A Geostatistical Inverse Method for Conditioning Randomly Heterogeneous Transmissivity Fields on Transient Hydraulic Head Data

by

Debra Lane Hughson

Submitted in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy in Earth and Environmental Science

New Mexico Institute of Mining and Technology

Socorro, New Mexico

May, 1997

ACKNOWLEDGEMENT

First of all, I would like to acknowledge God for making the universe random, or at least making it appear that way to me. Then I would like to thank all the people who've made computers and made them available to me[1], without which this project would have been impossible. Next to the Office of God there's one of the vice-presidents, Dr. Allan Gutjahr, who's been the main supporter of and co-conspirator with the author. Thanks also to my committee members. Thanks to John Wilson for many helpful discussions which have had a major influence on the direction of my research and for thoroughly reviewing this dissertation. Thanks to John Prentice for seeing me through some major life challenges and to Fred Phillips and Jim McCord. Special thanks to Uncle Sam for redistribution of wealth. The United States taxpayer has been a major source of funding for this project.

This dissertation was typeset with LaTeX[2] by the author.

[1]"Research sponsored in part by the Phillips Laboratory, Air Force Materiel Command, USAF, through the use of the MHPCC under cooperative agreement number F29601-93-2-0001. The U.S. Government is authorized to reproduce and distribute reprints notwithstanding any copyright notation thereon. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Phillips Laboratory or the U.S. Government."

[2]LaTeX document preparation system was developed by Leslie Lamport as a special version of Donald Knuth's TeX program for computer typesetting. TeX is a trademark of the American Mathematical Society. The LaTeX macro package for the New Mexico Institute of Mining and Technology dissertation format was adapted by Gerald Arnold from the LaTeX macro package for The University of Texas at Austin by Khe-Sing The.

ABSTRACT

I developed a geostatistical method for conditioning randomly heterogeneous transmissivity fields on time-dependent data of hydraulic head in bounded, two-dimensional, unsteady aquifer flow. I investigated conditioning on head data through time by Monte Carlo simulations using synthetic data sets for evaluating the methodology's ability to accurately model heterogeneous transmissivity fields. In general, the model shows that conditioning on time-dependent head data results in a reduction in the estimation variance of transmissivity and improvement in the replication of heterogeneities. Advective particle tracks are seen to be constrained to more narrow pathways by conditioning on head data in unsteady flow. Application of this stochastic inverse model to real aquifers is restricted primarily by the assumed knowledge of boundary conditions and parameter covariance structure. Through a case study using actual data from the Albuquerque Basin in Central New Mexico some of the difficulties inherent in applying stochastic modeling approaches to non-idealized situations are revealed. Despite these difficulties, the model can be useful for the design of data networks in evaluating the importance of regional transient flow on solute transport paths.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The ability to numerically solve the equations modeling groundwater flow became readily available with the advent of digital computers. Finite difference and finite element methods provide approximations to the partial differential equations governing flow with a precision proportional to the level of spatial and temporal discretization. Complicated boundary conditions and source/sink terms can be easily incorporated into the numerical schemes allowing the modeler to represent natural systems more closely than with analytical solutions, which require many simplifying assumptions. Increasing model complexity, however, demands more information about the model parameters and increases the effort of calibrating the parameters in order to predict (using the model) observations of the state of the natural system. For an aquifer which is areally extensive and artesian, the model may treat groundwater flow as essentially horizontal and the parameters, consisting of storage coefficient (S), and transmissivity (T), are vertically integrated. Additionally, parameters of boundary and initial conditions, and source/sink recharge or pumping or evapotranspiration type terms are needed. Given these parameters, the flow model can provide the state of the system, i.e. the vertically averaged hydraulic head, at every map location for all times – this is usually called the forward problem. The state of the real aquifer system is the water-level elevation measured

at well locations. Incorporating this head data into an estimate of the model parameters, in a way that is in some sense optimal, is the essence of the inverse problem. Solving directly for the parameters given data on head is an ill-posed problem afflicted by nonuniqueness and instability. Incorporating additional information or making restrictive assumptions to better condition the inverse problem is often left to the judgment of the modeler. The effect of these judgments on the reliability of the model's predictions is difficult to quantify.

Fundamental to the importance of models is not only the best prediction, but also an estimate of how good that best prediction is. While model results may be useful in understanding the natural system and its sensitivity to conditions represented by the input parameters, for the purpose of making decisions a model prediction must be accompanied by a measure of its reliability. This reliability or confidence can be expressed in terms of a probability. Instead of reporting that a groundwater contaminant will arrive at a regulation boundary in a given number of years, the model would give the probability that the contaminant will arrive at the regulation boundary in less than a given number of years. Instead of simply predicting the head at some location and time the model would also give the distribution of head or summary statistics for that distribution given, of course, model assumptions. Thus the first and second moments (the mean and the variance of the state of the system) are among the important results of a groundwater model. The mean hydraulic head represents the model's prediction of the most likely state of the aquifer flow system and the variance is a measure of the uncertainty in this prediction.

Uncertainty enters into the groundwater flow model from many sources.

First is the conceptual model applicable to the system being studied? Each additional complexity, be it leaky confining layers, vertical flow, consolidating aquifer matrix, etc. would need to be incorporated into the model to see if significant error is introduced by neglecting it, and each new model aspect would necessitate more parameter estimation with associated parameter uncertainty. The question of model adequacy in describing the salient features of the natural system being studied is beyond the scope of this research. Uncertainty as referenced here, then, should thus be interpreted to mean uncertainty given a particular conceptual model. Once the conceptual model has been chosen, types , locations, and magnitudes of boundary conditions must be determined. Spatial and temporal variability on a boundary could be a major feature of the natural system behavior. Significant errors may be introduced by attempting to model a third type boundary, for example, as either prescribed flux or prescribed head and uncertainty exists in exactly what flux or head along a boundary should be prescribed or exactly where to place the model boundary. Loads on the aquifer may occur in many forms including private and municipal wells, spatially distributed recharge, infiltration from unlined ponds or irrigation, phreatophyte evapotranspiration and so on. All of these loads must somehow be located and quantified in order to describe groundwater flow in the aquifer. Acknowledging all these sources of uncertainty still leaves uncertainty in estimating the storativity and transmissivity parameters. The second of these, uncertainty in the transmissivity parameter, is the main focus of this research.

