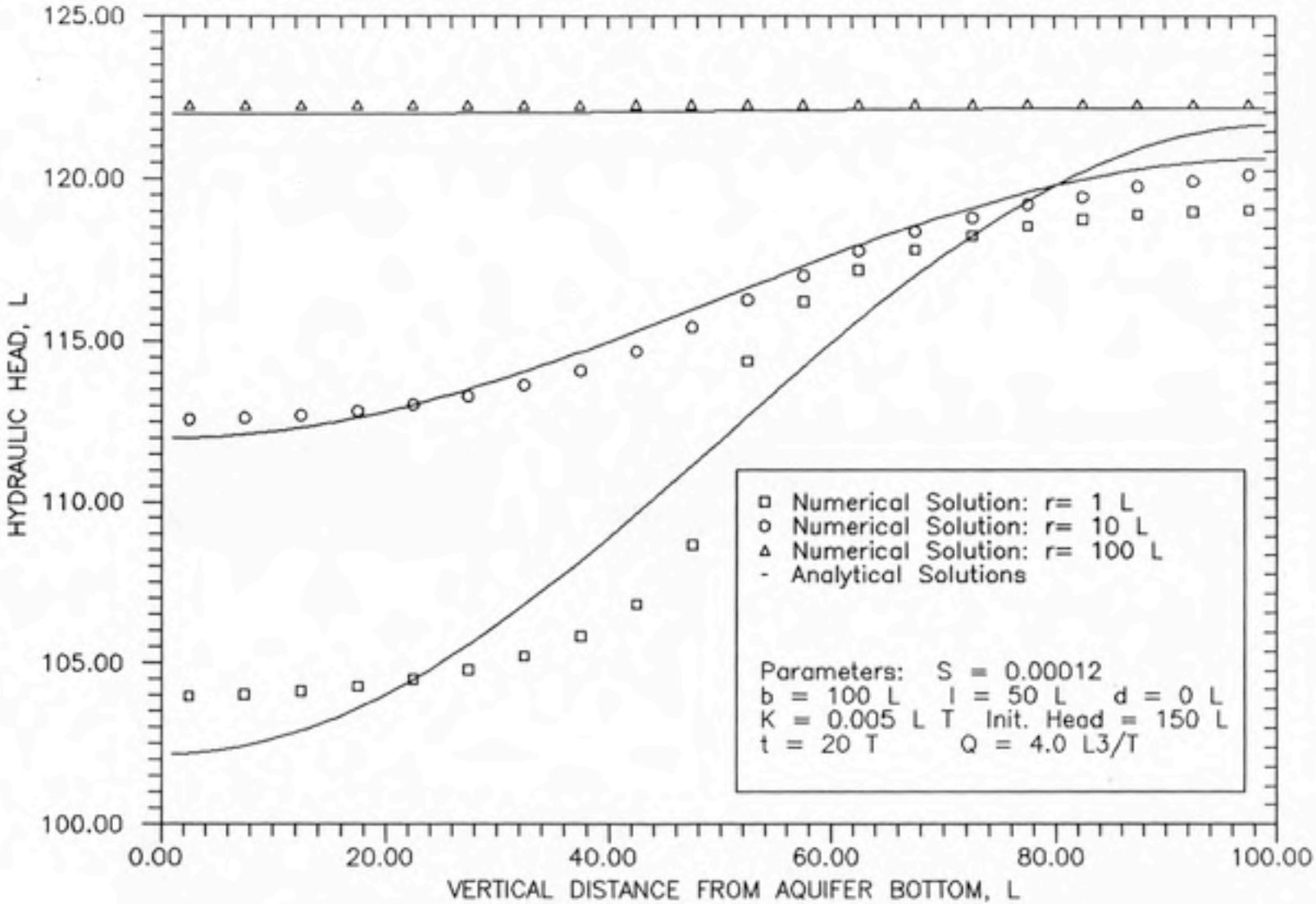


FIGURE 5.4: COMPARISON OF ANALYTICAL AND NUMERICAL SOLUTIONS--PARTIALLY PENETRATING WELL, CONFINED FLOW

5.2 Unconfined Flow

Model simulations of unconfined flow can be validated in a number of ways. In Chapter 2, the various approaches towards simulating unconfined flow with analytical solutions were discussed. The simplest of these approaches applies the Theis equation, using specific yield instead of storativity in the storage term. Since this approach is closest to the approach used in this numerical model, it is the most appropriate for validation of the REGFED model. This approach results in

$$s = \frac{Q}{4\pi T} W(u_u) \tag{5.3}$$

where

u_u = argument (modified for unconfined flow) of the well function (*dimensionless*)

$= \frac{r^2 S_y}{4Tt}$

S_y = specific yield (*dimensionless*)

However, the above solution ignores the fact that transmissivity changes with drawdown of the water table. The drawdown can be adjusted for transmissivity changes by applying the Jacob correction equation (Jacob, 1944)

$$s_c = s_o - \frac{s_o^2}{2b} \tag{5.4}$$

where

s_c = corrected drawdown (L)

s_o = drawdown calculated from Equation 5.3 (L)

b = aquifer thickness (L)

The results from the REGFED model and the adjusted Theis equation are compared in Figure 5.5. The parameters used in the comparison are also listed on the figure. Three different hydraulic conductivities were also used for the unconfined flow validation. The model agrees relatively well with the adjusted Theis equation. The accuracy of the model appears to decrease with hydraulic conductivity. The mass balance errors ranged from $9.82 \times 10^{-7}\%$to$9.55 \times 10^{-5}\%$ for these validations. The mass balance errors are not as good as for confined flow, but this is to be expected, given the difficulty of simulating unconfined flow.

A second validation of unconfined flow conditions was performed. This validation uses an exact analytical solution of the steady- state, unconfined version of the groundwater flow equation. This solution simulates one-dimensional, steady-state flow between Dirichlet (constant head) boundaries, as illustrated in Figure 5.6. The solution follows as

$$h^2(x) = \left(\frac{h_1^2 - h_o^2}{x_1}\right) + h_o^2 \tag{5.5}$$

where

h_o = head at up-gradient boundary (L)

h_1 = head at down-gradient boundary (L)

x = distance from up-gradient boundary (L)

x_1 = length of aquifer (L)

Equation IV-5 can be modified easily to account for the effects of constant recharge over the length of the aquifer

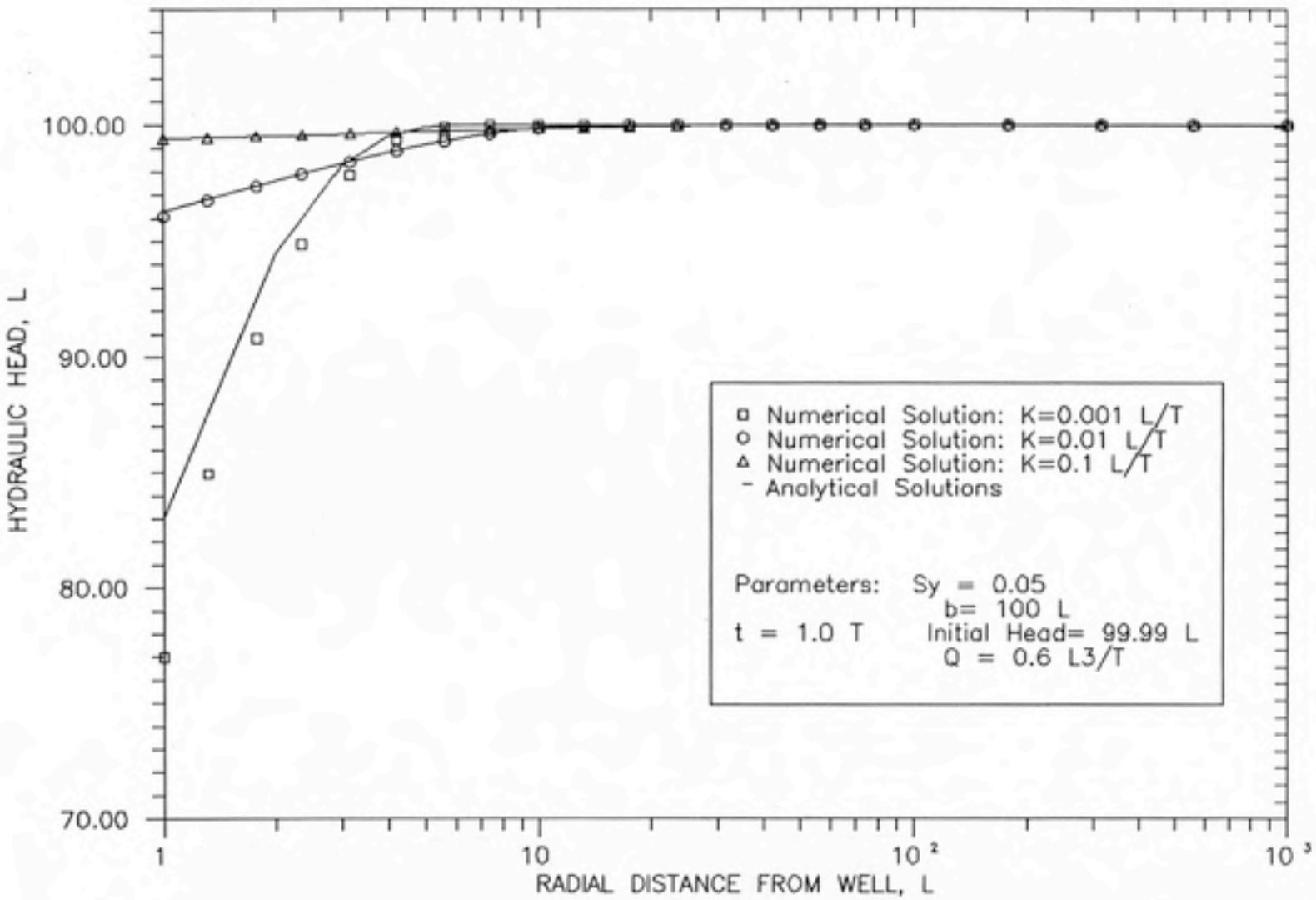
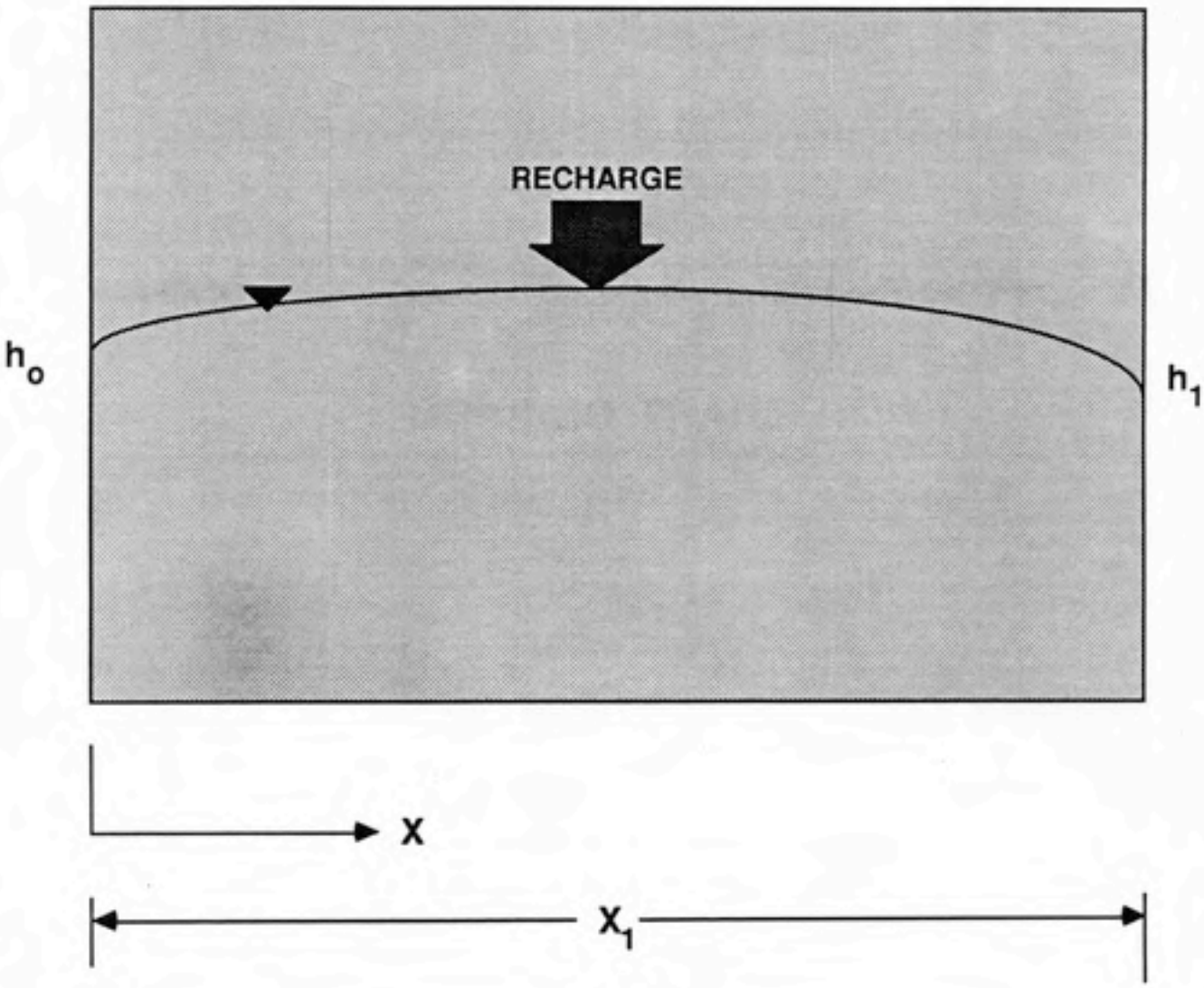


FIGURE 5.5: COMPARISON OF ANALYTICAL AND NUMERICAL SOLUTIONS-- TRANSIENT UNCONFINED FLOW

FIGURE 5.6

ILLUSTRATION OF ONE-DIMENSIONAL FLOW WITH DIRICHLET BOUNDARIES



$$h^2(x) = \left(\frac{h_1^2 - h_o^2}{x_1}\right) + h_o^2 + \left(\frac{\Gamma_r}{K}\right) x^2 \tag{5.6}$$

where

Γ_r = recharge rate (L/T)

Simulations of the numerical model and the analytical solution (Equation 5.5) are compared in Figure 5.7, for three different boundary conditions. The figure shows that steady-state unconfined flow can be accurately simulated by the model. Equation 5.6 provides a convenient way to test the recharge component of the model. Figure 5.8 shows a comparison of results from Equation 5.6 and the model for three different recharge rates. The addition of recharge does not appear to affect the accuracy of the model.

There are two other important model simulations of unconfined flow that should be considered. As hydraulic head declines as a result of a withdrawal well, it can fall below the top of a layer (see Figure 3.8). If the water table drops below the top of the uppermost layer, flow at the affected nodes changes from confined to unconfined. This transition is difficult to model because transmissivity and storage terms can change greatly within a single layer, for a single timestep (during the transition from confined to unconfined). These changes in parameters create a linear system that is difficult to solve. If the water table falls below the top of a layer below the uppermost layer, the affected nodes in that layer are effectively drained. The transition from full nodes to drained nodes creates a linear system that is even more difficult to solve than the confined-unconfined transition. Chapter 4 includes a discussion of algorithm modifications for this problem.

Because appropriate analytical solutions do not exist for these two problems, the problems were tested for mass balance errors. The relative thickness of the uppermost layer that becomes unconfined or drained was varied. It was hypothesized that the thicker the layer, the greater the change in transmissivity as a node switches from confined to

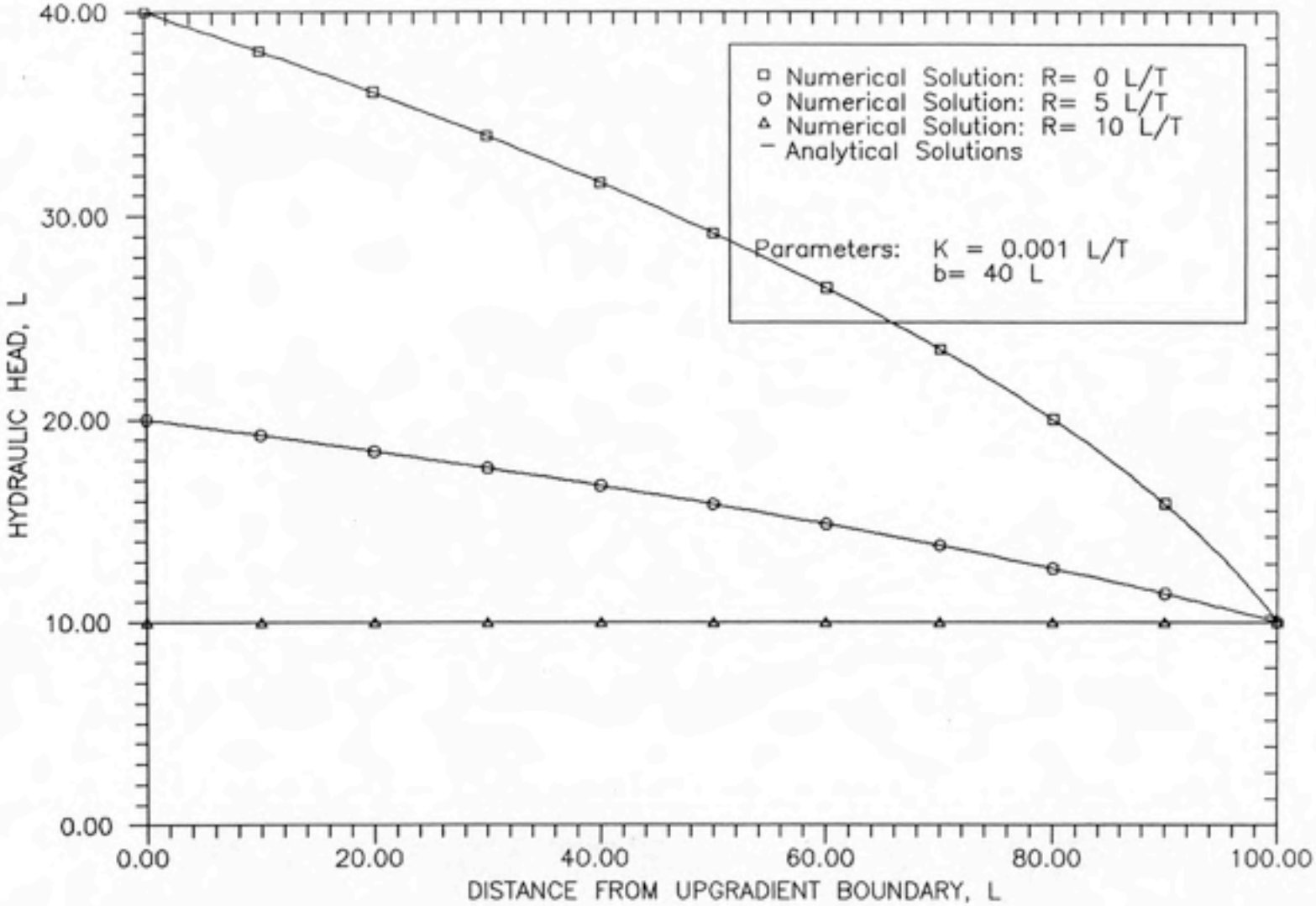


FIGURE 5.7: COMPARISON OF ANALYTICAL AND NUMERICAL SOLUTIONS-- STEADY-STATE UNCONFINED FLOW

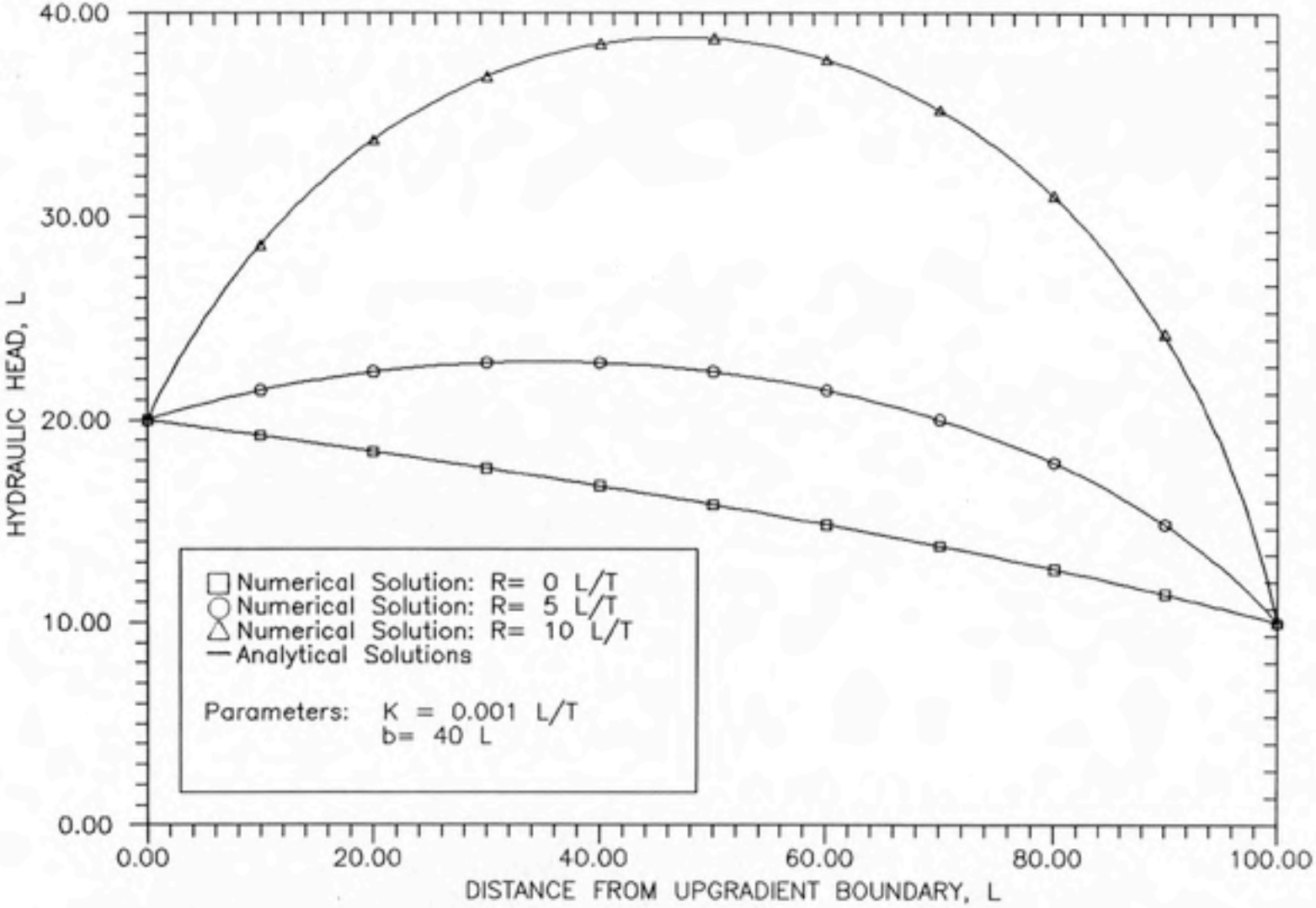


FIGURE 5.8: COMPARISON OF ANALYTICAL AND NUMERICAL SOLUTIONS-- STEADY-STATE UNCONFINED FLOW WITH RECHARGE

unconfined or from partially saturated to completely drained (with all other parameters held constant, including overall aquifer thickness). This assumption comes from the fact that transmissivity is a function of saturated thickness in an unconfined aquifer.

The results of these analyses are shown in Tables 5.1 and 5.2. These tables show that the mass balance error is much larger for the transitions (when compared to previous examples), but improves with increasing layer thickness, which is counter to the hypothesis. This indicates that thickness of the layers below the top may be dominating the mass balance accuracy. The overall high mass balance errors for the transitions are due again to steep vertical gradients.

5.3 Model Sensitivity

In the previous section, the REGFED model was subjected to variations of parameters that dealt with various hydrologic characteristics of groundwater flow, such as conductivities, recharge rates, etc. In this section, an analysis of the effect of varying parameters that deal strictly with operation of the model is performed. These parameters include timestep sizes, discretization schemes, and convergence criteria.

First, the size of individual timesteps was varied, under confined flow conditions. Figure 5.9 shows the model results for three different timestep sizes. These results are compared with an equivalent analytical solution, using the Theis equation. The figure indicates that increasing timestep size decreases accuracy.

The effect of different discretization schemes can be analyzed for the areal finite-element discretization and for the vertical finite-difference discretization. Given the same total area, the coarseness of the finite-element discretization scheme (or the total number of elements dividing the domain) was varied. The results for three different schemes are shown in Figure 5.10, along with an analytical solution for comparison. These results show that the accuracy of the model results decreases as the dicretization becomes more coarse.

Similarly, the vertical finite-difference discretization scheme was analyzed by varying

TABLE 5.1: MASS BALANCE ERRORS FOR CONFINED/UNCONFINED TRANSITION

Relative Thickness of Uppermost Layer	% Mass Balance Error
100 %	0.101
50 %	3.38
25 %	16.8

TABLE 5.2: MASS BALANCE ERRORS FOR PARTIALLY SATURATED/ DRAINED TRANSITION

Relative Thickness of Uppermost Layer	% Mass Balance Error
100 %	12.3
50 %	25.7
25 %	44.8

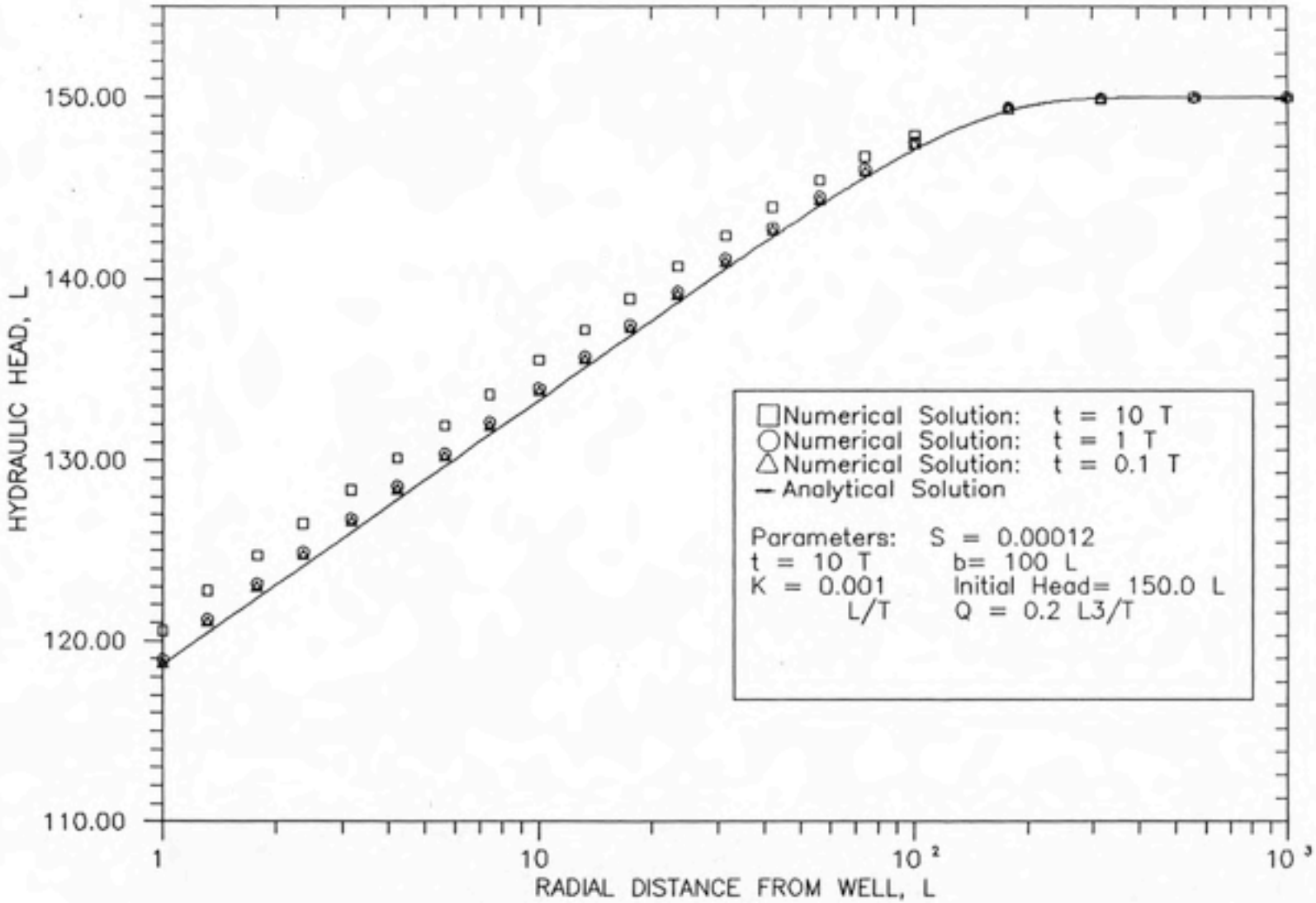


FIGURE 5.9: VARIATION OF TIMESTEP SIZE (CONFINED FLOW)

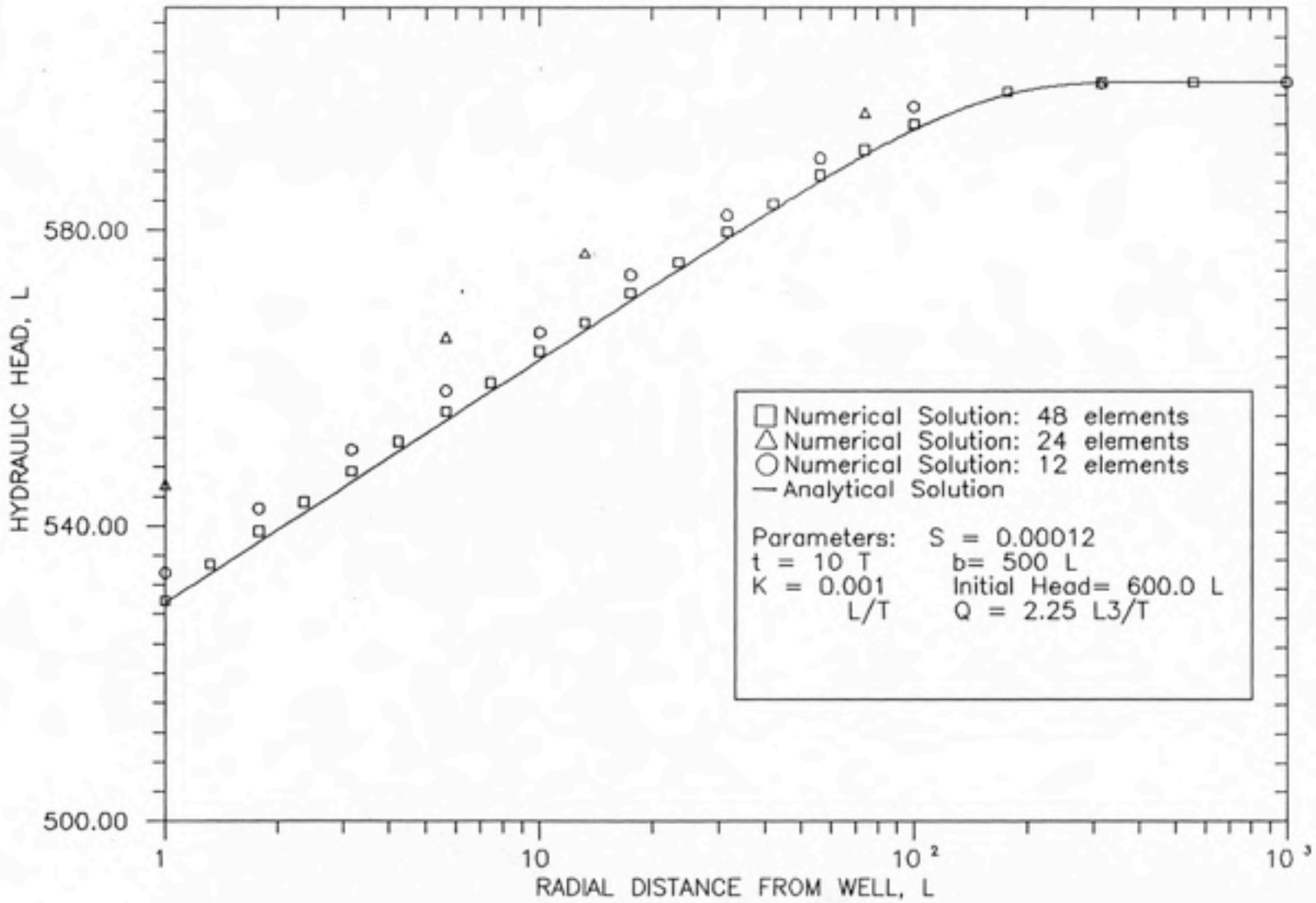


FIGURE 5.10: VARIATION OF HORIZONTAL DISCRETIZATION (CONFINED FLOW)

the number of layers in the vertical direction, given the same overall aquifer thickness. Again, the finer the discretization, the better the results, as shown in Figure 5.11. However, the model does not appear to be as sensitive to vertical discretization as for horizontal discretization, at least for these flow conditions.

The algorithm for solving unconfined flow problems is more complex than that for confined flow, and thus more model parameters are involved. These parameters include the convergence criteria (or maximum allowable error) and the maximum number of iterations allowed to reach convergence. Figure 5.12 shows the results of varying the maximum allowable error for an unconfined flow problem, with an analytical solution for comparison. As expected, model results improve with more stringent criteria. The figure also shows that, at least for this problem, there is a point where decreasing the maximum allowable error no longer significantly improves the accuracy of the solution.

Similar results are found when the maximum allowable number of iterations to achieve convergence is varied: the model is more accurate when more steps are allowed for convergence, given the same allowable error. This trend is illustrated in Figure 5.13.

The convergence properties for the model can be anlyzed by examining the magnitude of the errors from iteration to iteration and from timestep to timestep. Three different unconfined groundwater flow problems were considered: a single layered case, a five layer case where all nodes begin as unconfined, and a five layer case where the all nodes begin as confined, but some nodes eventually become unconfined. The last case tests the ability of each scheme to perform the transition between confined and unconfined nodes.

Figures 5.14A through 5.14C illustrate the characteristics of convergence for the three test cases. In these figures, percent residual is plotted against the number of iterations. The residual is calculated as

$$\text{percent residual} = \frac{h^{r+1} - h^r}{h^r} \times 100 \tag{5.7}$$

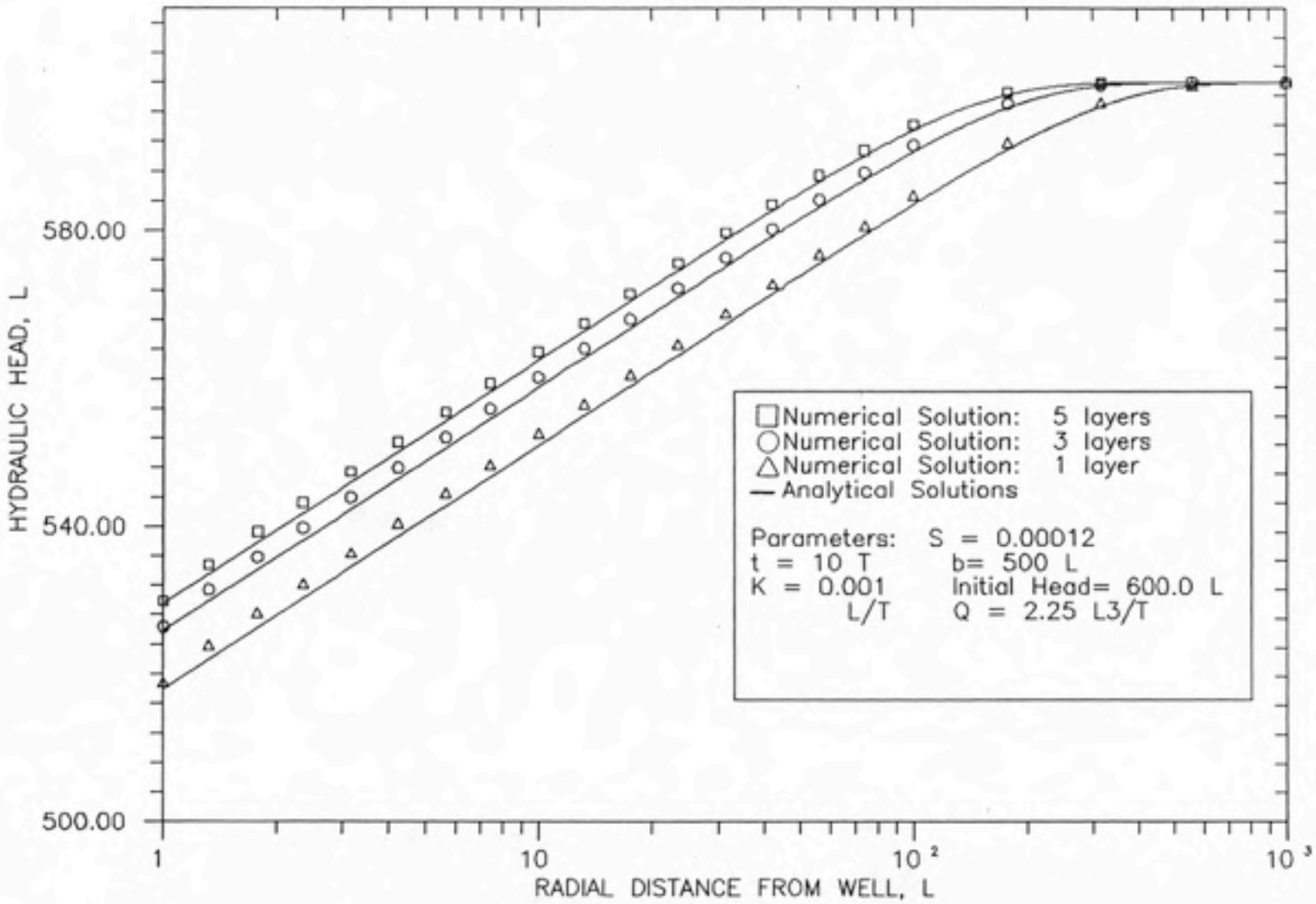


FIGURE 5.11: VARIATION OF VERTICAL DISCRETIZATION (CONFINED FLOW)

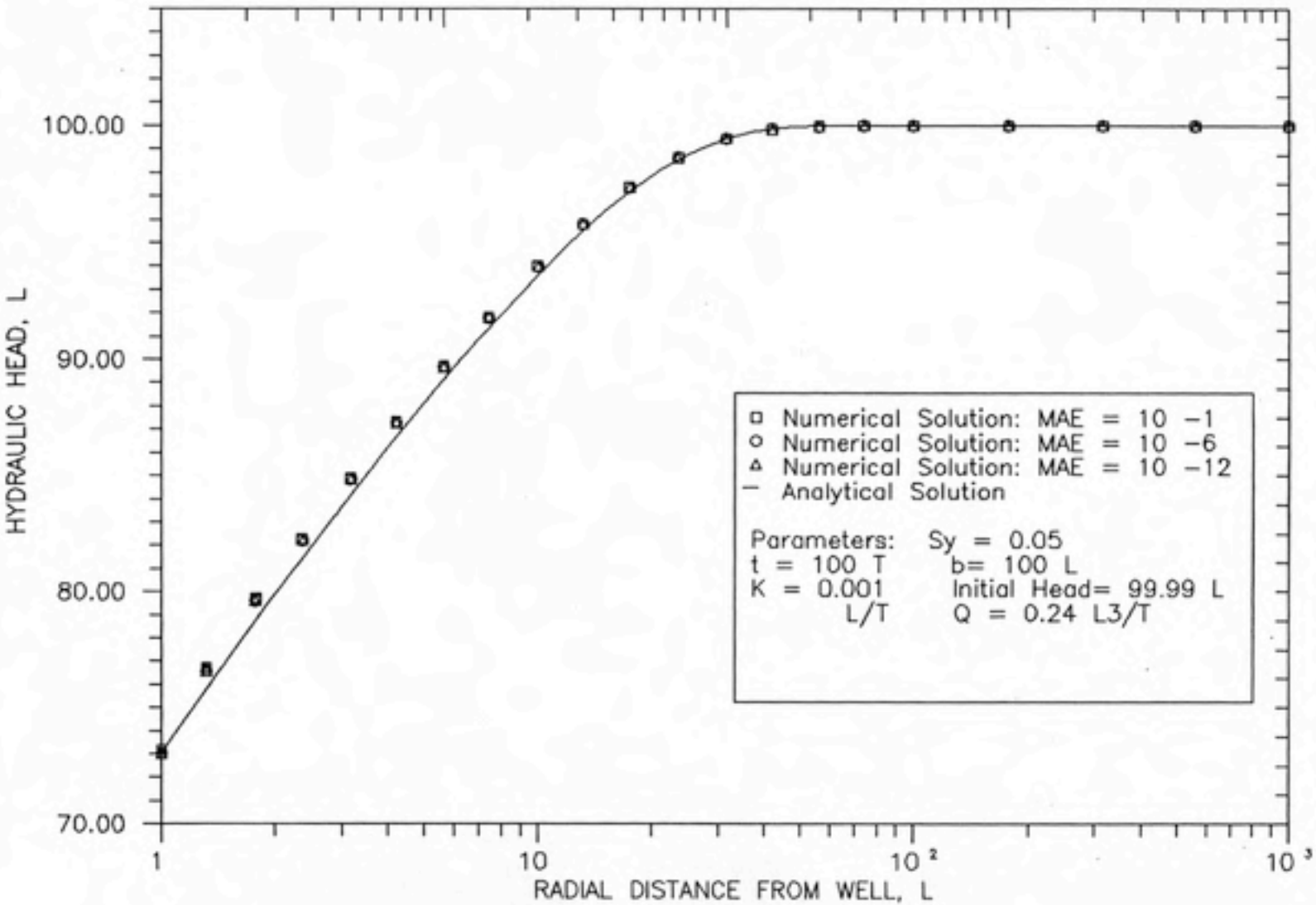


FIGURE 5.12: VARIATION OF MODEL PARAMETERS-- MAXIMUM ALLOWABLE ERROR (UNCONFINED FLOW)

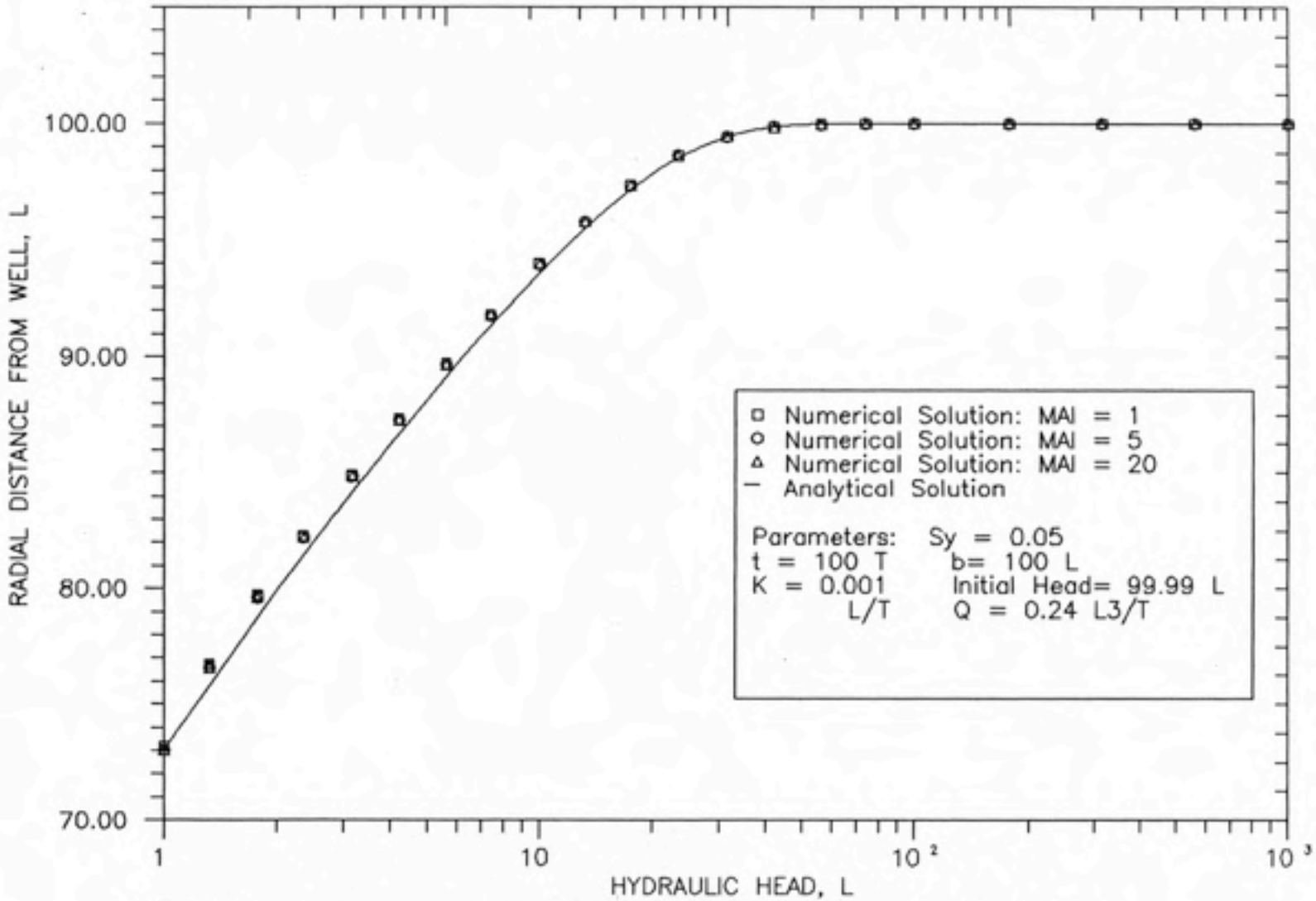


FIGURE 5.13: VARIATION OF MODEL PARAMETERS-- MAXIMUM ALLOWABLE ITERATIONS (UNCONFINED FLOW)

where

r = iteration index

The number of iterations represents the number of iterations performed to simulate the problem over the total number of timesteps. Thus, in Figure 5.14A, which exhibits stable convergence, iterations within a single timestep are found within a single "peak." At the beginning of a timestep, the residual starts high because the advancement of a timestep induces a large residual between the last iteration of the previous timestep and the first iteration of the present timestep. The residuals gradually diminish until the maximum allowable residual is achieved (at the bottom of a peak). The residual then increases to a high value at the beginning of the next timestep.