Aquifer transmissivity is most commonly determined by monitoring water levels during a pumping test. Long term pumping tests give an average of

permeability over a fairly large volume of aquifer but the actual sample volume is unknown. There is some uncertainty due to measurement and interpretation error but the main source of uncertainty results from the heterogeneity of the porous media which comprises the aquifer matrix. Spatial heterogeneity of the soil and rock through which water flows in the subsurface results in spatial variability of water level throughout the aquifer. Parameterizing a discretized finite difference or finite element model requires specifying a value of transmissivity for each block or element of the discretized grid. A complete characterization of aquifer transmissivity for the purpose of parameterizing a numerical model would require a pumping test for each element or block of the model such that the test measures the appropriate spatial average over the exact volume of each element. It is obvious that this is impossible. A common situation is that only a small number of pumping tests are available from a few widely scattered wells. Much more common are data on head in the aquifer which can be obtained by simply measuring water level in wells. It is desirable, therefore, to utilize all of the data on both head and transmissivity so that an estimate is made of transmissivity for every element in the numerical grid where the estimates agree with all of the available transmissivity data. The first moment, or mean of the estimated transmissivity field should match data values to within measurment error at the locations of pumping tests and the second moment, or variance at the data locations should represent measurement error. Away from the data locations the estimation variance represents uncertainty in the actual transmissivity value due to spatial heterogeneity. The numerical model, parameterized with this estimate of transmissivity, should then produce simulated hydraulic heads which closely match data on water levels in wells. More

precisely, the model gives the joint conditional distribution of the transmissivity and head given the data. One advantage of such a model is that, in order to reduce the uncertainty in the estimation of transmissivity, additional data from water level measurements could be included instead of more expensive, time-consuming pumping tests.

Spatial heterogeneity of the aquifer media, as represented by the transmissivity parameter, can be characterized by a mean, variance, and correlation structure or covariance. Similarly, the aquifer head has a mean, variance, and correlation behavior which is related to the parameters and the boundary conditions through the groundwater flow model. Uncertainty in the transmissivity parameter propagates through the groundwater flow model and results in uncertainty in the aquifer head. That component of uncertainty which is due to transmissivity heterogeneity can be isolated by considering boundary conditions, load, and storativity as deterministic with no random component. This uncertainty can be simulated by generating realizations of transmissivity with the variance and correlation structure believed to represent the aquifer. Each realization of transmissivity results in a realization of head when used to parameterize the groundwater flow model and, with many simulations, produces a probability distribution of transmissivity and head representing uncertainty due to heterogeneity of transmissivity. The transmissivity realization can be conditioned to match transmissivity data by means of the presumed covariance structure. Conditioning the transmissivity realization on head data, however, requires knowledge of the statistical relationship between head and transmissivity. That relationship can be approximated by using the cross-covariance between head and transmissivity and the covariance of the aquifer head.

The main contribution of this research is a numerical method for calculating an approximation to the cross-covariance and head covariance for a groundwater flow model in both time and space. Covariance and cross-covariance are then used to condition a realization of transmissivity on data of both transmissivity and head measurements recorded through time. Certainly the resulting probability distributions of transmissivity and head, jointly conditioned on data of head and transmissivity, do not represent a true measure of uncertainty in the groundwater flow model as it relates to model predictions or to the decisions made based on these predictions. The resulting probability distributions do, however, bear directly on the relative merit of data in model design since the uncertainty is conditioned by the data. With this caveat in mind then the 10 most significant things, listed in no particular order, from this research are as follows.

1. A new method is presented for calculating transient head covariances and head with transmissivity cross-covariances using a block-centered finite difference algorithm.

2. This block-centered finite difference algorithm takes advantage of the smoothness of the covariance functions to reduce the computations by means of a domain decomposition and quadratic interpolation.

3. Covariances obtained via first-order perturbation linearization of the governing equation neglect the nonlinearities of higher order terms. An iterative method is introduced which seeks to recapture these higher order terms by repeatedly solving the groundwater flow and cokriging equations.

4. A random realization of the spatial heterogeneous transmissivity parameter can then be conditioned to match head data to within a limited range of error.

5. Given simulations conditioned on head data, the effect of steady state and transient head data on advective transport predictions can be evaluated in a heterogeneous flow domain.

6. Advective transport predictions are seen to be more sensitive to head data conditioning than either head or transmissivity predictions. In particular, particle trajectories are focused into narrow zones by the head data conditioning.

7. The relationship between time of head data sampling and duration of the transient regime is important. Head data measurements collected early in the transient have much less conditioning impact on transport predictions than those collected midway into the transient flow regime.

8. Cross-covariances of head data locations in time extract more information from time-dependent head data than do merely the head covariances in space.

9. The trade-off is that cross-time head covariances expand the size of the cokriging matrices such that any marginal improvement from additional times of head data is overwhelmed by the additional computational cost.

10. Finally the choice of conceptual model, loads and boundary conditions is an important consideration in a practical application.

Chapter 2

Literature Review

Stochastic subsurface hydrology, that aspect of groundwater science which deals specifically with uncertainty in models of flow and transport, has undergone nearly exponential growth since its inception in the 1970's. *Dagan* [1986] estimated that over 100 articles related to stochastic subsurface hydrology were published between 1975 and 1983 in *Water Resources Research* alone. The journal *Stochastic Hydrology and Hydraulics* is devoted exclusively to publication of research in stochastic hydrology. Textbooks which provide excellent coverage on the scope of the subject have been authored by *Gelhar* [1993] and *Dagan* [1989]. Given this wealth of literature and reviews of the field by *Dagan* [1986], *Gelhar* [1986], *Ginn and Cushman* [1990], *Gutjahr and Bras* [1993] among others, the purpose of this review is merely to highlight work which is directly applicable to the present study. Emphasis in this review is on non-steady flow conditions, the geostatistical approach to the inverse problem, and the effect of conditioning on data.

Allan Freeze [1975] performed one of the early studies on the distribution of hydraulic head for transient flow through heterogeneous porous media using a numerical solution to the one-dimensional flow equation. *Freeze* found the interesting result that it may not be possible to define an equivalent uniform porous media with a single value for the permeability flow parameter

which represents a heterogeneous porous media under transient flow conditions. While this paper is a significant conceptual milestone in the treatment of porous media heterogeneity, *Gelhar et al.* [1977] pointed out that *Freeze* had neglected the important spatial correlation of the permeability parameter in his conceptual analysis. *Gelhar et al.* [1974], *Bakr et al.* [1978], and *Gutjahr et al.* [1978] developed spectral analysis techniques using a small perturbation linearization of the groundwater flow equations which specifically incorporates the spatial correlation of the permeability parameter by means of a spectral density function and the cross-spectral density of permeability with hydraulic head in the flow system. Under conditions of statistical homogeneity and for restricted forms of the spectral density function a unique representation of the variance and covariance of head in a two-dimensional aquifer can be obtained [*Mizell et al.*, 1982]. The small perturbation linearization limits this analysis to situations where the variance of the log transformed permeability parameter is less than 1.0 [*Dagan*, 1985]. Covariance of the longitudinal component of specific discharge has been seen to be reliable up to variance of 3.0 in the log conductivity, however, with the errors dependent on domain size [*Van Lent and Kitanidis*, 1996]. Restriction of spectral analysis to an infinite domain was removed by *Rubin and Dagan* [1988, 1989] who showed that the effect of variable boundary conditions are negligible about 3 correlation's lengths of the permeability parameter away from the boundary. Substantial limitations to the spectral approach are that flow conditions must be at steady state and the mean head gradient must be constant.