The flat portions of the curve found in Figure 5.14B occur where the convergence becomes unstable and the residuals tend to oscillate around a single value, for a certain number of iterations. Figure 5.14C exhibits this unstable behavior for more than half the total iterations. In additon, the residual increases sharply at about 300 iterations. This unstable behavior reflects the difficulty in solving the linear systems posed by node transitions.

These figures show that the model requires increasing numbers of iterations to achieve convergence, and that the convergence is less stable as the difficulty of the problem increases. The parameters for the test cases are listed in Table 5.3.

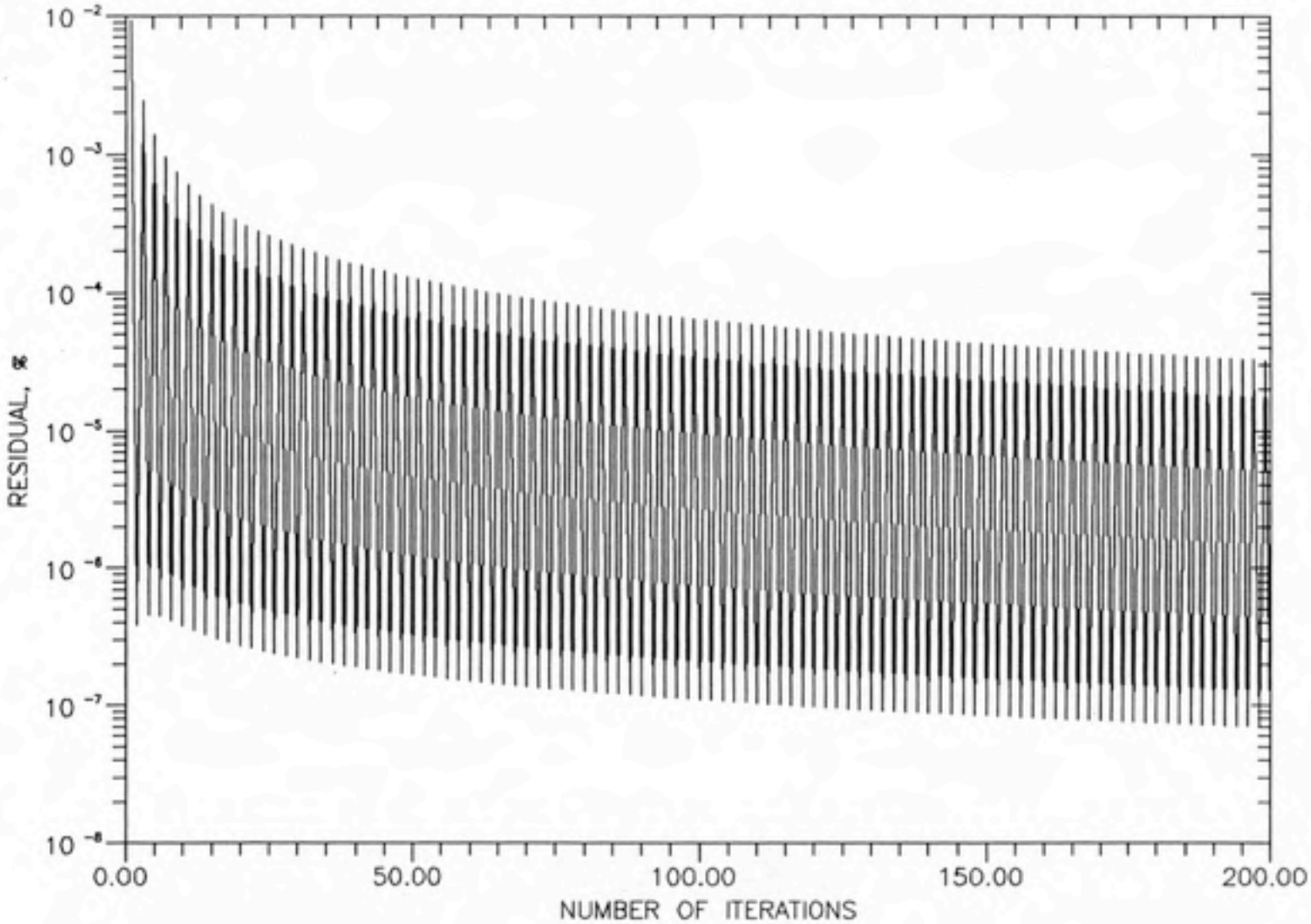


FIGURE 5.14A: RESIDUALS FOR TEST CASE 1

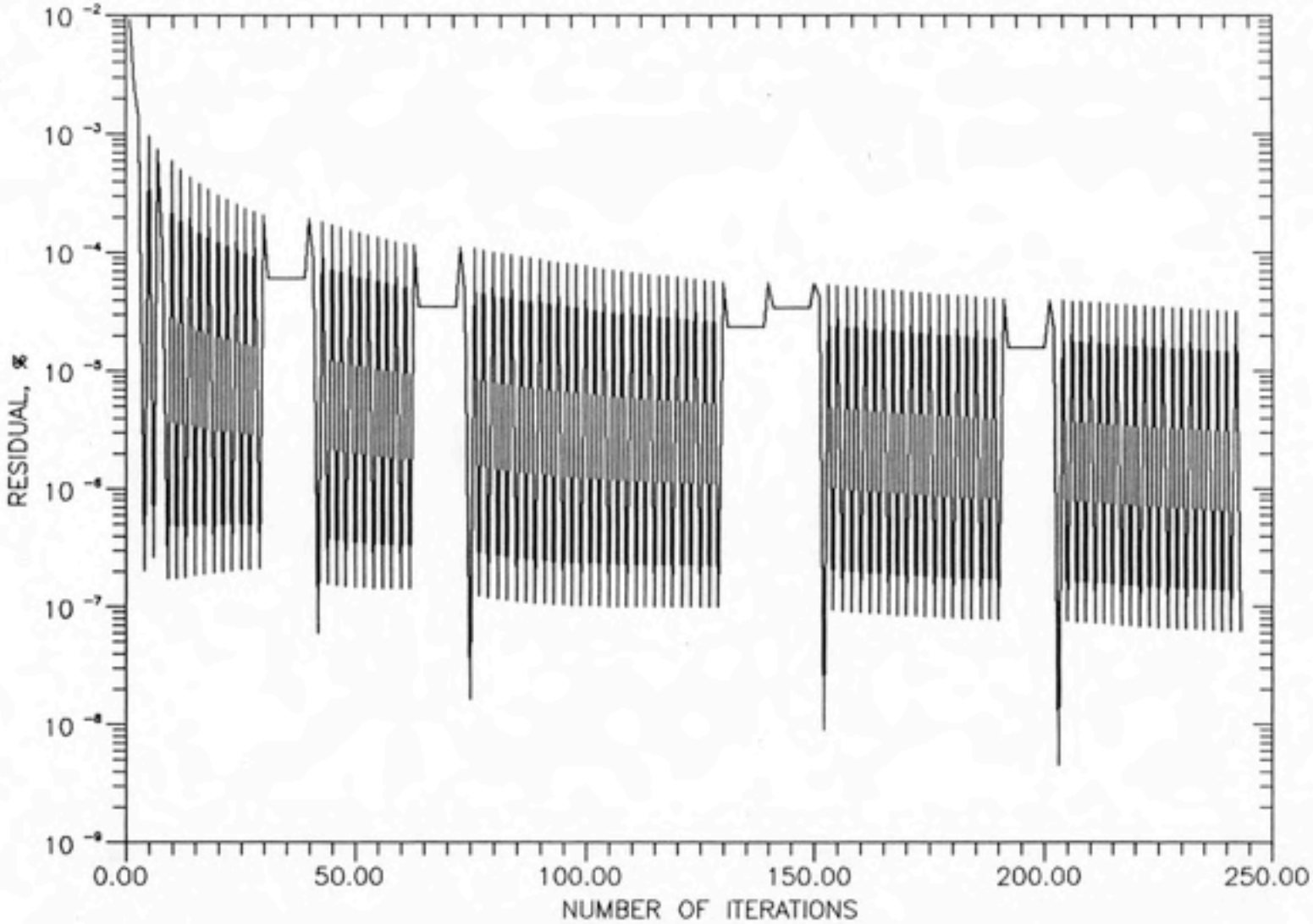


FIGURE 5.14B: RESIDUALS FOR TEST CASE 2

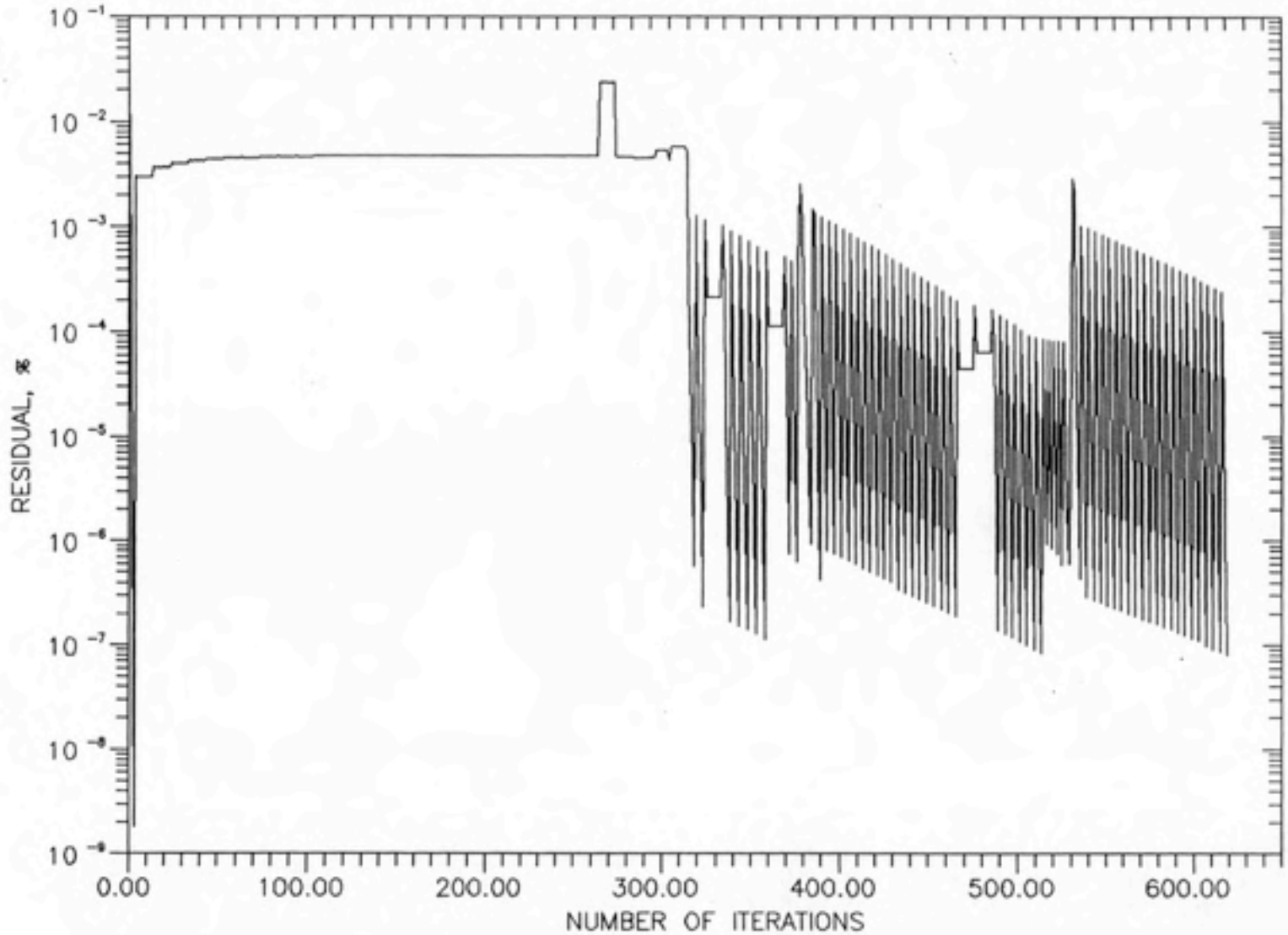


FIGURE 5.14C: RESIDUALS FOR TEST CASE 3

TABLE 5.3: PARAMETERS USED IN TEST CASES

SINGLE LAYERED CASE: ALL NODES BEGIN AS UNCONFINED	K = 0.001 L/T Sy = 0.05 b = 100 L Initial Head = 99.99 L Q = 2.0 L3/T t = 10 T
FIVE LAYER CASE: ALL NODES IN TOP LAYER BEGIN AS UNCONFINED	K = 0.001 L/T Sy = 0.05 b = 100 L Initial Head = 99.99 L Q = 2.0 L3/T t = 10 T
FIVE LAYER CASE: ALL NODES BEGIN AS CONFINED, SOME NODES BECOME UNCONFINED	K = 0.001 L/T Sy = 0.05 S = 0.00012 b = 100 L Initial Head = 99.99 L Q = 2.0 L3/T t = 10 T

6 MODEL APPLICATIONS: RESULTS AND DISCUSSIONS

This chapter describes some example applications that can be simulated with the REGFED model. The purpose of these applications is to demonstrate the computational efficiency of the model and to demonstrate that the model can handle groundwater flow problems that are more complex than those found in the validations of Chapter 5. The first two applications are related to contaminant transport problems. The third application is an analysis of an aquifer/aquitard groundwater system. The fourth application is a benchmark comparison with the most popular public domain three-dimensional flow model, the McDonald-Harbaugh model. All computer runs used in this chapter were performed on an IBM Personal Computer AT.

6.1 Two-Well Tracer Test

Studies relating to the analysis and prediction of solute transport between a recharging and discharging well pair have received considerable attention recently (Huyakorn et al., 1986). These studies are important from the standpoint of the design and analysis of two-well injection-withdrawal tracer tests in groundwater aquifers. Two-well tracer tests can provide several types of hydrodynamic data, including dispersion coefficients, velocity profiles, and contaminant travel times. For a conservative tracer, definition of the flow characteristics is most important. A schematic illustration of a two-well tracer test is shown in Figure 6.1.

In this application, the effects of a second withdrawal well on the performance of a two-well tracer test are also considered. It is hypothesized that the second withdrawal well captures a signifcant amount of the tracer flow. The aquifer is assumed to be unconfined. The discretization scheme for the application is shown in Figure 6.2, along with the various parameters used in the model simulation.

FIGURE 6.1
TWO-WELL INJECTION-WITHDRAWAL TRACER TEST

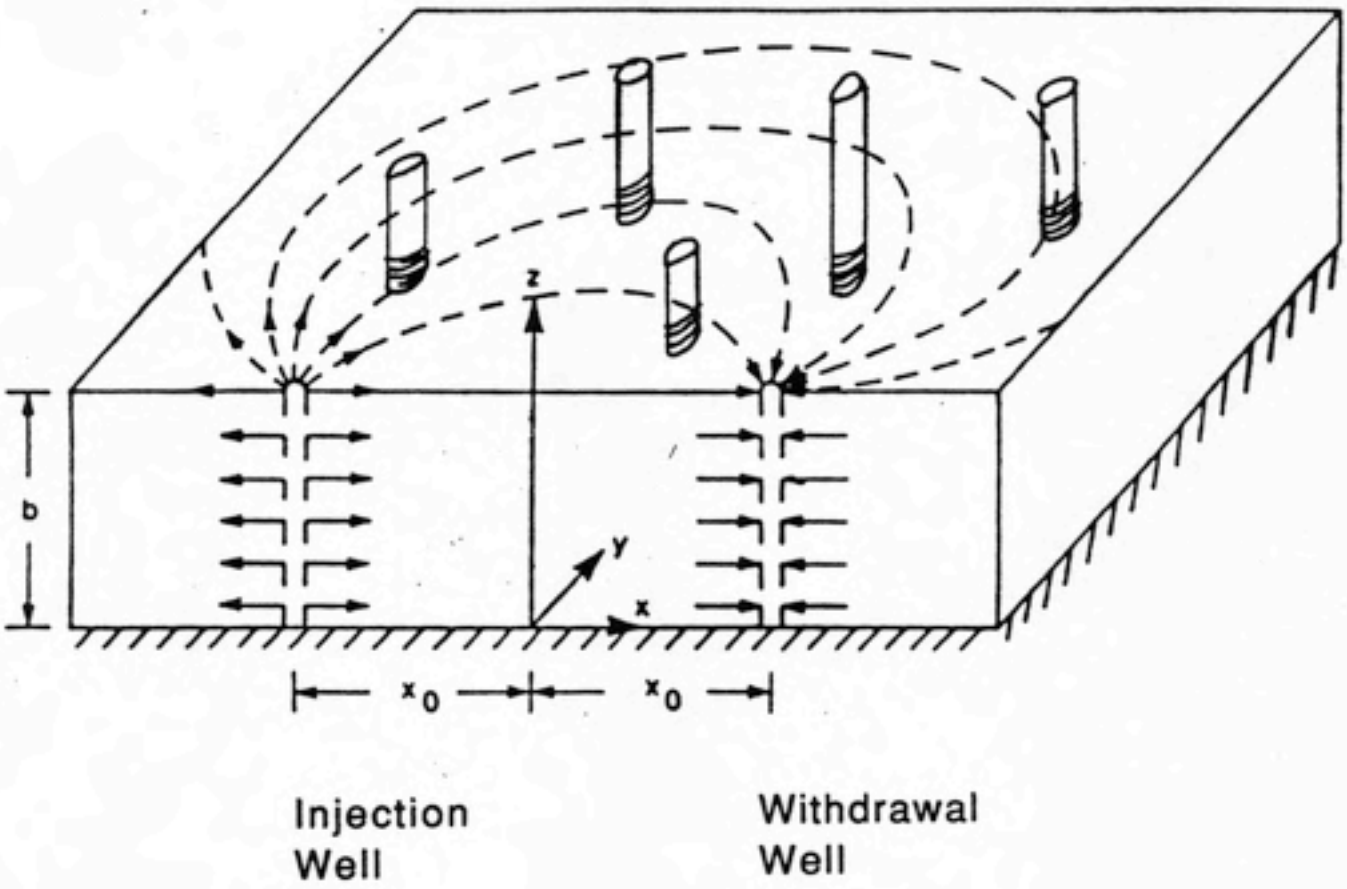
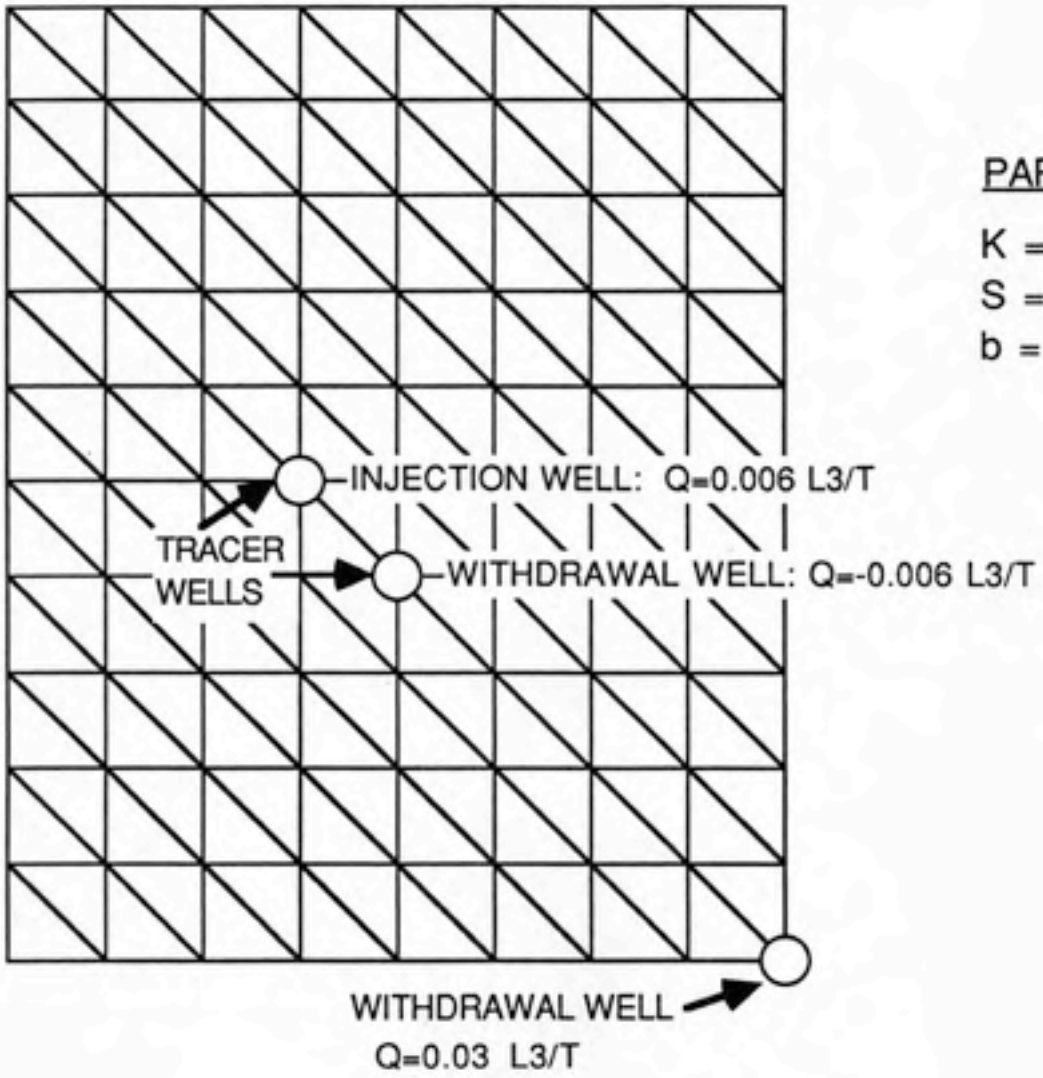


FIGURE 6.2
DISCRETIZATION SCHEME FOR TWO-WELL TRACER TEST



PARAMETERS

K = 0.0035 L/T
S = 0.0012
b = 300 L

The groundwater equipotentials at a vertical position aligned with the center of the screened portions of the tracer wells is shown in Figure 6.3. This figure demonstrates that the withdrawal well does not capture a significant amount of the injected tracer, for the particular parameters used here. But at later times, the influence of the second withdrawal well could extend to the pair of tracer wells.

6.2 Flow Within a Multi-Level Monitoring Well

Researchers often need to determine vertical head gradients, changes in water quality in the vertical section of an aquifer or changes between units of an interbedded aquifer system. This process requires that samples be taken at different subsurface observations. To accomplish this objective, a multi-level monitoring system can be installed, often consisting of a series of single boreholes containing several distinct monitoring locations (Pickens et al., 1981). A typical multi-level monitoring well installation is shown in Figure 6.4.

Howver, these multi-level wells may provide a conduit for contaminants to travel vertically through the aquifer, as shown in Figure 6.4. Thus, a groundwater sample taken from a particular vertical positon within the well may actually be a mixture of groundwater from different levels within the well. If a water quality sample from a discrete vertical postion in the aquifer is desired, the well may have to be modified.

In this application, the effects of a nearby pumping well (50 feet away) on vertical flow within a hypothetical monitoring well are simulated with the model. The pumping well effects are included as constant head boundary conditions that vary with depth. The simulated aquifer is a 200-foot deep unconfined aquifer. The monitoring well is screened from 178 feet to 190 feet above the aquifer bottom. The hydraulic characteristics of the monitoring well are approximated by setting extremely high hydraulic conductivities within the well (5-6 orders of magnitude higher than the aquifer media) and by setting the storativity and specific yields equal to one. The horizontal and vertical discretization schemes are shown in Figure 6.5, along with the various parameters used in the model

simulation.

The monitoring well application was simulated for three different time periods. The hydraulic heads near the centerline of the well for the three time periods are shown in Figure 6.6. This figure demonstrates that the differences in head in the monitoring well do not produce a significant vertical gradient, but that heads do change over time in the monitoring well, indicating that some flow in and out of the well occurs.

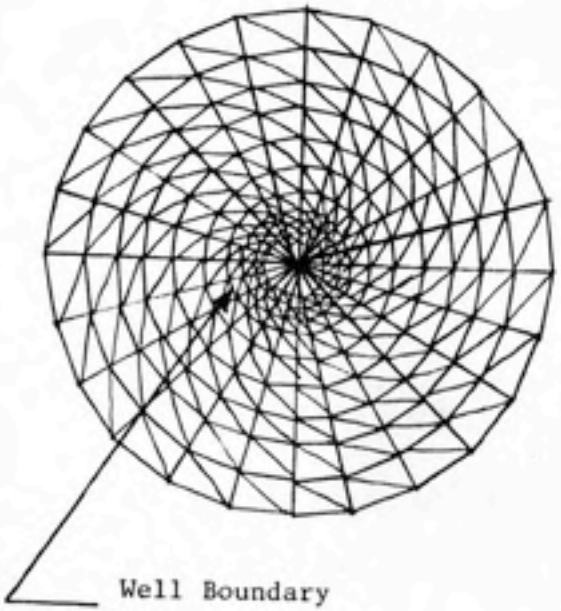
6.3 Flow Within an Aquifer/Aquitard System

In Chapter II, quasi three-dimensional flow models were discussed. Quasi three-dimensional models are suitable for simulating flow in groundwater systems where aquifers are separated by confining or semi-confining layers. Semi-confining layers are also known as aquitards. Such systems can be simplified by assuming that vertical components of flow within the aquifer are negligible and that the horizontal components of flow in the aquitard are negligible.

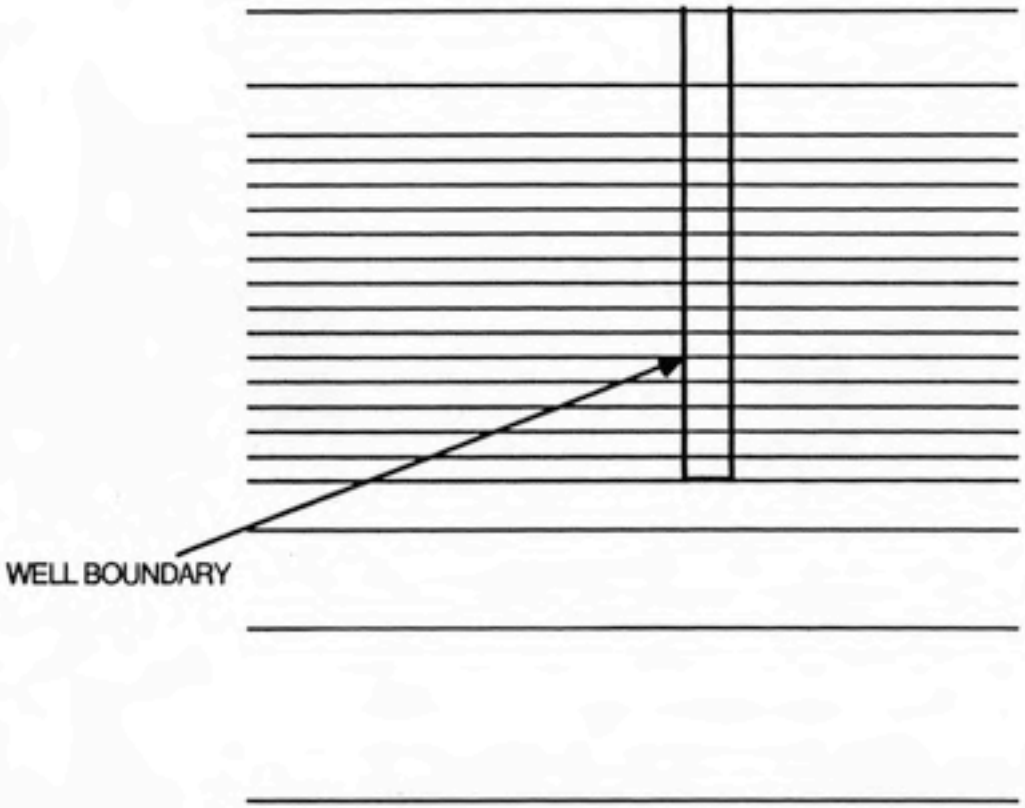
A schematic illustration of the simulated aquifer system is shown in Figure 2.1. As indicated in the figure, the contrast in hydraulic conductivities between the aquifer and the aquitard is two orders of magnitude. The aquifers are discretized into seven layers, while the aquitard is discretized into six layers. The model was not able to approximate horizontal flow in the aquifers and vertical flow in the aquitard within a reasonable amount of CPU time. The models inability to reproduce the problem is due to steep vertical gradients produced by large vertical changes in hydraulic conductivity. Smaller timesteps and finer vertical discretization may allow the model to overcome the vertical gradient problems. However, if the quasi three-dimensional assumptions are assumed to be correct, it may be advisable to use a quasi three-dimensional approach for this type of groundwater system. The quasi three-dimensional approach would reduce computational costs significantly, and should represent accurately the nature of flow in this type of groundwater system.

FIGURE 6.5

DISCRETIZATION SCHEME FOR
MULTI-LEVEL MONITORING WELL



PLAN VIEW



SECTION VIEW

PARAMETERS

K = 0.0035 L/T
Sy = 0.05
t = steady state

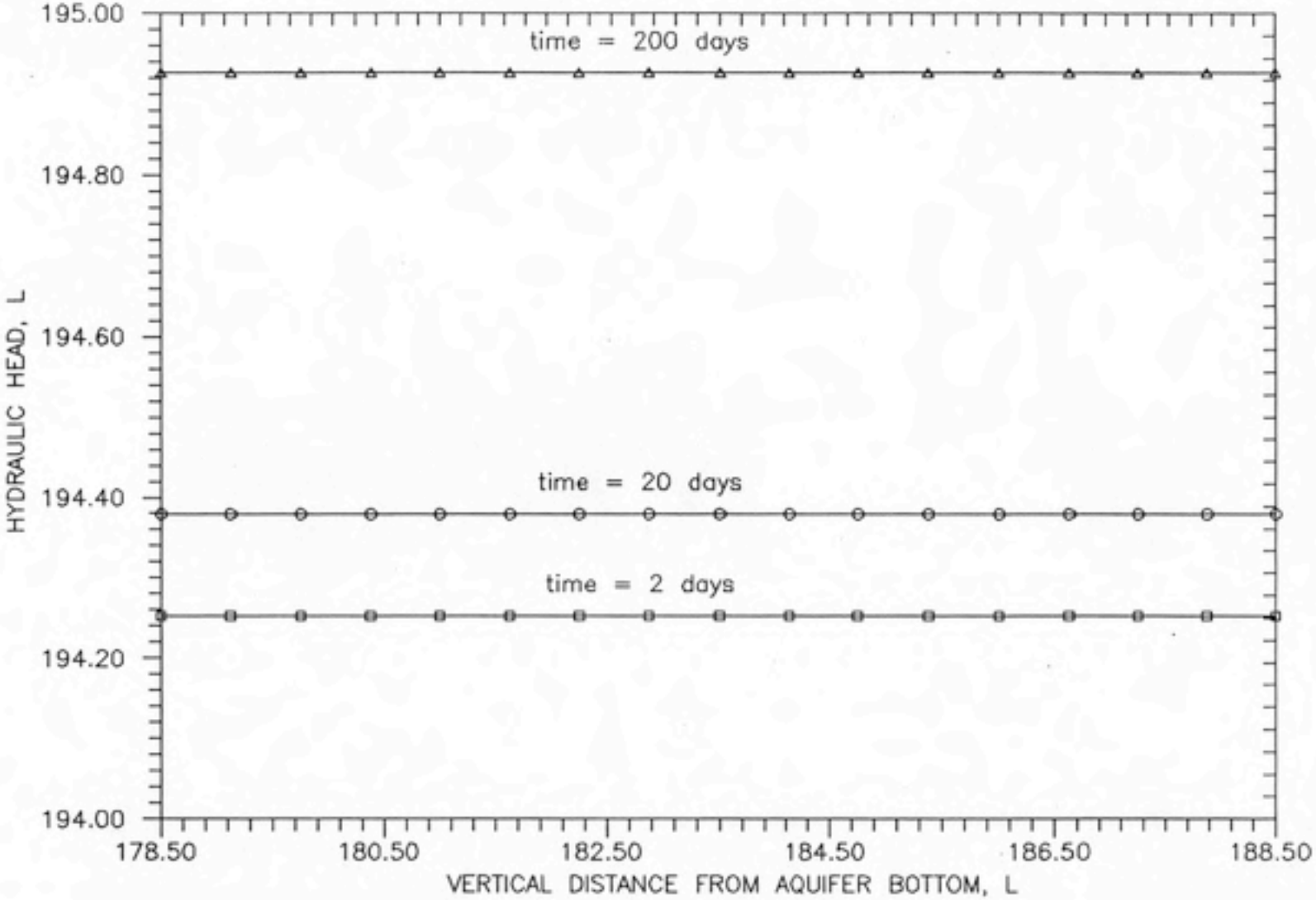


FIGURE 6.6: HYDRAULIC HEADS NEAR CENTERLINE OF MULTI-LEVEL MONITORING WELL

6.4 Comparison with McDonald-Harbaugh Model

The three-dimensional McDonald-Harbaugh model (McDonald and Harbaugh, 1984) represents the current state-of-the-art in public domain groundwater models. This model can provide a convenient benchmark for the REGFED model. The groundwater flow system chosen as a basis for comparison is a single pumped well in a confined aquifer, with constant head boundaries. The model simulations for this system can be validated by the Theis solution described in Chapter 5. The data sets submitted to the models contain an equal number of nodes and layers, except that the McDonald-Harbaugh model requires an extra row and column to simulate no-flow boundaries.

The model simulations are compared in Figure 6.7, along with the Theis simulation. The figure shows that both models accurately simulate the response of the groundwater system. The computational effort required to simulate the system is shown for both models in Table 6.1, along with the parmeters used in the sample problem. These results indicate that the REGFED model requires less CPU time than the McDonald-Harbaugh model to simulate the system, while providing better mass balance errors. The models were run on an IBM Personal Computer AT; the CPU time does not include input or ouptut of data.

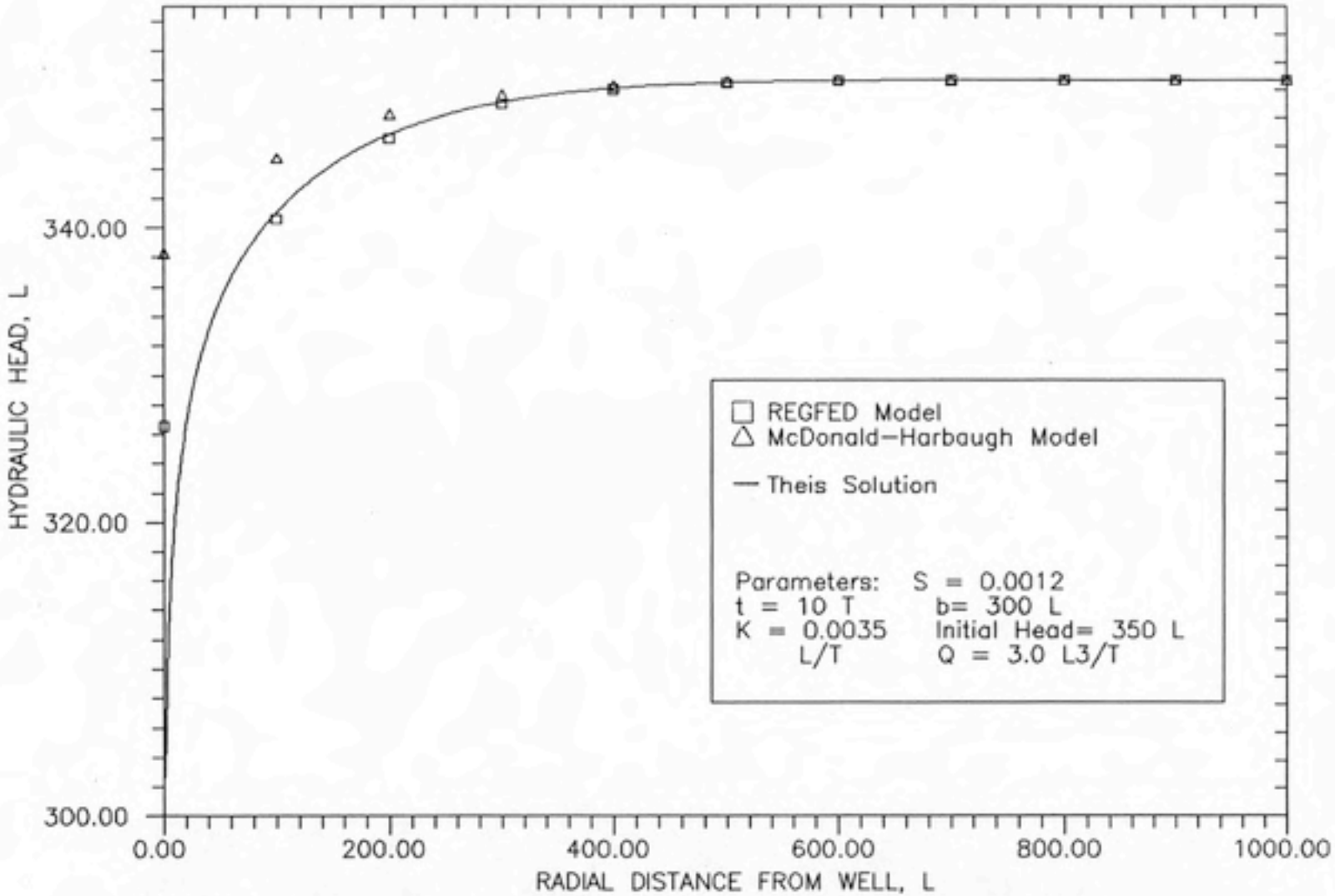


FIGURE 6.7: COMPARISON OF MODEL SIMULATIONS-- MCDONALD-HARBAUGH MODEL, REGFED MODEL, AND THEIS SOLUTION (CONFINED FLOW)

TABLE 6.1

COMPARISON OF PERFORMANCE OF REGFED MODEL AND MCDONALD-HARBAUGH MODEL

MODEL	CPU TIME (min.)	WATER BALANCE ERROR
MCDONALD-HARBAUGH	36.4	7.32×10^{-4}%
REGFED	28.5	2.21×10^{-5}%

SIMULATION CONDITIONS AND PARAMETERS

-Confined Flow
- 3 equally spaced layers
- 225 nodes
- 392 elements
- K = 0.0035 L/T
- b = 300 L
- S = 0.00012
- Q = 0.5 L3/T
- initial head = 350 L

7 CONCLUSIONS AND RECOMMENDATIONS

7.1 Conclusions

From the model development, and the tests and applications performed with the REGFED model, certain conclusions can be drawn. These conclusions include

- A mixed method consisting of finite-elements and finite-differences is an efficient and accurate method for modeling groundwater flow; the ALALS algorithm is a suitable example of such a method.
- The REGFED model compares favorably with the analytical solutions used in this report for model testing
- Mass balance errors are minimal for the test cases, except where drained node transitions occur.
- For situations where the model did not validate well, finer grid spacing or timestep sizes could improve model accuracy.
- The WELFED model can efficiently simulate some example applications that are relatively difficult, compared to the validation conditions.
- The model may not be able to accurately simulate aquifer/confining-layer conditions without significant computational efforts and storage requirements.
- Steep vertical gradients relative to nodal spacing have a deleterious effect on model accuracy.

7.2 Recommendations

The following recommendations can be made for improving the performance of the REGFED model.

- The model should be modified to include the quasi three-dimensional approach for modeling aquifer/confining layer conditions.
- Iteration schemes other than Picard Iteration should be explored, to see if convergence for unconfined flow conditions can be made quicker or more stable. Other schemes could include Newton-Raphson schemes or modified conjugate gradient schemes.
- Spacing criteria for nodal and timestep spacing should be specified such that model accuracy is optimized.
- Improvements to the confined/unconfined and drained node transitions should be made so that mass balance errors are minimized.
- In addition to excluding entire layers with confined elements from the matrix reforming process, the algorithm should be modified so that confined elements within layers that also include unconfined elements, can be excluded from matrix reforming.
- The model should be modified to include other types of finite elements that can fit various boundary or other conditions more efficiently and accurately.
- In general, steep vertical gradients should be avoided by utilizing finer discretization schemes or timestep sizes.
- An automatic timestep generator that minimizes water balance errors and steep vertical gradients should be included.
- An input data preprocessor should be added to the model in order to ease the burden of inputing data for large problems.

8 NOTATION

C = solution concentration (M/L^3) .
t = time (T) .
$\bar{v}$ = groundwater velocity (L/T) .
$\mathbf{D}$ = hydrodynamic dispersion tensor (L^2/T) .
$\nabla\cdot$ = divergence operator .
∇ = gradient operator .
$\left(\frac{\partial C}{\partial t}\right)_{rxn}$ = reactive term $(M/L^3/T)$.
$\Gamma(C)$ = source or sink term $(M/L^3/T)$.
D_{ij} = i,j term of dispersion tensor (L^2/T) .
i,j = components of Cartesian coordinate system .
α_T = transverse dispersivity (L) .
α_L = longitudinal dispersivity (L) .
$\bar{v}$ = average groundwater velocity (L/T) .
D^* = effective molecular diffusion coefficient (L^2/T) .
q = specific discharge (L/T) .
K = hydraulic conductivity (L/T) .
h = hydraulic head (L) .
$\frac{\partial h}{\partial x}$ = groundwater gradient (*dimensionless*) .
v = pore velocity (L/T) .
n = porosity (*dimensionless*) .
h = hydraulic head (L) .
$\mathbf{K}$ = hydraulic conductivity tensor (L/T) .
S_s = specific storage $(1/L)$.
$\Gamma(h)$ = source or sink term $(1/T)$.
K_x, K_y, K_z = components of conductivity in the x, y, and z directions, respectively (L/T) .
T_x, T_y = components of transmissivity in the x, and y directions, respectively, (L^2/T) .
S_y = specific yield (*dimensionless*) .
$\Gamma'(h)$ = vertically averaged source or sink term (L/T) .
Δx = distance between spatial locations in x-direction (L) .
Δh = change in hydraulic head from x_i to x_{i+1} (L) .
n = porosity (*dimensionless*) .
$O(\Delta x)$ = remainder of the Taylor series terms, including those with powers of Δx and higher .
$\hat{u}$ = trial function .
$a_0, a_1,$ and a_2 = coefficients related to element position and geometry .
$N_i^e, N_j^e,$ and N_m^e = basis functions .
$x_i, x_j, x_m, y_i, y_j,$ and y_m = coordinates of triangle vertices (L) .
A_e = area of triangle (L^2) .
N_n = basis function for node n .
N = number of nodes .

ε = error resulting from substitution of approximated form of u .
$\mathcal{R}$ = problem domain .
W_i = nodal weighting functions .
n_x and n_y = components of outward normal vector .
$\mathcal{B}$ = boundary of problem domain .
e = element region .
N = number of nodes .
$[G]$ = Global coefficient matrix .
$[E]$ = Element coefficient matrix .
$\{\vec{u}\}$ = dependent variable vector .
E_{ii}, etc. = integral terms found in Equation 3.20 for nodes in a triangular element .
$G_{m,i}$ = members of the Global coefficient matrix .
$\{\vec{b}\}$ = vector of coefficients representing boundary conditions .
h = hydraulic head (L) .
$\hat{h}$ = trial function for hydraulic head (L) .
$N_n(x,y)$ = two-dimensional basis function in the x-y plane .
h_n = nodal parameter dependent on z and time (L) .
n_{xy} = number of nodes in the x-y plane of each layer .
N_i = two-dimensional basis function in the x-y plane .
$\mathcal{R}$ = x-y problem domain .
$\mathcal{B}$ = boundary of the cross-section of $\mathcal{R}$.
$\frac{\partial h}{\partial n}$ = outward normal derivative on $\mathcal{B}$.
K_n = normal component of hydraulic conductivity on $\mathcal{B}$.
$k+1/2, k-1/2, k+1$, and $k-1$ = indices as shown in Figure 3.4 .
Δz terms are as shown in Figure 3.3 (L) .
K_{z+} = upper-weighted, harmonic-mean hydraulic conductivity (L/T) .
K_{z-} = lower-weighted, harmonic-mean hydraulic conductivity (L/T) .
l = index for present time step .
$l+1$ = index for next time step .
Δt = time increment for time step (T) .
$*$ = index referring to predicted solutions .
$[KH], \frac{[ST]}{\Delta t}, \{\vec{\Gamma}(h)\}, \{\vec{F}(h)\}, [KL]$, and $[KU]$ = matrices and vectors of integrals in Equation 3.11 .
n_i, n_j, and n_m = nodal indices on triangular element .
x_i, x_j, x_m, y_i, y_j, and y_m = coordinates of triangle vertices (L) .
A_e = area of triangle (L^2) .
a_{ij} = i,j element of $[A]$ matrix .
n = outward normal vector .
$\mathcal{B}$ = problem boundary .
g = arbitrary boundary function .
Γ_r = recharge rate (L/T) .
$u_{\mathcal{B}}$ = problem boundary .
f = arbitrary boundary function .
Γ_p = withdrawal or injection quantity (L^3/T) .
$[A]$ = sum of left-hand-side matrices .

$\{\vec{x}\}$ = the solution, or hydraulic head vector .
$\{\vec{b}\}$ = sum of right-hand-side vectors .
T = transmissivity = Kb (L^2/T) .
where b = saturated thickness in aquifer layer (L) .
S = specific yield (unconfined aquifer) or storativity (confined aquifer) (*dimensionless*) .
$(x_1, x_2, \ldots, x_{N_I})$ = unknowns .
r = iteration counter .
$[A]_{IJ}^{-1}$ = elements of the inverse of matrix $[A]$.
ε_b = prescribed residual tolerance .
max_J = maximum over all nodes .
Q = rate of change in storage of water in an element (L^3/T) .
top = elevation of the top of an element layer (L) .
S^l = storage factor (specific yield or storativity) in effect at time l (*dimensionless*) .
S^{l+1} = storage factor (specific yield or storativity) in effect at time $l+1$ (*dimensionless*) .
t_T = total time elapsed over simulation (T) .
ε_{wb} = water balance error (*dimensionless*) .
h_i = head at start of simulation (L) .
h_f = head at end of simulation (L) .
s = drawdown = initial head - new head (L) .
r = radial distance from well (L) .
$W(u_c)$ = well function for nonleaky aquifer (*dimensionless*) .
u_c = argument of the well function (*dimensionless*) .
T = transmissivity (L^2/T) .
S = storativity (*dimensionless*) .
Q = pumping rate (L^3/T) .
t = time (T) .
$W\left(u_c, \frac{n_s \pi r}{b}\right)$ = well function for leaky aquifer (*dimensionless*) .
$W(u_c)$ = well function for nonleaky aquifer (*dimensionless*) .
z = vertical distance from top of aquifer (L) .
n_s = summation index .
u_u = argument (modified for unconfined flow) of the well function (*dimensionless*) .
S_y = specific yield (*dimensionless*) .
s_c = corrected drawdown (L) .
s_o = drawdown calculated from Equation 4.3 (L) .
b = aquifer thickness (L) .
h_o = head at up-gradient boundary (L) .
h_1 = head at down-gradient boundary (L) .
x = distance from up-gradient boundary (L) .
x_1 = length of aquifer (L) .
Γ_r = recharge rate (L/T) .

9 REFERENCES

Abriola, L.M. and G.F. Pinder (1982), Calculation of Velocity in Three Space Dimensions from Hydraulic Head Measurements, Ground Water, 20(2), pp. 205-213.

Anderson, M.P. (1979), Using Models to Simulate the Movement of Contaminants through Groundwater Flow Systems, CRC Critical Reviews in Environmental Control, 9(2), pp. 97-156.

Anderson, M.A. (1984), Movement of Contaminants in Groundwater: Groundwater Transport–Advection and Dispersion, in Studies in Geophysics: Groundwater Contamination, National Academy Press, Washington, D.C., pp. 46-64.

Arnett, R.C., R.A. Deju, R.E. Gephart, and R.B. Lantz (1977), The Importance of Three Dimensional Groundwater Contaminant Modeling at the Hanford Waste Nuclear Facility, Transactions of the American Geophysical Union EOS, 58(Abstr.), p. 1138.

Babu, D.K., G.F. Pinder, and D.K. Sunada (1982), A Three Dimensional Hybrid Finite Element-Finite Difference Scheme for Groundwater Simulation, in 10th IMACS World Congress, National Association for Mathematics and Computers, pp. 292-294.

Batu, V. (1984), A Finite Element Dual Mesh Method to Calculate Nodal Darcy Velocities in Nonhomogeneous and Anisotropic Aquifers, Water Resources Research, 20(11), pp. 1705-1717.

Bear, J. (1972), Dynamics of Fluids in Porous Media, Elsevier Publishing Co., New York, 367 pp.

Bear, J. (1979), Hydraulics of Groundwater, McGraw-Hill, Inc., New York.

Bredehoeft, J.D. and G.F. Pinder (1970), Digital Analysis of Areal Flow in Multiaquifer Groundwater Systems: A Quasi Three- Dimensional Model, Water Resources Research, 6(3), pp. 883-888.

Cooley, R.L. (1971), A Finite-Difference Method for Unsteady Flow in Variably Saturated Porous Media: Application to a Single Pumping Well, Water Resources Research, 7(10), pp. 1607-1625.

de Marsily, G. (1986), Quantitative Hydrogeology: Groundwater Hydrology for Engineers, Academic Press, Inc., Orlando, FL, 440 pp.

Domenico, P.A. and G.A. Robbins (1984), A Dispersion Scale Effect in Model Calibrations and Field Tracer Experiments, Journal of Hydrology, 70, pp. 123-132.

Driscoll, F.G. (1986), Groundwater and Wells, Second Edition, Johnson Division, St. Paul, MN, 1108 pp.

Freeze, R.A. (1971), Three-Dimensional, Transient, Saturated-Unsaturated Flow in a Groundwater Basin, Water Resources Research, 7(2), pp. 347-366.

Freeze, R.A. (1975), A Stochastic-Conceptual Model of One Dimensional Groundwater Flow in Nonuniform Homogeneous Media, Water Resources Research, 11(2), pp. 725-732.

Freeze, R.A. and J.A. Cherry (1979), Groundwater Contamination, in Groundwater, Prentice-Hall, Inc., Englewood Cliffs, NJ, pp. 444- 447.

Fried, J.J. (1975), The Theory of Dispersion in Porous Media, in Groundwater Pollution, Elsevier Scientific Publishing Co., Amsterdam, pp. 5-46.

Fried, J.J. and M.A. Combarnous (1971), Dispersion in Porous Media, Advances in Hydroscience, edited by V.T. Chow, 7, pp. 169-282.

Frind, E.O. and G.B. Matanga (1985), The Dual Formulation of Flow for Contaminant Transport Modeling 1. Review of Theory and Accuracy Aspects, Water Resources Research, 21(2), pp. 159-169.

Frind, E.O. and M.J. Verge (1978), Three-Dimensional Modeling of Groundwater Flow Systems, Water Resources Research, 14(5), pp. 844- 856.

Frind, E.O., G.B. Matanga, and J.A. Cherry (1985), The Dual Formulation of Flow for Contaminant Transport Modeling 2. The Borden Aquifer, Water Resources Research, 21(2), pp. 170-182.

Gambolati, G. and F. Sartoretto (1986), A Conjugate Gradient Finite Element Model of Flow for Large Multiaquifer Systems, Water Resources Research, 22(7), pp. 1003-1015.

Gambolati, G., G. Pini, and T. Tucciarelli (1986), A 3-D Finite Element Conjugate Gradient Model of Subsurface Flow with Automatic Mesh Generation, Adavnces in Water Resources, 9, pp. 34-41.

Gelhar, L.W. (1976), Stochastic Analysis of Flow in Aquifers, in Advances in Groundwater Hydrology, Saleem, Z.A., ed., American Water Resources Association, Minnestoa, MN, pp. 57-78.

Gelhar, L.W. and C.L. Axness (1983), Three-Dimensional Stochastic Analysis of Macrodispersion in Aquifers, Water Resources Research, 19(1), pp. 161-180.

Gelhar, L.W., A. Mantoglou, C. Welty, and K.R. Rehfeldt (1985), A Review of Field-Sacle Physical Solute Transport Processes in Saturated and Unsaturated Porous Media, Electric Power Research Institute, Palo Alto, CA, Report EA-4190.

6 MODEL APPLICATIONS: RESULTS AND DISCUSSIONS

This chapter describes some example applications that can be simulated with the REGFED model. The purpose of these applications is to demonstrate the computational efficiency of the model and to demonstrate that the model can handle groundwater flow problems that are more complex than those found in the validations of Chapter 5. The first two applications are related to contaminant transport problems. The third application is an analysis of an aquifer/aquitard groundwater system. The fourth application is a benchmark comparison with the most popular public domain three-dimensional flow model, the McDonald-Harbaugh model. All computer runs used in this chapter were performed on an IBM Personal Computer AT.

6.1 Two-Well Tracer Test

Studies relating to the analysis and prediction of solute transport between a recharging and discharging well pair have received considerable attention recently (Huyakorn et al., 1986). These studies are important from the standpoint of the design and analysis of two-well injection-withdrawal tracer tests in groundwater aquifers. Two-well tracer tests can provide several types of hydrodynamic data, including dispersion coefficients, velocity profiles, and contaminant travel times. For a conservative tracer, definition of the flow characteristics is most important. A schematic illustration of a two-well tracer test is shown in Figure 6.1.

In this application, the effects of a second withdrawal well on the performance of a two-well tracer test are also considered. It is hypothesized that the second withdrawal well captures a signifcant amount of the tracer flow. The aquifer is assumed to be unconfined. The discretization scheme for the application is shown in Figure 6.2, along with the various parameters used in the model simulation.

FIGURE 6.1
TWO-WELL INJECTION-WITHDRAWAL TRACER TEST

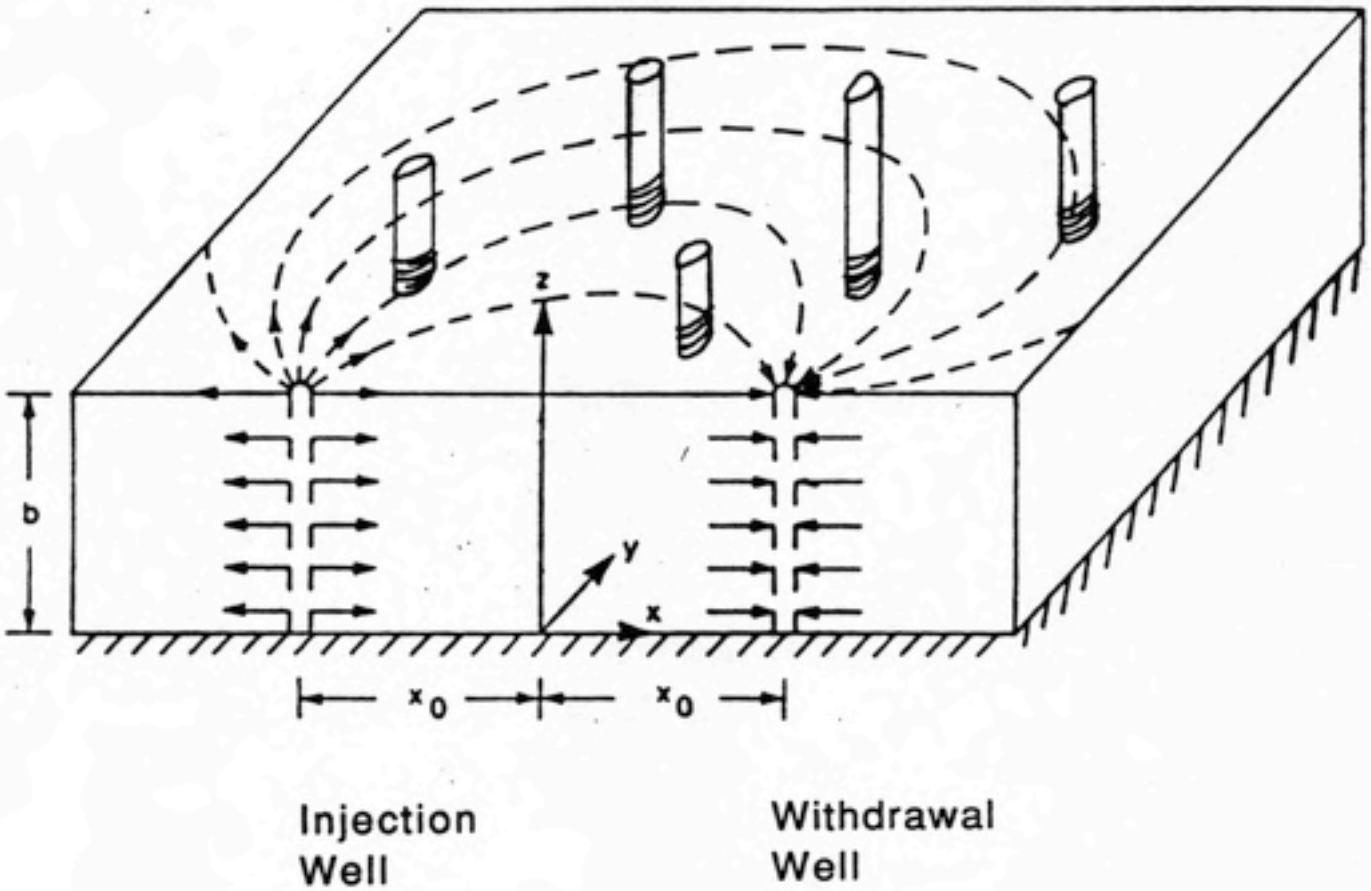
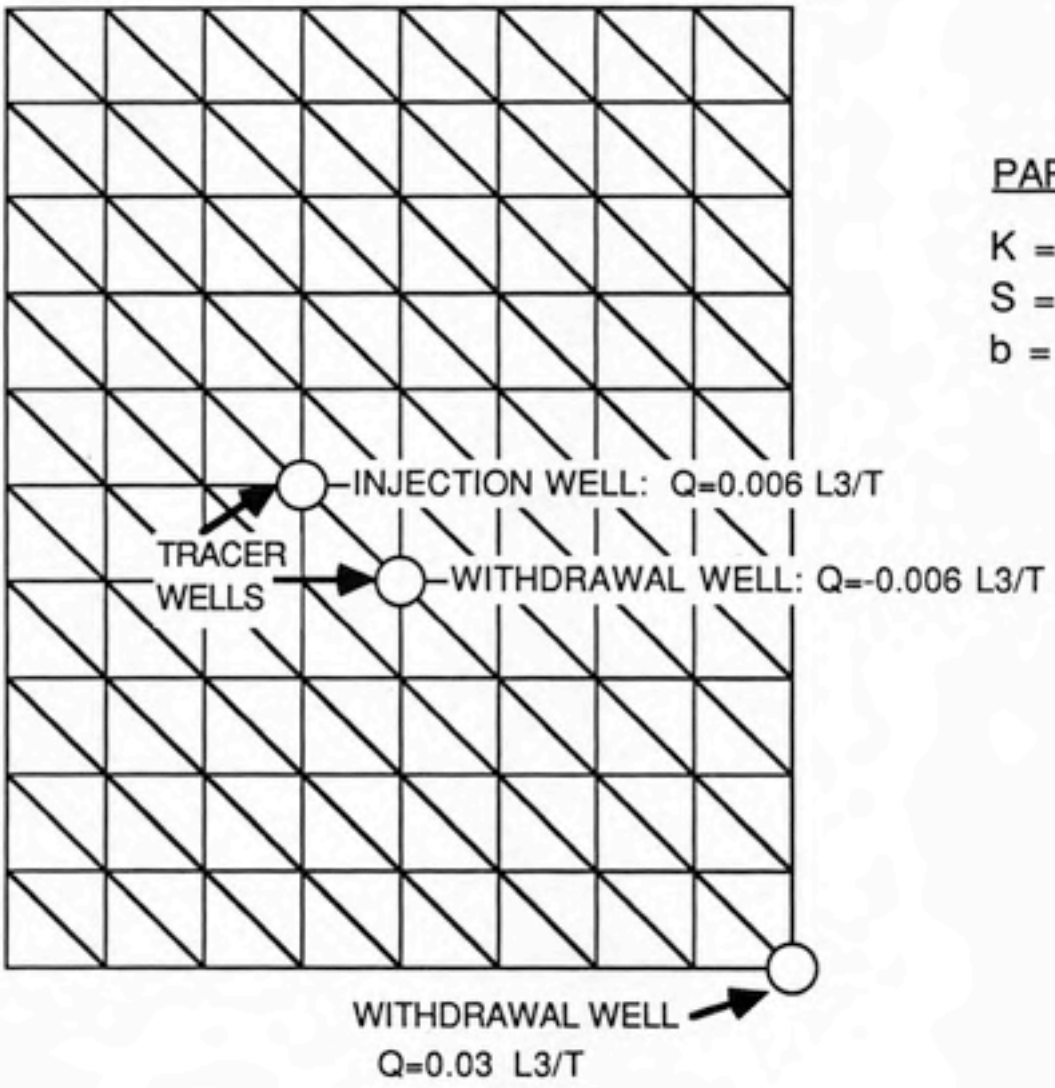


FIGURE 6.2
DISCRETIZATION SCHEME FOR TWO-WELL TRACER TEST



PARAMETERS

K = 0.0035 L/T
S = 0.0012
b = 300 L

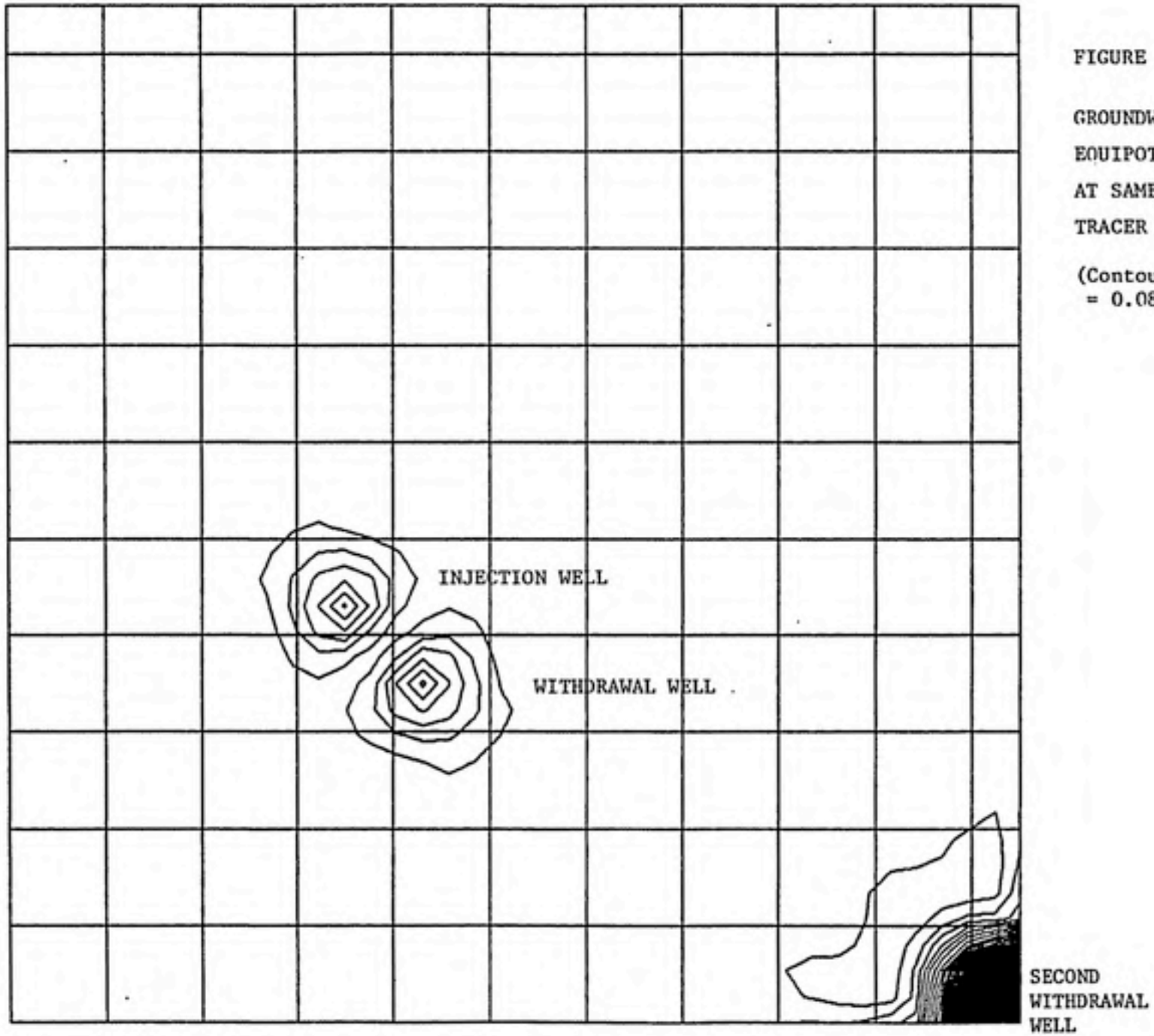


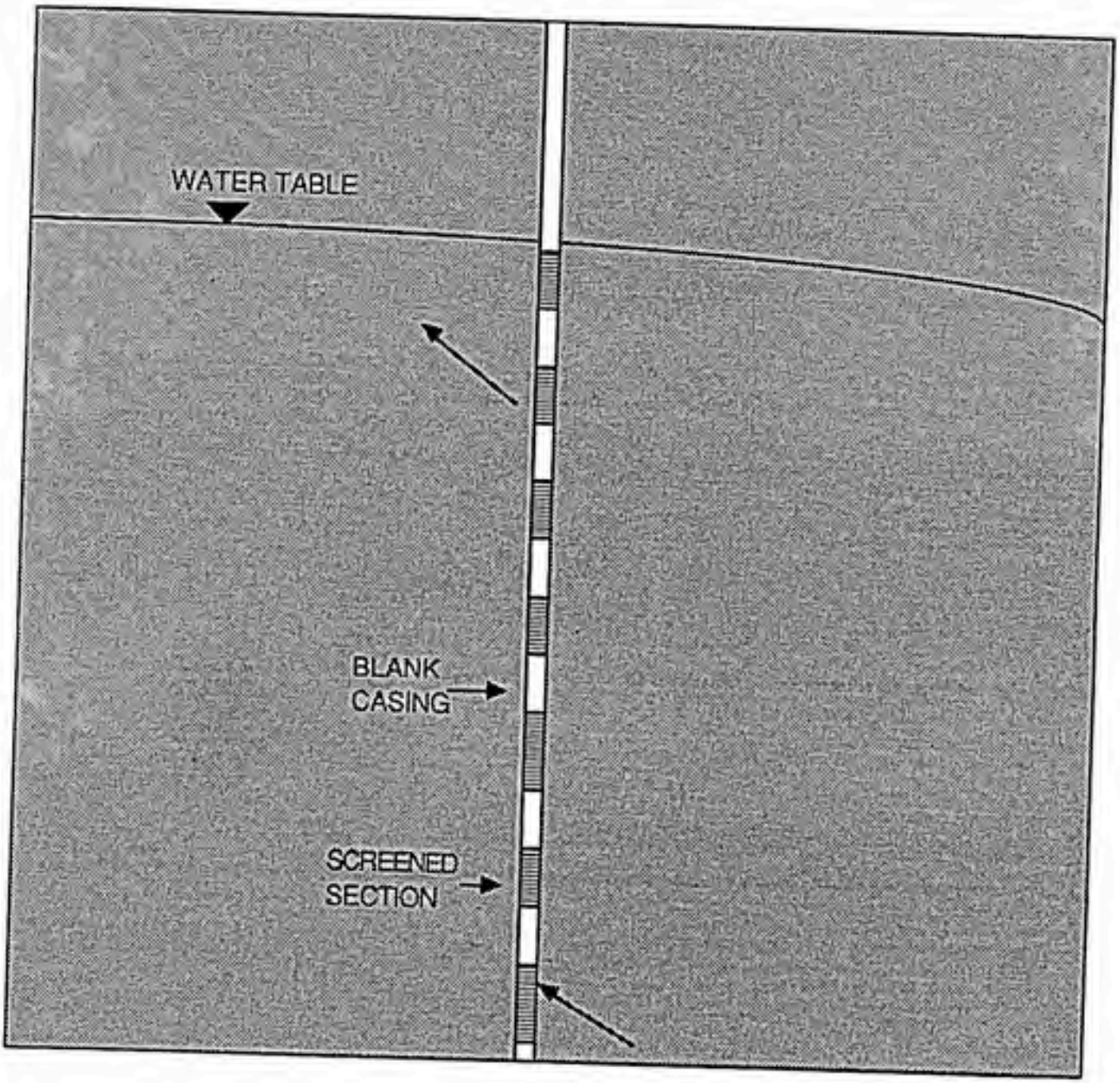
FIGURE 6.3

GROUNDWATER EQUIPOTENTIALS AT SAME DEPTH AS TRACER WELLS

(Contour Interval = 0.08 ft.)

FIGURE 6.4

ILLUSTRATION OF TYPICAL MULTI-LEVEL MONITORING WELL



The groundwater equipotentials at a vertical position aligned with the center of the screened portions of the tracer wells is shown in Figure 6.3. This figure demonstrates that the withdrawal well does not capture a significant amount of the injected tracer, for the particular parameters used here. But at later times, the influence of the second withdrawal well could extend to the pair of tracer wells.

6.2 Flow Within a Multi-Level Monitoring Well

Researchers often need to determine vertical head gradients, changes in water quality in the vertical section of an aquifer or changes between units of an interbedded aquifer system. This process requires that samples be taken at different subsurface observations. To accomplish this objective, a multi-level monitoring system can be installed, often consisting of a series of single boreholes containing several distinct monitoring locations (Pickens et al., 1981). A typical multi-level monitoring well installation is shown in Figure 6.4.

Howver, these multi-level wells may provide a conduit for contaminants to travel vertically through the aquifer, as shown in Figure 6.4. Thus, a groundwater sample taken from a particular vertical positon within the well may actually be a mixture of groundwater from different levels within the well. If a water quality sample from a discrete vertical postion in the aquifer is desired, the well may have to be modified.

In this application, the effects of a nearby pumping well (50 feet away) on vertical flow within a hypothetical monitoring well are simulated with the model. The pumping well effects are included as constant head boundary conditions that vary with depth. The simulated aquifer is a 200-foot deep unconfined aquifer. The monitoring well is screened from 178 feet to 190 feet above the aquifer bottom. The hydraulic characteristics of the monitoring well are approximated by setting extremely high hydraulic conductivities within the well (5-6 orders of magnitude higher than the aquifer media) and by setting the storativity and specific yields equal to one. The horizontal and vertical discretization schemes are shown in Figure 6.5, along with the various parameters used in the model

simulation.

The monitoring well application was simulated for three different time periods. The hydraulic heads near the centerline of the well for the three time periods are shown in Figure 6.6. This figure demonstrates that the differences in head in the monitoring well do not produce a significant vertical gradient, but that heads do change over time in the monitoring well, indicating that some flow in and out of the well occurs.

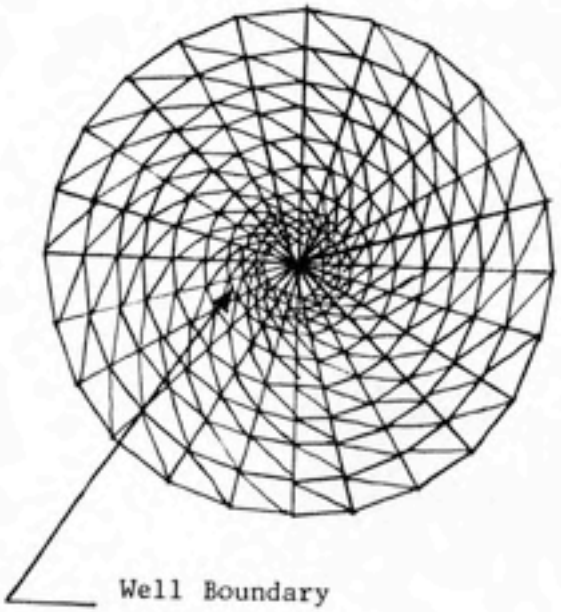
6.3 Flow Within an Aquifer/Aquitard System

In Chapter II, quasi three-dimensional flow models were discussed. Quasi three-dimensional models are suitable for simulating flow in groundwater systems where aquifers are separated by confining or semi-confining layers. Semi-confining layers are also known as aquitards. Such systems can be simplified by assuming that vertical components of flow within the aquifer are negligible and that the horizontal components of flow in the aquitard are negligible.

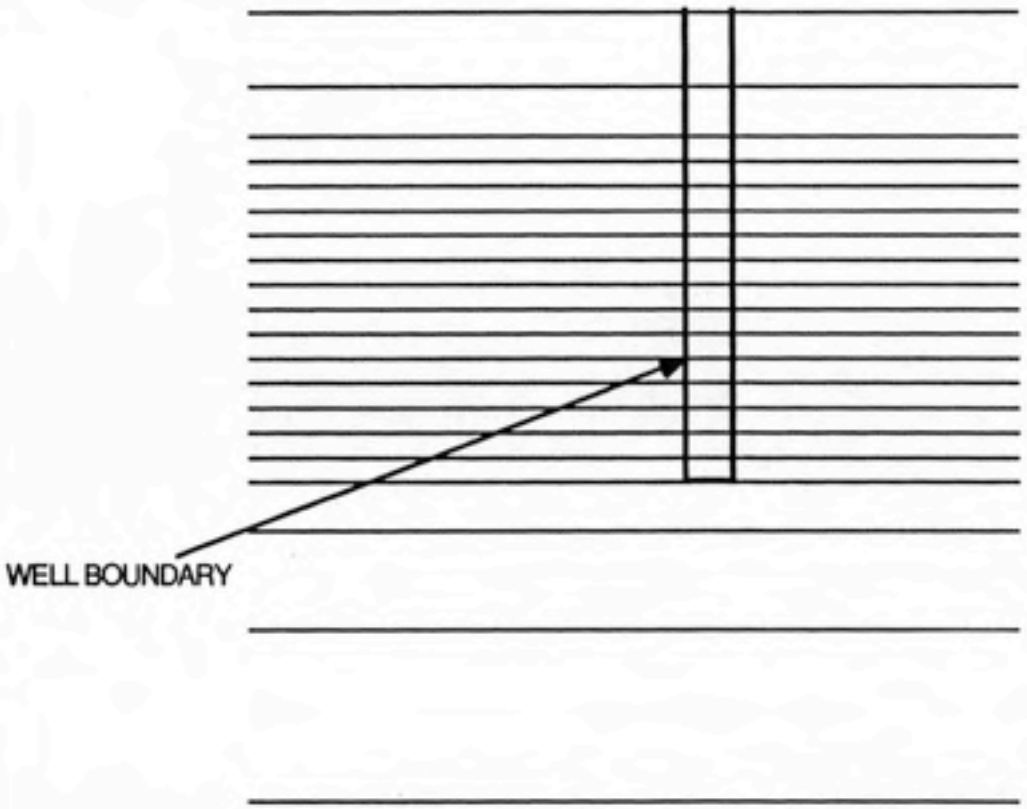
A schematic illustration of the simulated aquifer system is shown in Figure 2.1. As indicated in the figure, the contrast in hydraulic conductivities between the aquifer and the aquitard is two orders of magnitude. The aquifers are discretized into seven layers, while the aquitard is discretized into six layers. The model was not able to approximate horizontal flow in the aquifers and vertical flow in the aquitard within a reasonable amount of CPU time. The models inability to reproduce the problem is due to steep vertical gradients produced by large vertical changes in hydraulic conductivity. Smaller timesteps and finer vertical discretization may allow the model to overcome the vertical gradient problems. However, if the quasi three-dimensional assumptions are assumed to be correct, it may be advisable to use a quasi three-dimensional approach for this type of groundwater system. The quasi three-dimensional approach would reduce computational costs significantly, and should represent accurately the nature of flow in this type of groundwater system.