An alternative to the small perturbation linearization approach is the sensitivity of system state to parameters which can be obtained by truncation of

Taylor series expansions of parameter and state variables. A first order analysis of the second moment, i.e. uncertainty, and a second order analysis of the first moment was performed by *Dettinger and Wilson* [1981]. While they did not specifically include spatial correlation of the parameters as a random stochastic process their method can, in principle, incorporate spatial correlation and should yield results equivalent to the first-order perturbation linearization. The required sensitivity derivatives can be computed by an adjoint method [*Sykes et al.*, 1985]; however, since derivatives are required for head at every node of a discretized grid with respect to every parameter, for a numerical scheme with many elements the sensitivity matrices can become very large. *Townly and Wilson* [1985] address this concern and manage to reduce the number of terms actually stored in computer memory by taking advantage of the sparseness of the sensitivity matrix. Given a covariance matrix of the parameters, then the covariance of head at some time can be expressed in terms of the sensitivity matrix. Further simplifications are obtained by considering the effect of parameters on head for a single time step only, effectively treating the head at the previous time step as being independent of the current time step value. Whether this simplification is justified in all cases is not discussed nor can the cross-covariance of head at one time with head at another time be obtained. *Wilson and Dettinger* investigated the information content of head measurements for both steady state and transient flow conditions with a Kalman filter and concluded that in rapidly responding, highly stressed systems the information content of transient head measurements is significantly increased. *Townly and Wilson* [1983] conditioned on transmissivity data by kriging and incorporated head data with a weighted least-squares method. The addition of head

data was seen to reduce the estimation variance of transmissivity and improve model predictions.

The inverse problem of incorporating head data (i.e. information on the system state) into estimation of parameters and estimation of prediction uncertainty has been approached several different ways. There is both the problem of optimal estimation of parameters, and thereby optimal model prediction, and the problem of parameter estimation uncertainty and the resulting model prediction uncertainty. Furthermore, a conditioned model should actually duplicate measurements such that the optimal parameter estimate matches measured parameter data, the model produces heads which match measured head data, and estimation or prediction variance is equal to measurement error at the data locations. The optimal parameter estimate tends to be a smoothly varying field whether determined by weighted least squares, maximum likelihood, or some other statistical technique, while the true underlying reality tends to be irregularly variable. Since the detail of the actual variability is unknown it is characterized as a spatially distributed random variable [*Matheron*, 1973] with a probability function multivariate on spatial location. Stochastic simulation is the process of generating equally probable realizations from the governing random function which exhibit irregular variability characteristic of the natural system. The geostatistical method of stochastic simulation which is the focus of this research differs philosophically from the weighted least squares methods of, for example, *Cooley* [1985] or the maximum likelihood method of, for example, *Carrera and Neuman* [1986] which strive to optimize the best parameter estimate. The inverse problem of incorporating head data in this context can be stated then as drawing random simulations from the probabil-

ity distribution of the parameter field which match all data on that parameter and, given the groundwater flow model, result in a system state that matches all data on hydraulic head [*Gutjahr et al.*, 1994].

Conditioning a single random field on data is a straight-forward geostatistical routine [*Deutsch and Journel*, 1992]. Jointly conditioning correlated random fields on data of each, however, such that statistical and physical relationships between the fields are maintained, is much more difficult. An early attempt to jointly condition random transmissivity fields on data of both transmissivity and head was that of *Delhomme* [1979]. The basic idea of *Delhomme's* method was to repeatedly generate random simulations, condition them on the transmissivity data, parameterize a groundwater flow model with the conditioned transmissivity field, and save only those fields which resulted in heads which matched the head data sufficiently closely. Needless to say, this method is extremely demanding of computer time in order to obtain a statistically significant number of head fields which match the data. The method is used in the research presented here as a check on conditional probability distributions only for steady state head, as its use for transient flow conditions was seen to currently be beyond the capabilities of the computer hardware. Another method by *Clifton and Neuman* [1982] used the difference in kriged head and transmissivity at the data locations in the minimization of an objective function. While this method results in significant reduction in conditional head variance, it ignores the cross-covariance structure of head and transmissivity and results in overly smoothed head and transmissivity fields. The pilot-point method of *Marsily et al.* [1984] and *LaVenue and Pickens* [1992] involves adding artificial data of transmissivity at points where the model is most sensitive and adjusting

them to improve the match to head data. This method has the advantage of being able to incorporate time-dependent head data but is, again, demanding of computational resources. *Dagan* [1982] investigated the effects of conditioning on head and log transmissivity variance for steady flow using a small perturbation analytical technique. While useful analytical expressions are given, the conditioning emphasis is primarily on transmissivity data. Conditional variance of hydraulic head is given in terms of the Green's function which may, for some two-dimensional bounded domains, not be particularly easy to compute. The effect of head data on the inverse problem of identifying transmissivity has been examined by *Dagan* [1985] through small perturbation development of the head - log transmissivity cross-covariance relationship for steady flow. The difficulty of evaluating the Green's function for bounded domains is circumvented by localizing the problem and substituting the Green's function for an infinite domain. While transmissivity data is seen to have the most influence on reducing variance, the effect of head data on reducing uncertainty in log transmissivity is also shown to be important.

Two-dimensional transient aquifer flow was specifically examined for an unbounded aquifer domain undergoing slowly varying transients by *Dagan and Rubin* [1988]. Their method allows for the stochastic estimation of aquifer recharge as a function of time only and storativity is considered constant. Assuming, however, a quadratic trend in mean head and unconditional head moments effectively constant in time appears to be a significant restriction. Earlier work by *Dagan* [1982] indicated that steady state analysis of three-dimensional flow may be adequate but for two-dimensional flow where there is a large correlation scale for the transmissivity parameter, such as regional aquifer flow under