FIGURE 6.5

DISCRETIZATION SCHEME FOR
MULTI-LEVEL MONITORING WELL



PLAN VIEW



SECTION VIEW

PARAMETERS

K = 0.0035 L/T
Sy = 0.05
t = steady state

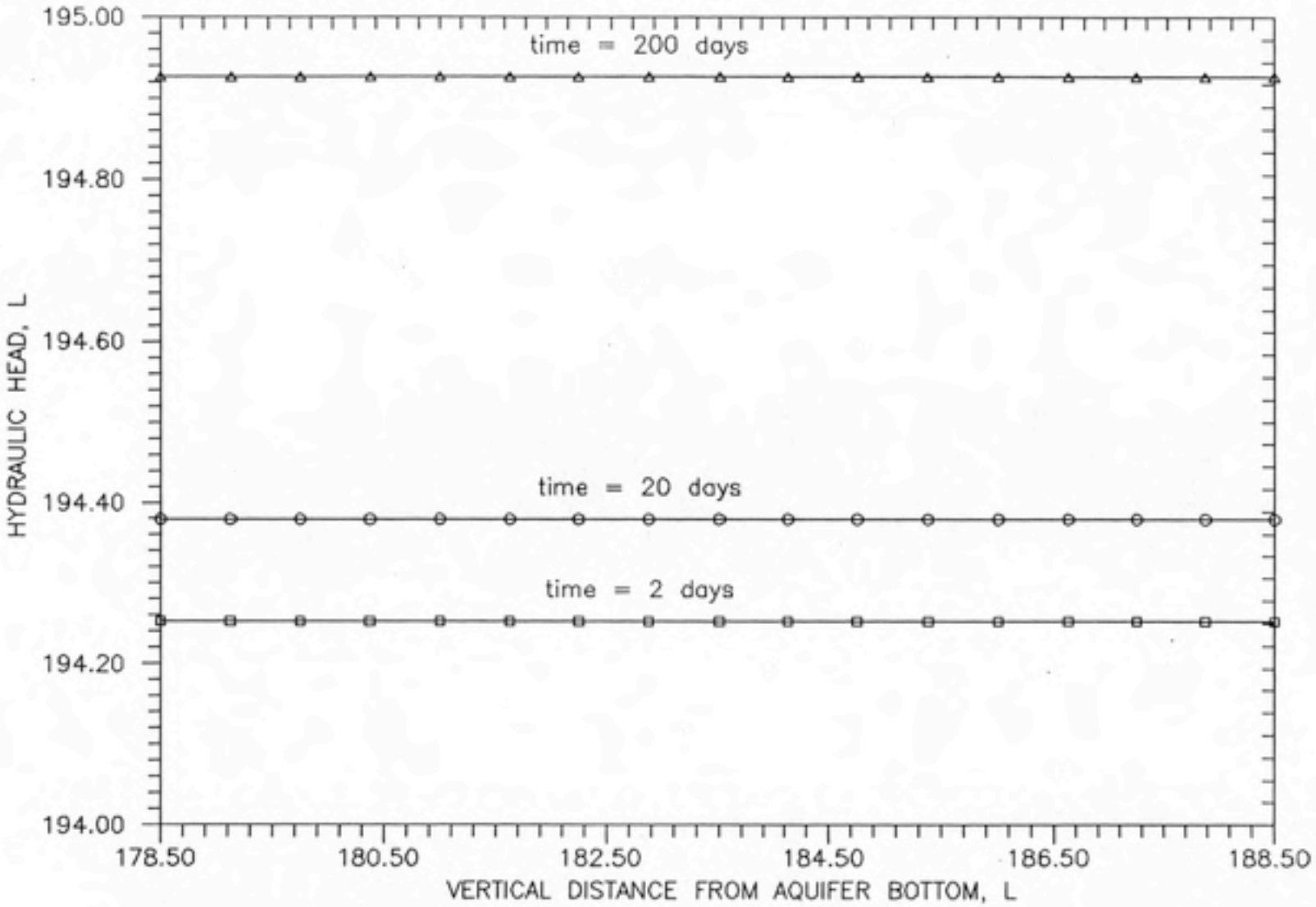


FIGURE 6.6: HYDRAULIC HEADS NEAR CENTERLINE OF MULTI-LEVEL MONITORING WELL

6.4 Comparison with McDonald-Harbaugh Model

The three-dimensional McDonald-Harbaugh model (McDonald and Harbaugh, 1984) represents the current state-of-the-art in public domain groundwater models. This model can provide a convenient benchmark for the REGFED model. The groundwater flow system chosen as a basis for comparison is a single pumped well in a confined aquifer, with constant head boundaries. The model simulations for this system can be validated by the Theis solution described in Chapter 5. The data sets submitted to the models contain an equal number of nodes and layers, except that the McDonald-Harbaugh model requires an extra row and column to simulate no-flow boundaries.

The model simulations are compared in Figure 6.7, along with the Theis simulation. The figure shows that both models accurately simulate the response of the groundwater system. The computational effort required to simulate the system is shown for both models in Table 6.1, along with the parmeters used in the sample problem. These results indicate that the REGFED model requires less CPU time than the McDonald-Harbaugh model to simulate the system, while providing better mass balance errors. The models were run on an IBM Personal Computer AT; the CPU time does not include input or ouptut of data.

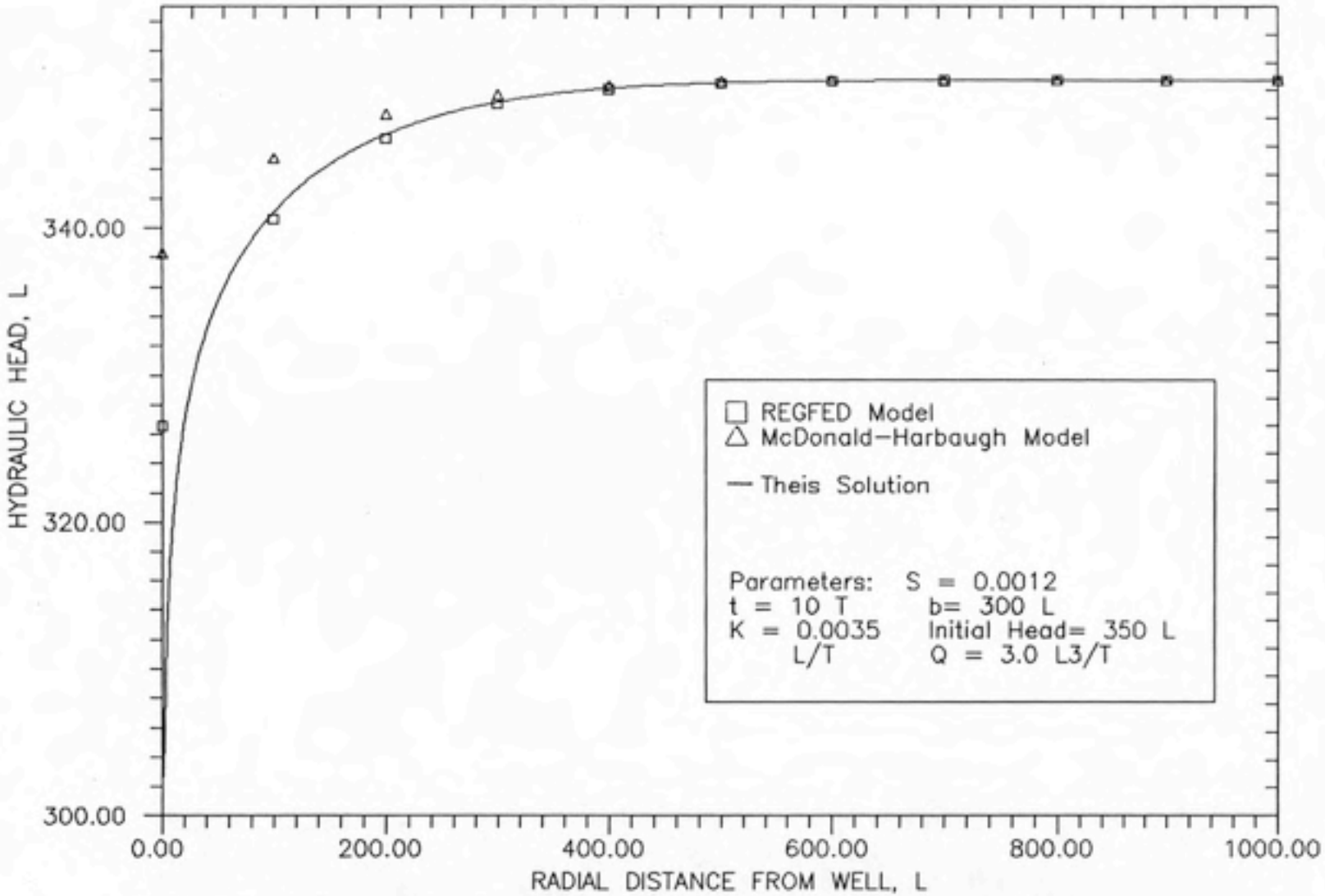


FIGURE 6.7: COMPARISON OF MODEL SIMULATIONS--MCDONALD-HARBAUGH MODEL, REGFED MODEL, AND THEIS SOLUTION (CONFINED FLOW)

TABLE 6.1

COMPARISON OF PERFORMANCE OF REGFED MODEL AND MCDONALD-HARBAUGH MODEL

MODEL	CPU TIME (min.)	WATER BALANCE ERROR
MCDONALD-HARBAUGH	36.4	7.32×10^{-4}%
REGFED	28.5	2.21×10^{-5}%

SIMULATION CONDITIONS AND PARAMETERS

-Confined Flow
- 3 equally spaced layers
- 225 nodes
- 392 elements
- K = 0.0035 L/T
- b = 300 L
- S = 0.00012
- Q = 0.5 L3/T
- initial head = 350 L

7 CONCLUSIONS AND RECOMMENDATIONS

7.1 Conclusions

From the model development, and the tests and applications performed with the REGFED model, certain conclusions can be drawn. These conclusions include

- A mixed method consisting of finite-elements and finite-differences is an efficient and accurate method for modeling groundwater flow; the ALALS algorithm is a suitable example of such a method.
- The REGFED model compares favorably with the analytical solutions used in this report for model testing
- Mass balance errors are minimal for the test cases, except where drained node transitions occur.
- For situations where the model did not validate well, finer grid spacing or timestep sizes could improve model accuracy.
- The WELFED model can efficiently simulate some example applications that are relatively difficult, compared to the validation conditions.
- The model may not be able to accurately simulate aquifer/confining-layer conditions without significant computational efforts and storage requirements.
- Steep vertical gradients relative to nodal spacing have a deleterious effect on model accuracy.

7.2 Recommendations

The following recommendations can be made for improving the performance of the REGFED model.

- The model should be modified to include the quasi three-dimensional approach for modeling aquifer/confining layer conditions.
- Iteration schemes other than Picard Iteration should be explored, to see if convergence for unconfined flow conditions can be made quicker or more stable. Other schemes could include Newton-Raphson schemes or modified conjugate gradient schemes.
- Spacing criteria for nodal and timestep spacing should be specified such that model accuracy is optimized.
- Improvements to the confined/unconfined and drained node transitions should be made so that mass balance errors are minimized.
- In addition to excluding entire layers with confined elements from the matrix reforming process, the algorithm should be modified so that confined elements within layers that also include unconfined elements, can be excluded from matrix reforming.
- The model should be modified to include other types of finite elements that can fit various boundary or other conditions more efficiently and accurately.
- In general, steep vertical gradients should be avoided by utilizing finer discretization schemes or timestep sizes.
- An automatic timestep generator that minimizes water balance errors and steep vertical gradients should be included.
- An input data preprocessor should be added to the model in order to ease the burden of inputing data for large problems.

8 NOTATION

C = solution concentration (M/L^3) .
t = time (T) .
$\overline{v}$ = groundwater velocity (L/T) .
$\mathbf{D}$ = hydrodynamic dispersion tensor (L^2/T) .
$\nabla\cdot$ = divergence operator .
∇ = gradient operator .
$\left(\frac{\partial C}{\partial t}\right)_{rxn}$ = reactive term $(M/L^3/T)$.
$\Gamma(C)$ = source or sink term $(M/L^3/T)$.
D_{ij} = i,j term of dispersion tensor (L^2/T) .
i,j = components of Cartesian coordinate system .
α_T = transverse dispersivity (L) .
α_L = longitudinal dispersivity (L) .
$\overline{v}$ = average groundwater velocity (L/T) .
D^* = effective molecular diffusion coefficient (L^2/T) .
q = specific discharge (L/T) .
K = hydraulic conductivity (L/T) .
h = hydraulic head (L) .
$\frac{\partial h}{\partial x}$ = groundwater gradient (*dimensionless*) .
v = pore velocity (L/T) .
n = porosity (*dimensionless*) .
h = hydraulic head (L) .
$\mathbf{K}$ = hydraulic conductivity tensor (L/T) .
S_s = specific storage $(1/L)$.
$\Gamma(h)$ = source or sink term $(1/T)$.
K_x, K_y, K_z = components of conductivity in the x, y, and z directions, respectively (L/T) .
T_x, T_y = components of transmissivity in the x, and y directions, respectively, (L^2/T) .
S_y = specific yield (*dimensionless*) .
$\Gamma'(h)$ = vertically averaged source or sink term (L/T) .
Δx = distance between spatial locations in x-direction (L) .
Δh = change in hydraulic head from x_i to x_{i+1} (L) .
n = porosity (*dimensionless*) .
$O(\Delta x)$ = remainder of the Taylor series terms, including those with powers of Δx and higher .
$\hat{u}$ = trial function .
$a_0, a_1,$ and a_2 = coefficients related to element position and geometry .
$N_i^e, N_j^e,$ and N_m^e = basis functions .
$x_i, x_j, x_m, y_i, y_j,$ and y_m = coordinates of triangle vertices (L) .
A_e = area of triangle (L^2) .
N_n = basis function for node n .
N = number of nodes .

ε = error resulting from substitution of approximated form of u .
$\mathcal{R}$ = problem domain .
W_i = nodal weighting functions .
n_x and n_y = components of outward normal vector .
$\mathcal{B}$ = boundary of problem domain .
e = element region .
N = number of nodes .
$[G]$ = Global coefficient matrix .
$[E]$ = Element coefficient matrix .
$\{\vec{u}\}$ = dependent variable vector .
E_{ii}, etc. = integral terms found in Equation 3.20 for nodes in a triangular element .
$G_{m,i}$ = members of the Global coefficient matrix .
$\{\vec{b}\}$ = vector of coefficients representing boundary conditions .
h = hydraulic head (L) .
$\hat{h}$ = trial function for hydraulic head (L) .
$N_n(x,y)$ = two-dimensional basis function in the x-y plane .
h_n = nodal parameter dependent on z and time (L) .
n_{xy} = number of nodes in the x-y plane of each layer .
N_i = two-dimensional basis function in the x-y plane .
$\mathcal{R}$ = x-y problem domain .
$\mathcal{B}$ = boundary of the cross-section of $\mathcal{R}$.
$\frac{\partial \hat{h}}{\partial n}$ = outward normal derivative on $\mathcal{B}$.
K_n = normal component of hydraulic conductivity on $\mathcal{B}$.
$k+1/2, k-1/2, k+1$, and $k-1$ = indices as shown in Figure 3.4 .
Δz terms are as shown in Figure 3.3 (L) .
K_{z+} = upper-weighted, harmonic-mean hydraulic conductivity (L/T) .
K_{z-} = lower-weighted, harmonic-mean hydraulic conductivity (L/T) .
l = index for present time step .
$l+1$ = index for next time step .
Δt = time increment for time step (T) .
$*$ = index referring to predicted solutions .
$[KH], \frac{[ST]}{\Delta t}, \{\vec{\Gamma}(h)\}, \{\vec{F}(h)\}, [KL]$, and $[KU]$ = matrices and vectors of integrals in Equation 3.11 .
n_i, n_j, and n_m = nodal indices on triangular element .
x_i, x_j, x_m, y_i, y_j, and y_m = coordinates of triangle vertices (L) .
A_e = area of triangle (L^2) .
a_{ij} = i,j element of $[A]$ matrix .
n = outward normal vector .
$\mathcal{B}$ = problem boundary .
g = arbitrary boundary function .
Γ_r = recharge rate (L/T) .
$u_{\mathcal{B}}$ = problem boundary .
f = arbitrary boundary function .
Γ_p = withdrawal or injection quantity (L^3/T) .
$[A]$ = sum of left-hand-side matrices .

$\{\vec{x}\}$ = the solution, or hydraulic head vector .
$\{\vec{b}\}$ = sum of right-hand-side vectors .
T = transmissivity = Kb (L^2/T) .
where b = saturated thickness in aquifer layer (L) .
S = specific yield (unconfined aquifer) or storativity (confined aquifer) (*dimensionless*) .
$(x_1, x_2, \ldots, x_{N_l})$ = unknowns .
r = iteration counter .
$[A]^{-1}_{IJ}$ = elements of the inverse of matrix $[A]$.
ε_b = prescribed residual tolerance .
max_J = maximum over all nodes .
Q = rate of change in storage of water in an element (L^3/T) .
top = elevation of the top of an element layer (L) .
S^l = storage factor (specific yield or storativity) in effect at time l (*dimensionless*) .
S^{l+1} = storage factor (specific yield or storativity) in effect at time $l+1$ (*dimensionless*) .
t_T = total time elapsed over simulation (T) .
ε_{wb} = water balance error (*dimensionless*) .
h_i = head at start of simulation (L) .
h_f = head at end of simulation (L) .
s = drawdown = initial head - new head (L) .
r = radial distance from well (L) .
$W(u_c)$ = well function for nonleaky aquifer (*dimensionless*) .
u_c = argument of the well function (*dimensionless*) .
T = transmissivity (L^2/T) .
S = storativity (*dimensionless*) .
Q = pumping rate (L^3/T) .
t = time (T) .
$W\left(u_c, \frac{n_s \pi r}{b}\right)$ = well function for leaky aquifer (*dimensionless*) .
$W(u_c)$ = well function for nonleaky aquifer (*dimensionless*) .
z = vertical distance from top of aquifer (L) .
n_s = summation index .
u_u = argument (modified for unconfined flow) of the well function (*dimensionless*) .
S_y = specific yield (*dimensionless*) .
s_c = corrected drawdown (L) .
s_o = drawdown calculated from Equation 4.3 (L) .
b = aquifer thickness (L) .
h_o = head at up-gradient boundary (L) .
h_1 = head at down-gradient boundary (L) .
x = distance from up-gradient boundary (L) .
x_1 = length of aquifer (L) .
Γ_r = recharge rate (L/T) .

9 REFERENCES

Abriola, L.M. and G.F. Pinder (1982), Calculation of Velocity in Three Space Dimensions from Hydraulic Head Measurements, Ground Water, 20(2), pp. 205-213.

Anderson, M.P. (1979), Using Models to Simulate the Movement of Contaminants through Groundwater Flow Systems, CRC Critical Reviews in Environmental Control, 9(2), pp. 97-156.

Anderson, M.A. (1984), Movement of Contaminants in Groundwater: Groundwater Transport–Advection and Dispersion, in Studies in Geophysics: Groundwater Contamination, National Academy Press, Washington, D.C., pp. 46-64.

Arnett, R.C., R.A. Deju, R.E. Gephart, and R.B. Lantz (1977), The Importance of Three Dimensional Groundwater Contaminant Modeling at the Hanford Waste Nuclear Facility, Transactions of the American Geophysical Union EOS, 58(Abstr.), p. 1138.

Babu, D.K., G.F. Pinder, and D.K. Sunada (1982), A Three Dimensional Hybrid Finite Element-Finite Difference Scheme for Groundwater Simulation, in 10th IMACS World Congress, National Association for Mathematics and Computers, pp. 292-294.

Batu, V. (1984), A Finite Element Dual Mesh Method to Calculate Nodal Darcy Velocities in Nonhomogeneous and Anisotropic Aquifers, Water Resources Research, 20(11), pp. 1705-1717.

Bear, J. (1972), Dynamics of Fluids in Porous Media, Elsevier Publishing Co., New York, 367 pp.

Bear, J. (1979), Hydraulics of Groundwater, McGraw-Hill, Inc., New York.

Bredehoeft, J.D. and G.F. Pinder (1970), Digital Analysis of Areal Flow in Multiaquifer Groundwater Systems: A Quasi Three- Dimensional Model, Water Resources Research, 6(3), pp. 883-888.

Cooley, R.L. (1971), A Finite-Difference Method for Unsteady Flow in Variably Saturated Porous Media: Application to a Single Pumping Well, Water Resources Research, 7(10), pp. 1607-1625.

de Marsily, G. (1986), Quantitative Hydrogeology: Groundwater Hydrology for Engineers, Academic Press, Inc., Orlando, FL, 440 pp.

Domenico, P.A. and G.A. Robbins (1984), A Dispersion Scale Effect in Model Calibrations and Field Tracer Experiments, Journal of Hydrology, 70, pp. 123-132.

Driscoll, F.G. (1986), Groundwater and Wells, Second Edition, Johnson Division, St. Paul, MN, 1108 pp.

Freeze, R.A. (1971), Three-Dimensional, Transient, Saturated-Unsaturated Flow in a Groundwater Basin, Water Resources Research, 7(2), pp. 347-366.

Freeze, R.A. (1975), A Stochastic-Conceptual Model of One Dimensional Groundwater Flow in Nonuniform Homogeneous Media, Water Resources Research, 11(2), pp. 725-732.

Freeze, R.A. and J.A. Cherry (1979), Groundwater Contamination, in Groundwater, Prentice-Hall, Inc., Englewood Cliffs, NJ, pp. 444- 447.

Fried, J.J. (1975), The Theory of Dispersion in Porous Media, in Groundwater Pollution, Elsevier Scientific Publishing Co., Amsterdam, pp. 5-46.

Fried, J.J. and M.A. Combarnous (1971), Dispersion in Porous Media, Advances in Hydroscience, edited by V.T. Chow, 7, pp. 169-282.

Frind, E.O. and G.B. Matanga (1985), The Dual Formulation of Flow for Contaminant Transport Modeling 1. Review of Theory and Accuracy Aspects, Water Resources Research, 21(2), pp. 159-169.

Frind, E.O. and M.J. Verge (1978), Three-Dimensional Modeling of Groundwater Flow Systems, Water Resources Research, 14(5), pp. 844- 856.

Frind, E.O., G.B. Matanga, and J.A. Cherry (1985), The Dual Formulation of Flow for Contaminant Transport Modeling 2. The Borden Aquifer, Water Resources Research, 21(2), pp. 170-182.

Gambolati, G. and F. Sartoretto (1986), A Conjugate Gradient Finite Element Model of Flow for Large Multiaquifer Systems, Water Resources Research, 22(7), pp. 1003-1015.

Gambolati, G., G. Pini, and T. Tucciarelli (1986), A 3-D Finite Element Conjugate Gradient Model of Subsurface Flow with Automatic Mesh Generation, Adavnces in Water Resources, 9, pp. 34-41.

Gelhar, L.W. (1976), Stochastic Analysis of Flow in Aquifers, in Advances in Groundwater Hydrology, Saleem, Z.A., ed., American Water Resources Association, Minnestoa, MN, pp. 57-78.

Gelhar, L.W. and C.L. Axness (1983), Three-Dimensional Stochastic Analysis of Macrodispersion in Aquifers, Water Resources Research, 19(1), pp. 161-180.

Gelhar, L.W., A. Mantoglou, C. Welty, and K.R. Rehfeldt (1985), A Review of Field-Sacle Physical Solute Transport Processes in Saturated and Unsaturated Porous Media, Electric Power Research Institute, Palo Alto, CA, Report EA-4190.

Gillham, R.W., E.A. Sudicky, J.A. Cherry, and E.O. Frind (1984), An Advection-Diffusion Concept for Solute Transport in Heterogeneous Unconsolidated Geological Deposits, Water Resources Research, 20(3), pp. 369-378.

Gupta, S.K. and K.K. Tanji (1976), A Three-Dimensional Galerkin Finite Element Solution of Flow through Multiaquifers in Sutter Basin, California, Water Resources Research, 12(2), pp. 155-162.

Gupta, S.K., C.R. Cole, and G.F. Pinder (1984), A Finite-Element Three-Dimensional Groundwater (FE3DGW) Model for a Multiaquifer System, Water Resources Research, 20(5), pp. 553-563.

Guvanasen, V. and R.E. Volker (1981), Simulating Mass Transport in Unconfined Aquifers, Journal of the Hydraulics Division, HY4, pp. 461-477.

Hantush, M.S. (1961), Drawdown Around a Partially Penetrating Well, Journal of the Hydraulics Division, Proceedings ASCE, HY4, pp. 83-98.

Huang, Y.H. and S.L. Sonnenfeld (1974), Analysis of Unsteady Flow Toward an Artesian Well by Three-Dimensional Finite Elements, Water Resources Research, 10(3), pp. 591-596.

Huyakorn, P.S. and G.F. Pinder (1983), Computational Methods in Subsurface Flow, Academic Press, New York, 473 pp.

Huyakorn, P.S., B.G. Jones, and P.F. Andersen (1986), Finite Element Algorithms for Simulating Three-Dimensional Groundwater Flow and Solute Transport in Multilayer Systems, Water Resources Research, 22(3), pp. 361-374.

Huyakorn, P.S., P.F. Andersen, F.J. Molz, O. Guven, and J.G. Melv ille (1986), Simulation of a Two-Well Tracer Tests in Stratified Aquifers at the Chalk River and the Mobile Sites, Water Resources Research, 22(7), pp. 1016-1030.

Huyakorn, P.S., P.F. Andersen, O. Guven, and F.J. Molz (1986), A Curvilinear Finite Element Model for Simulating Two-Well Tracer Tests and Transport in Stratified Aquifers, Water Resources Research, 22(5), pp. 663-678.

Intera Environmental Consultants, Inc. (1983), SWENT: A Three-Dimensional Finite Difference Code for the Simulation of Fluid, Energy, and Solute Radionuclide Transport, Report ONWI-457, Office of Nuclear Waste Isolation, Batelle Research, Col

International Cround Water Modeling Center (1987), Personal Communication.

Jacob, C.E. (1950), Flow of Groundwater, in Engineering Hydraulics, Rouse, H., ed., John Wiley and Sons, New York, pp. 321-386.

Krosyzinski, U.I. and G. Dagan (1975), Well Pumping in Unconfined Aquifers: The Influ-

ence of the Unsaturated Zone, Water Resources Research, 11(4), pp. 479-490.

Lennon, G.P., P.L.-F. Liu, and J.A. Liggett (1980), Boundary Integral Solutions to Three-Dimensional Unconfined Darcy's Flow, Water Resources Research, 16(4), 651-658.

Liggett, J.A. (1977), Location of Free Surface in Porous Media, Journal of the Hydraulics Division, ASCE, 103(HY4), pp. 353-365.

McDonald, M.G. and A.W. Harbaugh (1984), A Modular Three-Dimensional Finite Difference Groundwater-Flow Model, U.S. Geological Survey, Scientific Publications Co., Washington, D.C., 527 pp.

Mercer, J.W. and C.R. Faust (1980), Ground-Water Modeling: Mathematical Models, Ground Water, 18(3), pp. 212-227.

Molz, F.J., O. Guven, and J.G. Melville (1983), An Examination of Scale-Dependent Dispersion Coefficients, Ground Water, 21(6), pp. 715-725.

Molz, F.J., O. Guven, J.G. Melville, R.D. Crocker, and K.T. Matteson (1986), Performance, Analysis, and Simulation of a Two-Well Tracer Test at the Mobile Site, Water Resources Research, 22(7), pp. 1031-1037.

Narasimhan, T.N. and P.A. Witherspoon (1976), An Integrated Finite Difference Method for Analyzing Fluid Flow in Porous Media, Water Resources Research, 12(1), pp. 57-64.

Narasimhan, T.N., S.P. Neuman, and P.A. Witherspoon (1978), Finite Element Method for Subsurface Hydrology Using a Mixed Explicit- Implicit Scheme, Water Resources Research, 14(5), pp. 863-877.

Neuman, S.P. (1972), Theory of Flow in Unconfined Aquifers Considering Delayed Response of the Water Table, Water Resources Research, 8(4), pp. 1031-1045.

Neuman, S.P. and P.A. Witherspoon (1971), Analysis of Unsteady Flow with a Free Surface Using the Finite Element Method, Water Resources Research, 7(3), pp. 611-623.

Neuman, S.P., Preller, C., and T.N. Narasimhan (1982), Adaptive Explicit-Implicit Quasi Three-Dimensional Finite Element Model of Flow and Subsidence in Multiaquifer Systems, Water Resources research, 18(5), pp. 1551-1561.

Peaudecerf, P. and J.-P. Sauty (1978), Application of a Mathematical Model to the Characterization of Dispersion Effects on Groundwater, Progress in Water Technology, 10(5/6), pp. 443-454.

Pickens, J.F. and G.E. Grisak (1981), Modeling of Scale-Dependent Dispersion in Hydrogeologic Systems, Water Resources Research, 17(6), pp. 1701-1711.

Pickens, J.F., J.A. Cherry, R.M. Coupland, G.E. Grisak, W.F. Merr it, and B.A. Risto

(1981), A Multilevel Device for Ground Water Sampling, Ground Water Monitoring Review, Spring, pp. 48-51.

Pinder, G.F. (1973), A Galerkin-Finite Element Simulation of Groundwater Contamination on Long Island, New York, Water Resources Research, 9(6), pp. 1657-1669.

Pinder, G.F., M. Celia, and W.G. Gray (1981), Velocity Calculation from Randomly Located Hydraulic Heads, Ground Water, 19(3), pp. 262-264.

Prickett, T.A., and C.G. Lonnquist (1971), Selected Digital Computer Techniques for Groundwater Resource Evaluation, Illinois State Water Survey Bulletin 55, 62 pp.

Reeves, M. and O.F. Duguid (1975), Water Movement through Saturated-Unsaturated Porous Media: A Finite Element Galerkin Model, Report ORNL-4927, Oak Ridge National Laboratory, Oak Ridge, TN, 1975.

Reeves, M. and R.M. Cramwell (1981), User's Manual for the Sandia Waste-Isolation Flow and Transport Model (SWIFT), Report NUREG/CR- 2324, U.S. Nuclear regulatory Commission, Washington ,D.C.

Scheidegger, A.E. (1961), General Theory of Dispersion in Porous Media, Journal of Geophysical Research, 66(10), pp. 3273-3278.

Schwartz, F.W. (1977), Macroscopic Dispersion in Porous Media: The Controlling Factors, Water Resources Research, 13(4), pp. 743- 752.

Segol, G., G.F. Pinder, and W.G. Gray (1975), A Galerkin-Finite Element Technique for Calculating the Transient Position of the Saltwater Front, Water Resources Research, 11(2), pp. 343-347.

Skibitzke, H.E. and G.M. Robinson (1963), Dispersion in Ground Water Flowing Through Heterogeneous Materials, Geological Survey Professional Paper 386-B, U.S. Government Priiting Office, Washington D.C., 3 pp.

Smith, L. and F.W. Schwartz (1981), Mass Transport 3. Role of Hydraulic Conductivity Data in Prediction, Water Resources Research, 17(5), pp. 1463-1479.

Strang, G. (1986), Introduction to Applied Mathematics, Wellesley-Cambridge Press, Wellesley, MA, 758 pp.

Streltsova, T.D. (1972), Unsteady Radial Flow in an Unconfined Aquifer, Water Resources Research, 8(4), pp. 1059-1066.

Sudicky, E.A. (1986), A Natural Gradient Experiment on Solute Transport in a Sand Aquifer, 5, Spatial Variability of Hydraulic Conductivity and Its Role in the Dispersion Process, Water Resources Research, 22(13), pp. 2069-2082.

Sudicky, E.A. and J.A. Cherry (1979), Field Observations of Tracer Dispersion Under Natural Flow Conditions in an Unconfined Sandy Aquifer, Water Pollution Research in Canada, 14, pp. 1-17.

Taylor, G.S. and J.N. Luthin (1969), Computer Methods for Transient Analysis of Water-Table Aquifers, Water Resources Research, 5(1), pp. 144-152.

Theis, C.V. (1935), The Relation Between the Lowering of the Piezometric Surface and the Rate and Duration of Discharge of a Well Using Ground-Water Storage, Reports and Papers, Hydrology, pp. 519-524.

Tompson, A.F.B. and W.G. Gray (1986), A Second-Order Approach for the Modeling of Dispersive Transport in Porous Media 1. Theoretical Development, Water Resources Research, 22(5), pp. 591-599.

Trescott, P.C. and S.P. Larson (1977), Comparison of Iterative Methods of Solving Two-Dimensional Groundwater Flow Equations, Water Resources Research, 13(1), pp. 125-136.

Trescott, P.C., G.F. Pinder, and S.P. Larson (1976), Finite Difference Model for Aquifer Simulation in Two-Dimensions with Results of Numerical Experiments, Techniques of Water Resources Investigations of U.S. Geol. Surv., Book 7, Chapt. C1, 116 pp.

Trescott, P.C., G.F. Pinder, and S.P. Larson (1976), Documentation of Finite Difference Model for Simulation of Three Dimensional Groundwater Flow, U.S. Geological Survey Open File Report, 75-438, 32 pp.

van Genuchten, M.Th., G.F. Pinder, and E.O. Frind (1977), Simulation of Two-Dimensional Contaminant Transport With Isoparametric Hermitian Finite Elements, Water Resources Research, 13(2), pp. 451-458.

Wang, H.F. and M.P. Anderson (1982), Introduction to Groundwater Modeling: Finite Difference and Finite Element Methods, W.H. Freeman and Co., San Francisco, 237 pp.

Winter, C.L., C.M. Newman, and S.P. Neumann (1984), A Perturbation Expansion for Diffusion in a Random Velocity Field, SIAM Journal of Applied Mathematics, 44(2), pp. 411-424.

Winter, T.C. (1978), Numerical Simulation of Steady State Three-Dimensional Groundwater Flow Near Lakes, Water Resources Research, 14(2), pp. 245-254.

Yeh, G.-T. (1981), On the Computation of Darcian Velocity and Mass Balance in the Finite Element Modeling of Groundwater Flow, Water Resources Research, 17(5), pp. 1529-1534.

Dimensional Groundwater Flow, Journal of Hydrology, 85, pp. 349-365.

APPENDIX 1:

REVIEW OF FINITE DIFFERENCE AND FINITE ELEMENT METHODS

1 Finite Difference Method

The finite difference method is a numerical method for solving differential equations. Application of the method results in the discretization of a problem domain into a finite number of predetermined points. Values of the dependent variable can be approximated at these points.

The accuracy of finite difference approximations is a function of differential equation type, discretization, and method used to approximate the relevant derivatives. Boundary conditions and the temporal nature of the problem (i.e. steady-state or transient conditions) also affect the accuracy. As a simple example of a finite difference approximation, one can examine the first-order ordinary derivative du/dx.

This derivative can be approximated by expanding the Taylor series definition of the derivative of the function u(x) with respect to linear distance, x. Figure A1.1 illustrates the discretization of u(x). For an n+1 term Taylor series expansion about the point x = a, the series can be written as

$$\left.\frac{du}{dx}\right|_{x=a} = \frac{u(b)-u(a)}{\Delta x} - \frac{\Delta x}{2!}\frac{\mathrm{d}^2u}{\mathrm{d}x^2} - \cdots - \frac{(\Delta x)^{n-1}}{n!}\frac{\mathrm{d}^nu}{\mathrm{d}x^n} \qquad (A1.1)$$

where

$\Delta x = b - a$, distance between spatial location b and a (L)

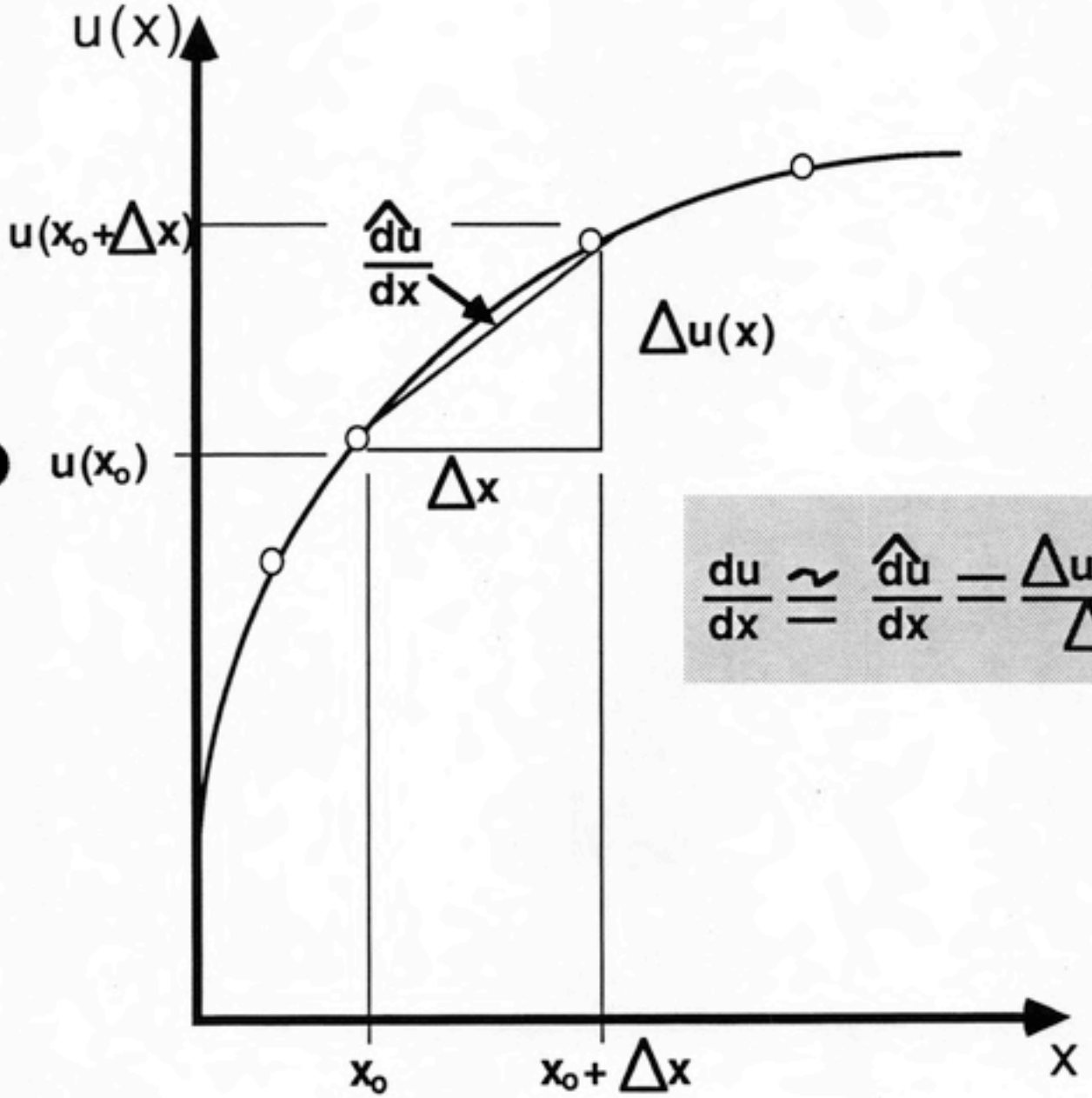
Equation III-2 can be written as sum of the following terms

$$\left.\frac{du}{dx}\right|_{x=a} = \frac{u(b)-u(a)}{\Delta x} + O(\Delta x) \qquad (A1.2)$$

where

FIGURE A1.1

ILLUSTRATION OF FINITE DIFFERENCE APPROXIMATION



$$\frac{du}{dx} \simeq \frac{\widehat{du}}{dx} = \frac{\Delta u(x)}{\Delta x}$$

$O(\Delta x)$ = remainder of the Taylor series terms, including those with powers of Δx and higher

By truncating the series to the first term on the right hand side, a first- order approximation is achieved. The forward-difference approximation takes the form of

$$\left.\frac{du}{dx}\right|_x \approx \frac{u(x+\Delta x)-u(x)}{\Delta x} \qquad (A1.3)$$

The backward-difference approximation is written as

$$\left.\frac{du}{dx}\right|_x \approx \frac{u(x)-u(x-\Delta x)}{\Delta x} \qquad (A1.4)$$

By subtracting the full Taylor series expansion for Equation A1.3 from the full expansion series for Equation A1.4 and solving for the first-order derivative, the central-difference approximation results.

$$\left.\frac{du}{dx}\right|_x = \frac{u(x+\Delta x)-u(x-\Delta x)}{2\Delta x} + O[(\Delta x)^2] \qquad (A1.5)$$

The truncated form of this equation is more accurate than Equations A1.3 and A1.4, because of the higher order terms that are eliminated in the subtraction. Only second-order and higher terms remain, thus Equation A1.5 is a second-order approximation.

The second order derivative can be approximated by summing the full series expansions for Equations A1.3 and A1.4, and rearranging, resulting in

$$\left.\frac{d^2u}{dx^2}\right|_x \approx \frac{u(x+\Delta x)-2u(x)+u(x-\Delta x)}{(\Delta x)^2} \qquad (A1.6)$$

Higher order derivatives can be obtained from similar Taylor series applications.

2 Finite Element Method

The finite-element method is also a numerical method for approximating differential equations. The problem domain is divided into a finite number of small, interconnected subregions (finite-elements). The dependent variable can be approximated at the nodes that interconnect the elements or anywhere over the domain of an element.

Galerkin's finite-element method is frequently the method of choice when modeling groundwater flow (and contaminant transport). Galerkin's method is a weighted-residual method, which leads to similar equations as variational principles (Wang and Anderson, 1982). Variational principles imply that a physical quantity, such as energy potential, is minimized over the problem domain. If the potential, which is analogous to hydraulic head, is expressed in terms of its nodal values, algebraic equations result.

The weighted-residual is a measure of the degree to which the nodal values of hydraulic heads do not satisfy the governing equation. If a particular weighted residual is forced to vanish, the heads at the nodes can be obtained from a system of algebraic equations.

In the Galerkin finite-element method, the type of element discretization determines the trial solutions that are employed. These trial solutions can be polynomials that are piecewise continuous over the individual elements. Nodes are located along the boundaries of each subdomain or in the interior of the subdomain. The basis function is obtained from the trial function. A basis function is associated with each specific node. For example, a linear triangular element (see Figure A1.2) has a trial function defined by a first-order interpolating polynomial of the form

$$\hat{u}(x, y) = a_0 + a_1 x + a_2 y \qquad (A1.7)$$

where

$\hat{u}$ = trial function

$a_0, a_1,$ and a_2 = coefficients related to element position and geometry

and the basis functions N , associated with this trial function are

$$\hat{u}(x,y) = N^e_{n_i}(x,y)u_{n_i} + N^e_{n_j}(x,y)u_{n_j} + N^e_{n_m}(x,y)u_{n_m} \qquad (A1.8)$$

$$N^e_{n_i} = \frac{1}{2A_e}\left[(x_{n_j}y_{n_m} - x_{n_m}y_{n_j}) + (y_{n_j} - y_{n_m})x + (x_{n_m} - x_{n_j})y\right]$$

$$N^e_{n_j} = \frac{1}{2A_e}\left[(x_{n_m}y_{n_i} - x_{n_i}y_{n_m}) + (y_{n_m} - y_{n_i})x + (x_{n_i} - x_{n_m})y\right]$$

$$N^e_{n_m} = \frac{1}{2A_e}\left[(x_{n_i}y_{n_j} - x_{n_j}y_{n_i}) + (y_{n_i} - y_{n_j})x + (x_{n_j} - x_{n_i})y\right]$$

where

$N^e_{n_i}, N^e_{n_j},$ and $N^e_{n_m}$ = basis functions

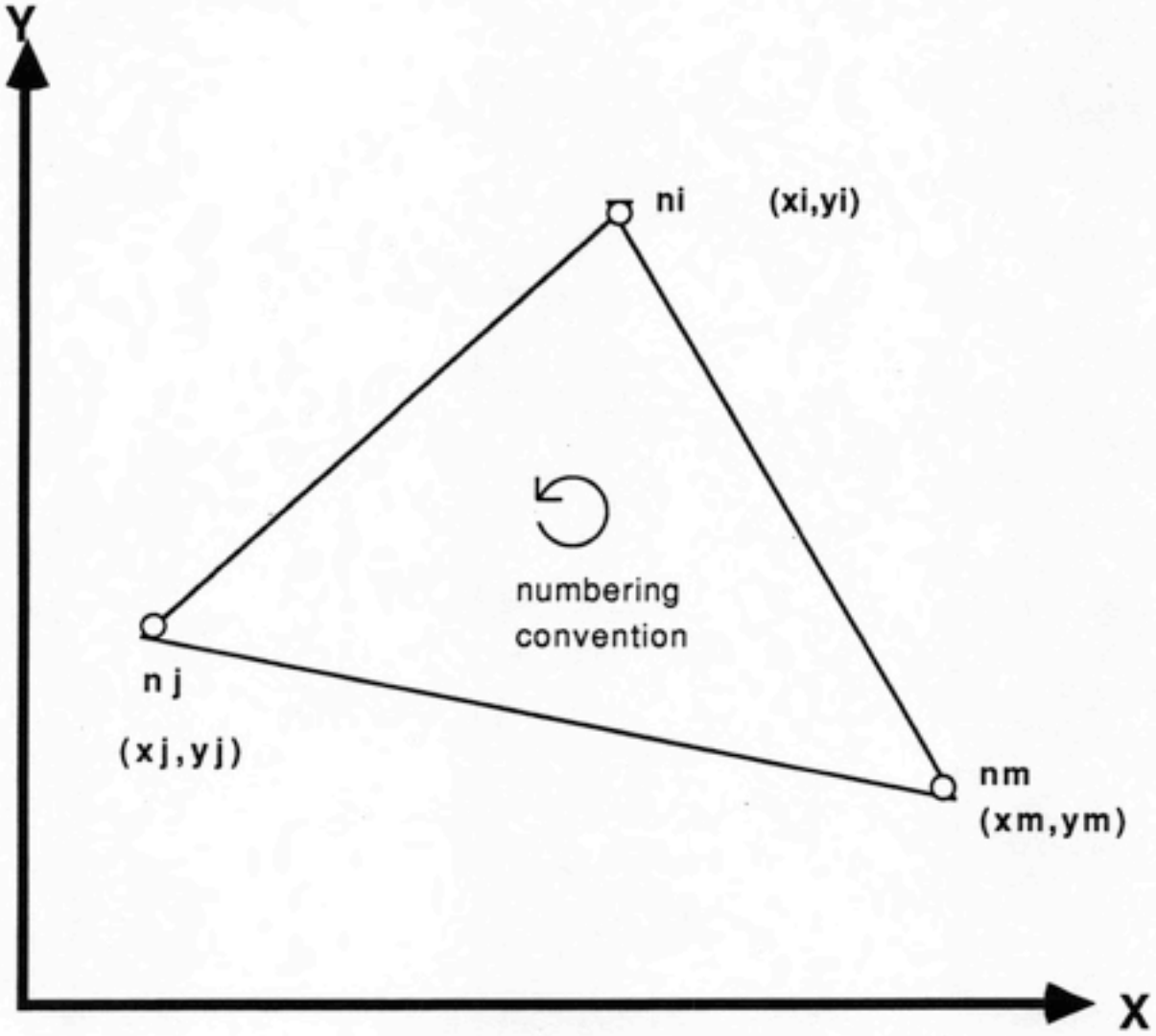
$x_{n_i}, x_{n_j}, x_{n_m}, y_{n_i}, y_{n_j},$ and y_{n_m} = coordinates of triangle vertices (L)

A_e = area of triangle (L^2)

Basis functions such as these are substituted into Galerkin finite-element solutions.

The Galerkin finite-element method can be applied to the Laplace equation as an example. The Laplace equation is usually expressed as

FIGURE A1.2
LINEAR TRIANGULAR ELEMENT



$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \qquad (A1.9)$$

Equation A1.9 can be restated in the form of differential operator as

$$L(u) = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \qquad (A1.10)$$

The dependent variable, u, may be approximated over the domain by introducing a set of nodal basis functions

$$\hat{u}(x, y) = \sum_{n=1}^{N} N_n u_n \qquad (A1.11)$$

where

N_n = basis function for node n

N = number of nodes

Substituting the approximate value of u into the differential operator gives

$$L[\hat{u}(x, y)] = \varepsilon \qquad (A1.12)$$

where

ε = error resulting from substitution of approximated form of u

The objective of the weighted residual priciple implies that

$$\iint_{\mathcal{R}} W_i(x, y)\varepsilon \;\; d\mathcal{R} = 0 \qquad (A1.13)$$

where

$\mathcal{R}$ = problem domain

W_i = nodal weighting functions

The Galerkin method imposes the condition that the weighting functions are equivalent to the nodal basis functions, resulting in

$$\iint_{\mathcal{R}} N_i(x,y)\varepsilon \;\; d\mathcal{R} = 0 \tag{A1.14}$$

or, restating (A1.14) in terms of the differential operator

$$\iint_{\mathcal{R}} N_i(x,y)L(\hat{u}) \;\; d\mathcal{R} = 0 \tag{A1.15}$$

and replacing the differential operator with the governing equation gives

$$\iint_{\mathcal{R}} N_i(x,y)(\frac{\partial^2 \hat{u}}{\partial x^2} + \frac{\partial^2 \hat{u}}{\partial y^2}) \;\; dx \;\; dy = 0 \tag{A1.16}$$

Equation A1.16 represents the Galerkin approximation to the Laplace equation. The second order derivatives in Equation A1.16 can be reduced by applying Green's formula, which in this case is essentially integration by parts. The resulting equation is

$$\begin{aligned} &\iint_{\mathcal{R}} N_i \left(\frac{\partial^2 \hat{u}}{\partial x^2} + \frac{\partial^2 \hat{u}}{\partial y^2}\right) \; dx \; dy \\ &= \oint_{B} N_i \left(\frac{\partial \hat{u}}{\partial x} n_x + \frac{\partial \hat{u}}{\partial y} n_y\right) \; ds - \iint_{\mathcal{R}} \left(\frac{\partial \hat{u}}{\partial x}\frac{\partial N_i}{\partial x} + \frac{\partial \hat{u}}{\partial y}\frac{\partial N_i}{\partial y}\right) \; dx \; dy \end{aligned} \tag{A1.17}$$

where

n_x and n_y = components of outward normal vector

β = boundary of problem domain

The basis functions, Ni, are defined to vanish for nodes that are outside of the given element. Applying this definition to (A1.17) results in

$$\sum_{l=1}^{elements} \left[\iint_{l} \left(\frac{\partial \hat{u}_l}{\partial x} \frac{\partial N_i}{\partial x} + \frac{\partial \hat{u}_l}{\partial y} \frac{\partial N_i}{\partial y} \right) dx\, dy \right] = \oint_{\beta} N_i \left(\frac{\partial \hat{u}_l}{\partial x} n_x + \frac{\partial \hat{u}_l}{\partial y} n_y \right) ds \quad (A1.18)$$

where

l = element

The derivative terms in (A1.18) can be simplified such that

$$\begin{aligned} &\sum_{l=1}^{elements} \left[\iint_{l} \left(\frac{\partial \hat{u}_l}{\partial x} \frac{\partial N_i}{\partial x} + \frac{\partial \hat{u}_l}{\partial y} \frac{\partial N_i}{\partial y} \right) dx\, dy \right] \\ &= \sum_{l=1}^{elements} \left[\iint_{l} \left[\sum_{n=1}^{N} \left(\sum_{i=1}^{N} \frac{\partial N_i}{\partial x} u_n \frac{\partial N_n}{\partial x} \right) + \sum_{n=1}^{N} \left(\sum_{i=1}^{N} \frac{\partial N_i}{\partial y} u_n \frac{\partial N_n}{\partial y} \right) \right] dx\, dy \right] \end{aligned} \quad (A1.19)$$

Next, the dependent variable is separated from the integral terms. The remaining integral terms can be condensed to matrix notation to yield

$$\begin{aligned} \sum_{l=1}^{elements} \left[\iint_{l} \left(\frac{\partial \hat{u}_l}{\partial x} \frac{\partial N_i}{\partial x} + \frac{\partial \hat{u}_l}{\partial y} \frac{\partial N_i}{\partial y} \right) dx\, dy \right] &= \left(\sum_{e} [E] \right) \{\vec{u}\} \\ &= [G] \{\vec{u}\} \end{aligned} \quad (A1.20)$$

where

$[G]$ = Global coefficient matrix

$[E]$ = Element coefficient matrix

$\{\vec{u}\}$ = dependent variable vector

The element coefficient matrix, E, can be thought of as a three-by-three matrix

$$[E] = \begin{pmatrix} E_{ii} & E_{ij} & E_{im} \\ E_{ji} & E_{jj} & E_{jm} \\ E_{mi} & E_{mj} & E_{mm} \end{pmatrix} \qquad (A1.21)$$

where

E_{ii}, etc. = integral terms found in Equation A1.21 for nodes in a triangular element

The members in the global coefficient matrix can be represented by the following summation.

$$G_{m,i} = \sum_{e} E_{m,i} \qquad (A1.22)$$

where

$G_{m,i}$ = members of the Global coefficient matrix

m and i = row and column indices, respectively

The boundary term from Equation A1.18 can be condensed into a vector as follows

$$\oint_{B} N_i \left(\frac{\partial \hat{u}}{\partial x} n_x + \frac{\partial \hat{u}}{\partial y} n_y \right) \, ds = \{\vec{b}\} \qquad (A1.23)$$

where

$\{\vec{b}\}$ = vector of coefficients representing boundary conditions

The final, condensed matrix form of the Galerkin solution to the Laplace equation is then

$$[G]\{\vec{u}\} = \{\vec{b}\} \tag{A1.24}$$

APPENDIX 2:

FORTRAN CODE FOR REGFED

```
$DEBUG
$LARGE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                        C
C     PROGRAM NAME : UNCONF                                              C
C                                                                        C
C     PROGRAM PURPOSE : THREE-DIMENSIONAL, TRANSIENT, UNCONFINED         C
C                       AND CONFINED GROUNDWATER FLOW SIMULATION         C
C                                                                        C
C     WRITTEN BY : ALEX MAYER                                            C
C                  WATER RESOURCES ENGINEERING PROGRAM                   C
C                  ENVIRONMENTAL SCIENCES AND ENGINEERING DEPARTMENT     C
C                  SCHOOL OF PUBLIC HEALTH                               C
C                  UNIVERSITY OF NORTH CAROLINA                          C
C                                                                        C
C     LATEST VERSION : 10-04-87                                          C
C                                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                        C
C     INPUT AND OUTPUT UNIT ASSIGNMENTS                                  C
C                                                                        C
C     UNIT   TYPE     DESCRIPTION                                        C
C     ----   ------   ---------------------------------------------      C
C     1      INPUT    SIMULATION CONTROL PARAMETERS: NUMBER OF           C
C                     NODES, LAYERS, ELEMENTS; BANDWIDTH;                C
C                     TIMESTEP SIZE AND NUMBER; MAXIMUM NUMBER           C
C                     OF ITERATIONS AND ALLOWABLE ERROR; SWITCH FOR      C
C                     READ IN HEADS AND HEAD ACCLERATION.                C
C                                                                        C
C     2      INPUT    INITIAL HEADS (BY LAYERS), STRESSES (BY NODES),    C
C                     CONSTANT HEAD BOUNDARIES (BY NODES, LAYERS)        C
C$DEBUG
$LARGE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                        C
C     PROGRAM NAME : UNCONF                                              C
C                                                                        C
C     PROGRAM PURPOSE : THREE-DIMENSIONAL, TRANSIENT, UNCONFINED         C
C                       AND CONFINED GROUNDWATER FLOW SIMULATION         C
C                                                                        C
C     WRITTEN BY : ALEX MAYER                                            C
C                  WATER RESOURCES ENGINEERING PROGRAM                   C
C                  ENVIRONMENTAL SCIENCES AND ENGINEERING DEPARTMENT     C
C                  SCHOOL OF PUBLIC HEALTH                               C
C                  UNIVERSITY OF NORTH CAROLINA                          C
C                                                                        C
C     LATEST VERSION : 10-04-87                                          C
C                                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                        C
C     INPUT AND OUTPUT UNIT ASSIGNMENTS                                  C
C                                                                        C
C     UNIT   TYPE     DESCRIPTION                                        C
C     ----   ------   ---------------------------------------------      C
C     1      INPUT    SIMULATION CONTROL PARAMETERS: NUMBER OF           C
C                     NODES, LAYERS, ELEMENTS; BANDWIDTH;                C
C                     TIMESTEP SIZE AND NUMBER; MAXIMUM NUMBER           C
C                     OF ITERATIONS AND ALLOWABLE ERROR; SWITCH FOR      C
C                     READ IN HEADS AND HEAD ACCLERATION.                C
C                                                                        C
```

```
C     2       INPUT     INITIAL HEADS (BY LAYERS), STRESSES (BY NODES),  C
C                       CONSTANT HEAD BOUNDARIES (BY NODES, LAYERS)      C
C                                                                        C
C     3       INPUT     INITIAL HEADS (BY NODES, LAYERS), NODAL X-Y      C
C                       COORDINATES (BY NODES)                           C
C                                                                        C
C     4       INPUT     ELEMENT PROPERTIES: TOP; BOTTOM; HYDRAULIC       C
C                       CONDUCTIVITIES IN X,Y,Z DIRECTIONS; STORATIVITY. C
C                                                                        C
C     5       INPUT     ELEMENT PROPERTIES: SPECIFIC YIELD, RECHARGE     C
C                                                                        C
C     6       OUTPUT    WRITE SUBROUTINE (ECHO OF READ SUBROUTINE);      C
C                       PROGRESS OF PROGRAM (TIMESTEP NUMBER,            C
C                       ITERATION NUMBER, ETC.)                          C
C                                                                        C
C     7       OUTPUT    SUMMARY OF INPUT DATA, WATER BALANCE ERROR;      C
C                       NUMBER OF ITERATIONS PERFORMED; LARGEST ERROR    C
C                       ENCOUNTERED; FINAL HEAD RESULTS                  C
C                                                                        C
C     8       OUTPUT    STATUS OF NODES (UNCONFINED,CONFINED, ETC.);     C
C                       VARIOUS TEMPORARY OUTPUTS TO CHECK PROGRAM       C
C                                                                        C
C     9       INPUT     ELEMENT NODE ASSIGNMENTS                         C
C                                                                        C
C    10       OUTPUT    ERROR STATUS: TIMESTEP NUMBER; ITERATION         C
C                       NUMBER, MAXIMUM ERROR SIZE, LAYER AND NODE       C
C                       WHERE MAXIMUM ERROR OCCURED                      C
C                                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                        C
C     VARIABLE LISTING: MAIN PROGRAM                                     C
C                                                                        C
C     NAME    TYPE        DESCRIPTION                                    C
C     ------  ---------   ---------------------------------------------  C
C     EXMAX   REAL*8      MAXIMUM ERROR ENCOUNTERED BETWEEN ITERATIONS   C
C     IT      INTEGER*2   TIMESTEP LOOP COUNTER                          C
C     ITERTT  INTEGER*2   COUNTER FOR TOTAL NUMBER OF ITERATIONS         C
C                         PERFORMED IN A RUN                             C
C     ITER    INTEGER*2   ITERATION LOOP COUNTER                         C
C     IZ      INTEGER*2   LAYER LOOP COUNTER                             C
C     MXITER  INTEGER*2   MAXIMUM ALLOWABLE ITERATIONS                   C
C     NDRY    INTEGER*2   FLAG FOR PRESENCE OF DRY NODE (IF GREATER      C
C                         THAN ZERO, DRY NODE PRESENT)                   C
C     NCONT   INTEGER*2   FLAG FOR CONTINUATION OF ITERATION             C
C                         LOOP (IF 1, CONTINUE ITERATING, ELSE GO TO     C
C                         NEXT TIMESTEP)                                 C
C     NLAY    INTEGER*2   NUMBER OF LAYERS                               C
C     NLAY1   INTEGER*2   NUMBER OF LAYERS PLUS ONE                      C
C     NLSTRT  INTEGER*2   LAYER NUMBER WHERE LAYER IS UNCONFINED         C
C     NRAD    INTEGER*2   SWITCH FOR RADIAL OUTPUT (IF 1, RADIAL OUTPUT)C
C     NSS     INTEGER*2   SWITCH FOR STEADY STATE CASE (IF 1, STEADY     C
C                         STATE)                                         C
C     NSTART  INTEGER*2   BEGINNING LAYER NUMBER FOR LAYER LOOP          C
C     NTIMST  INTEGER*2   TOTAL NUMBER OF TIMESTEPS                      C
C     NXON    INTEGER*2   SWITCH FOR ESTIMATING STORAGE FOR WATER        C
C                         BALANCE (ENSURES THAT STORAGE WILL BE          C
C                         CALCULATED FOR WATER BALANCE) IF 1, CALCULATE C
C                                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                        C
```

```
C                    MAIN PROGRAM                                          C
C                                                                          C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4ERR(51,20),NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NECHO,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
      CALL READ
      IF (NECHO.EQ.1) THEN
           CALL WRITE
      ENDIF
C.....CALL DATA PROCESSING SUBROUTINES
      CALL BANDNO
      CALL TOPZ
      CALL RECHG
C.....CALL DATA SUMMARY SUBROUTINE
      CALL WRITFL
C.....INITIALIZE TOTAL ITERATION COUNTER
      ITERTT=0
C.....INITIALIZE MAXIMUM ERROR FOR RUN
      EMXMAX=0.D0
C.....BEGIN TIME LOOP
      DO 60 IT=1,NTIMST
           WRITE (6,1000) IT,NTIMST
C..........INITIALIZE ITERATION LOOP
           ITER=0
C..........BEGIN ITERATION LOOP, START ITERATION COUNTER
   21      ITER=ITER+1
C..........START TOTAL ITERATION COUNTER
           ITERTT=ITERTT+1
           WRITE (6,1003) ITER,MXITER
C..........CHECK STATUS OF HEADS
           CALL CHKHED(2)
C..........EXCHANGE NEW HEADS FOR OLD HEADS
           CALL EXCHNG
C..........CHECK STATUS OF HEADS
           CALL CHKHED(3)
C..........IF NECESSARY, CHANGE STRESSES TO ACCOUNT FOR DRY NODES
           IF (NDRY.GT.0) THEN
                CALL QRDRY
           ENDIF
C..........BEGIN Z LAYER LOOP FOR PREDICTOR EQUATIONS:
C..........PERFORMANCE OF LOOP DEPENDS ON TIMESTEP, ITERATION,
C..........CONFINED OR UNCONFINED STATUS.
C..........SET NLAY1
           NLAY1=NLAY+1
C..........FIRST TIMESTEP, FIRST ITERATION, CONFINED OR UNCONFINED
           IF ( (IT.EQ.1) .AND. (ITER.EQ.1) ) THEN
                DO 20 IZ=1,NLAY
                     CALL FORMKH
                     CALL FORMST(0)
                     CALL LHSPRD
                     CALL LHSDIR
```

```
C.....................IF NECESSARY, ACCOUNT FOR DRY NODES
                      IF (NDRY.GT.0) THEN
                           CALL LHSDRY
                      ENDIF
                      CALL FACTOR
   20                 CALL FORMKV(1,1)
C..........FIRST TIMESTEP, SECOND OR MORE ITERATIONS, UNCONFINED
           ELSEIF ( (IT.EQ.1) .AND. (ITER.GT.1) ) THEN
C...............EXECUTE LOOP ONLY FOR UNCONFINED LAYERS
                NSTART=NLSTRT
                DO 25 IZ=NSTART,NLAY
                      CALL FORMKH
                      CALL FORMST(0)
                      CALL LHSPRD
                      CALL LHSDIR
C.....................IF NECESSARY, ACCOUNT FOR DRY NODES
                      IF (NDRY.GT.0) THEN
                           CALL LHSDRY
                      ENDIF
                      CALL FACTOR
   25                 CALL FORMKV(1,1)
C..........SECOND OR MORE TIMESTEPS, FIRST ITERATION, UNCONFINED
           ELSEIF ( (IT.GT.1) .AND. (ITER.EQ.1) .AND.
     1               (NLSTRT.LT.NLAY1) ) THEN
                DO 30 IZ=1,NLAY
                      CALL FORMKH
                      CALL FORMST(0)
                      CALL LHSPRD
                      CALL LHSDIR
C.....................IF NECESSARY, ACCOUNT FOR DRY NODES
                      IF (NDRY.GT.0) THEN
                           CALL LHSDRY
                      ENDIF
                      CALL FACTOR
   30                 CALL FORMKV(1,1)
C..........SECOND OR MORE TIMESTEPS, SECOND OR MORE ITERATIONS,
C..........UNCONFINED
           ELSEIF ( (IT.GT.1) .AND. (ITER.GT.1) ) THEN
C...............EXECUTE LOOP ONLY FOR UNCONFINED LAYERS
                NSTART=NLSTRT
                DO 35 IZ=1,NLAY
                      CALL FORMKH
                      CALL FORMST(0)
                      CALL LHSPRD
                      CALL LHSDIR
C.....................IF NECESSARY, ACCOUNT FOR DRY NODES
                      IF (NDRY.GT.0) THEN
                           CALL LHSDRY
                      ENDIF
                      CALL FACTOR
   35                 CALL FORMKV(1,1)
           ENDIF
C..........BEGIN Z LAYER LOOP FOR REMAINING PREDICTOR EQUATIONS
           DO 50 IZ=1,NLAY
                CALL RHSPRD
                CALL RHSDIR
C...............IF NECESSARY, ACCOUNT FOR DRY NODES
                IF (NDRY.GT.0) THEN
                      CALL RHSDRY
                ENDIF
```

```
                CALL SOLVE
C               WRITE (6,1001) IZ,NLAY
   50     CONTINUE
C.........CHECK FOR STEADY STATE CASE
          IF (NSS.NE.1) THEN
C.........CALCULATE KV FOR CORRECTOR LOOP
          DO 53 IZ=1,NLAY
   53         CALL FORMKV(3,2)
C.........BEGIN CORRECTOR LOOP FOR NODES
          DO 55 INCOR=1,NNODE
               CALL LHSCOR
C..............IF NECESSARY, ACCOUNT FOR DRY NODES ON LEFT-HAND SIDE
               IF (NDRY.GT.0) THEN
                    CALL LCORDR
               ENDIF
               CALL RHSCOR
C..............IF NECESSARY, ACCOUNT FOR DRY NODES ON RIGHT-HAND SIDE
               IF (NDRY.GT.0) THEN
                    CALL RCORDR
               ENDIF
               CALL THMALG
C              WRITE (6,1002) INCOR,NNODE
   55     CONTINUE
          ENDIF
C.........CHECK CONVERGENCE
          CALL CONVER(NCONT)
          IF (NCONT.EQ.1) THEN
              GO TO 21
          ENDIF
   60 CONTINUE
C.....WATER BALANCE
      CALL WATBAL
C.....WRITE TOTAL NUMBER OF ITERATIONS
      WRITE (7,1011) ITERTT
      WRITE (7,1012)
C.....OUTPUT
      IF (NRAD.EQ.1) THEN
C..........WRITE OUT HEADS IN "R",HEAD,Z FORMAT
           CALL OUTRAD
      ELSE
C..........WRITE OUT HEADS IN X,Y,HEAD,Z FORMAT
C..........BEGIN Z LAYER LOOP
           DO 70 IZ=1,NLAY
   70      CALL OUTCOL
      ENDIF
 1000 FORMAT (' TIMESTEP=',I4,' OF',I4)
 1001 FORMAT (' LAYER=',I4,' OF',I4)
 1002 FORMAT (' NODE=',I4,' OF',I4)
 1003 FORMAT (' ITERATION=',I4,' OF',I4,' MAX')
 1008 FORMAT (' ITER=',I4)
 1011 FORMAT (' TOTAL NUMBER OF ITERATIONS PERFORMED = ',I6)
 1012 FORMAT (' ')
      STOP
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     VARIABLE LISTING: SUBROUTINE READ                                C
C                                                                      C
C     NAME    TYPE       DESCRIPTION                                   C
C     ------  ---------  --------------------------------------------- C
```

```
C     DELTIM  REAL*8      TIME STEP SIZE [T]                              C
C     ELMPRP  REAL*8      TOP OF ELEMENT (ELEMENTS,1,LAYERS); [L]         C
C     ELMPRP  REAL*8      BOTTOM OF ELEMENT (ELEMENTS,2,LAYERS); [L]      C
C     ELMPRP  REAL*8      HYDRAULIC CONDUCTIVITY IN X DIRECTION           C
C                         (ELEMENTS,3,LAYERS); [L/T]                      C
C     ELMPRP  REAL*8      HYDRAULIC CONDUCTIVITY IN Y DIRECTION           C
C                         (ELEMENTS,4,LAYERS); [L/T]                      C
C     ELMPRP  REAL*8      HYDRAULIC CONDUCTIVITY IN Z DIRECTION           C
C                         (ELEMENTS,5,LAYERS); [L/T]                      C
C     ELMPRP  REAL*8      STORATIVITY (ELEMENTS,6,LAYERS); [D]            C
C     ELMPRP  REAL*8      SPECIFIC YIELD (ELEMENTS,7,LAYERS); [L/T]       C
C     ELMPRP  REAL*8      RECHARGE (ELEMENTS,8,LAYERS); [L2/T]            C
C     ERRALL  REAL*8      ALLOWABLE ERROR BETWEEN TIMESTEPS [D]           C
C     HEAD    REAL*8      HEADS FROM OLD TIMESTEP (NODES,LAYERS,1); [L] C
C     HEAD    REAL*8      HEADS FROM NEW ITERATION (NODES,LAYERS,2); [L]C
C     HEAD    REAL*8      HEADS FROM OLD ITERATION (NODES,LAYERS,3); [L]C
C     IE      INTEGER*2   LOOP COUNTER FOR ELEMENTS                       C
C     IN      INTEGER*2   LOOP COUNTER FOR NODES                          C
C     IP      INTEGER*2   LOOP COUNTER FOR INDEX IN ELEMENT PROPERTIES,   C
C                         NODAL COORDINATES, NODE ASSIGNMENTS             C
C     IZ      INTEGER*2   LOOP COUNTER FOR LAYERS                         C
C     MXITER  INTEGER*2   MAXIMUM ALLOWABLE ITERATIONS                    C
C     NACCL   INTEGER*2   SWITCH FOR ACCELERATING HEADS BETWEEN           C
C                         TIMESTEPS (IF 1, ACCELERATE HEADS)              C
C     NECHO   INTEGER*2   SWITCH FOR ECHOING OUT READ IN DATA             C
C                         (IF 1, ECHO)                                    C
C     NELEM   INTEGER*2   TOTAL NUMBER OF ELEMENTS                        C
C     NEM     INTEGER*2   NODES ASSIGNED TO ELEMENT (ELEMENTS,(I,J,K))    C
C     NLAY    INTEGER*2   NUMBER OF LAYERS                                C
C     NNODE   INTEGER*2   TOTAL NUMBER OF NODES                           C
C     NODCOR  REAL*8      X COORDINATES OF NODES (NODES,1); [L]           C
C     NODCOR  REAL*8      Y COORDINATES OF NODES (NODES,2); [L]           C
C     NODFLG  INTEGER*2   NODAL FLAG FOR CONSTANT HEAD (NODES,LAYERS,1)   C
C     NODFLG  INTEGER*2   NODAL FLAG FOR HEAD STATUS (NODES,LAYERS,2)     C
C                         SEE SUBROUTINE CHKHED FOR EXPLANATION OF        C
C                         FLAG NUMBERS                                    C
C     NRAD    INTEGER*2   SWITCH FOR RADIAL OUTPUT (IF 1, RADIAL OUTPUT)C
C     NSS     INTEGER*2   SWITCH FOR STEADY STATE CASE (IF 1, STEADY      C
C                         STATE)                                          C
C     NTIMST  INTEGER*2   TOTAL NUMBER OF TIMESTEPS                       C
C     Q       REAL*8      NODAL STRESS (NODES,LAYERS); [L3/T]             C
C                                                                         C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                         C
C  SUBROUTINE READ DESCRIPTION:                                           C
C  READS IN ALL RELEVANT INPUT DATA.  SEE MAIN PROGRAM                    C
C  FOR UNIT ASSIGNMENTS FOR THIS SUBROUTINE                               C
C                                                                         C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                         C
C            SUBROUTINE READ                                              C
C                                                                         C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE READ
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
```

```
     4NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NECHO,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
      READ (1,1000) NNODE,NELEM,NLAY
      READ (1,1001) DELTIM,NTIMST,NECHO,NACCL,NSS,NRAD
      READ (1,1002) ERRALL,MXITER
      DO 200 IN=1,NNODE
  200 READ (2,1004) ( Q(IN,IZ), IZ=1,NLAY )
      DO 300 IN=1,NNODE
  300 READ (2,1005) ( NODFLG(IN,IZ,1), IZ=1,NLAY )
      DO 400 IN=1,NNODE
  400 READ (3,1006) ( HEAD(IN,IZ,2), IZ=1,NLAY )
      DO 500 IN=1,NNODE
  500 READ (3,1007) ( NODCOR(IN,IP), IP=1,2 )
      DO 550 IZ=1,NLAY
          DO 550 IE=1,NELEM
  550 READ (4,1008) ( ELMPRP(IE,IP,IZ), IP=1,6 )
      DO 600 IZ=1,NLAY
          DO 600 IE=1,NELEM
  600 READ (5,1009) ( ELMPRP(IE,IP,IZ), IP=7,8 )
      DO 700 IE=1,NELEM
  700 READ (9,1010) ( NEM(IE,IP), IP=1,3 )
 1000 FORMAT (3I4)
 1001 FORMAT (F10.4,5I4)
 1002 FORMAT (E12.5,I4)
 1004 FORMAT ((7F10.4))
 1005 FORMAT ((15I4))
 1006 FORMAT ((7F10.4))
 [illegible]07 FORMAT ((7F10.4))
 1008 FORMAT ((6E12.5))
 1009 FORMAT ((2E12.5))
 1010 FORMAT ((4I4))
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     VARIABLE LISTING: SUBROUTINE WRITE                               C
C                                                                      C
C     NAME    TYPE       DESCRIPTION                                   C
C     ------  ---------  --------------------------------------------  C
C     DELTIM  REAL*8     TIME STEP SIZE [T]                            C
C     ELMPRP  REAL*8     TOP OF ELEMENT (ELEMENTS,1,LAYERS); [L]       C
C     ELMPRP  REAL*8     BOTTOM OF ELEMENT (ELEMENTS,2,LAYERS); [L]    C
C     ELMPRP  REAL*8     HYDRAULIC CONDUCTIVITY IN X DIRECTION         C
C                        (ELEMENTS,3,LAYERS); [L/T]                    C
C     ELMPRP  REAL*8     HYDRAULIC CONDUCTIVITY IN Y DIRECTION         C
C                        (ELEMENTS,4,LAYERS); [L/T]                    C
C     ELMPRP  REAL*8     HYDRAULIC CONDUCTIVITY IN Z DIRECTION         C
C                        (ELEMENTS,5,LAYERS); [L/T]                    C
C     ELMPRP  REAL*8     STORATIVITY (ELEMENTS,6,LAYERS); [D]          C
C     ELMPRP  REAL*8     SPECIFIC YIELD (ELEMENTS,7,LAYERS); [L/T]     C
C     ELMPRP  REAL*8     RECHARGE (ELEMENTS,8,LAYERS); [L2/T]          C
C     ERRALL  REAL*8     ALLOWABLE ERROR BETWEEN TIMESTEPS [D]         C
C     HEAD    REAL*8     HEADS FROM OLD TIMESTEP (NODES,LAYERS,1); [L] C
C     HEAD    REAL*8     HEADS FROM NEW ITERATION (NODES,LAYERS,2); [L]C
C     HEAD    REAL*8     HEADS FROM OLD ITERATION (NODES,LAYERS,3); [L]C
C     IE      INTEGER*2  LOOP COUNTER FOR ELEMENTS                     C
C     IN      INTEGER*2  LOOP COUNTER FOR NODES                        C
```

```
C     IP       INTEGER*2  LOOP COUNTER FOR INDEX IN ELEMENT PROPERTIES, C
C                         NODAL COORDINATES, NODE ASSIGNMENTS          C
C     IZ       INTEGER*2  LOOP COUNTER FOR LAYERS                      C
C     MXITER   INTEGER*2  MAXIMUM ALLOWABLE ITERATIONS                 C
C     NACCL    INTEGER*2  SWITCH FOR ACCELERATING HEADS BETWEEN        C
C                         TIMESTEPS (IF 1, ACCELERATE HEADS)           C
C     NECHO    INTEGER*2  SWITCH FOR ECHOING OUT READ IN DATA          C
C                         (IF 1, ECHO)                                 C
C     NELEM    INTEGER*2  TOTAL NUMBER OF ELEMENTS                     C
C     NEM      INTEGER*2  NODES ASSIGNED TO ELEMENT (ELEMENTS,(I,J,K)) C
C     NLAY     INTEGER*2  NUMBER OF LAYERS                             C
C     NNODE    INTEGER*2  TOTAL NUMBER OF NODES                        C
C     NODCOR   REAL*8     X COORDINATES OF NODES (NODES,1): [L]        C
C     NODCOR   REAL*8     Y COORDINATES OF NODES (NODES,2): [L]        C
C     NODFLG   INTEGER*2  NODAL FLAG FOR CONSTANT HEAD (NODES,LAYERS,1) C
C     NODFLG   INTEGER*2  NODAL FLAG FOR HEAD STATUS (NODES,LAYERS,2)  C
C                         SEE SUBROUTINE CHKHED FOR EXPLANATION OF     C
C                         FLAG NUMBERS                                 C
C     NRAD     INTEGER*2  SWITCH FOR RADIAL OUTPUT (IF 1, RADIAL OUTPUT)C
C     NSS      INTEGER*2  SWITCH FOR STEADY STATE CASE (IF 1, STEADY   C
C                         STATE)                                       C
C     NTIMST   INTEGER*2  TOTAL NUMBER OF TIMESTEPS                    C
C     Q        REAL*8     NODAL STRESS (NODES,LAYERS): [L3/T]          C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C  SUBROUTINE WRITE DESCRIPTION:                                       C
C  ECHOS OUT ALL DATA READ IN FROM SUBROUTINE READ.                    C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C            SUBROUTINE WRITE                                          C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE WRITE
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NECHO,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
      WRITE (6,1000) NNODE,NELEM,NLAY
      WRITE (6,1001) DELTIM,NTIMST,NECHO,NACCL,NSS,NRAD
      WRITE (6,1002) ERRALL,MXITER
      DO 200 IN=1,NNODE
  200 WRITE (6,1004) ( Q(IN,IZ), IZ=1,NLAY )
      DO 300 IN=1,NNODE
  300 WRITE (6,1005) ( NODFLG(IN,IZ,1), IZ=1,NLAY )
      DO 400 IN=1,NNODE
  400 WRITE (6,1006) ( HEAD(IN,IZ,2), IZ=1,NLAY )
      DO 500 IN=1,NNODE
  500 WRITE (6,1007) ( NODCOR(IN,IP), IP=1,2 )
      DO 550 IZ=1,NLAY
          DO 550 IE=1,NELEM
  550 WRITE (6,1008) ( ELMPRP(IE,IP,IZ), IP=1,6 )
```

```
      DO 600 IZ=1,NLAY
          DO 600 IE=1,NELEM
  600 WRITE (6,1009) ( ELMPRP(IE,IP,IZ), IP=7,8 )
      DO 700 IE=1,NELEM
  [illegible]00 WRITE (6,1010) ( NEM(IE,IP), IP=1,3 )
 [illegible]000 FORMAT (3I4)
 1001 FORMAT (F10.4,5I4)
 1002 FORMAT (E12.5,I4)
 1004 FORMAT ((7F10.4))
 1005 FORMAT ((15I4))
 1006 FORMAT ((7F10.4))
 1007 FORMAT ((7F10.4))
 1008 FORMAT ((6E12.5))
 1009 FORMAT ((2E12.5))
 1010 FORMAT ((4I4))
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     VARIABLE LISTING: SUBROUTINE BANDNO                              C
C                                                                      C
C     NAME   TYPE       DESCRIPTION                                    C
C     ------  ---------  --------------------------------------------- C
C     D      REAL*8     NUMBER OF NODES BETWEEN ANY TWO NODES (3)      C
C     DMAX   REAL*8     MAXIMUM NUMBER OF NODES BETWEEN ANY TWO NODES C
C     I      INTEGER*2  NODE LOOP COUNTER (FOR EACH ELEMENT)           C
C     IE     INTEGER*2  ELEMENT LOOP COUNTER                           C
C     IP     INTEGER*2  COUNTER FOR NEXT NODE ON ELEMENT               C
C     NBAND  INTEGER*2  BANDWIDTH                                      C
C     NEM    INTEGER*2  NODES ASSIGNED TO ELEMENT (ELEMENTS,(I,J,K))   C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C  SUBROUTINE BANDNO DESCRIPTION:                                      C
C  CALCULATES BANDWIDTH OF NODAL DISCRETIZATION BY LOOKING AT MAXIMUM  C
C  DISTANCE (IN TERMS OF NODES) BEWTEEN ANY TWO NODES                  C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C            SUBROUTINE BANDNO                                         C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE BANDNO
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4D(3),NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NECHO,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
      DMAX=0
      DO 50 I=1,3
   50 D(I)=0
      DO 100 IE=1,NELEM
           DO 100 I=1,3
                IF (I.LE.2) THEN
```

```
                    IP=I+1
               ELSE
                    IP=1
               ENDIF
               D(I)=ABS(NEM(IE,I)-NEM(IE,IP))
               IF (D(I).GT.DMAX) THEN
                     DMAX=D(I)
               ENDIF
  100 CONTINUE
      NBAND=(2*DMAX)+1
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     VARIABLE LISTING: SUBROUTINE TOPZ                                 C
C                                                                       C
C     NAME    TYPE       DESCRIPTION                                    C
C     ------  ---------  ---------------------------------------------- C
C     ELMPRP  REAL*8     TOP OF ELEMENT (ELEMENTS,1,LAYERS); [L]        C
C     INT     INTEGER*2  NODE LOOP COUNTER (FOR EACH ELEMENT)           C
C     IET     INTEGER*2  ELEMENT LOOP COUNTER                           C
C     IZT     INTEGER*2  LAYER LOOP COUNTER                             C
C     NELEM   INTEGER*2  TOTAL NUMBER OF ELEMENTS                       C
C     NEM     INTEGER*2  NODES ASSIGNED TO ELEMENT (ELEMENTS,(I,J,K))   C
C     NLAY    INTEGER*2  NUMBER OF LAYERS                               C
C     NODE    INTEGER*2  NODE NUMBER ON ELEMENT (3)                     C
C     TOP     REAL*8     TOP OF NODE (NODES,LAYERS) [L]                 C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C  SUBROUTINE TOPZ DESCRIPTION:                                         C
C  ASSIGNS TOPS OF LAYERS TO NODES BY TRANSFERRING TOPS OF LAYERS       C
C  DATA FROM ELEMENTS.  DATA SAVED IN "TOP" ARRAY.                      C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C            SUBROUTINE TOPZ                                            C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE TOPZ
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NODE(3),NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NECHO,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
C.....ASSIGN ELEMENT LAYER TOPS TO NODE LAYER TOPS
      DO 50 IZT=1,NLAY
           DO 50 IET=1,NELEM
                DO 50 INT=1,3
                     NODE(1)=NEM(IET,1)
                     NODE(2)=NEM(IET,2)
                     NODE(3)=NEM(IET,3)
   50 TOP(NODE(INT),IZT)=ELMPRP(IET,1,IZT)
      RETURN
```

```
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     VARIABLE LISTING: SUBROUTINE RECHG                                C
                                                                        C
C     NAME    TYPE       DESCRIPTION                                    C
C     ------  ---------  ---------------------------------------------  C
C     AE      REAL*8     AREA OF TRIANGULAR ELEMENT [L2]                C
C     ELMPRP  REAL*8     RECHARGE (ELEMENTS,8,LAYERS); [L2/T]           C
C     IE      INTEGER*2  ELEMENT LOOP COUNTER                           C
C     IIZ     INTEGER*2  LAYER LOOP COUNTER                             C
C     IN      INTEGER*2  NODE LOOP COUNTER                              C
C     IZT     INTEGER*2  LAYER LOOP COUNTER                             C
C     NELEM   INTEGER*2  TOTAL NUMBER OF ELEMENTS                       C
C     NEM     INTEGER*2  NODES ASSIGNED TO ELEMENT (ELEMENTS,(I,J,K))   C
C     NLAY    INTEGER*2  NUMBER OF LAYERS                               C
C     NNODE   INTEGER*2  TOTAL NUMBER OF NODES                          C
C     NLAY    INTEGER*2  NUMBER OF LAYERS                               C
C     NODCOR  REAL*8     X COORDINATES OF NODES (NODES,1); [L]          C
C     NODCOR  REAL*8     Y COORDINATES OF NODES (NODES,2); [L]          C
C     NODE    INTEGER*2  NODE NUMBER ON ELEMENT (3)                     C
C     RECHGN  REAL*8     RECHARGE AS A NODAL QUANTITY (NODES,LAYERS)    C
C                        [L3/T]                                         C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C  SUBROUTINE RECHG DESCRIPTION:                                        C
C  ASSIGNS RECHRGE TO NODES BY INTEGRATING RECHARGE OVER EACH ELEMENT.  C
C  CHANGES RECHARGE FROM [L/T] TO [L3/T]. ASSIGNS RECHARGE TO           C
C  "RECHGN" ARRAY.                                                      C
                                                                        C
C     ***NOTE THAT THIS SUBROUTINE COMPUTES RECHARGE SUCH               C
C     ***THAT RECHARGE IS A NEGATIVE QUANTITY WHEN IT                   C
C     ***LEAVES THIS SUBROUTINE                                         C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C            SUBROUTINE RECHG                                           C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE RECHG
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NODE(3),NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NECHO,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
      DO 50 IN=1,NNODE
           DO 50 IIZ=1,NLAY
   50 RECHGN(IN,IIZ)=0.D00
      DO 100 IE=1,NELEM
C         CALCULATE AREA OF TRIANGULAR ELEMENT
          AE=0.5*((NODCOR(NEM(IE,1),1)*NODCOR(NEM(IE,2),2)
     1       -NODCOR(NEM(IE,2),1)*NODCOR(NEM(IE,1),2))
     2       +(NODCOR(NEM(IE,3),1)*NODCOR(NEM(IE,1),2)
```

```
     3       -NODCOR(NEM(IE,1),1)*NODCOR(NEM(IE,3),2))
     4       +(NODCOR(NEM(IE,2),1)*NODCOR(NEM(IE,3),2)
     5       -NODCOR(NEM(IE,3),1)*NODCOR(NEM(IE,2),2)))
           DO 100 IN=1,3
                NODE(1)=NEM(IE,1)
                NODE(2)=NEM(IE,2)
                NODE(3)=NEM(IE,3)
  100           RECHGN(NODE(IN),NLAY)=RECHGN(NODE(IN),NLAY)+
     1                            (ELMPRP(IE,8,NLAY)*AE/3)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     VARIABLE LISTING: SUBROUTINE WRITFL                              C
C                                                                      C
C     NAME    TYPE       DESCRIPTION                                   C
C     ------  ---------  --------------------------------------------  C
C     DELTIM  REAL*8     TIME STEP SIZE [T]                            C
C     ELMPRP  REAL*8     TOP OF ELEMENT (ELEMENTS,1,LAYERS); [L]       C
C     ELMPRP  REAL*8     HYDRAULIC CONDUCTIVITY IN X DIRECTION         C
C                        (ELEMENTS,3,LAYERS); [L/T]                    C
C     ELMPRP  REAL*8     HYDRAULIC CONDUCTIVITY IN Y DIRECTION         C
C                        (ELEMENTS,4,LAYERS); [L/T]                    C
C     ELMPRP  REAL*8     HYDRAULIC CONDUCTIVITY IN Z DIRECTION         C
C                        (ELEMENTS,5,LAYERS); [L/T]                    C
C     ELMPRP  REAL*8     STORATIVITY (ELEMENTS,6,LAYERS); [D]          C
C     ELMPRP  REAL*8     SPECIFIC YIELD (ELEMENTS,7,LAYERS); [L/T]     C
C     ELMPRP  REAL*8     RECHARGE (ELEMENTS,8,LAYERS); [L2/T]          C
C     ERRALL  REAL*8     ALLOWABLE ERROR BETWEEN TIMESTEPS [D]         C
C     HEAD    REAL*8     HEADS FROM NEW ITERATION (NODES,LAYERS,2); [L]C
C     IE      INTEGER*2  LOOP COUNTER FOR ELEMENTS                     C
C     IN      INTEGER*2  LOOP COUNTER FOR NODES                        C
C     IZ      INTEGER*2  LOOP COUNTER FOR LAYERS                       C
C     MXITER  INTEGER*2  MAXIMUM ALLOWABLE ITERATIONS                  C
C     NACCL   INTEGER*2  SWITCH FOR ACCELERATING HEADS BETWEEN         C
C                        TIMESTEPS (IF 1, ACCELERATE HEADS)            C
C     NELEM   INTEGER*2  TOTAL NUMBER OF ELEMENTS                      C
C     NLAY    INTEGER*2  NUMBER OF LAYERS                              C
C     NNODE   INTEGER*2  TOTAL NUMBER OF NODES                         C
C     NSS     INTEGER*2  SWITCH FOR STEADY STATE CASE (IF 1, STEADY    C
C                        STATE)                                        C
C     NTIMST  INTEGER*2  TOTAL NUMBER OF TIMESTEPS                     C
C     Q       REAL*8     NODAL STRESS (NODES,LAYERS); [L3/T]           C
C     QTOT    REAL*8     TOTAL STRESS  [L3/T]                          C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C  SUBROUTINE WRITFL DESCRIPTION:                                      C
C  WRITES OUT A SUMMARY OF INPUT DATA TO HEAD OUTPUT FILE.             C
C  APPROPRIATE ONLY FOR HOMOGENEOUS DATA SET BECAUSE SUBROUTINE ONLY   C
C  LOOKS AT FIRST DATA ITEM FOR EACH PARAMTER.                         C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C            SUBROUTINE WRITFL                                         C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE WRITFL
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
```

```
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NECHO,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
      WRITE (7,997)
      WRITE (7,998)
      WRITE (7,1000) NNODE
      WRITE (7,1001) NLAY
      WRITE (7,1002) NELEM
      WRITE (7,1003) NBAND
      WRITE (7,1004) DELTIM
      WRITE (7,1005) NTIMST
      WRITE (7,1006) ERRALL
      WRITE (7,1007) MXITER
      DO 200 IN=1,NNODE
      DO 200 IZ=1,NLAY
  200 QTOT=QTOT+Q(IN,IZ)
      WRITE (7,1008) QTOT
      WRITE (7,1009) HEAD(1,1,2)
      WRITE (7,1010) ELMPRP(1,3,1)
      WRITE (7,1011) ELMPRP(1,4,1)
      WRITE (7,1012) ELMPRP(1,5,1)
      WRITE (7,1013) ELMPRP(1,6,1)
      WRITE (7,1014) ELMPRP(1,7,1)
      WRITE (7,1015) ELMPRP(1,8,1)
      DO 300 IZ=1,NLAY
  300 WRITE (7,1016) IZ,ELMPRP(1,1,IZ)
      WRITE (7,999)
      IF (NACCL.EQ.0) THEN
           WRITE (7,1017)
      ELSE IF (NACCL.EQ.1) THEN
           WRITE (7,1018)
      ENDIF
      WRITE (7,999)
      IF (NSS.EQ.0) THEN
           WRITE (7,1021)
      ELSE IF (NSS.EQ.1) THEN
           WRITE (7,1022)
      ENDIF
      WRITE (7,999)
      WRITE (7,1019)
      WRITE (7,1020)
  997 FORMAT (' INPUT DATA')
  998 FORMAT (' ===== ====')
  999 FORMAT (' ')
 1000 FORMAT (' NUMBER OF NODES = ',I4)
 1001 FORMAT (' NUMBER OF LAYERS = ',I4)
 1002 FORMAT (' NUMBER OF ELEMENTS= ',I4)
 1003 FORMAT (' BANDWIDTH = ',I4)
 1004 FORMAT (' DELTA TIME = ',F10.4)
 1005 FORMAT (' NUMBER OF TIMESTEPS = ',I4)
 1006 FORMAT (' MAXIMUM ALLOWABLE ERROR = ',E12.5)
 1007 FORMAT (' MAXIMUM NUMBER OF ITERATIONS = ',I4)
 1008 FORMAT (' TOTAL STRESS = ',F10.4)
 1009 FORMAT (' INITIAL HEAD = ',F10.4)
```

```
 1010 FORMAT (' HYDRAULIC CONDUCTIVITY IN X DIRECTION = ',E12.5)
 1011 FORMAT (' HYDRAULIC CONDUCTIVITY IN Y DIRECTION = ',E12.5)
 1012 FORMAT (' HYDRAULIC CONDUCTIVITY IN Z DIRECTION = ',E12.5)
 1013 FORMAT (' SPECIFIC STORAGE = ',E12.5)
 1014 FORMAT (' SPECIFIC YIELD = ',E12.5)
 1015 FORMAT (' RECHARGE = ',E12.5)
 1016 FORMAT (' TOP OF LAYER ',I4,' = ',E12.5)
 1017 FORMAT (' ACCELERATION OF HEAD ESTIMATES IS ---OFF---')
 1018 FORMAT (' ACCELERATION OF HEAD ESTIMATES IS ---ON---')
 1019 FORMAT (' OUTPUT DATA')
 1020 FORMAT (' ====== ====')
 1021 FORMAT (' TRANSIENT CASE')
 1022 FORMAT (' STEADY STATE CASE')
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     VARIABLE LISTING: SUBROUTINE EXCHNG                               C
C                                                                       C
C     NAME    TYPE        DESCRIPTION                                   C
C     ------  ---------   --------------------------------------------  C
C     HEAD    REAL*8      HEADS FROM OLD TIMESTEP (NODES,LAYERS,1); [L] C
C     HEAD    REAL*8      HEADS FROM NEW ITERATION (NODES,LAYERS,2); [L]C
C     HEAD    REAL*8      HEADS FROM OLD ITERATION (NODES,LAYERS,3); [L]C
C     IN      INTEGER*2   LOOP COUNTER FOR NODES                        C
C     IPT     INTEGER*2   LOOP COUNTER FOR INDEX IN HEADS               C
C     IT      INTEGER*2   TIMESTEP LOOP COUNTER                         C
C     ITER    INTEGER*2   ITERATION LOOP COUNTER                        C
C     IZ      INTEGER*2   LOOP COUNTER FOR LAYERS                       C
C     NACCL   INTEGER*2   SWITCH FOR ACCELERATING HEADS BETWEEN         C
C                         TIMESTEPS (IF 1, ACCELERATE HEADS)            C
C     NLAY    INTEGER*2   NUMBER OF LAYERS                              C
C     NLAY1   INTEGER*2   NUMBER OF LAYERS PLUS ONE                     C
C     NLSTRT  INTEGER*2   LAYER NUMBER WHERE LAYER IS UNCONFINED        C
C     NLSTR1  INTEGER*2   LAYER NUMBER WHERE LAYER IS CONFINED          C
C     NNODE   INTEGER*2   TOTAL NUMBER OF NODES                         C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C  SUBROUTINE EXCHNG DESCRIPTION:                                       C
C  EXCHANGES HEADS FROM OLD ITERATION TO NEW ITERATION OR FROM OLD      C
C  TIMESTEP TO NEW TIMESTEP.  ACCELERATES HEAD ESTIMATES FOR FIRST      C
C  ITERATION IN A TIMESTEP IF FLAG "NACCL" = 1.  HEAD ACCELERATOR       C
C  IS A FIRST-ORDER APPROXIMATION OF DH/DT.  ONLY UNCONFINED HEADS      C
C  ARE ACCELERATED.                                                     C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C            SUBROUTINE EXCHNG                                          C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE EXCHNG
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
```

```
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NECHO,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
C.....FIRST TWO TIME STEPS (UNCONFINED OR CONFINED)
      IF ( (IT.LE.2) .OR. (NACCL.EQ.0) ) THEN
           DO 10 IZ=1,NLAY
                DO 10 IN=1,NNODE
                     HEAD(IN,IZ,3)=HEAD(IN,IZ,2)
                     IF (ITER.EQ.1) THEN
                          HEAD(IN,IZ,1)=HEAD(IN,IZ,2)
                     ENDIF
   10      CONTINUE
C.....TIME STEPS GREATER THAN TWO
      ELSE IF ( (IT.GT.2) .AND. (NACCL.EQ.1) ) THEN
C..........COMPUTE NUMBER OF LAYERS PLUS ONE
           NLAY1=NLAY+1
C..........EXCHANGE AND ACCELERATE HEADS ONLY IF FIRST
C..........ITERATION IN A TIMESTEP AND ONLY IF UNCONFINED HEADS EXIST
           IF ( (ITER.EQ.1) .AND. (NLSTRT.LT.NLAY1) ) THEN
C...............UNCONFINED LAYERS
                DO 30 IZ=NLSTRT,NLAY
                     DO 30 IN=1,NNODE
                          HEAD(IN,IZ,1)=(2*HEAD(IN,IZ,2))-HEAD(IN,IZ,1)
                          HEAD(IN,IZ,3)=HEAD(IN,IZ,1)
   30           WRITE (8,1000) IT,IZ,IN,( HEAD(IN,IZ,IPT), IPT=1,3 )
C...............CONFINED LAYERS
                NLSTR1=NLSTRT-1
                DO 35 IZ=1,NLSTR1
                     DO 35 IN=1,NNODE
                          HEAD(IN,IZ,1)=HEAD(IN,IZ,2)
   35           HEAD(IN,IZ,3)=HEAD(IN,IZ,2)
C..........EXCHANGE HEADS IF GREATER THAN FIRST ITERATION IN A
C..........TIMESTEP AND ONLY IF UNCONFINED HEADS EXIST
           ELSE IF ( (ITER.GT.1) .AND. (NLSTRT.NE.NLAY1) ) THEN
                DO 40 IZ=1,NLAY
                     DO 40 IN=1,NNODE
   40           HEAD(IN,IZ,3)=HEAD(IN,IZ,2)
C..........EXCHANGE HEADS ONLY IF FIRST ITERATION IN A
C..........TIMESTEP AND ONLY IF ALL HEADS CONFINED
           ELSE IF ( (ITER.EQ.1) .AND. (NLSTRT.EQ.NLAY1) ) THEN
                DO 50 IZ=1,NLAY
                     DO 50 IN=1,NNODE
                          HEAD(IN,IZ,1)=HEAD(IN,IZ,2)
   50                     HEAD(IN,IZ,3)=HEAD(IN,IZ,2)
           ENDIF
      ENDIF
 1000 FORMAT (3I4,(3F10.4))
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     VARIABLE LISTING: SUBROUTINE CHKHED                              C
C                                                                      C
C     NAME    TYPE       DESCRIPTION                                   C
C     ------  ---------  --------------------------------------------  C
C     ELMPRP  REAL*8     TOP OF ELEMENT (ELEMENTS,1,LAYERS); [L]       C
C     ELMPRP  REAL*8     BOTTOM OF ELEMENT (ELEMENTS,2,LAYERS); [L]    C
C     HEAD    REAL*8     HEADS FROM OLD TIMESTEP (NODES,LAYERS,1); [L] C
C     HEAD    REAL*8     HEADS FROM NEW ITERATION (NODES,LAYERS,2); [L]C
C     HEAD    REAL*8     HEADS FROM OLD ITERATION (NODES,LAYERS,3); [L]C
```

```
C     INCK     INTEGER*2  LOOP COUNTER FOR NODES                          C
C     INO      INTEGER*2  LOOP COUNTER FOR NODES                          C
C     IPCK     INTEGER*2  INDEX FOR HEADS (***PASSED***)                  C
C     ITER     INTEGER*2  ITERATION LOOP COUNTER                          C
C     IZCK     INTEGER*2  LOOP COUNTER FOR LAYERS                         C
C     IZO      INTEGER*2  LOOP COUNTER FOR LAYERS                         C
C     NACCL    INTEGER*2  SWITCH FOR ACCELERATING HEADS BETWEEN           C
C                         TIMESTEPS (IF 1, ACCELERATE HEADS)              C
C     NDRY     INTEGER*2  FLAG FOR PRESENCE OF DRY NODE (IF GREATER       C
C                         THAN ZERO, DRY NODE PRESENT)                    C
C     NLAY     INTEGER*2  NUMBER OF LAYERS                                C
C     NLSTRT   INTEGER*2  LAYER NUMBER WHERE LAYER IS UNCONFINED          C
C     NNODE    INTEGER*2  TOTAL NUMBER OF NODES                           C
C     NODFLG   INTEGER*2  NODAL FLAG FOR HEAD STATUS (NODES,LAYERS,2)     C
C                         SEE SUBROUTINE CHKHED FOR EXPLANATION OF        C
C                         FLAG NUMBERS                                    C
C     NS       INTEGER*2  COUNTER FOR UNCONFINED OR DRY NODES (IF         C
C                         GREATER THAN 1, IMPLIES UNCONF. OR DRY          C
C                         NODES PRESENT IN LAYER                          C
C                                                                         C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                         C
C  SUBROUTINE CHKHED DESCRIPTION:                                         C
C  CHECKS STATUS OF HEADS AND ASSIGNS INDEX NUMBER ACCORDING TO           C
C  STATUS:                                                                C
C          OLD CONFINED -- NODFLG(NODES,LAYERS,1)                         C
C          NEW CONFINED -- NODFLG(NODES,LAYERS,2)                         C
C          NEW UNCONFINED -- NODFLG(NODES,LAYERS,3)                       C
C          OLD UNCONFINED -- NODFLG(NODES,LAYERS,4)                       C
C          COMPLETELY UNSATURATED -- NODFLG(NODES,LAYERS,5)               C
C                                                                         C
C  ASSIGNS VALUE OF DRY NODE FLAG:                                        C
C          NO DRY NODES PRESENT IN SYSTEM -- 0                            C
C          DRY NODES PRESENT IN SYSTEM -- 1                               C
C                                                                         C
C  FINDS LAYER WHERE UNCONFINED HEADS EXIST AND ASSIGNS THAT LAYER        C
C  NUMBER TO VARIABLE "NLSTRT".                                           C
C                                                                         C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                         C
C            SUBROUTINE CHKHED                                            C
C                                                                         C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE CHKHED(IPCK)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NODE(3),NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
C.....INITIALIZE DRY NODE FLAG
      NDRY=0
C.....CHECK CONDITION OF HEADS AND ASSIGN CODE
      NLSTRT=NLAY+1
      DO 150 IZCK=1,NLAY
```

```
          NS=0
          DO 100 IECK=1,NELEM
C.........IDENTIFY NODE NUMBER ON TRIANGULAR ELEMENT
         NODE(1)=NEM(IECK,1)
         NODE(2)=NEM(IECK,2)
         NODE(3)=NEM(IECK,3)
          HSUM = 0.D00
          DO  50 INCK = 1,3
             HSUM = HSUM+HEAD(NODE(INCK),IZCK,IPCK)
   50     CONTINUE
          HBAR = HSUM/3.D00
C...............COMPLETELY SATURATED (OLD-LAST TIME STEP)
                IF ((HBAR.GE.ELMPRP(IECK,1,IZCK))
     1              .AND. (NELFLG(IECK,IZCK).LE.2)) THEN
                    NELFLG(IECK,IZCK)=1
C...............COMPLETELY SATURATED (NEW-THIS TIME STEP)
               ELSEIF ((HBAR.GE.ELMPRP(IECK,1,IZCK))
     1              .AND. (NELFLG(IECK,IZCK).GT.2)) THEN
                    NELFLG(IECK,IZCK)=2
                    IF ( (IT.EQ.1) .AND. (ITER.EQ.1) ) THEN
                         NELFLG(IECK,IZCK)=1
                    ENDIF
C...............PARTIALLY SATURATED (NEW-THIS TIME STEP)
               ELSEIF ((HBAR.LT.ELMPRP(IECK,1,IZCK))
     1              .AND.
     2             (HBAR.GT.ELMPRP(IECK,2,IZCK))
     3              .AND.
     4              (NELFLG(IECK,IZCK).LE.2)) THEN
                    NELFLG(IECK,IZCK)=3
                    IF ( (IT.EQ.1) .AND. (ITER.EQ.1) ) THEN
                         NELFLG(IECK,IZCK)=4
                    ENDIF
C...............PARTIALLY SATURATED (OLD-LAST TIME STEP)
               ELSEIF ((HBAR.LT.ELMPRP(IECK,1,IZCK))
     1              .AND.
     2             (HBAR.GT.ELMPRP(IECK,2,IZCK))
     3              .AND.
     4              (NELFLG(IECK,IZCK).GE.3)) THEN
                    NELFLG(IECK,IZCK)=4
C...............COMPLETELY UNSATURATED
               ELSEIF (HBAR.LE.ELMPRP(IECK,2,IZCK))
     1         THEN
                    NELFLG(IECK,IZCK)=5
                    NDRY=NDRY+1
               ENDIF
C...............CHECK FOR UNCONFINED HEADS
               IF ( (NELFLG(IECK,IZCK).GE.2) .AND.
     1              (NELFLG(IECK,IZCK).LE.5) ) THEN
                    NS=NS+1
               ENDIF
  100     CONTINUE
          IF (NS.GE.1) THEN
               NLSTRT=NLSTRT-1
          ENDIF
  150 CONTINUE
C     WRITE (8,1001) ITER
C     DO 200 INO=1,NNODE
C 200 WRITE (8,1000) ( NELFLG(INO,IZO), IZO=1,NLAY )
 1000 FORMAT (15I4)
 1001 FORMAT (' ITERATION=',I4)
```

```
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                        C
C     VARIABLE LISTING: SUBROUTINE QRDRY                                 C
C                                                                        C
C     NAME    TYPE        DESCRIPTION                                    C
C     ------  ----------  ---------------------------------------------  C
C     IN      INTEGER*2   LOOP COUNTER FOR NODES                         C
C     IZ      INTEGER*2   LOOP COUNTER FOR LAYERS                        C
C     IZM     INTEGER*2   LOOP COUNTER FOR LAYERS, MINUS 1  (IZ-1)       C
C     NLAY    INTEGER*2   NUMBER OF LAYERS                               C
C     NNODE   INTEGER*2   TOTAL NUMBER OF NODES                          C
C     NODFLG  INTEGER*2   NODAL FLAG FOR HEAD STATUS (NODES,LAYERS,2)    C
C                         SEE SUBROUTINE CHKHED FOR EXPLANATION OF       C
C                         FLAG NUMBERS                                   C
C     Q       REAL*8      NODAL STRESS (NODES,LAYERS); [L3/T]            C
C     QDRY    REAL*8      NODAL STRESS ADJUSTED FOR DRY LAYERS           C
C                         (NODES,LAYERS); [L3/T]                         C
C                                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                        C
C  SUBROUTINE QRDRY DESCRIPTION:                                         C
C  REASSIGNS NODAL STRESSES Q TO QRDRY IF UNSATURATED LAYERS EXIST.      C
C  IF LAYER IZ IS UNSATURATED, ASSIGN Q TO LAYER IZ-1.                   C
C                                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                        C
C             SUBROUTINE QRDRY                                           C
C                                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE QRDRY
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NODE(3),NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
C.....INITIALIZE QDRY
      DO 50 IN=1,NNODE
           DO 50 IZ=1,NLAY
   50 QDRY(IN,IZ)=0.D00
C.....MOVE STRESSES FROM DRY LAYERS TO LOWER LAYERS
      DO 100 IE=1,NELEM
           DO 100 IN3=1,3
C.........IDENTIFY NODE NUMBER ON TRIANGULAR ELEMENT
          NODE(1)=NEM(IE,1)
          NODE(2)=NEM(IE,2)
          NODE(3)=NEM(IE,3)
          DO 90 IZ=NLAY,2,-1
               IZM=IZ-1
               IF (NELFLG(IE,IZ).EQ.5) THEN
                 QDRY(NODE(IN3),IZM)=Q(NODE(IN3),IZ)+QDRY(NODE(IN3),IZ)
                 QDRY(NODE(IN3),IZ)=0.D00
               ELSE
```

```
                QDRY(NODE(IN3),IZ)=Q(NODE(IN3),IZ)+QDRY(NODE(IN3),IZ)
              ENDIF
C             WRITE (8,1000) IT,IZ,QDRY(NODE(IN3),IZ)
   90       CONTINUE
            QDRY(NODE(IN3),1)=Q(NODE(IN3),1)
  100 CONTINUE
C1000 FORMAT (' TIMESTEP=',I4,' LAYER',I4,' QDRY=',F10.4)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                            C
C     VARIABLE LISTING: SUBROUTINE FORMKH                                    C
C                                                                            C
C     NAME    TYPE        DESCRIPTION                                        C
C     ------  ---------   ------------------------------------------         C
C     AE      REAL*8      AREA OF TRIANGULAR ELEMENT [L2]                    C
C     ELMPRP  REAL*8      TOP OF ELEMENT (ELEMENTS,1,LAYERS); [L]            C
C     ELMPRP  REAL*8      BOTTOM OF ELEMENT (ELEMENTS,2,LAYERS); [L]         C
C     ELMPRP  REAL*8      HYDRAULIC CONDUCTIVITY IN X DIRECTION              C
C                         (ELEMENTS,3,LAYERS); [L/T]                         C
C     ELMPRP  REAL*8      HYDRAULIC CONDUCTIVITY IN Y DIRECTION              C
C                         (ELEMENTS,4,LAYERS); [L/T]                         C
C     HEAD    REAL*8      HEADS FROM OLD ITERATION (NODES,LAYERS,3); [L]C
C     IB      INTEGER*2   LOOP COUNTER FOR BANDWIDTH                         C
C     IE      INTEGER*2   LOOP COUNTER FOR ELEMENTS                          C
C     IN      INTEGER*2   LOOP COUNTER FOR NODES                             C
C     IZ      INTEGER*2   LOOP COUNTER FOR LAYERS                            C
C     J       INTEGER*2   INDEX FOR BANDED COLUMN POSITION IN MATRIX KH C
C     KH      REAL*8      HORIZONTAL FLOW TERMS (NODES,BANDED INDEX,         C
C                         LAYERS); [L2/T]                                    C
C     NBAND   INTEGER*2   BANDWIDTH                                          C
C     NELEM   INTEGER*2   TOTAL NUMBER OF ELEMENTS                           C
C     NEM     INTEGER*2   NODES ASSIGNED TO ELEMENT (ELEMENTS,(I,J,K))       C
C     NLAY    INTEGER*2   NUMBER OF LAYERS                                   C
C     NNODE   INTEGER*2   TOTAL NUMBER OF NODES                              C
C     NODCOR  REAL*8      X COORDINATES OF NODES (NODES,1); [L]              C
C     NODCOR  REAL*8      Y COORDINATES OF NODES (NODES,2); [L]              C
C     NODE    REAL*8      NODAL POSITION ON ELEMENT (3)                      C
C     NODFLG  INTEGER*2   NODAL FLAG FOR HEAD STATUS (NODES,LAYERS,2)        C
C                         SEE SUBROUTINE CHKHED FOR EXPLANATION OF           C
C                         FLAG NUMBERS                                       C
C     SX      REAL*8      SECOND X DERIVATIVE OF BASIS FUNCTION; [L3/T] C
C     SY      REAL*8      SECOND Y DERIVATIVE OF BASIS FUNCTION; [L3/T] C
C     TRANSX  REAL*8      TRANSMISSIVITY IN X DIRECTION; [L2/T]              C
C     TRANSY  REAL*8      TRANSMISSIVITY IN Y DIRECTION; [L2/T]              C
C                                                                            C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                            C
C  SUBROUTINE FORMKH DESCRIPTION:                                            C
C  FORMS MATRIX OF HORIZONTAL FLOW COMPONENTS (KH).  INTEGRATES OVER         C
C  ELEMENTS AND ASSIGNS TO MATRIX KH IN BANDED FORM.  TRANSMISSIVITY         C
C  CALCULATIONS BASED ON STATUS OF HEADS (CONFINED, UNCONFINED, ETC.)        C
C                                                                            C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                            C
C            SUBROUTINE FORMKH                                               C
C                                                                            C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE FORMKH
      IMPLICIT REAL*8 (A-H,O-Z)
```

```
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4SX(3),SY(3),NODE(3),NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
C.....INITIALIZE [KH]
      DO 20 IN=1,NNODE
      DO 20 IB=1,NBAND
   20 KH(IN,IB,IZ)=0.0
C.....BEGIN ELEMENT LOOP
      DO 40 IE=1,NELEM
C.........CALCULATE AREA OF TRIANGULAR ELEMENT
          AE=0.5*((NODCOR(NEM(IE,1),1)*NODCOR(NEM(IE,2),2)
     1       -NODCOR(NEM(IE,2),1)*NODCOR(NEM(IE,1),2))
     2       +(NODCOR(NEM(IE,3),1)*NODCOR(NEM(IE,1),2)
     3       -NODCOR(NEM(IE,1),1)*NODCOR(NEM(IE,3),2))
     4       +(NODCOR(NEM(IE,2),1)*NODCOR(NEM(IE,3),2)
     5       -NODCOR(NEM(IE,3),1)*NODCOR(NEM(IE,2),2)))
C.........FORM SPATIAL DERIVATIVES OF INTERPOLATION FUNCTIONS
          SX(1)=0.5*(NODCOR(NEM(IE,2),2)-NODCOR(NEM(IE,3),2))/AE
          SX(2)=0.5*(NODCOR(NEM(IE,3),2)-NODCOR(NEM(IE,1),2))/AE
          SX(3)=0.5*(NODCOR(NEM(IE,1),2)-NODCOR(NEM(IE,2),2))/AE
          SY(1)=0.5*(NODCOR(NEM(IE,3),1)-NODCOR(NEM(IE,2),1))/AE
          SY(2)=0.5*(NODCOR(NEM(IE,1),1)-NODCOR(NEM(IE,3),1))/AE
          SY(3)=0.5*(NODCOR(NEM(IE,2),1)-NODCOR(NEM(IE,1),1))/AE
C.........IDENTIFY NODE NUMBER ON TRIANGULAR ELEMENT
          NODE(1)=NEM(IE,1)
          NODE(2)=NEM(IE,2)
          NODE(3)=NEM(IE,3)

C.............CALCULATE TRANSMISSIVITIES, DEPENDING ON HEAD CONDITION

           HSUM = 0.D00
           DO 100 IN = 1,3
              HSUM = HSUM+HEAD(NODE(IN),IZ,3)
  100      CONTINUE
           HBAR = HSUM/3.D00
           IF(HBAR.GT.ELMPRP(IE,1,IZ)) THEN
              TOPAQ = ELMPRP(IE,1,IZ)
           ELSE
              TOPAQ = HBAR
           ENDIF
           IF(TOPAQ.GT.ELMPRP(IE,2,IZ)) THEN
              TRANSX = (TOPAQ-ELMPRP(IE,2,IZ))*ELMPRP(IE,3,IZ)
              TRANSY = (TOPAQ-ELMPRP(IE,2,IZ))*ELMPRP(IE,4,IZ)
           ELSE
              TRANSX = 1.D-10
              TRANSY = 1.D-10
           ENDIF
C.........BEGIN NODE LOOP FOR EACH ELEMENT
          DO 40 IN=1,3
C.............CALCULATE [KH]
C.............NODE I
              J=NEM(IE,1)+((NBAND+1)/2-NODE(IN))
              KH(NODE(IN),J,IZ)=KH(NODE(IN),J,IZ)+(AE*
```

```
     1        ((TRANSX*SX(1)*SX(IN))
     2        +(TRANSY*SY(1)*SY(IN))))
C............NODE J
              J=NEM(IE,2)+((NBAND+1)/2-NODE(IN))
              KH(NODE(IN),J,IZ)=KH(NODE(IN),J,IZ)+(AE*
     1        ((TRANSX*SX(2)*SX(IN))
     2        +(TRANSY*SY(2)*SY(IN))))
C............NODE K
              J=NEM(IE,3)+((NBAND+1)/2-NODE(IN))
              KH(NODE(IN),J,IZ)=KH(NODE(IN),J,IZ)+(AE*
     1        ((TRANSX*SX(3)*SX(IN))
     2        +(TRANSY*SY(3)*SY(IN))))
C             WRITE (8,1000) NODE(IN),IT,ITER,TRANSX,KH(NODE(IN),J,IZ)
   40 CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     VARIABLE LISTING: SUBROUTINE FORMST                              C
C                                                                      C
C     NAME    TYPE       DESCRIPTION                                   C
C     ------  ---------- --------------------------------------------  C
C     AE      REAL*8     AREA OF TRIANGULAR ELEMENT [L2]               C
C     CNST    REAL*8     CONSTANT RESULTING FROM INTEGRATION OF        C
C                        STORAGE TERMS, FUNCTION OF KRONECKER DELTA    C
C                        FUNCTION (3,3); [D]                           C
C     ELMPRP  REAL*8     STORATIVITY (ELEMENTS,6,LAYERS); [D]          C
C     ELMPRP  REAL*8     SPECIFIC YIELD (ELEMENTS,7,LAYERS); [L/T]     C
C     IE      INTEGER*2  LOOP COUNTER FOR ELEMENTS                     C
C     IN      INTEGER*2  LOOP COUNTER FOR NODES                        C
C     IPN     INTEGER*2  INDEX FOR TYPE OF STORAGE TERM                C
C     IT      INTEGER*2  LOOP COUNTER FOR TIME                         C
C     IZ      INTEGER*2  LOOP COUNTER FOR LAYERS                       C
C     I       INTEGER*2  INDEX FOR CNST MATRIX                         C
C     J       INTEGER*2  INDEX FOR CNST MATRIX                         C
C     NELEM   INTEGER*2  TOTAL NUMBER OF ELEMENTS                      C
C     NEM     INTEGER*2  NODES ASSIGNED TO ELEMENT (ELEMENTS,(I,J,K))  C
C     NLAY    INTEGER*2  NUMBER OF LAYERS                              C
C     NNODE   INTEGER*2  TOTAL NUMBER OF NODES                         C
C     NODCOR  REAL*8     X COORDINATES OF NODES (NODES,1); [L]         C
C     NODCOR  REAL*8     Y COORDINATES OF NODES (NODES,2); [L]         C
C     NODE    REAL*8     NODAL POSITION ON ELEMENT (3)                 C
C     NODFLG  INTEGER*2  NODAL FLAG FOR HEAD STATUS (NODES,LAYERS,2)   C
C                        SEE SUBROUTINE CHKHED FOR EXPLANATION OF      C
C                        FLAG NUMBERS                                  C
C     NSS     INTEGER*2  SWITCH FOR STEADY STATE CASE (IF 1, STEADY    C
C                        STATE)                                        C
C     NTIMST  INTEGER*2  TOTAL NUMBER OF TIMESTEPS                     C
C     NXON    INTEGER*2  SWITCH FOR ESTIMATING STORAGE FOR WATER       C
C                        BALANCE (ENSURES THAT STORAGES WILL BE        C
C                        CALCULATED FOR WATER BALANCE) IF 1, CALCULATE C
C     ST      REAL*8     STORAGE TERMS TO BE APPLIED TO NEW HEADS      C
C                        (NODES,LAYERS,1); [L2]                        C
C     ST      REAL*8     STORAGE TERMS TO BE APPLIED TO TOPS OF NODES  C
C                        (NODES,LAYERS,2); [L2]                        C
C     ST      REAL*8     STORAGE TERMS TO BE APPLIED TO OLD HEADS      C
C                        (NODES,LAYERS,3); [L2]                        C
C     STONEW  REAL*8     STORAGE COEFFICIENT TO BE APPLIED TO NEW      C
C                        HEADS, [D]                                    C
C     STOOLD  REAL*8     STORAGE COEFFICIENT TO BE APPLIED TO OLD      C
```

```
C                           HEADS, [D]                                      C
C     STOTOP  REAL*8        STORAGE COEFFICIENT TO BE APPLIED TO TOPS OF    C
C                           NODES, [D]                                      C
C     SX      REAL*8        SECOND X DERIVATIVE OF BASIS FUNCTION; [L3/T]   C
C     SY      REAL*8        SECOND Y DERIVATIVE OF BASIS FUNCTION; [L3/T]   C
C                                                                           C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                           C
C  SUBROUTINE FORMST DESCRIPTION:                                           C
C  FORMS MATRIX OF STORAGE COMPONENTS (ST).  INTEGRATES OVER ELEMENTS       C
C  ELEMENTS AND LUMP DIAGONALIZES ST.  STORAGE CALCULATIONS BASED ON        C
C  STATUS OF HEADS (CONFINED, UNCONFINED, ETC.).  THREE DIFFERENT           C
C  STORAGE TERMS ARE CALCULATED:                                            C
C       STORAGE TERMS APPLIED TO NEW HEADS -- (NODES,LAYERS,1)              C
C       STORAGE TERMS APPLIED TO TOPS OF NODES -- (NODES,LAYERS,2)          C
C       STORAGE TERMS APPLIED TO OLD HEADS -- (NODES,LAYERS,3)              C
C                                                                           C
C  THREE TERMS ARE NECESSARY TO ACCOUNT FOR TIMESTEPS WHERE HEAD            C
C  STATUS CHANGES FROM CONFINED TO UNCONFINED OR VICE VERSA.                C
C                                                                           C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                           C
C            SUBROUTINE FORMST                                              C