Chapter 3

Methods

The quest of this research effort is for a stochastic conditional simulation model of transient flow in a regional basin aquifer. Boundary conditions, storativity, and source/sink loads will be treated deterministically so that the only random parameter is spatially heterogeneous transmissivity. Transmissivity represents the permeability of the porous aquifer matrix, with respect to the fluid properties of water, integrated vertically over the thickness of the saturated zone of the aquifer. Data on transmissivity is commonly obtained from pumping tests where a well is pumped for some period of time and water levels in the well or nearby observation wells are monitored. Measurements thus obtained often show less variability than, for example, measurements made in the laboratory on samples of the undisturbed aquifer matrix or measurements obtained by slug injection or withdrawal, because the pump test represents an average over a greater volume of aquifer porous material. While the true sample volume of a pumping test is unknown, treating transmissivity data as point measurements makes sense only if the scale of the model is much greater than the scale encompassed by the pumping test. A heterogeneous property averaged over larger volumes of sample shows less variability and a longer correlation scale [*Journel and Huijbreghts*, 1978] than does the theoretical underlying point-scale random field. Using a finite element or finite difference discretiza-

tion for a numerical solution can introduce an artificial correlation between transmissivity values unless the element size is much smaller than the correlation scale of the transmissivity field [*Dagan*, 1985]. An assumption of statistical second-order stationarity is made here in describing the spatial heterogeneity of the transmissivity. Essentially this means that the statistical properties are not a function of location but only of separation distance between locations in the field. In an attempt to circumvent some problems of scale, the method described here discretizes a block-centered finite difference grid so that the block size approximates the scale of the sample volume of pumping tests. Alternatively, the point-scale process could be integrated over the block domain but there remains the problem of initially determining that point scale covariance. Concerning spatial discretization, there needs to be at least 10 blocks of the finite difference grid per correlation scale to adequately model the spatial variability [*Van Lent and Kitanidis*, 1996] and several correlation lengths per scale of the bounded domain.

This chapter develops the theory and an algorithm for a geostatistical inverse method of conditioning random simulations of perturbations in the log-transformed transmissivity parameter on measurements of head made in both space and time. Equations for covariances are approximated by a small-perturbation linearization of the two-dimensional vertically integrated groundwater flow equation. The covariance equations are then solved, along with equations for head and mean head, by a block-centered finite difference numerical method, taking advantage of the smoothness of the covariances to reduce matrix dimensions. Conditioning on data is accomplished by cokriging using the covariances obtained from the numerical solution, and head is solved

directly from the cokriged transmissivities and the groundwater flow model. An improvement to the match between heads resulting from this solution to the flow model and actual head data is accomplished by iteratively cokriging on the differences between model heads and the data and solving the flow equation.

3.1 Mathematical Development

A model for two-dimensional laminar flow through a porous media saturated with water expressing conservation of mass and momentum is

$$S\frac{\partial\phi(\mathbf{x},t)}{\partial t} - \nabla\cdot(T(\mathbf{x})\nabla\phi(\mathbf{x},t)) = R(\mathbf{x},t) \tag{3.1}$$

where

S is storage coefficient, storativity,

$\phi(\mathbf{x},t)$ is hydraulic head $[L]$,

$T(\mathbf{x})$ is spatially heterogeneous transmissivity, $\left[\frac{L^2}{t}\right]$,

$R(\mathbf{x},t)$ is a spatially uniform source or sink, $\left[\frac{L}{t}\right]$,

$\mathbf{x}$ is the coordinate vector of spatial location, and

t is time.

Storativity in this model is taken to be a constant. Sources and sinks, such as recharge or evapotranspiration, can vary in space and time but are presumed to be known. Transmissivity is a spatially heterogeneous parameter which is unknown, except at data locations of pumping tests, and is modeled statistically

as a second-order stationary random field. Uncertainty in the transmissivity parameter then propagates through the model and results in uncertainty in the hydraulic head. Head also is measured only at well locations but through time as well as in space. This model can be applied to basin-scale unsteady flow in confined aquifers or in unconfined aquifers where the change in saturated thickness is negligible. This model is not valid where there is a significant component of vertical flow, such as may be the case near point sources, sinks, or boundaries or where there is a significant change in saturated thickness.

Boundary and initial conditions are required in order to complete the model. The general boundary condition can be formulated as

$$a\nabla\phi \cdot \mathbf{n} + b\phi = c \tag{3.2}$$

in which $\mathbf{n}$ is a unit vector outward normal to the boundary and a, b, and c are constants whose values determine the boundary conditions. The types of boundaries which will be considered here are where $a = 0$ and $b = 1$ (type 1), and where $a = 1$ and $b = 0$ (type 2). The value of c will be treated as deterministic along the boundary but will be allowed to possibly change with time to create transient flow conditions. Spatial location of the model boundaries is an extremely important aspect of designing a model to represent some real aquifer system. For this research, however, boundaries are considered to be of known location, type, and magnitude. The initial condition will be determined by applying the load, R, given boundary conditions, and solving (3.1) with $S = 0$.

Randomness is introduced and the model is linearized by a first-order small perturbation expansion. The log-transformed transmissivity is expanded

into the sum of a constant mean, $F = E\{ln(T(\mathbf{x}))\}$, and small mean zero perturbation, $f(\mathbf{x})$. Likewise, hydraulic head is the sum of the mean head, $H(\mathbf{x},t) = E\{\phi(\mathbf{x},t)\}$, and a small mean zero perturbation in head, $h(\mathbf{x},t)$. Substituting the expressions $ln(T) = F + f$ and $\phi = H + h$ into (3.1) and dropping all products of the perturbed quantities, a first-order expression for the mean head is obtained as

$$S\frac{\partial H(\mathbf{x},t)}{\partial t} - T_G\nabla^2 H(\mathbf{x},t) = R(\mathbf{x},t) \tag{3.3}$$

and a stochastic partial differential equation for the head perturbation as

$$S\frac{\partial h(\mathbf{x},t)}{\partial t} - T_G\nabla^2 h(\mathbf{x},t) = T_G\nabla\cdot(f(\mathbf{x})\nabla H(\mathbf{x},t)) \tag{3.4}$$

where $T_G = e^F$ is the geometric mean of transmissivity. The f perturbations in equation (3.4) are a function of $\mathbf{x}$ only and the h perturbation are a function of $\mathbf{x}$ and t. Multiplying equation (3.4) by $f(\xi)$, a function of a different space variable, and taking the expectation gives a first order approximation to the cross-covariance, $C_{fh} = E\{f(\xi)h(\mathbf{x},t)\}$.

$$S\frac{\partial C_{fh}}{\partial t} - T_G\nabla^2 C_{fh} = T_G\nabla\cdot(C_{ff}\nabla H) \tag{3.5}$$

Similarly, multiplying equation (3.4) by $h(\xi,\tau)$, a function of different space and time variables, and taking the expectation gives a first-order approximation to the head covariance $C_{hh} = E\{h(\xi,\tau)h(\mathbf{x},t)\}$.

$$S\,E\left\{h\frac{\partial h}{\partial t}\right\} - T_G\nabla^2 C_{hh} = T_G\nabla\cdot(C_{hf}\nabla H) \tag{3.6}$$

In this notation, the ∂ and ∇ operate on t and $\mathbf{x}$ of the second subscripted variable in C_{fh}, C_{hh}, C_{hf}, and C_{ff}. The time derivative in (3.6) is written as

an expectation to show that $h(\xi,\tau)$ cannot be moved inside the time derivative term when $\tau = t$.

Boundary conditions for the covariance approximations in equations (3.5) and (3.6) are obtained from the boundary conditions specified for equation (3.1). Type 1 conditions specify a constant head along the boundary. In (3.2) when $a = 0$, $b = 1$, and c is a constant prescribed head on the boundary the expectation

$$E\{f(\xi)h(\mathbf{x},t)\} = E\{f(\xi)\mathbf{c}\} = c\,E\{f(\xi)\} = 0$$

and thus the boundary conditions for (3.5) and (3.6) are prescribed constant covariance equal to zero. Type 2 conditions specify a prescribed flux along the boundary so $\frac{\partial\phi}{\partial\eta} = c$ where $\frac{\partial\phi}{\partial\eta}$ represents the component of $\nabla\phi$ normal to the boundary and c is a prescribed constant. Thus the expectation

$$E\left\{f(\xi)\frac{\partial h(\mathbf{x},t)}{\partial\eta}\right\} = \frac{\partial E\{fh\}}{\partial\eta} = c\,E\{f\}$$

implying that the type 2 boundary condition for equations (3.5) and (3.6) is such that the derivative normal to the boundary is equal to zero.

The load on the system of equations (3.5) and (3.6) is in the form of $T_G\nabla\cdot(C_{ff}\nabla H(\mathbf{x},t))$ where $C_{ff} = E\{f(\xi)f(\mathbf{x})\}$ is the covariance of the second-order stationary field of random perturbations in $ln(T(\mathbf{x}))$. Some stationary covariance models are the gaussian, also referred to as bell-shaped,

$$C_{ff}(\xi,\mathbf{x}) = \sigma_f^2 \exp\left\{-\left(\frac{\mathrm{x}_1-\xi_1}{\lambda_1}\right)^2 + \left(\frac{\mathrm{x}_2-\xi_2}{\lambda_2}\right)^2\right\} \tag{3.7}$$

the exponential,

$$C_{ff}(\xi,\mathbf{x}) = \sigma_f^2 \exp\left\{-\sqrt{\left(\frac{\mathrm{x}_1-\xi_1}{\lambda_1}\right)^2 + \left(\frac{\mathrm{x}_2-\xi_2}{\lambda_2}\right)^2}\right\} \tag{3.8}$$

and the modified Mizell-A [*Mizell et al.*, 1982].

$$C_{ff}(\xi,\mathbf{x}) = \sigma_f^2 \left[\Theta \mathrm{K}_1(\Theta) - \tfrac{1}{2}\Theta^2 \mathrm{K}_0(\Theta)\right]$$
$$\Theta = \tfrac{1.33\pi}{4\lambda}\sqrt{(x_1-\xi_1)^2+(x_2-\xi_2)^2} \tag{3.9}$$

In all of the above models σ_f^2 is the variance of the $\ln(T(\mathbf{x}))$ perturbations and $\mathbf{x} = [\mathbf{x_1}, \mathbf{x_2}]$ and $\xi = [\xi_1, \xi_2]$ are the spatial coordinates of two locations in the field. In the bell (3.7) and exponential (3.8) models λ_1 and λ_2 are the correlation lengths along axes 1 and 2 respectively. The modified Mizell-A (3.9) must be isotropic, $\lambda = \lambda_1 = \lambda_2$, and K_1 and K_0 are modified Bessel's functions of the second kind order 1 and 0. The method presented here is not restricted to just these models but is generally applicable for any valid covariance function.

3.2 Numerical Solution

A solution to the system of equations presented in section 3.1 may be obtained by the standard and well-developed finite difference and finite element methods. Discretizing the bounded domain into N blocks or elements and relating discharge through each results in a system of N equations with N unknown heads which can be expressed in matrix form for a fully implicit time step as

$$\tilde{\mathbf{A}}\,\phi_{\Delta \mathrm{t}} = \mathbf{b} + S\,\phi \tag{3.10}$$

This is the discretized form of equation (3.1) where ϕ is a vector of length N heads at time t, $\phi_{\Delta \mathrm{t}}$ is a vector of N heads at time $t + \Delta t$, $\tilde{\mathbf{A}}$ is a matrix of coefficients incorporating heterogeneous transmissivity, and $\mathbf{b}$ is a vector containing source and sink terms for each block and terms incorporating boundary conditions, all at time $t+\Delta t$ for a fully implicit time step. The storativity term

S is taken to be a scalar constant in this model but could, in general, represent a diagonal matrix of storativity terms, one for each block or element, if such detailed information were available. The time step increment is represented as Δt and is allowed to change with each time step. A changing time step increment permits the use of a smaller time step to capture detail when the system is changing rapidly and the use of a larger time step when the system is changing slowly to avoid extra computations, although there are restrictions on how much the time increment is allowed to change with each step. The matrix $\tilde{\mathbf{A}}$ contains transmissivity, storativity, time step increment, and information on spatial discretization. A detailed description of the matrix equations for solving equation (3.10) by five-point stencil block-centered finite difference solution is presented later in this dissertation. The solution for the mean head, equation (3.3), is similar to (3.10) with ϕ replaced by $\underline{H}$, a vector of length N mean head values, and the matrix $\tilde{\mathbf{A}}$ incorporating the mean transmissivity, T_G, which is constant over the domain.

Spatial and temporal discretization of the small-perturbation linearization equation (3.4) results in a matrix equation of the form

$$\mathbf{A}\,\mathbf{h}_{\Delta t} = \mathbf{W}\,\mathbf{f} + \mathbf{B}\,\mathbf{h} \tag{3.11}$$

where $\mathbf{h}_{\Delta t}$ is a vector of N head perturbations at time $t+\Delta t$, $\mathbf{h}$ is a vector of N head perturbations at time Δt, and $\mathbf{f}$ is a vector of perturbations in $ln(T(\mathbf{x}))$. Matrices $\mathbf{A}$ and $\mathbf{B}$ depend on the method of time integration. I used a fully implicit time step to solve for head and a combination of fully implicit and fully explicit time steps to solve the covariance equations. For a fully implicit time step $\mathbf{A}$ is pentadiagonal with elements consisting of S, geometric mean log

transmissivity, T_G, and grid block dimensions and matrix $\mathbf{B}$ is all zeros except for N diagonal elements of S. Matrix $\mathbf{A}$ becomes diagonal with all elements S for a fully explicit time step and the matrix $\mathbf{B}$ contains S, T_G, and the block dimensions. The product of the matrix $\mathbf{W}$ and the vector $\mathbf{f}$ represents an approximation of the derivative of $f\nabla H$ by finite differencing. A matrix equation approximation to (3.5) is then obtained by multiplying the vector $\mathbf{f}^{\mathrm{T}}$, where $^{\mathrm{T}}$ indicates transpose, by $\mathbf{h}_{\Delta t} = \mathbf{A}^{-1}\mathbf{W}\mathbf{f} + \mathbf{A}^{-1}\mathbf{B}\mathbf{h}$ and taking the expectation.