C                                                                           C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE FORMST(NXON)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4SX(3),SY(3),NODE(3),CNST(3,3),NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
C.....INITIALIZE [ST]
      DO 20 IN=1,NNODE
      DO 20 IPN=1,3
   20 ST(IN,IZ,IPN)=0.0
C.....CHECK FOR STEADY STATE CASE
      IF ( (NSS.EQ.0) .OR. (NXON.EQ.1) ) THEN
C.....BEGIN ELEMENT LOOP
      DO 40 IE=1,NELEM
C.........CALCULATE AREA OF TRIANGULAR ELEMENT
         AE=0.5*((NODCOR(NEM(IE,1),1)*NODCOR(NEM(IE,2),2)
     1       -NODCOR(NEM(IE,2),1)*NODCOR(NEM(IE,1),2))
     2       +(NODCOR(NEM(IE,3),1)*NODCOR(NEM(IE,1),2)
     3       -NODCOR(NEM(IE,1),1)*NODCOR(NEM(IE,3),2))
     4       +(NODCOR(NEM(IE,2),1)*NODCOR(NEM(IE,3),2)
     5       -NODCOR(NEM(IE,3),1)*NODCOR(NEM(IE,2),2)))
C.........IDENTIFY NODE NUMBER ON TRIANGULAR ELEMENT
         NODE(1)=NEM(IE,1)
         NODE(2)=NEM(IE,2)
         NODE(3)=NEM(IE,3)
C.........SET DELTA FUNCTION (TIMES SIX)
         DO 30 I=1,3
               DO 30 J=1,3
                     IF (I.EQ.J) THEN
```

```
                          CNST(I,J)=6.D00
                     ELSE
                          CNST(I,J)=12.D00
                     ENDIF
   30      CONTINUE
C.........BEGIN NODE LOOP FOR EACH ELEMENT
          DO 40 IN=1,3
C.............CALCULATE [ST] AS A LUMPED DIAGONALIZED MATRIX
C.............CALCULATE STORAGE TERMS BASED ON HEAD CONDITION
              IF (NELFLG(IE,IZ).EQ.1) THEN
              STONEW=ELMPRP(IE,6,IZ)
              STOTOP=0.D00
              STOOLD=ELMPRP(IE,6,IZ)
              ELSE IF (NELFLG(IE,IZ).EQ.2) THEN
              STONEW=ELMPRP(IE,6,IZ)
              STOTOP=ELMPRP(IE,6,IZ)-ELMPRP(IE,7,IZ)
              STOOLD=ELMPRP(IE,7,IZ)
              ELSE IF (NELFLG(IE,IZ).EQ.3) THEN
              STONEW=ELMPRP(IE,7,IZ)
              STOTOP=ELMPRP(IE,7,IZ)-ELMPRP(IE,6,IZ)
              STOOLD=ELMPRP(IE,6,IZ)
              ELSE IF (NELFLG(IE,IZ).EQ.4) THEN
              STONEW=ELMPRP(IE,7,IZ)
              STOTOP=0.D00
              STOOLD=ELMPRP(IE,7,IZ)
              ELSE IF (NELFLG(IE,IZ).EQ.5) THEN
              STONEW=1.0E-10
              STOTOP=0.D00
              STOOLD=1.0E-10
              ENDIF
              DO 40 IC=1,3
                   ST(NODE(IN),IZ,1)=ST(NODE(IN),IZ,1)
     1             +(STONEW*AE/CNST(IC,IN))
                   ST(NODE(IN),IZ,2)=ST(NODE(IN),IZ,2)
     1             +(STOTOP*AE/CNST(IC,IN))
                   ST(NODE(IN),IZ,3)=ST(NODE(IN),IZ,3)
     1             +(STOOLD*AE/CNST(IC,IN))
   40 CONTINUE
      ENDIF
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     VARIABLE LISTING: SUBROUTINE LHSPRD                              C
C                                                                      C
C     NAME    TYPE       DESCRIPTION                                   C
C     ------  ---------  --------------------------------------------  C
C     IN      INTEGER*2  LOOP COUNTER FOR NODES                        C
C     IZ      INTEGER*2  LOOP COUNTER FOR LAYERS                       C
C     KH      REAL*8     HORIZONTAL FLOW TERMS (NODES,BANDED INDEX,    C
C                        LAYERS); [L2/T]                               C
C     MID     INTEGER*2  INDEX FOR BANDED COLUMN MIDPOINT IN MATRIX KH C
C     NBAND   INTEGER*2  BANDWIDTH                                     C
C     NNODE   INTEGER*2  TOTAL NUMBER OF NODES                         C
C     ST      REAL*8     STORAGE TERMS TO BE APPLIED TO NEW HEADS      C
C                        (NODES,LAYERS,1); [L2]                        C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C  SUBROUTINE LHSPRD DESCRIPTION:                                      C
```

```
C  FORMS LEFT HAND SIDE MATRIX FOR PREDICTOR EQUATIONS.  ADDS ST        C
C  MATRIX TO KH MATRIX, LEAVING A NEW KH MATRIX.                        C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C             SUBROUTINE LHSPRD                                         C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE LHSPRD
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
C.....BEGIN NODE LOOP
      DO 40 IN=1,NNODE
            MID=(NBAND+1)/2
            KH(IN,MID,IZ)=KH(IN,MID,IZ)+(ST(IN,IZ,1)/DELTIM)
   40 CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     VARIABLE LISTING: SUBROUTINE LHSDIR                               C
C                                                                       C
C     NAME    TYPE        DESCRIPTION                                   C
C     ------  ---------   ---------------------------------------------C
C     IB      INTEGER*2   LOOP COUNTER FOR BANDWIDTH                    C
C     IN      INTEGER*2   LOOP COUNTER FOR NODES                        C
C     IZ      INTEGER*2   LOOP COUNTER FOR LAYERS                       C
C     KH      REAL*8      HORIZONTAL FLOW TERMS (NODES,BANDED INDEX,    C
C                         LAYERS); [L2/T]                               C
C     MID     INTEGER*2   INDEX FOR BANDED COLUMN MIDPOINT IN MATRIX KH C
C     NBAND   INTEGER*2   BANDWIDTH                                     C
C     NNODE   INTEGER*2   TOTAL NUMBER OF NODES                         C
C     NODFLG  INTEGER*2   NODAL FLAG FOR BOUNDARY CONDITION: IF 1,      C
C                         CONSTANT HEAD BOUNDARY (NODES,LAYERS,1)       C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C  SUBROUTINE LHSDIR DESCRIPTION:                                       C
C  INCORPORATES CONSTANT HEAD BOUNDARIES INTO KH MATRIX. AT NODES       C
C  WHERE CONSTANT HEADS ARE SPECIFIED, ROW IN KH IS ASSIGNED ALL        C
C  0'S, EXCEPT AT NODE LOCATION, WHERE A 1 IS ASSIGNED.                 C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C             SUBROUTINE LHSDIR                                         C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE LHSDIR
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
```

```
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
      DO 40 IN=1,NNODE
           IF ( NODFLG(IN,IZ,1).EQ.1 ) THEN
                DO 20 IB=1,NBAND
   20                KH(IN,IB,IZ)=0.D00
                MID=(NBAND+1)/2
                KH(IN,MID,IZ)=1.D00
           ENDIF
   40 CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                              C
C     VARIABLE LISTING: SUBROUTINE LHSDRY                                      C
C                                                                              C
C     NAME    TYPE        DESCRIPTION                                          C
C     ------  ---------   ---------------------------------------------        C
C     IB      INTEGER*2   LOOP COUNTER FOR BANDWIDTH                           C
C     IN      INTEGER*2   LOOP COUNTER FOR NODES                               C
C     IZ      INTEGER*2   LOOP COUNTER FOR LAYERS                              C
C     KH      REAL*8      HORIZONTAL FLOW TERMS (NODES,BANDED INDEX,           C
C                         LAYERS); [L2/T]                                      C
C     MID     INTEGER*2   INDEX FOR BANDED COLUMN MIDPOINT IN MATRIX KH        C
C     NBAND   INTEGER*2   BANDWIDTH                                            C
C     NNODE   INTEGER*2   TOTAL NUMBER OF NODES                                C
C     NODFLG  INTEGER*2   NODAL FLAG FOR HEAD STATUS (NODES,LAYERS,2)          C
C                         SEE SUBROUTINE CHKHED FOR EXPLANATION OF             C
C                         FLAG NUMBERS                                         C
C                                                                              C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                              C
C  SUBROUTINE LHSDRY DESCRIPTION:                                              C
C  INCORPORATES DRY NODES INTO KH MATRIX. AT NODES                             C
C  WHERE NODES ARE DRY, ROW IN KH IS ASSIGNED ALL                              C
C  0'S, EXCEPT AT NODE LOCATION, WHERE A 1 IS ASSIGNED.                        C
C                                                                              C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                              C
C            SUBROUTINE LHSDRY                                                 C
C                                                                              C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE LHSDRY
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NODE(3),NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
```

```
      DO 40 IE=1,NELEM
      DO 40 IN3=1,3
C.........IDENTIFY NODE NUMBER ON TRIANGULAR ELEMENT
          NODE(1)=NEM(IE,1)
          NODE(2)=NEM(IE,2)
          NODE(3)=NEM(IE,3)
           IF ( NELFLG(IE,IZ).EQ.5 ) THEN
                DO 20 IB=1,NBAND
   20                 KH(NODE(IN3),IB,IZ)=0.D00
                MID=(NBAND+1)/2
                KH(NODE(IN3),MID,IZ)=1.D00
           ENDIF
   40 CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     VARIABLE LISTING: SUBROUTINE FACTOR                               C
C                                                                       C
C     NAME    TYPE       DESCRIPTION                                    C
C     ------  ---------  ---------------------------------------------  C
C     I       INTEGER*2  LOOP COUNTER FOR COLUMNS                       C
C     IZ      INTEGER*2  LOOP COUNTER FOR LAYERS                        C
C     J       INTEGER*2  LOOP COUNTER FOR ROWS                          C
C     KH      REAL*8     HORIZONTAL FLOW TERMS (NODES,BANDED INDEX,     C
C                        LAYERS); [L2/T]                                C
C     MID     INTEGER*2  INDEX FOR BANDED COLUMN MIDPOINT IN MATRIX KH C
C     NBAND   INTEGER*2  BANDWIDTH                                      C
C     NDIAG   INTEGER*2  COLUMN INDEX                                   C
C     NDO     INTEGER*2  ROW INDEX                                      C
C     NEQN    INTEGER*2  ROW INDEX                                      C
C     NHIGH   INTEGER*2  COLUMN INDEX                                   C
C     NITER   INTEGER*2  COLUMN INDEX                                   C
C     NLOW    INTEGER*2  COLUMN INDEX                                   C
C     NMAX    INTEGER*2  COLUMN INDEX                                   C
C     NNODE   INTEGER*2  TOTAL NUMBER OF NODES                          C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C  SUBROUTINE FACTOR DESCRIPTION:                                       C
C  FACTORS KH MATRIX INTO A LOWER DECOMPOOSED MATRIX FOR USE IN A       C
C  GAUSS ELIMINATION SOLUTION ALGORITHM.  FULL KH MATRIX IS NOT SAVED. C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C             SUBROUTINE FACTOR                                         C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE FACTOR
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
```

```
      NDIAG = NBAND/2 + 1
      NMAX = NBAND - NDIAG
      NDO = NNODE - 1
      DO 10 I = 1,NDO
      NITER = NMAX
      IF( NNODE - I .LT. NITER ) NITER = NNODE - I
      DO 10 J = 1,NITER
      NEQN = I + J
      KH(NEQN,NDIAG - J,IZ)=-KH(NEQN,NDIAG - J,IZ)/
     1KH(I,NDIAG,IZ)
      NLOW = NDIAG - J + 1
      NHIGH = NLOW + NITER - 1
      DO 10 K = NLOW,NHIGH
   10 KH(NEQN,K,IZ)=KH(NEQN,K,IZ)+KH(NEQN,NDIAG-J,IZ)
     1*KH(I,NDIAG+1-NLOW+K,IZ)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     VARIABLE LISTING: SUBROUTINE FORMKV                              C
C                                                                      C
C     NAME    TYPE       DESCRIPTION                                   C
C     ------  ---------- --------------------------------------------  C
C     AE      REAL*8     AREA OF TRIANGULAR ELEMENT [L2]               C
C     CNST    REAL*8     CONSTANT RESULTING FROM INTEGRATION OF        C
C                        STORAGE TERMS, FUNCTION OF KRONECKER DELTA    C
C                        FUNCTION (3,3); [D]                           C
C     DELZ    REAL*8     CENTRAL DIFFERENCE AROUND LAYER IZ; [L]       C
C                        (SATURATED THICKNESS)                         C
C     DELZM   REAL*8     CENTRAL DIFFERENCE AROUND LAYER IZ-1; [L]     C
C                        (SATURATED THICKNESS)                         C
C     DELZP   REAL*8     CENTRAL DIFFERENCE AROUND LAYER IZ+1; [L]     C
C                        (SATURATED THICKNESS)                         C
C     DELZMT  REAL*8     SATURATED THICKNESS IN LAYER IZ-1; [L]        C
C     DELZPT  REAL*8     SATURATED THICKNESS IN LAYER IZ+1; [L]        C
C     DELZT   REAL*8     SATURATED THICKNESS IN LAYER IZ; [L]          C
C     ELMPRP  REAL*8     TOP OF ELEMENT (ELEMENTS,1,LAYERS); [L]       C
C     ELMPRP  REAL*8     BOTTOM OF ELEMENT (ELEMENTS,2,LAYERS); [L]    C
C     ELMPRP  REAL*8     HYDRAULIC CONDUCTIVITY IN X DIRECTION         C
C                        (ELEMENTS,5,LAYERS); [L/T]                    C
C     HEAD    REAL*8     HEADS FROM OLD TIMESTEP (NODES,LAYERS,1); [L] C
C     HEAD    REAL*8     HEADS FROM NEW ITERATION (NODES,LAYERS,2); [L]C
C     I       INTEGER*2  INDEX FOR CNST MATRIX                         C
C     IP      INTEGER*2  INDEX FOR KU, KU+KL, OR KL                    C
C     IE      INTEGER*2  LOOP COUNTER FOR ELEMENTS                     C
C     IN      INTEGER*2  LOOP COUNTER FOR NODES                        C
C     IXF     INTEGER*2  INDEX FOR PREDICTOR (=1) OR CORRECTOR (=2)    C
C                        SET OF EQUATIONS                              C
C     IXH     INTEGER*2  INDEX FOR HEADS                               C
C     IZ      INTEGER*2  LOOP COUNTER FOR LAYERS                       C
C     IZM     INTEGER*2  LOOP COUNTER FOR LAYERS (IZ-1)                C
C     IZP     INTEGER*2  LOOP COUNTER FOR LAYERS (IZ+1)                C
C     J       INTEGER*2  INDEX FOR CNST MATRIX                         C
C     KV      REAL*8     UPPER SET OF VERTICAL FLOW TERMS FOR USE IN   C
C                        PREDICTOR EQUATIONS (NODES,LAYERS,1,1); [L2/T]C
C     KV      REAL*8     UPPER SET OF VERTICAL FLOW TERMS FOR USE IN   C
C                        CORRECTOR EQUATIONS (NODES,LAYERS,1,2); [L2/T]C
C     KV      REAL*8     UPPER+LOWER SET OF VERTICAL FLOW TERMS FOR    C
C                        USE IN PREDICTOR EQUATIONS (NODES,LAYERS,2,1);C
C                        [L2/T]                                        C
```

```
C     KV       REAL*8      UPPER+LOWER SET OF VERTICAL FLOW TERMS FOR    C
C                          USE IN CORRECTOR EQUATIONS (NODES,LAYERS,2,2);C
C                          [L2/T]                                        C
C     KV       REAL*8      LOWER SET OF VERTICAL FLOW TERMS FOR USE IN   C
C                          PREDICTOR EQUATIONS (NODES,LAYERS,3,1); [L2/T]C
C     KV       REAL*8      LOWER SET OF VERTICAL FLOW TERMS FOR USE IN   C
C                          CORRECTOR EQUATIONS (NODES,LAYERS,3,2); [L2/T]C
C     NBAND    INTEGER*2   BANDWIDTH                                     C
C     NELEM    INTEGER*2   TOTAL NUMBER OF ELEMENTS                      C
C     NEM      INTEGER*2   NODES ASSIGNED TO ELEMENT (ELEMENTS,(I,J,K))  C
C     NLAY     INTEGER*2   NUMBER OF LAYERS                              C
C     NNODE    INTEGER*2   TOTAL NUMBER OF NODES                         C
C     NODCOR   REAL*8      X COORDINATES OF NODES (NODES,1); [L]         C
C     NODCOR   REAL*8      Y COORDINATES OF NODES (NODES,2); [L]         C
C     NODE     REAL*8      NODAL POSITION ON ELEMENT (3)                 C
C     NODFLG   INTEGER*2   NODAL FLAG FOR HEAD STATUS (NODES,LAYERS,2)   C
C                          SEE SUBROUTINE CHKHED FOR EXPLANATION OF      C
C                          FLAG NUMBERS                                  C
C     SX       REAL*8      SECOND X DERIVATIVE OF BASIS FUNCTION; [L3/T] C
C     SY       REAL*8      SECOND Y DERIVATIVE OF BASIS FUNCTION; [L3/T] C
C     TRANSZM  REAL*8      UPPER HARMONIC AVERAGE OF TRANSMISSIVITY      C
C                          IN Z DIRECTION; [L2/T]                        C
C     TRANSZP  REAL*8      LOWER HARMONIC AVERAGE OF TRANSMISSIVITY      C
C                          IN Z DIRECTION; [L2/T]                        C
C                                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                        C
C  SUBROUTINE FORMKV DESCRIPTION:                                        C
C  CALCULATES VERTICAL FLOW COMPONENTS. ESTIMATES TRANSMISSIVITIES       C
C  ACCORDING TO HEAD STATUS, USING A BLOCK-CENTERED FINITE DIFFERENCE    C
C  APPROXIMATION.  TRANSMISSIVITY ESTIMATES ALSO BASED ON POSITON        C
C  OF LAYER.  FLOW COMPONENTS CALCULATED BY INTEGRATING OVER ELEMENTS    C
C  ELEMENTS.  THREE DIFFERENT VERTICAL FLOW COMPONENTS ARE CALCULATED:   C
C     UPPER SET OF VERTICAL FLOW TERMS (NODES,LAYERS,1,IXF)              C
C     UPPER+LOWER SET OF VERTICAL FLOW TERMS (NODES,LAYERS,2,IXF)        C
C     LOWER SET OF VERTICAL FLOW TERMS  (NODES,LAYERS,3,IXF)             C
C                                                                        C
C     INDEX IXF = 1 -- KV FOR PREDICTOR EQUATIONS                        C
C     INDEX IXF = 2 -- KV FOR PREDICTOR EQUATIONS                        C
C                                                                        C
C  INDEX IXH ALLOWS FOR DIFFERENT SET OF HEADS TO BE USED IN             C
C  SUBROUTINE:                                                           C
C     INDEX IXH = 1 -- HEADS FROM OLD TIMESTEP                           C
C     INDEX IXH = 3 -- HEADS FROM NEW ITERATION                          C
C                                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                        C
C            SUBROUTINE FORMKV                                           C
C                                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE FORMKV(IXH,IXF)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NODE(3),CNST(3,3),NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
```

```
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
C.....INITIALIZE [KV]
      DO 20 IN=1,NNODE
           DO 20 IP=1,3
   20 KV(IN,IZ,IP,IXF)=0.0
C.....CHECK FOR SINGLE LAYER
      IF (NLAY.NE.1) THEN
C..........BEGIN ELEMENT LOOP
           DO 40 IE=1,NELEM
C.........CALCULATE AREA OF TRIANGULAR ELEMENT
          AE=0.5*((NODCOR(NEM(IE,1),1)*NODCOR(NEM(IE,2),2)
     1       -NODCOR(NEM(IE,2),1)*NODCOR(NEM(IE,1),2))
     2       +(NODCOR(NEM(IE,3),1)*NODCOR(NEM(IE,1),2)
     3       -NODCOR(NEM(IE,1),1)*NODCOR(NEM(IE,3),2))
     4       +(NODCOR(NEM(IE,2),1)*NODCOR(NEM(IE,3),2)
     5       -NODCOR(NEM(IE,3),1)*NODCOR(NEM(IE,2),2)))
C.........IDENTIFY NODE NUMBER ON TRIANGULAR ELEMENT
          NODE(1)=NEM(IE,1)
          NODE(2)=NEM(IE,2)
          NODE(3)=NEM(IE,3)
C.........
C.........CALCULATE TRANSMISSIVITIES AND DELTA Z'S IN
C.........Z-DIRECTION USING BLOCK-CENTERED APPROACH,
C.........ACCOUNTING FOR BOUNDARY CONDITIONS IN Z-DIRECTION,
C.........AND ACCOUNTING FOR SATURATED AND UNSATURATED
C.........CONDITIONS
C.........
C.........BOTTOM LAYER
C.........
          IF (IZ.EQ.1) THEN
          IZP=IZ+1
          IZM=IZ
C.........AVERAGE HEADS OVER EACH ELEMENT
          HSUM = 0.D00
          DO 100 IN=1,3
               HSUMP = HSUMP+HEAD(NODE(IN),IZP,IXH)
               HSUM = HSUM+HEAD(NODE(IN),IZ,IXH)
               HSUMM = HSUMM+HEAD(NODE(IN),IZM,IXH)
  100     CONTINUE
          HBARP = HSUMP/3.D00
          HBAR = HSUM/3.D00
          HBARM = HSUMM/3.D00
C.........CHECK SATURATED/UNSATURATED CONDITION
          IF (HBAR.GT.ELMPRP(IE,1,IZ)) THEN
               DELZP=(ELMPRP(IE,1,IZP)-ELMPRP(IE,2,IZ))/2
               DELZ=ELMPRP(IE,1,IZ)-ELMPRP(IE,2,IZ)
               DELZM=DELZ
          ELSEIF ( (HBAR.LT.ELMPRP(IE,1,IZ)) .AND.
     1              (HBAR.GT.ELMPRP(IE,2,IZ)) ) THEN
               DELZP=1.0E-08
               DELZ=HBAR-ELMPRP(IE,2,IZ)
               DELZM=DELZ
          ELSEIF (HBAR.LT.ELMPRP(IE,2,IZ)) THEN
               DELZP=(ELMPRP(IE,1,IZP)-ELMPRP(IE,2,IZ))/2
               DELZ=ELMPRP(IE,1,IZ)-ELMPRP(IE,2,IZ)
               DELZM=DELZ
          ENDIF
C.........CALCULATE CONDUCTIVITIES
          HCONZP=((DELZ/2)+(DELZP/2))/
```

```
     1            ((DELZ/2/(ELMPRP(IE,5,IZ)))
     2           +(DELZP/2/(ELMPRP(IE,5,IZP))))
          HCONZM=0.0
C.........
C.........TOP LAYER
C.........
          ELSEIF (IZ.EQ.NLAY) THEN
          IZP=IZ
          IZM=IZ-1
C.........AVERAGE HEADS OVER EACH ELEMENT
          HSUM = 0.D00
          DO 200 IN=1,3
               HSUMP = HSUMP+HEAD(NODE(IN),IZP,IXH)
               HSUM = HSUM+HEAD(NODE(IN),IZ,IXH)
               HSUMM = HSUMM+HEAD(NODE(IN),IZM,IXH)
  200     CONTINUE
          HBARP = HSUMP/3.D00
          HBAR = HSUM/3.D00
          HBARM = HSUMM/3.D00
C.........CHECK SATURATED/UNSATURATED CONDITION
          IF (HBAR.GT.ELMPRP(IE,1,IZ)) THEN
               DELZ=ELMPRP(IE,1,IZ)-ELMPRP(IE,2,IZ)
               DELZP=DELZ
               DELZM=(ELMPRP(IE,1,IZ)-ELMPRP(IE,2,IZM))/2
          ELSEIF ( (HBAR.LT.ELMPRP(IE,1,IZ)) .AND.
     1              (HBAR.GT.ELMPRP(IE,2,IZ)) ) THEN
               DELZ=HBAR-ELMPRP(IE,2,IZ)
               DELZP=DELZ
               DELZM=(HBAR-ELMPRP(IE,2,IZM))/2
          ELSEIF (HBAR.LT.ELMPRP(IE,2,IZ)) THEN
               DELZ=ELMPRP(IE,1,IZ)-ELMPRP(IE,2,IZ)
               DELZP=DELZ
               DELZM=(ELMPRP(IE,1,IZ)-ELMPRP(IE,2,IZM))/2
          ENDIF
C.........CALCULATE CONDUCTIVITIES
          HCONZP=0.0
          HCONZM=((DELZ/2)+(DELZM/2))/
     1            ((DELZ/2/(ELMPRP(IE,5,IZ)))
     2           +(DELZM/2/(ELMPRP(IE,5,IZM))))
          ELSE
C.........
C.........OTHER LAYERS
C.........
          IZP=IZ+1
          IZM=IZ-1
C.........AVERAGE HEADS OVER EACH ELEMENT
          HSUM = 0.D00
          DO 300 IN=1,3
               HSUMP = HSUMP+HEAD(NODE(IN),IZP,IXH)
               HSUM = HSUM+HEAD(NODE(IN),IZ,IXH)
               HSUMM = HSUMM+HEAD(NODE(IN),IZM,IXH)
  300     CONTINUE
          HBARP = HSUMP/3.D00
          HBAR = HSUM/3.D00
          HBARM = HSUMM/3.D00
C.........CHECK SATURATED/UNSATURATED CONDITION
          IF (HBAR.GT.ELMPRP(IE,1,IZ)) THEN
               DELZP=(ELMPRP(IE,1,IZP)-ELMPRP(IE,2,IZ))/2
               DELZ=ELMPRP(IE,1,IZ)-ELMPRP(IE,2,IZ)
               DELZM=(ELMPRP(IE,1,IZ)-ELMPRP(IE,2,IZM))/2
```

```
          ELSEIF ( (HBAR.LT.ELMPRP(IE,1,IZ)) .AND.
     1             (HBAR.GT.ELMPRP(IE,2,IZ)) ) THEN
               DELZP=1.0E-08
               DELZ=HBAR-ELMPRP(IE,2,IZ)
               DELZM=(HBAR-ELMPRP(IE,2,IZM))/2
          ELSEIF (HBAR.LT.ELMPRP(IE,2,IZ)) THEN
               DELZP=(ELMPRP(IE,1,IZP)-ELMPRP(IE,2,IZ))/2
               DELZ=ELMPRP(IE,1,IZ)-ELMPRP(IE,2,IZ)
               DELZM=(ELMPRP(IE,1,IZ)-ELMPRP(IE,2,IZM))/2
          ENDIF
C.........CALCULATE CONDUCTIVITIES
          HCONZP=((DELZ/2)+(DELZP/2))/
     1           ((DELZ/2/(ELMPRP(IE,5,IZ)))
     2          +(DELZP/2/(ELMPRP(IE,5,IZP))))
          HCONZM=((DELZ/2)+(DELZM/2))/
     1           ((DELZ/2/(ELMPRP(IE,5,IZ)))
     2          +(DELZM/2/(ELMPRP(IE,5,IZM))))
          ENDIF
C.........BEGIN NODE LOOP FOR EACH ELEMENT
          DO 40 IN=1,3
C.............CALCULATE [KU],[KU]+[KL],[KL] AS LUMPED DIAGONALIZED
C.............MATRICES
              DO 40 IC=1,3
C.............--KU--
              KV(NODE(IN),IZ,1,IXF)=KV(NODE(IN),IZ,1,IXF)
     1                +((HCONZP/(DELZ))*AE/3.D00)
C.............--KU+KL--
              KV(NODE(IN),IZ,2,IXF)=KV(NODE(IN),IZ,2,IXF)
     1                +((HCONZP/(DELZ))*AE/3.D00)
     2                +((HCONZM/(DELZ))*AE/3.D00)
C.............--KL--
   40         KV(NODE(IN),IZ,3,IXF)=KV(NODE(IN),IZ,3,IXF)
     1                +((HCONZM/(DELZ))*AE/3.D00)
      ENDIF
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C  SUBROUTINE RHSPRD DESCRIPTION:                                      C
C  FORMS RIGHT HAND SIDE MATRIX FOR PREDICTOR EQUATIONS.  FIRST       C
C  MULTIPLIES KV TIMES APPROPRIATE HEADS TO FORM RHS2 MATRIX.  THIS    C
C  MATRIX IS CALCULATED SEPARATELY BECAUSE IT WILL BE USED AGAIN IN    C
C  CORRECTOR EQUATIONS.  SECOND, MULTIPLIES ST TIMES APPROPRIATE       C
C  HEADS AND ADDS NODAL STRESSES (PUMPING AND RECHARGE).  IF DRY       C
C  NODES ARE PRESENT, USE QDRY INSTEAD OF Q.                           C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     VARIABLE LISTING: SUBROUTINE RHSPRD                              C
C                                                                      C
C     NAME    TYPE       DESCRIPTION                                   C
C     ------  ---------  --------------------------------------------- C
C     HEAD    REAL*8     HEADS FROM OLD TIMESTEP (NODES,LAYERS,1); [L] C
C     IN      INTEGER*2  LOOP COUNTER FOR NODES                        C
C     IZ      INTEGER*2  LOOP COUNTER FOR LAYERS                       C
C     IZM     INTEGER*2  LOOP COUNTER FOR LAYERS (IZ-1)                C
C     IZP     INTEGER*2  LOOP COUNTER FOR LAYERS (IZ+1)                C
C     NDRY    INTEGER*2  FLAG INDICATING DRY NODES PRESENT             C
C     NLAY    INTEGER*2  NUMBER OF LAYERS                              C
C     NNODE   INTEGER*2  TOTAL NUMBER OF NODES                         C
```

```
C     NODFLG  INTEGER*2  NODAL FLAG FOR HEAD STATUS (NODES,LAYERS,2)   C
C                        SEE SUBROUTINE CHKHED FOR EXPLANATION OF      C
C                        FLAG NUMBERS                                  C
C     NZM     INTEGER*2  CONSTANTS FOR ZEROING OUT TERMS IN RHS2 IF    C
C                        NECESSARY                                     C
C     NZP     INTEGER*2  CONSTANTS FOR ZEROING OUT TERMS IN RHS2 IF    C
C                        NECESSARY                                     C
C     Q       REAL*8     NODAL STRESS (NODES,LAYERS); [L3/T]           C
C     QDRY    REAL*8     NODAL STRESS ADJUSTED FOR DRY LAYERS          C
C                        (NODES,LAYERS); [L3/T]                        C
C     RHS     REAL*8     RIGHT HAND SIDE FOR PREDICTOR EQUATIONS       C
C                        (NODES,LAYERS); [L3/T]                        C
C     RHS2    REAL*8     VERTICAL FLOW*HEAD TERMS ON RIGHT HAND SIDE;  C
C                        (NODES,LAYERS); [L3/T]                        C
C     ST      REAL*8     STORAGE TERMS TO BE APPLIED TO TOPS OF NODES  C
C                        (NODES,LAYERS,2); [L2]                        C
C     ST      REAL*8     STORAGE TERMS TO BE APPLIED TO OLD HEADS      C
C                        (NODES,LAYERS,3); [L2]                        C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C            SUBROUTINE RHSPRD                                         C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE RHSPRD
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NODE(3),NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
C.....CHECK FOR LAYER POSITION
C.....TOP LAYER
      IF (IZ.EQ.NLAY) THEN
          IZM=IZ-1
          IZP=IZ
          NZM=1
          NZP=0
C.....BOTTOM LAYER
      ELSEIF (IZ.EQ.1) THEN
          IZM=IZ
          IZP=IZ+1
          NZM=0
          NZP=1
C.....LAYERS OTHER THAN TOP OR BOTTOM
      ELSE
          IZM=IZ-1
          IZP=IZ+1
          NZM=1
          NZP=1
      ENDIF
C.....CALCULATE RIGHT HAND SIDE FOR PREDCITOR EQUATIONS
      DO 40 IN=1,NNODE
C..........SINGLE LAYER
           IF (NLAY.EQ.1) THEN
```

```
                RHS2(IN,IZ)=0.0
C..........OTHER THAN SINGLE LAYER
           ELSE
                RHS2(IN,IZ)=(NZP*KV(IN,IZ,1,1)*HEAD(IN,IZP,1))
     1                     -(KV(IN,IZ,2,1)*HEAD(IN,IZ,1))
     2                     +(NZM*KV(IN,IZ,3,1)*HEAD(IN,IZM,1))
           ENDIF
C..........CHECK FOR DRY NODES
           IF (NDRY.GT.0) THEN
                RHS(IN,IZ)=((ST(IN,IZ,3)/DELTIM)*HEAD(IN,IZ,1))
     1                    +((ST(IN,IZ,2)/DELTIM)*TOP(IN,IZ))
     2                    +RHS2(IN,IZ)+QDRY(IN,IZ)
           ELSEIF (NDRY.EQ.0) THEN
                RHS(IN,IZ)=((ST(IN,IZ,3)/DELTIM)*HEAD(IN,IZ,1))
     1                    +((ST(IN,IZ,2)/DELTIM)*TOP(IN,IZ))
     2                    +RHS2(IN,IZ)+Q(IN,IZ)-RECHGN(IN,IZ)
           ENDIF
   40 CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     VARIABLE LISTING: SUBROUTINE RHSDIR                              C
C                                                                      C
C     NAME    TYPE       DESCRIPTION                                   C
C     ------  ---------  --------------------------------------------  C
C     HEAD    REAL*8     HEADS FROM OLD TIMESTEP (NODES,LAYERS,1); [L] C
C     IN      INTEGER*2  LOOP COUNTER FOR NODES                        C
C     IZ      INTEGER*2  LOOP COUNTER FOR LAYERS                       C
C     IZM     INTEGER*2  LOOP COUNTER FOR LAYERS (IZ-1)                C
C     NNODE   INTEGER*2  TOTAL NUMBER OF NODES                         C
C     NODFLG  INTEGER*2  NODAL FLAG FOR HEAD STATUS (NODES,LAYERS,2)   C
C                        SEE SUBROUTINE CHKHED FOR EXPLANATION OF      C
C                        FLAG NUMBERS                                  C
C     RHS     REAL*8     RIGHT HAND SIDE FOR PREDICTOR EQUATIONS       C
C                        (NODES,LAYERS); [L3/T]                        C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C  SUBROUTINE RHSDIR DESCRIPTION:                                      C
C  INCORPORATES CONSTANT HEAD BOUNDARIES INTO RHS VECTOR. AT NODES     C
C  WHERE CONSTANT HEADS ARE SPECIFIED, SET RHS EQUAL TO HEADS FROM     C
C  OLD TIMESTEP.                                                       C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C            SUBROUTINE RHSDIR                                         C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE RHSDIR
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
```

```
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
      DO 20 IN=1,NNODE
           IF ( NODFLG(IN,IZ,1).EQ.1 ) THEN
                RHS(IN,IZ)=HEAD(IN,IZ,1)
           ENDIF
   20 CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                                C
C     VARIABLE LISTING: SUBROUTINE RHSDRY                                        C
C                                                                                C
C     NAME    TYPE        DESCRIPTION                                            C
C     ------  ---------   ------------------------------------------------       C
C     HEAD    REAL*8      HEADS FROM OLD ITERATION (NODES,LAYERS,3); [L]C
C     IN      INTEGER*2   LOOP COUNTER FOR NODES                                 C
C     IZ      INTEGER*2   LOOP COUNTER FOR LAYERS                                C
C     NNODE   INTEGER*2   TOTAL NUMBER OF NODES                                  C
C     NODFLG  INTEGER*2   NODAL FLAG FOR HEAD STATUS (NODES,LAYERS,2)            C
C                         SEE SUBROUTINE CHKHED FOR EXPLANATION OF               C
C                         FLAG NUMBERS                                           C
C     RHS     REAL*8      RIGHT HAND SIDE FOR PREDICTOR EQUATIONS                C
C                         (NODES,LAYERS); [L3/T]                                 C
C                                                                                C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                                C
C  SUBROUTINE RHSDRY DESCRIPTION:                                                C
C  INCORPORATES DRY NODES INTO RHS VECTOR AT NODES                               C
C  WHERE DRY NODES ARE SPECIFIED, SET RHS EQUAL TO HEADS FROM                    C
C  OLD TIMESTEP AND LAYER BELOW.                                                 C
C                                                                                C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                                C
C            SUBROUTINE RHSDRY                                                   C
C                                                                                C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE RHSDRY
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NODE(3),NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
      DO 20 IE=1,NELEM
           IF ( (IZ.EQ.1) .AND. (NODFLG(IE,IZ,2).EQ.5) ) THEN
                WRITE (6,1000)
           ENDIF
           IF ( (IZ.GT.1) .AND. (NODFLG(IE,IZ,2).EQ.5) ) THEN
                IZM=IZ-1
                DO 10 IN3=1,3
C.....................IDENTIFY NODE NUMBER ON TRIANGULAR ELEMENT
                     NODE(1)=NEM(IE,1)
                     NODE(2)=NEM(IE,2)
                     NODE(3)=NEM(IE,3)
                     RHS(NODE(IN3),IZ)=HEAD(NODE(IN3),IZM,1)
```

```
   10          CONTINUE
            ENDIF
   20 CONTINUE
 1000 FORMAT (' BOTTOM NODE IN AQUIFER GOES DRY')
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     VARIABLE LISTING: SUBROUTINE SOLVE                                C
C                                                                       C
C     NAME    TYPE        DESCRIPTION                                   C
C     ------  ----------  --------------------------------------------  C
C     HEAD    REAL*8      HEADS FROM NEW TIMESTEP (NODES,LAYERS,2); [L] C
C     I       INTEGER*2   LOOP COUNTER FOR COLUMNS                      C
C     IZ      INTEGER*2   LOOP COUNTER FOR LAYERS                       C
C     J       INTEGER*2   LOOP COUNTER FOR ROWS                         C
C     KH      REAL*8      HORIZONTAL FLOW TERMS (NODES,BANDED INDEX,    C
C                         LAYERS); [L2/T]                               C
C     NBAND   INTEGER*2   BANDWIDTH                                     C
C     NDIAG   INTEGER*2   COLUMN INDEX                                  C
C     NEQN    INTEGER*2   ROW INDEX                                     C
C     NITER   INTEGER*2   COLUMN INDEX                                  C
C     NMAX    INTEGER*2   COLUMN INDEX                                  C
C     NNODE   INTEGER*2   TOTAL NUMBER OF NODES                         C
C     NODE    INTEGER*2   ROW INDEX                                     C
C     RHS     REAL*8      RIGHT HAND SIDE FOR PREDICTOR EQUATIONS       C
C                         (NODES,LAYERS); [L3/T]                        C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C  SUBROUTINE SOLVE DESCRIPTION:                                        C
C  SOLVES FOR HEADS FROM PREDICTOR EQUATIONS USING BACKWARD              C
C  SUBSTITUTION (GAUSS ELIMINATION).  USES BANDED FORM OF MATRIX.       C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C             SUBROUTINE SOLVE                                          C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE SOLVE
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
      NDIAG = NBAND/2 + 1
      NMAX=NBAND-NDIAG
      NODE = NNODE - 1
      DO 10 I = 1,NODE
      NITER=NMAX
      IF(NITER.GT. NNODE - I) NITER = NNODE - I
      DO 10 J = 1,NITER
   10 RHS(I+J,IZ) = RHS(I+J,IZ) + KH(I+J,NDIAG-J,IZ)*RHS(I,IZ)
      HEAD(NNODE,IZ,2) = RHS(NNODE,IZ)/KH(NNODE,NDIAG,IZ)
```

```
      NODE=NNODE-1
      DO 20 J = NODE,1,-1
      NITER=NMAX
      IF(NITER.GT.NNODE-J) NITER=NNODE-J
      DO 30 K=1,NITER
   30 RHS(J,IZ)=RHS(J,IZ)-HEAD(J+K,IZ,2)*KH(J,NDIAG+K,IZ)
   20 HEAD(J,IZ,2)=RHS(J,IZ)/KH(J,NDIAG,IZ)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     VARIABLE LISTING: SUBROUTINE LHSCOR                              C
C                                                                      C
C     NAME    TYPE       DESCRIPTION                                   C
C     ------  ---------  --------------------------------------------- C
C     CORLHA  REAL*8     COLUMN A OF LEFT HAND SIDE OF CORRECTOR       C
C                        (LAYERS); [L2/T]                              C
C     CORLHB  REAL*8     COLUMN B OF LEFT HAND SIDE OF CORRECTOR       C
C                        (LAYERS); [L2/T]                              C
C     CORLHC  REAL*8     COLUMN C OF LEFT HAND SIDE OF CORRECTOR       C
C                        (LAYERS); [L2/T]                              C
C     INCOR   INTEGER*2  LOOP COUNTER FOR NODES                        C
C     IZCOR   INTEGER*2  LOOP COUNTER FOR LAYERS                       C
C     KV      REAL*8     UPPER SET OF VERTICAL FLOW TERMS FOR USE IN   C
C                        CORRECTOR EQUATIONS (NODES,LAYERS,1,2); [L2/T]C
C     KV      REAL*8     UPPER+LOWER SET OF VERTICAL FLOW TERMS FOR    C
C                        USE IN CORRECTOR EQUATIONS (NODES,LAYERS,2,2);C
C                        [L2/T]                                        C
C     KV      REAL*8     LOWER SET OF VERTICAL FLOW TERMS FOR USE IN   C
C                        CORRECTOR EQUATIONS (NODES,LAYERS,3,2); [L2/T]C
C     NLAY    INTEGER*2  NUMBER OF LAYERS                              C
C     ST      REAL*8     STORAGE TERMS TO BE APPLIED TO OLD HEADS      C
C                        (NODES,LAYERS,3); [L2]                        C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C  SUBROUTINE LHSCOR DESCRIPTION:                                      C
C  CALCULATES LEFT HAND SIDE VECTORS FOR CORRECTOR EQUATIONS.  ASSIGNS C
C  APPROPRIATE KV OR KV PLUS ST TO THREE VECTORS.                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C            SUBROUTINE LHSCOR                                         C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE LHSCOR
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
C     FORM LEFT HAND SIDE MATRIX OF CORRECTOR
      DO 50 IZCOR=1,NLAY
          CORLHC(IZCOR)=-KV(INCOR,IZCOR,1,2)
```

```
          CORLHB(IZCOR)=KV(INCOR,IZCOR,2,2)+(ST(INCOR,IZCOR,1)/DELTIM)
   50 CORLHA(IZCOR)=-KV(INCOR,IZCOR,3,2)
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C                                                                      C
C     VARIABLE LISTING: SUBROUTINE LCORDR                              C
C                                                                      C
C     NAME    TYPE        DESCRIPTION                                  C
C     ------  ---------   ---------------------------------------      C
C     CORLHA  REAL*8      COLUMN A OF LEFT HAND SIDE OF CORRECTOR      C
C                         (LAYERS); [L2/T]                             C
C     CORLHB  REAL*8      COLUMN B OF LEFT HAND SIDE OF CORRECTOR      C
C                         (LAYERS); [L2/T]                             C
C     CORLHC  REAL*8      COLUMN C OF LEFT HAND SIDE OF CORRECTOR      C
C                         (LAYERS); [L2/T]                             C
C     INCOR   INTEGER*2   LOOP COUNTER FOR NODES                       C
C     IZCOR   INTEGER*2   LOOP COUNTER FOR LAYERS                      C
C     NODFLG  INTEGER*2   NODAL FLAG FOR HEAD STATUS (NODES,LAYERS,2)  C
C                         SEE SUBROUTINE CHKHED FOR EXPLANATION OF     C
C                         FLAG NUMBERS                                 C
C     NLAY    INTEGER*2   NUMBER OF LAYERS                             C
C                                                                      C
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C                                                                      C
C  SUBROUTINE LCORDR DESCRIPTION:                                      C
C  INCORPORATES DRY NODES INTO CORRECTOR VECTORS. AT NODES             C
C  WHERE DRY NODES ARE SPECIFIED, SET CORLHA, CORLHC EQUAL TO 0,       C
C  CORLHB EQUAL TO 1.                                                  C
C                                                                      C
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C                                                                      C
C            SUBROUTINE LCORDR                                         C
C                                                                      C
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE LCORDR
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NODE(3),NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
      DO 50 IZCOR=1,NLAY
            DO 25 IE=1,NELEM
                  DO 25 IN3=1,3
C.....................IDENTIFY NODE NUMBER ON TRIANGULAR ELEMENT
                        NODE(1)=NEM(IE,1)
                        NODE(2)=NEM(IE,2)
                        NODE(3)=NEM(IE,3)
                        IF (NODE(IN3).EQ.INCOR) THEN
                             IF (NELFLG(IE,IZCOR).EQ.5) THEN
                                 CORLHC(IZCOR)=0.D00
                                 CORLHB(IZCOR)=1.D00
                                 CORLHA(IZCOR)=0.D00
                             ENDIF
```

```
                      ENDIF
  25      CONTINUE
  50  CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     VARIABLE LISTING: SUBROUTINE RHSCOR                              C
C                                                                      C
C     NAME    TYPE       DESCRIPTION                                   C
C     ------  ---------  --------------------------------------------  C
C     CORRHS  REAL*8     RIGHT HAND SIDE OF CORRECTOR (LAYERS); [L3/T] C
C     HEAD    REAL*8     HEADS FROM NEW TIMESTEP (NODES,LAYERS,2); [L] C
C     INCOR   INTEGER*2  LOOP COUNTER FOR NODES                        C
C     IZCOR   INTEGER*2  LOOP COUNTER FOR LAYERS                       C
C     NLAY    INTEGER*2  NUMBER OF LAYERS                              C
C     ST      REAL*8     STORAGE TERMS TO BE APPLIED TO TOPS OF NODES  C
C                        (NODES,LAYERS,2); [L2]                        C
C     ST      REAL*8     STORAGE TERMS TO BE APPLIED TO NEW HEADS      C
C                        (NODES,LAYERS,3); [L2]                        C
C     TOP     REAL*8     TOP OF NODES (NODES,LAYERS); [L]              C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C  SUBROUTINE RHSCOR DESCRIPTION:                                      C
C  FORMS RIGHT HAND SIDE MATRIX FOR CORRECTOR EQUATIONS.               C
C  MULTIPLIES ST TIMES APPROPRIATE HEADS TO FORM AND ADDS RHS2         C
C  VECTOR.                                                             C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C            SUBROUTINE RHSCOR                                         C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE RHSCOR
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
C     BEGIN LAYER LOOP
      DO 40 IZCOR=1,NLAY
      CORRHS(IZCOR)=((ST(INCOR,IZCOR,3)/DELTIM)*HEAD(INCOR,IZCOR,2))
     1             +((ST(INCOR,IZCOR,2)/DELTIM)*TOP(INCOR,IZCOR))
     2             -RHS2(INCOR,IZCOR)
  40  CONTINUE
          RETURN
          END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     VARIABLE LISTING: SUBROUTINE RCORDR                              C
C                                                                      C
C     NAME    TYPE       DESCRIPTION                                   C
C     ------  ---------  --------------------------------------------  C
```

```
C     CORRHS  REAL*8      RIGHT HAND SIDE OF CORRECTOR (LAYERS); [L3/T] C
C     HEAD    REAL*8      HEADS FROM OLD ITERATION (NODES,LAYERS,3); [L]C
C     INCOR   INTEGER*2   LOOP COUNTER FOR NODES                        C
C     IZCOR   INTEGER*2   LOOP COUNTER FOR LAYERS                       C
C     IZMCOR  INTEGER*2   LOOP COUNTER FOR LAYERS (IZCOR-1)             C
C     NODFLG  INTEGER*2   NODAL FLAG FOR HEAD STATUS (NODES,LAYERS,2)   C
C                         SEE SUBROUTINE CHKHED FOR EXPLANATION OF      C
C                         FLAG NUMBERS                                  C
C     NLAY    INTEGER*2   NUMBER OF LAYERS                              C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C  SUBROUTINE RCORDR DESCRIPTION:                                       C
C  INCORPORATES DRY NODES INTO RCORDR VECTOR.  AT NODES                 C
C  WHERE DRY NODES ARE SPECIFIED, SET RHS EQUAL TO HEADS FROM           C
C  OLD TIMESTEP AND LAYER BELOW.                                        C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C            SUBROUTINE RCORDR                                          C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE RCORDR
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NODE(3),NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
      DO 50 IZCOR=1,NLAY
            DO 50 IE=1,NELEM
                  IF ( (IZCOR.EQ.1) .AND.
     1                 (NELFLG(IE,IZCOR).EQ.5) ) THEN
                       WRITE (6,1000)
                  ENDIF
                  IF ( (IZCOR.GT.1) .AND.
     1                 (NELFLG(IE,IZCOR).EQ.5) ) THEN
C.....................IDENTIFY NODE NUMBER ON TRIANGULAR ELEMENT
                       DO 25 IN3=1,3
                            NODE(1)=NEM(IE,1)
                            NODE(2)=NEM(IE,2)
                            NODE(3)=NEM(IE,3)
                            IF (NODE(IN3).EQ.INCOR) THEN
                                 IZMCOR=IZCOR-1
                                 CORRHS(IZCOR)=HEAD(NODE(IN3),IZMCOR,1)
                            ENDIF
   25                  CONTINUE
                  ENDIF
   50 CONTINUE
 1000 FORMAT (' BOTTOM NODE IN AQUIFER GOES DRY')
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     VARIABLE LISTING: SUBROUTINE OUTRAD                               C
```

```
C                                                                       C
C     NAME    TYPE        DESCRIPTION                                   C
C     ------  ---------   ---------------------------------------------C
C     HEAD    REAL*8      HEADS FROM NEW ITERATION (NODES,LAYERS,2); [L]C
C     INN     INTEGER*2   LOOP COUNTER FOR NODES                        C
C     IN2     INTEGER*2   LOOP COUNTER FOR EVERY OTHER NODE             C
C     IZ      INTEGER*2   LOOP COUNTER FOR LAYERS                       C
C     NLAY    INTEGER*2   NUMBER OF LAYERS                              C
C     NNODE   INTEGER*2   TOTAL NUMBER OF NODES                         C
C     NNODE2  INTEGER*2   HALF THE TOTAL NUMBER OF NODES                C
C     NODCOR  REAL*8      Y COORDINATES OF NODES (NODES,2); [L]         C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C  SUBROUTINE OUTRAD DESCRIPTION:                                       C
C  OUTPUTS FINAL HEADS IN RADIAL FORMAT, LAYER BY LAYER.  APPROPRIATE   C
C  ONLY FOR PIE SHAPED INPUT DATA SET.                                  C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C             SUBROUTINE OUTRAD                                         C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE OUTRAD
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
      WRITE (7,1000) (IZ,IZ=1,NLAY)
      NNODE2=NNODE/2
      DO 20 INN=1,NNODE2
      IN2=((INN-1)*2)+1
   20 WRITE (7,1001) NODCOR(IN2,2),( HEAD(IN2,IZ,2), IZ=1,NLAY)
 1000 FORMAT ('           ',(7I10))
 1001 FORMAT (F10.4,(7F10.4))
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     VARIABLE LISTING: SUBROUTINE OUTCOL                               C
C                                                                       C
C     NAME    TYPE        DESCRIPTION                                   C
C     ------  ---------   ---------------------------------------------C
C     HEAD    REAL*8      HEADS FROM NEW ITERATION (NODES,LAYERS,2); [L]C
C     IN      INTEGER*2   LOOP COUNTER FOR NODES                        C
C     IZ      INTEGER*2   LOOP COUNTER FOR LAYERS                       C
C     NLAY    INTEGER*2   NUMBER OF LAYERS                              C
C     NNODE   INTEGER*2   TOTAL NUMBER OF NODES                         C
C     NODCOR  REAL*8      X COORDINATES OF NODES (NODES,1); [L]         C
C     NODCOR  REAL*8      Y COORDINATES OF NODES (NODES,2); [L]         C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
```

```
C  SUBROUTINE OUTCOL DESCRIPTION:                                        C
C  OUTPUTS FINAL HEADS IN NODE BY NODE, LAYER BY LAYER FORMAT.           C
C  APPROPRIATE FOR ALL DATA SETS.                                        C
C                                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                        C
C             SUBROUTINE OUTCOL                                          C
C                                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE OUTCOL
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
      WRITE (7,1000) IZ,NLAY
      DO 20 IN=1,NNODE
   20 WRITE (7,1001) ( NODCOR(IN,IP), IP=1,2 ),HEAD(IN,IZ,2)
 1000 FORMAT (' LAYER=',I4,' OF',I4)
 1001 FORMAT (3F10.4)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                        C
C     VARIABLE LISTING: SUBROUTINE THMALG                                C
C                                                                        C
C     NAME    TYPE        DESCRIPTION                                  C
C     ------  ---------   ---------------------------------------------C
C     AA      REAL*8      COLUMN A OF LEFT HAND SIDE OF CORRECTOR      C
C                         (LAYERS); [L3/T]                             C
C     BB      REAL*8      COLUMN B OF LEFT HAND SIDE OF CORRECTOR      C
C                         (LAYERS); [L3/T]                             C
C     BETA    REAL*8      TEMPORARY COLUMN VECTOR                      C
C     CC      REAL*8      COLUMN C OF LEFT HAND SIDE OF CORRECTOR      C
C                         (LAYERS); [L3/T]                             C
C     GAMMA   REAL*8      TEMPORARY COLUMN VECTOR                      C
C     HEAD    REAL*8      HEADS FROM NEW ITERATION (NODES,LAYERS,2); [L]C
C     INCOR   INTEGER*2   LOOP COUNTER FOR NODES                       C
C     I       INTEGER*2   LOOP COUNTER FOR LAYERS                      C
C     NLAY    INTEGER*2   NUMBER OF LAYERS                             C
C     NNODE   INTEGER*2   TOTAL NUMBER OF NODES                        C
C     RR      REAL*8      RIGHT HAND SIDE OF CORRECTOR (LAYERS); [L3/T] C
C                                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                        C
C  SUBROUTINE THMALG DESCRIPTION:                                        C
C  SOLVES FOR HEADS FROM CORRECTOR EQUATIONS USING THOMAS ALGORTITHM.    C
C  APPROPRIATE FOR BANDED MATRIX WITH ONLY THREE COLUMNS (OR VECTORS).   C
C                                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                        C
C             SUBROUTINE THMALG                                          C
C                                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
      SUBROUTINE THMALG
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION AA(20), BB(20), CC(20), RR(20)
      DIMENSION BETA(20), GAMMA(20)
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),RHS2(51,20),TOP(51,20),
     3QDRY(51,20),NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2AA,BB,CC,RR,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
      BETA(1) = BB(1)
      GAMMA(1) = RR(1) / BETA(1)
      DO 10 I = 2, NLAY
        BETA(I) = BB(I) - AA(I) * CC(I - 1) / BETA(I - 1)
   10 GAMMA(I) = (RR(I) - AA(I)*GAMMA(I - 1)) / BETA(I)
      HEAD(INCOR,NLAY,2) = GAMMA(NLAY)
      N1 = NLAY - 1
      DO 20 J = 1, N1
        I = NLAY - J
   20 HEAD(INCOR,I,2)=GAMMA(I)-CC(I)*HEAD(INCOR,I + 1,2)/BETA(I)
 1000 FORMAT (4E12.6)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                            C
C     VARIABLE LISTING: SUBROUTINE CONVER                                    C
C                                                                            C
C     NAME    TYPE        DESCRIPTION                                        C
C     ------  ---------   ------------------------------------------------   C
C     ERRALL  REAL*8      ERROR BETWEEN TIMESTEPS [D]                        C
C     ERRALL  REAL*8      ALLOWABLE ERROR BETWEEN TIMESTEPS [D]              C
C     ERRMAX  REAL*8      MAXIMUM ERROR BETWEEN ITERATIONS [D]               C
C     EMXMAX  REAL*8      MAXIMUM ERROR FOR ENTIRE RUN [D]                   C
C     HEAD    REAL*8      HEADS FROM NEW ITERATION (NODES,LAYERS,2); [L]C
C     HEAD    REAL*8      HEADS FROM OLD ITERATION (NODES,LAYERS,3); [L]C
C     INE     INTEGER*2   LOOP COUNTER FOR NODES                             C
C     IT      INTEGER*2   LOOP COUNTER FOR TIME                              C
C     ITER    INTEGER*2   LOOP COUNTER FOR ITERATIONS                        C
C     IZE     INTEGER*2   LOOP COUNTER FOR LAYERS                            C
C     INEMAX  INTEGER*2   NODE INDEX FOR LOCATION OF ERRMAX                  C
C     IZEMAX  INTEGER*2   LAYER INDEX FOR LOCATION OF ERRMAX                 C
C     MXITER  INTEGER*2   MAXIMUM ALLOWABLE ITERATIONS                       C
C     NCONT   INTEGER*2   FLAG FOR CONTINUING ITERATIONS (=1 CONTINUE)       C
C     NLAY    INTEGER*2   NUMBER OF LAYERS                                   C
C     NLAY1   INTEGER*2   NUMBER OF LAYERS PLUS ONE (NLAY+1)                 C
C     NLSTRT  INTEGER*2   LAYER NUMBER OF UNCONFINED LAYER                   C
C     NNODE   INTEGER*2   TOTAL NUMBER OF NODES                              C
C     NTIMST  INTEGER*2   TOTAL NUMBER OF TIMESTEPS                          C
C                                                                            C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                            C
C  SUBROUTINE CONVER DESCRIPTION:                                            C
C  CHECKS FOR CONVERGENCE OF HEAD SOLUTION BY FINDING MAXIMUM ERROR          C
C  BETWEEEN HEADS FROM NEW ITERATION AND HEADS FROM OLD ITERATION.           C
C  ERROR IS CALCULATED AS ABSOLUTE VALUE OF RELATIVE ERROR.                 C
C  ERROR IS COMPARED TO MAXIMUM ALLOWABLE ERROR AND NUMBER OF                C
C  ITERATIONS IS COMPARED TO MAXIMUM ALLOWABLE ITERATIONS.  IF               C
```

```
C  CONVERGENCE IS REACHED BEFORE MAXIMUM ALLOWABLE ITERATIONS,          C
C  "NCONT" FLAG IS 0, AND MAIN PROGRAM IS ALLOWED TO GO TO NEXT         C
C  TIMESTEP.  OTHERWISE, "NCONT" FLAG IS SET TO 1 AND ITERATIONS        C
C  CONTINUE.  IF ERROR IS TOO LARGE AT MAXIMUM ALLOWABLE ITERATIONS,    C
C  SEND ERROR MESSAGE AND PROCEED TO NEXT TIMESTEP.                     C
C  ALSO, KEEPS TRACK OF POSITION (NODE,LAYER) OF MAXIMUM ERROR FROM     C
C  ANY TIMESTEP.  MAXIMUM ERRROR AND POSTION ARE OUTPUT TO UNIT 10.     C
C  MAXIMUM ERROR OVER ALL TIMESTEPS IS ALSO IDENTIFIED AND SENT TO      C
C  UNIT 7.                                                              C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C            SUBROUTINE CONVER                                          C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE CONVER(NCONT)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
C.....INITIALIZE NCONT
      NCONT=0
C.....FIND GREATEST ERROR
      ERRMAX=0.D0
C.....SET NLAY+1=NLAY1
      NLAY1=NLAY+1
      DO 100 INE=1,NNODE
           DO 100 IZE=1,NLAY
                ERR=DABS(HEAD(INE,IZE,2)-HEAD(INE,IZE,3))
     1                 /HEAD(INE,IZE,3)
                IF (ERR.GT.ERRMAX) THEN
                      ERRMAX=ERR
                      INEMAX=INE
                      IZEMAX=IZE
                ENDIF
                IF (ERR.GT.EMXMAX) THEN
                      EMXMAX=ERR
                ENDIF
  100 CONTINUE
C.....WRITE OUT ERROR (DIFFERENCE BETWEEN OLD AND NEW)
      WRITE (10,1007) IT,ITER,ERRMAX,INEMAX,IZEMAX
C.....CHECK FOR CONVERGENCE OR ALL CONFINED
C.....AND CONTINUE ITERATION OR GO TO NEXT TIME STEP
      IF ( (ITER.LT.MXITER) .AND. (ERRMAX.GT.ERRALL)
     1.AND. (NLSTRT.NE.NLAY1) ) THEN
          NCONT=1
      ELSE IF (ITER.EQ.MXITER) THEN
          WRITE (6,1004)
      ELSE IF ( (ITER.LT.MXITER) .AND. (ERRMAX.LT.ERRALL)
     1.AND. (NLSTRT.NE.NLAY1) ) THEN
           WRITE (6,1005) ITER
      ELSE IF (NLSTRT.EQ.NLAY1) THEN
          WRITE (6,1009)
```

```
      ENDIF
C.....WRITE LARGEST ERROR ENCOUNTERED
      IF (IT.EQ.NTMSTP) THEN
      WRITE (7,1013) EMXMAX
      WRITE (7,1012)
      ENDIF
 1004 FORMAT (' FAILED TO MEET CONVERGENCE CRITERIA WITHIN MAXIMUM
     1          ALLOWABLE ITERATIONS')
 1005 FORMAT (' SUCCEEDED IN MEETING CONVERGENCE CRITERIA WITHIN'
     1,I4,' ITERATIONS')
 1007 FORMAT (' ',I4,I4,E12.6,I4,I4)
 1009 FORMAT (' ALL LAYERS CONFINED, NO ITERATIONS NECESSARY')
 1012 FORMAT (' ')
 1013 FORMAT (' LARGEST ERROR ENCOUNTERED BETWEEN STEPS = ',E12.6)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     VARIABLE LISTING: SUBROUTINE WATBAL                              C
C                                                                      C
C     NAME    TYPE        DESCRIPTION                                  C
C     ------  ----------  -------------------------------------------  C
C     BALLHS  REAL*8      LEFT HAND SIDE OF BALANCE EQUATION: STRESSES C
C                         [L3/T]                                       C
C     BALRHS  REAL*8      RIGHT HAND SIDE OF BALANCE EQUATION: CHANGES C
C                         IN WATER LEVEL [L3/T]                        C
C     DELTIM  REAL*8      TIME STEP SIZE [T]                           C
C     ERRBAL  REAL*8      ERROR IN WATER BALANCE [D]                   C
C     HEAD    REAL*8      HEADS FROM OLD TIMESTEP (NODES,LAYERS,1); [L] C
C     HEAD    REAL*8      HEADS FROM NEW ITERATION (NODES,LAYERS,2); [L]C
C     IN      INTEGER*2   LOOP COUNTER FOR NODES                       C
C     IZ      INTEGER*2   LOOP COUNTER FOR LAYERS                      C
C     MXITER  INTEGER*2   MAXIMUM ALLOWABLE ITERATIONS                 C
C     NLAY    INTEGER*2   NUMBER OF LAYERS                             C
C     NNODE   INTEGER*2   TOTAL NUMBER OF NODES                        C
C     NTIMST  INTEGER*2   TOTAL NUMBER OF TIMESTEPS                    C
C     ST      REAL*8      STORAGE TERMS TO BE APPLIED TO NEW HEADS     C
C                         (NODES,LAYERS,1); [L2]                       C
C     ST      REAL*8      STORAGE TERMS TO BE APPLIED TO TOPS OF NODES C
C                         (NODES,LAYERS,2); [L2]                       C
C     ST      REAL*8      STORAGE TERMS TO BE APPLIED TO OLD HEADS     C
C                         (NODES,LAYERS,3); [L2]                       C
C     TOP     REAL*8      TOP OF NODE (NODES,LAYERS); [L]              C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C  SUBROUTINE WATBAL DESCRIPTION:                                      C
C  PERFORMS WATER BALANCE OVER ENTIRE DOMAIN.  CALCULATES WATER OUT    C
C  BY SUBTRACTING FINAL HEADS FROM INITIAL HEADS AND INTEGRATING       C
C  OVER EACH ELEMENT.  IN ORDER TO GET INITIAL HEADS INPUT DATA        C
C  FILE IS REWOUND AND THESE HEADS ARE CHECKED FOR STATUS (CONFINED,   C
C  UNCONFINED, ETC.).  FINAL HEADS ARE ALSO CHECKED FOR STATUS, AND    C
C  STORAGE TERMS ARE CALCULATED ON STATUS OF HEADS. WATER OUT IS       C
C  COMPARED TO WATER OUT AS A RESULT OF NODAL STRESSES.  ERROR IN      C
C  WATER BALANCE IS THEN CALCULATED AS DIFFERENCE BETWEEN WATER OUTS.  C
C  ERROR IS CALCULATED AS ABSOLUTE VALUE OF RELATIVE DIFFERENCE.       C
C  WATER BALANCE ERROR IS OUTPUT TO UNITS 20 AND 7.                     C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
```

```
C             SUBROUTINE WATBAL                                          C
C                                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE WATBAL
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH,KV,NODCOR
      DIMENSION Q(51,20),NODFLG(51,20,2),HEAD(51,20,3),
     1NODCOR(51,2),ELMPRP(51,8,20),NEM(51,4),KH(51,10,20),ST(51,20,3),
     2RHS(51,20),RECHGN(51,20),KV(51,20,3,2),CORLHA(20),CORLHB(20),
     3CORLHC(20),CORRHS(20),RHS2(51,20),TOP(51,20),QDRY(51,20),
     4NELFLG(51,20)
      COMMON Q,NODFLG,HEAD,NODCOR,ELMPRP,NEM,KH,ST,RHS,KV,
     1RECHGN,NNODE,NELEM,NLAY,NBAND,DELTIM,NTIMST,NSWTCH,IT,IZ,INCOR,
     2CORLHA,CORLHB,CORLHC,CORRHS,MXITER,ERRALL,ITER,RHS2,TOP,NLSTRT,
     3QDRY,NDRY,NACCL,NSS,NRAD,NELFLG
C.....READ BACK IN INITIAL HEADS
      REWIND (3)
      DO 80 IN=1,NNODE
   80 READ (3,1010) ( HEAD(IN,IZ,1), IZ=1,NLAY )
C.....CHECK CONDITION OF OLD HEADS
      CALL CHKHED(1)
C.....CHECK CONDITION OF NEW HEADS WITH INITIAL HEADS AS REFERENCE
      CALL CHKHED(2)
      DO 100 IZ=1,NLAY
  100 CALL FORMST(1)
      DO 40 IZ=1,NLAY
           DO 40 IN=1,NNODE
                BALRHS=BALRHS+(-(ST(IN,IZ,3)*HEAD(IN,IZ,1))
     1                         -(ST(IN,IZ,2)*TOP(IN,IZ))
     2                         +(ST(IN,IZ,1)*HEAD(IN,IZ,2)) )
   40 BALLHS=BALLHS+(Q(IN,IZ)*NTIMST*DELTIM)
      ERRBAL=DABS((BALRHS-BALLHS)/BALRHS)
      WRITE (6,1000) ERRBAL
      WRITE (7,1000) ERRBAL
      WRITE (7,1001)
 1000 FORMAT (' RELATIVE ERROR IN WATER BALANCE=',E12.5)
 1001 FORMAT (' ')
 1010 FORMAT ((7F10.4))
      RETURN
      END
```