$$E\left[\mathbf{h}_{\Delta t}\,\mathbf{f}^{\mathrm{T}}\right] = \mathbf{A}^{-1}\mathbf{W}\,E\left[\mathbf{f}\,\mathbf{f}^{\mathrm{T}}\right] + \mathbf{A}^{-1}\mathbf{B}\,E\left[\mathbf{h}\,\mathbf{f}^{\mathrm{T}}\right] \tag{3.12}$$

Similarly, multiplying the head perturbation vector $\mathbf{h}^{\mathrm{T}}$ at time t by $\mathbf{h}_{\Delta t} = \mathbf{A}^{-1}\mathbf{W}\mathbf{f} + \mathbf{A}^{-1}\mathbf{B}\mathbf{h}$ and taking the expectation gives a matrix form of (3.6)

$$E\left[\mathbf{h}_{\Delta t}\,\mathbf{h}^{\mathrm{T}}\right] = \mathbf{A}^{-1}\mathbf{W}\,E\left[\mathbf{f}\,\mathbf{h}^{\mathrm{T}}\right] + \mathbf{A}^{-1}\mathbf{B}\,E\left[\mathbf{h}\,\mathbf{h}^{\mathrm{T}}\right] \tag{3.13}$$

where $E\left[\mathbf{f}\,\mathbf{h}^{\mathrm{T}}\right] = \left(E\left[\mathbf{h}\,\mathbf{f}^{\mathrm{T}}\right]\right)^{T}$. The head covariance at time step $t+\Delta t$ can then be obtained by the multiplying the vector $\mathbf{h}_{\Delta t}^{\mathrm{T}}$ by $\mathbf{h}_{\Delta t} = \mathbf{A}^{-1}\mathbf{W}\mathbf{f} + \mathbf{A}^{-1}\mathbf{B}\mathbf{h}$ and taking the expectation

$$E\left[\mathbf{h}_{\Delta t}\,\mathbf{h}_{\Delta t}{}^{\mathrm{T}}\right] = \mathbf{A}^{-1}\mathbf{W}\,E\left[\mathbf{f}\,\mathbf{h}_{\Delta t}{}^{\mathrm{T}}\right] + \mathbf{A}^{-1}\mathbf{B}\,E\left[\mathbf{h}\,\mathbf{h}_{\Delta t}{}^{\mathrm{T}}\right] \tag{3.14}$$

The solution to these equations can be obtained in steps starting with an initial steady state condition. Using one of the input C_{ff} covariance models (3.7), (3.8), or (3.9) to construct the matrix $E\left[\mathbf{f}\,\mathbf{f}^{\mathrm{T}}\right]$ the matrix $E\left[\mathbf{h}_{\Delta t}\,\mathbf{f}^{\mathrm{T}}\right]$ can be obtained from (3.12) with storativity S set equal to zero. At each subsequent time step, the solution $E\left[\mathbf{h}_{\Delta t}\,\mathbf{f}^{\mathrm{T}}\right]$ is used as input to the next time step with storativity set to S. The transpose of the solution to (3.12)

is used as input to (3.14) with S set to zero to obtain the initial steady state head covariance. This initial head covariance matrix and the transpose of the solution from (3.12) are used as input in (3.13) to obtain the cross-time head covariance matrix $E\left[\mathbf{h}_{\Delta t}\,\mathbf{h}^{\mathrm{T}}\right]$. The transpose of this cross-time head covariance matrix finally becomes input into (3.14) to arrive at the head covariance for the next time step. Note that the cross-time head covariance matrices for head perturbations at some time t with all time steps up to another time $\kappa\Delta t$ can be obtained from (3.13).

3.3 Conditioning by Cokriging

One of the applications of the covariance approximations developed in section 3.1 is that they can be used for conditioning randomly simulated fields jointly on data of head and transmissivity by linear cokriging estimation. Given a data vector of transmissivity and hydraulic head measured at spatial locations, the cokriging estimator of the perturbation in $ln(T(\mathbf{x}))$ at some location $\mathbf{x_0}$ is the sum of the product of the data and the weight applied to each datum.

$$\hat{f}(\mathbf{x_0}) = \mathbf{f_d}\,\lambda_\mathbf{f} + \mathbf{h_d}\,\lambda_\mathbf{h} \tag{3.15}$$

The vectors $\lambda_\mathbf{f}$ and $\lambda_\mathbf{h}$ contain cokriging weights applied to the perturbation data in $\mathbf{f_d}$ and $\mathbf{h_d}$. The perturbation data in $\mathbf{f_d}$ are obtained by subtracting F from each pumping test datum location

$$\mathbf{f_d} = [ln(T(\mathbf{x_1})) - F, \ldots, ln(T(\mathbf{x_{n_f}})) - F] \tag{3.16}$$

where n_f is the number of spatial locations of pumping test data. Similarly, the head perturbation data is obtained by subtracting the mean from head data at

each water level measurement location,

$$\mathbf{h_d} = [\phi(\mathbf{x_1}, t_k) - H(\mathbf{x_1}, t_k), \ldots, \phi(\mathbf{x_{n_h}}, t_k) - H(\mathbf{x_{n_h}}, t_k)] \tag{3.17}$$

where n_h is the number of water level measurement locations and t_k indicates the time of measurement. Mean head at each measurement location, $H(\mathbf{x_d}, t_k)$, is determined from the numerical solution to (3.3). Conditioning on time-dependent head data in $\mathbf{h_d}$ is accomplished by two different methods developed in the following sections. In the sequential iteration approach the head data in $\mathbf{h_d}$ is replaced with current time data at each time step, while in the collateral or simultaneous approach all head data for all time steps is included.

Conditioning on data using the cokriging estimator is accomplished by a standard geostatistical procedure. Let $\mathbf{u}$ represent a random field of transmissivity perturbations with a mean of zero, a specified correlation scale, and specified variance. This field could be generated by turning bands [*Mantoglou and Wilson*, 1982], matrix decomposition [*Davis*, 1987], or other methods. A spectral approach utilizing the Fast Fourier Transform (FFT) [*Gutjahr*, 1989] is used here due primarily to its computational speed advantage and the ease of incorporating an existing subroutine into the algorithm. The cokriged estimator (3.15) at every location in the field is a smooth interpolator which, with no measurement error, passes through all data, $\hat{\mathbf{v}} = \hat{\mathbf{f}}(\mathbf{x_i})$ for $i = 1, \ldots, N$. Sampling the random field $\mathbf{u}$ at the same locations as the f data and replacing the vectors $\mathbf{f_d}$ and $\mathbf{h_d}$ in (3.15) by the samples of the random field and resulting head perturbations results in a smooth cokriged interpolator, $\hat{\mathbf{u}}$, which matches the random field samples at the data locations as shown schematically in Figure 3.1. A random field which is conditioned, i.e. matches the data and

possesses the correct statistical properties, is obtained as

$$\mathbf{v} = \hat{\mathbf{v}} + (\mathbf{u} - \hat{\mathbf{u}}) \tag{3.18}$$

3.3.1 A time-sequential approach to conditioning on head data

Two different approaches to conditioning on time dependent head data are investigated in this research. The method described in this section is a direct extension of the iterative technique of *Gutjahr et al.* [1994] from steady state analysis to time-step conditioning. At the initial steady state, cokriging weights are determined by solving the system of equations

$$\begin{bmatrix} \mathbf{R_{ff}} & \mathbf{R_{fh^k}} \\ \mathbf{R_{h^kf}} & \mathbf{R_{h^kh^k}} \end{bmatrix} \begin{bmatrix} \lambda_{\mathbf{f_i}} \\ \lambda_{\mathbf{h_i^k}} \end{bmatrix} = \begin{bmatrix} \mathbf{R_{ff_i}} \\ \mathbf{R_{h^kf_i}} \end{bmatrix} \tag{3.19}$$

where $\mathbf{R_{ff}}$, $\mathbf{R_{h^kf}}$, $\mathbf{R_{fh^k}}$, and $\mathbf{R_{h^kh^k}}$ are the covariance matrix between f data locations, cross-covariance matrix between f and h data locations at the initial steady state time $k = 0$, and head covariance matrix between h data locations. The matrix $\mathbf{R_{h^kf}}$ is just the transpose of $\mathbf{R_{fh^k}}$. The vectors $\lambda_\mathbf{f}$ and $\lambda_\mathbf{h}^\mathbf{k}$ contain the cokriging weights applied to the f data and steady state h data in estimating a value of $\hat{f}(\mathbf{x_l})$ at location $\mathbf{x_l}$ and vectors $\mathbf{R_{ff_i}}$ and $\mathbf{R_{h^kf_i}}$ are the covariance and cross-covariance between the estimation location and the f and h data locations. Using the cokriging weights from (3.19), the random field, $\mathbf{u}$ is then conditioned on f and initial steady state h data by equation (3.18). The cokriging equations could be reformulated to estimate a head perturbation field but, due to the first-order linearized approximation, continuity between the $ln(T(\mathbf{x}))$ and head perturbations would not be preserved [*Gutjahr et al.*, 1994].

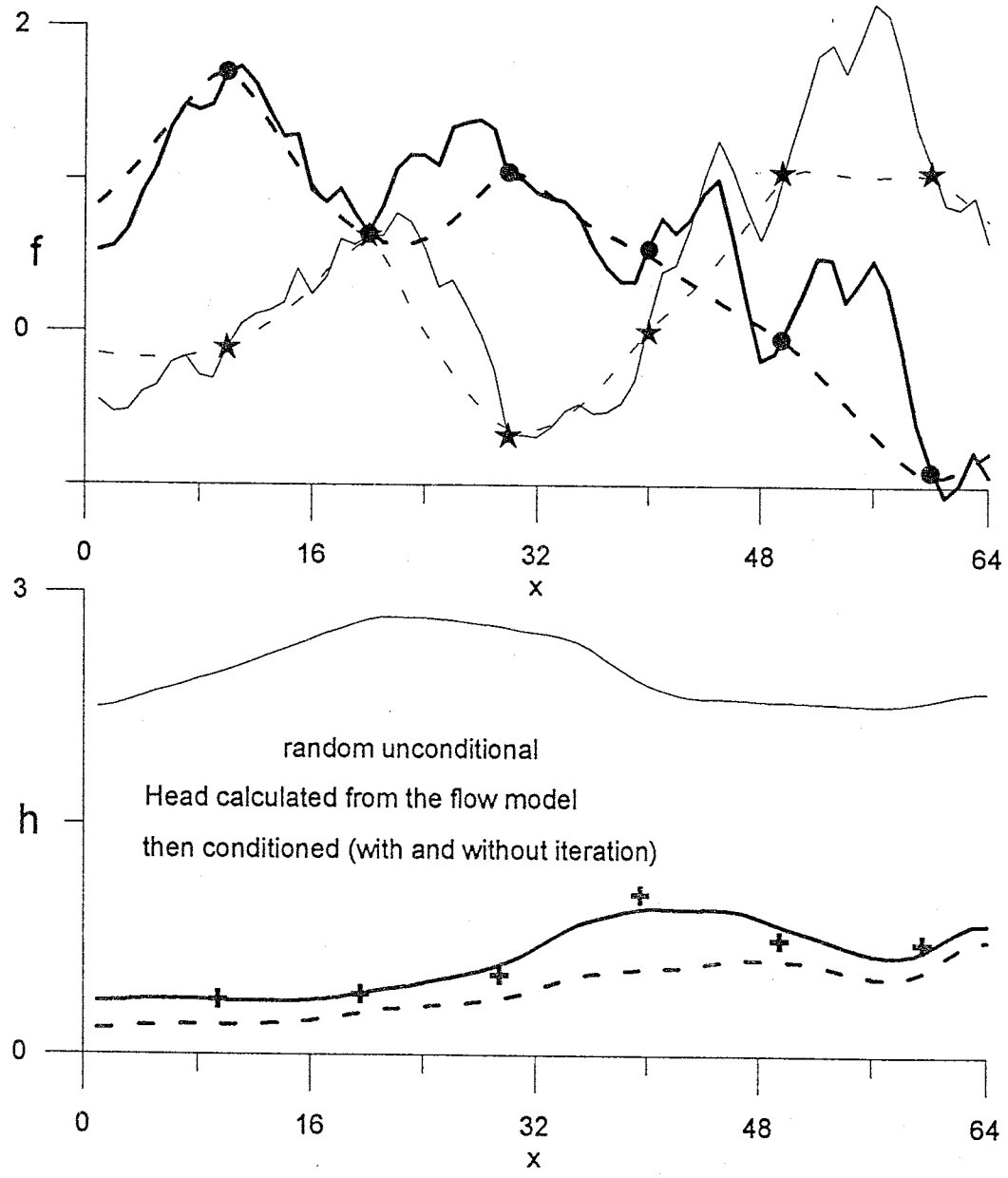


Figure 3.1: A one-dimensional illustration of conditioning a random field on data. The light solid line in the top figure is $\mathbf{u}$ in equation (3.18) and the light solid line in the lower figure is the resulting head. Data are indicated by a $\bullet$ symbol for $\mathbf{f_d}$ and $+$ symbols for $\mathbf{h_d}$. Samples of $\mathbf{u}$ at the $\mathbf{f_d}$ locations are indicated with $\star$ symbols. The cokriged fields $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ are the light and heavy dashed lines in the upper figure and $\mathbf{v}$ is the heavy solid line. Conditioned h fields with and without iteration are shown with solid and dashed lines respectively in the lower figure.