APPENDIX 3:

FORTRAN CODES FOR VALIDATION PROGRAMS

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C        PROGRAM FOR STEADY-STATE, ONE-DIMENSIONAL FLOW IN AN           C
C        UNCONFINED AQUIFER WITH TWO DIRICHLET BOUNDARIES (WITH         C
C        OR WITHOUT RECHARGE)                                           C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      IMPLICIT REAL*8 (A-H,O-Z)
C.....READ IN DATA
      READ (1,1000) IXMAX,XDEL
      READ (1,1001) HEAD0,HEAD1,X1
      READ (1,1002) RECHGE,CONDX
C.....ECHO OUT DATA
      WRITE (3,1000) IXMAX,XDEL
      WRITE (3,1001) HEAD0,HEAD1,X1
      WRITE (3,1002) RECHGE,CONDX
C.....BEGIN X LOOP
      DO 100 IX=1,IXMAX
          X=XDEL*(IX-1)
C.........CALCULATE HEADS
          HEADSQ=((RECHGE/CONDX)*(X**2))
     1          + ( ( (((HEAD1**2)-(HEAD0**2))/X1)
     2          - (RECHGE*X1/CONDX) ) *X )
     3          + (HEAD0**2)
          HEAD=HEADSQ**0.5D0
C.........WRITE HEADS,X
  100     WRITE (6,1003) X,HEAD
 1000 FORMAT (I4,F10.4)
 1001 FORMAT (3F10.4)
 1002 FORMAT (2E12.6)
 [illegible]03 FORMAT (2F10.4)
      STOP
      END
```

```
$LARGE
$DEBUG
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                  PROGRAM FOR ANALYTIC SOLUTION FOR
CC                  FLOW IN CONFINED AQUIFERS, INFINITE RADIUS
CC                  -- THEIS EQUATION
CC                  OR FLOW IN UNCONFINED AQUIFERS WHERE DRAWDOWN
CC                  IS SMALL COMPARED TO AQUIFER THICKNESS
CC                  -- THEIS EQUATION WITH SPECIFIC YIELD;
CC                     JACOB DRAWDOWN CORRECTION
CC
CC                  ..............................................
CC
CC                  0=UNCONFINED
CC                  1=CONFINED
CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH
      COMMON NRMAX,B,RDEL,KH,S,Q,IR,TIME,NCONF,HINIT
      CALL READ
      CALL WRITE
C     IF ( NCONF.EQ.0 ) THEN
C          WRITE (3,1002)
C     ELSE IF ( NCONF.EQ.1 ) THEN
C          WRITE (3,1003)
C     ENDIF
      BEGIN R (RADIAL) LOOP
      DO 100 IR=1,NRMAX
           WRITE (2,1000) IR,NRMAX
           R=IR*RDEL
C.....CALCULATE WELL FUNCTION ARGUMENT
           U1=((R**2)*S)/(4*(KH*B)*TIME)
C.....CALCULATE W(U)
           CALL WELLFN (U1,WEL1)
           PI = 3.14159265358979323846264 3D0
C.....CALCULATE DRAWDOWN AND CORRECT FOR UNCONFINED AQUIFER
C.....WITH JACOB CORRECTION
           SDWN=(Q/(4*PI*KH*B))*WEL1
           IF ( NCONF.EQ.0 ) THEN
               DRWDWN=B*(1-((1-(2*SDWN/B))**0.5D0))
           ELSE IF ( NCONF.EQ.1 ) THEN
               DRWDWN=SDWN
           ENDIF
C.....CALCULATE HEAD
      HEAD=HINIT-DRWDWN
  100 WRITE (3,1001) R,HEAD
 1001 FORMAT (2F10.4)
 1000 FORMAT (' IR STEP',I4,' OF',I4)
 1002 FORMAT (' UNCONFINED AQUIFER')
 1003 FORMAT (' CONFINED AQUIFER')
      STOP
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                   SUBROUTINE READ
C
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccC
      SUBROUTINE READ
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH
      COMMON NRMAX,B,RDEL,KH,S,Q,IR,TIME,NCONF,HINIT
      READ (1,1001) B,RDEL,NRMAX,NCONF
      READ (1,1002) KH,S,Q,TIME
      READ (1,1003) HINIT
 1001 FORMAT (2F10.4,2I4)
 1002 FORMAT (4E12.6)
 1003 FORMAT (F10.4)
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccC
C
C                 SUBROUTINE WRITE
C
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccC
      SUBROUTINE WRITE
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 KH
      COMMON NRMAX,B,RDEL,KH,S,Q,IR,TIME,NCONF,HINIT
      WRITE (2,1001) B,RDEL,NRMAX,NCONF
      WRITE (2,1002) KH,S,Q,TIME
      WRITE (2,1003) HINIT
 1001 FORMAT (2F10.4,2I4)
 1002 FORMAT (4E12.6)
 1003 FORMAT (F10.4)
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccC
C
C                 SUBROUTINE WELLFN
C
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccC
      SUBROUTINE WELLFN (U,WFNANS)
      IMPLICIT REAL*8 (A-H,O-Z)
      IF (U.GT.50.00D0) THEN
           U=50.00D0
      ENDIF
      IF (U.LT.1.00D0) THEN
C          SOLVE WELL FUNCTION WHEN U < 1
           WFNANS=-LOG(U)-.57721566D0+.99999193D0*U-.24991055D0*U**2
     1          +5.519968E-02*U**3-9.76004E-03*U**4+1.07857E-03*U**5
      ELSE
C          SOLVE WELL FUNCTION WHEN U > 1
           WN=U**4+8.5733287401D0*U**3+18.059016973D0*U**2
     1            +8.6347608925D0*U+.2677737343D0
           WD=U**4+9.5733223454D0*U**3+25.6329561486D0*U**2
     1              +21.0996530827D0*U+3.9584969228D0
           WFNANS=WN/(WD*U*EXP(U))
      ENDIF
      RETURN
      END
```

```
$DEBUG
$LARGE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC
CC                    PROGRAM FOR ANALYTIC SOLUTION FOR
CC                    PARTIALLY SCREENED WELLS
CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 L,KV,KH
      COMMON N1MAX,NRMAX,NZMAX,L,D,B,RDEL,ZDEL,KV,KH,
     1STOR,Q,H,TIME,HINIT,ERRALL
      CALL READ
      CALL WRITE
C.....BEGIN R (RADIAL) LOOP
      DO 100 IR=1,NRMAX
           WRITE (2,1000) IR,NRMAX
           R=IR*RDEL
C..........BEGIN Z (VERTICAL) LOOP
           DO 100 IZ=1,NZMAX
                WRITE (2,1001) IZ,NZMAX
                Z=IZ*ZDEL
C...............CALCULATE WELL FUNCTION ARGUMENT
                U1=((R**2)*STOR)/(4*(KH*B)*TIME)
                WRITE (3,1002) U1
                CALL WELLFN (U1,WEL1)
                WRITE (3,1002) U1
                CALL SUMMAT(R,Z,U1,WEL3,N1MAX,L,D,B,ERRALL)
100        CALL DRAWDN(WEL1,WEL3,R,Z,H,Q,KH,B,HINIT)
C     CALL OUTPUT(H,NRMAX,NZMAX,RDEL,ZDEL)
 1000 FORMAT (' IR STEP',I4,' OF',I4)
 1001 FORMAT (' IZ STEP',I4,' OF',I4)
 1002 FORMAT (' U1=',E12.6)
      STOP
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                    SUBROUTINE READ
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE READ
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 L,KV,KH
      COMMON N1MAX,NRMAX,NZMAX,L,D,B,RDEL,ZDEL,KV,KH,
     1STOR,Q,H,TIME,HINIT,ERRALL
      READ (1,1000) N1MAX,NRMAX,NZMAX
      READ (1,1001) L,D,B,RDEL,ZDEL
      READ (1,1002) KV,KH,STOR,Q,TIME
      READ (1,1003) ERRALL,HINIT
 1000 FORMAT (3I4)
 1001 FORMAT (5F10.4)
 1002 FORMAT (6E12.6)
 1003 FORMAT (E12.6,F10.4)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                    SUBROUTINE WRITE
```

```
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE WRITE
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 L,KV,KH
      COMMON N1MAX,NRMAX,NZMAX,L,D,B,RDEL,ZDEL,KV,KH,
     1STOR,Q,H,TIME,HINIT,ERRALL
      WRITE (2,1000) N1MAX,NRMAX,NZMAX
      WRITE (2,1001) L,D,B,RDEL,ZDEL
      WRITE (2,1002) KV,KH,STOR,Q,TIME
      WRITE (2,1003) ERRALL,HINIT
 1000 FORMAT (3I4)
 1001 FORMAT (5F10.4)
 1002 FORMAT (6E12.6)
 1003 FORMAT (E12.6,F10.4)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                 SUBROUTINE WELLFN
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE WELLFN (U,WFNANS)
      IMPLICIT REAL*8 (A-H,O-Z)
               WRITE (3,1002) U
 1002 FORMAT (' U=',E12.6)
      IF (U.GT.50.00D0) THEN
           U=50.00D0
      ENDIF
      IF (U.LT.1.00D0) THEN
           SOLVE WELL FUNCTION WHEN U < 1
           WFNANS=-LOG(U)-.57721566D0+.99999193D0*U-.24991055D0*U**2
     1          +5.519968E-02*U**3-9.76004E-03*U**4+1.07857E-03*U**5
      ELSE
C          SOLVE WELL FUNCTION WHEN U > 1
           WN=U**4+8.5733287401D0*U**3+18.059016973D0*U**2
     1            +8.6347608925D0*U+.2677737343D0
           WD=U**4+9.5733223454D0*U**3+25.6329561486D0*U**2
     1              +21.0996530827D0*U+3.9584969228D0
           WFNANS=WN/(WD*U*EXP(U))
      ENDIF
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                 SUBROUTINE LKWF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE LKWF(I1,U,BETA,LKWFT)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 LKWFT,LKWF1,LKWF2,LKWF3
      DIMENSION Z1(256),W1(256),Z2(256),W2(256),Z3(256),W3(256)
      LKWFT=0.D0
      LKWF1=0.D0
      LKWF2=0.D0
      LKWF3=0.D0
      ULIM1=1.D-05
      ULIM2=1.D-01
      ULIM3=1.D+20
      IF (U.LE.ULIM1) THEN
```

```
           IF (I1.EQ.1) THEN
           CALL DGAUSS(U,ULIM1,Z1,W1,256)
           ENDIF
           DO 100 M1=1,256
                LKWF1=LKWF1+W1(M1)*((1/Z1(M1))*EXP(-Z1(M1)
     1               -(BETA**2/(4*Z1(M1)))))
  100      CONTINUE
           IF (I1.EQ.1) THEN
           CALL DGAUSS(ULIM1,ULIM2,Z2,W2,256)
           ENDIF
           DO 200 M2=1,256
                LKWF2=LKWF2+W2(M2)*((1/Z2(M2))*EXP(-Z2(M2)
     1               -(BETA**2/(4*Z2(M2)))))
  200      CONTINUE
           IF (I1.EQ.1) THEN
           CALL DGAUSS(ULIM2,ULIM3,Z3,W3,256)
           ENDIF
           DO 300 M3=1,256
                LKWF3=LKWF3+W3(M3)*((1/Z3(M3))*EXP(-Z3(M3)
     1               -(BETA**2/(4*Z3(M3)))))
  300      CONTINUE
           LKWFT=LKWF1+LKWF2+LKWF3
      ELSEIF ( (U.LE.ULIM2) .AND. (U.GT.ULIM1) ) THEN
           IF (I1.EQ.1) THEN
           CALL DGAUSS(U,ULIM2,Z2,W2,256)
           ENDIF
           DO 400 M2=1,256
                LKWF2=LKWF2+W2(M2)*((1/Z2(M2))*EXP(-Z2(M2)
     1               -(BETA**2/(4*Z2(M2)))))
  400      CONTINUE
           IF (I1.EQ.1) THEN
           CALL DGAUSS(ULIM2,ULIM3,Z3,W3,256)
           ENDIF
           DO 500 M3=1,256
                LKWF3=LKWF3+W3(M3)*((1/Z3(M3))*EXP(-Z3(M3)
     1               -(BETA**2/(4*Z3(M3)))))
  500      CONTINUE
           LKWFT=LKWF2+LKWF3
      ELSEIF (U.GT.ULIM2) THEN
           IF (I1.EQ.1) THEN
           CALL DGAUSS(U,ULIM3,Z3,W3,256)
           ENDIF
           DO 600 M3=1,256
                LKWF3=LKWF3+W3(M3)*((1/Z3(M3))*EXP(-Z3(M3)
     1               -(BETA**2/(4*Z3(M3)))))
  600      CONTINUE
           LKWFT=LKWF3
      ENDIF
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                 SUBROUTINE SUMMAT
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE SUMMAT(R,Z,U1,WEL3,N1MAX,L,D,B,ERRALL)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 L,KV,KH
      PI = 3.14159265358979323846264 3D0
      I1=0
```

```
C.....BEGIN SUMMATION LOOP
C.....INITIALIZE WEL3
      WEL3=0.D0
   50 I1=I1+1
           XI1=FLOAT(I1)
           XI1=DBLE(XI1)
           WRITE (2,1002) I1,N1MAX,XI1
C..........CALCULATE LEAKY WELL FUNCTION ARGUMENT
           U2=XI1*PI*R/B
           CALL LKWF(I1,U1,U2,WEL2)
           WRITE (3,1003) U1,U2,WEL2
           XSIN=DSIN(XI1*PI*L/B)-DSIN(XI1*PI*D/B)
           XCOS=DCOS(XI1*PI*Z/B)
           WRITE (3,1001) Z
           WEL3IN = (1.D0/XI1)*XCOS*XSIN*WEL2
           WEL3=WEL3+(((2*B)/(PI*(L-D)))*WEL3IN)
           WRITE (3,1000) WEL3IN,WEL3
           ARGMIN=1.D-15
C.....CHECK FOR CONVERGENCE OF SUMMATION
      WEL3ER=ABS(WEL3IN/WEL3)
      IF ((WEL3ER.GT.ERRALL) .AND. (I1.LT.N1MAX) .AND.
     1    (DABS(WEL3IN).GT.ARGMIN) ) THEN
           GO TO 50
      ELSEIF ((WEL3ER.GT.ERRALL) .AND. (I1.EQ.N1MAX) ) THEN
           WRITE (2,1004)
      ELSEIF  ( (WEL3ER.LT.ERRALL) .AND. (I1.LT.N1MAX) ) THEN
           WRITE (2,1005)
      ELSEIF ( (DABS(WEL3IN).LT.ARGMIN) .AND. (I1.LT.N1MAX) ) THEN
           WRITE (2,1006)
      ENDIF
 [illegible]00 FORMAT (' WEL3IN, WEL3 = ',2E12.6)
 1001 FORMAT (' Z =',F10.4)
 1002 FORMAT (' I1 STEP',I4,' OF',I4,F10.4)
 1003 FORMAT (' U1, U2, WEL2, = ',3E12.6)
 1004 FORMAT (' SUMMATION FAILED TO CONVERGE')
 1005 FORMAT (' SUMMATION CONVERGES')
 1006 FORMAT (' ARGUMENT TOO SMALL FOR LEAKY WELL FN')
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                 SUBROUTINE DRAWDN
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE DRAWDN(WEL1,WEL3,R,Z,H,Q,KH,B,HINIT)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 L,KV,KH
      PI = 3.14159265358979323846264 3D0
      H = HINIT-((Q/(4*PI*(KH*B)))*(WEL1+WEL3))
      WRITE (6,1000) R,Z,H
 1000 FORMAT (2F10.4,E12.6)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                 SUBROUTINE OUTPUT
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE OUTPUT(H,NRMAX,NZMAX,RDEL,ZDEL)
      IMPLICIT REAL*8 (A-H,O-Z)
```

```
      REAL*8 L,KV,KH
      DO 100 IZO=1,NZMAX
           Z=IZO*ZDEL
           WRITE (6,1001) Z
           DO 100 IRO=1,NRMAX
                R=IRO*RDEL
  100 WRITE (6,1000) R,H(IRO,IZO)
 1000 FORMAT (F10.4,E12.6)
 1001 FORMAT (12X,F10.4)
      RETURN
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C        PROGRAM FOR STEADY-STATE, ONE-DIMENSIONAL FLOW IN AN          C
C        UNCONFINED AQUIFER WITH TWO DIRICHLET BOUNDARIES (WITH        C
C        OR WITHOUT RECHARGE)                                          C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      IMPLICIT REAL*8 (A-H,O-Z)
C.....READ IN DATA
      READ (1,1000) IXMAX,XDEL
      READ (1,1001) HEAD0,HEAD1,X1
      READ (1,1002) RECHGE,CONDX
C.....ECHO OUT DATA
      WRITE (3,1000) IXMAX,XDEL
      WRITE (3,1001) HEAD0,HEAD1,X1
      WRITE (3,1002) RECHGE,CONDX
C.....BEGIN X LOOP
      DO 100 IX=1,IXMAX
          X=XDEL*(IX-1)
C.........CALCULATE HEADS
          HEADSQ=((RECHGE/CONDX)*(X**2))
     1          + ( ( (((HEAD1**2)-(HEAD0**2))/X1)
     2          - (RECHGE*X1/CONDX) ) *X )
     3          + (HEAD0**2)
          HEAD=HEADSQ**0.5D0
C.........WRITE HEADS,X
  100     WRITE (6,1003) X,HEAD
 1000 FORMAT (I4,F10.4)
 1001 FORMAT (3F10.4)
 1002 FORMAT (2E12.6)
 1003 FORMAT (2F10.4)
      STOP
      END
```