In order to obtain a head perturbation field from the conditioned random f field, the flow model (3.1) is solved via (3.10) where each block or element is parameterized with transmissivity $T(\mathbf{x_i}) = \mathbf{exp}\{v(\mathbf{x_i}) + F\}$ and the mean head from (3.3) is subtracted. Calculating the head perturbation field in this manner retains products of perturbations which were dropped in the development of equation (3.11). Equation (3.11) is never actually solved but is shown as a step in the development of equations (3.12), (3.13), and (3.14).

3.3.2 Iterative improvement of the match to head data

The head perturbations obtained from the flow model parameterized with the conditioned f field, do not exactly match the head perturbation data. This is due, in part, to the first-order linearization approximation [*Dagan*, 1985]. As the input variance of the f field, σ_f^2, is reduced the discrepancy between data and the model result also diminishes. A departure from a multivariate gaussian distribution of the correlated f and h variables may also affect this discrepancy [*Gutjahr*, 1984]. The match between head perturbation data and the head perturbations resulting from the flow model parameterized with the conditioned f field can be improved somewhat by an *ad hoc* iterative technique [*Gutjahr, et al.* 1994]. The strategy is to take the difference between the h data and the model results and add this difference to the unconditioned h field samples used in (3.18) to obtain $\hat{\mathbf{u}}$. First of all, the unconditioned random field $\mathbf{u}$ is substituted into the flow model equation solver (3.10) and the resulting perturbations around the mean head are sampled. Call these samples $\mathbf{h_s^{(0)}}$. These samples, and samples of the unconditioned f field, $\mathbf{u}$, are used to create a smooth cokriged interpolator $\hat{u}_i = \mathbf{f_u}\,\lambda_{\mathbf{f_i}} + \mathbf{h_s^{(0)}}\,\lambda_{\mathbf{h_i}}$ for $i = 1, \ldots, N$

and the cokriging weights, λ_{f_i} and λ_{h_i}, are determined from (3.19) for each location i. The first conditioned f field from (3.18) is then substituted into the model equation solver (3.10) and the resulting discrepancy between head perturbation data and the model results are added to $\mathbf{h_u}$. At iteration j, the conditioned f field of perturbations in log transmissivity at location i is

$$\begin{aligned} \mathbf{v_i^{(j)}} &= \left(\lambda_{f_i}\mathbf{f_d} - \lambda_{f_i}\mathbf{f_u} + \lambda_{h_i}\left(\textstyle\sum_{l=0}^{j-1}\mathbf{h_d} - \mathbf{h_s^{(l)}}\right)\right) + \mathbf{u_i} \\ &= \mathbf{v_i^{(j-1)}} + \lambda_{h_i}\left(\mathbf{h_d} - \mathbf{h_s^{(j-1)}}\right) \end{aligned} \tag{3.20}$$

for $i = 1, \ldots N$ where $\mathbf{h_s^{(l)}}$ is a vector of samples of head perturbations resulting from the flow model at iteration l. Once the iteration has converged to some prescribed convergence criterion, the conditioned field is treated as the unconditioned field at the next time step. Conditioning on head data is thus done sequentially since an f field conditioned on all previous times is not conditioned on head data at the next time. For conditioning at the next time step, k in (3.19) is replaced by $k+1$ in order to determine the cokriging weights for each estimation location at that time step.

This approach is adaptive in that the f field changes, or adapts, with each time step in order to get a good match between the head perturbations resulting from the flow model and the head perturbation data. An iteration at each time step is performed to improve the model h and data agreement. Some global convergence properties of this method could be said to exist if the changes in the conditioned f field become smaller with each time step and if the conditioned f field from the last time step resulted in a close match with h data when used in the flow model at earlier times. While the changes in

the f fields between subsequent time steps are small, there seems to be an approximately equivalent change from time step to time step. The final conditioned f, while resulting in a match to h data at the final time step within the convergence criterion, does not result in a close match to h data at any of the earlier time steps.

3.3.3 A collateral approach to conditioning on head data

An adaptive sequential conditioning method, described above, allows for conditioning on head data through an iterative procedure whereby the f field conditioned on f data produces, through the flow model, an h field which very closely matches h data; however, the f field changes at each time step. Furthermore, the f field conditioned on the final time step h data does not accurately reproduce h data at any of the earlier time steps. A more satisfactory solution would be to obtain one f field conditioned on all the h data through time which closely reproduces this h data when used to parameterize the flow model. An approach to arriving at this one conditioned f field is described in this section.

Suppose that there are $0, 1, \ldots, \kappa$ times at which head data measurements are available. A matrix of cokriging equations, similar to (3.19), is

$$\begin{bmatrix} \mathbf{R_{ff}} & \mathbf{R_{fh^0}} & \mathbf{R_{fh^1}} & \cdots & \mathbf{R_{fh^\kappa}} \\ \mathbf{R_{h^0f}} & \mathbf{R_{h^0h^0}} & \mathbf{R_{h^0h^1}} & \cdots & \mathbf{R_{h^0h^\kappa}} \\ \mathbf{R_{h^1f}} & \mathbf{R_{h^1h^0}} & \mathbf{R_{h^1h^1}} & \cdots & \mathbf{R_{h^1h^\kappa}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{R_{h^\kappa f}} & \mathbf{R_{h^\kappa h^0}} & \mathbf{R_{h^\kappa h^1}} & \cdots & \mathbf{R_{h^\kappa h^\kappa}} \end{bmatrix} \begin{bmatrix} \lambda_{\mathbf{f}} \\ \lambda_{\mathbf{h^0}} \\ \lambda_{\mathbf{h^1}} \\ \vdots \\ \lambda_{\mathbf{h^\kappa}} \end{bmatrix} = \begin{bmatrix} \mathbf{R_{ff_i}} \\ \mathbf{R_{h^0f_i}} \\ \mathbf{R_{h^1f_i}} \\ \vdots \\ \mathbf{R_{h^\kappa f_i}} \end{bmatrix} \tag{3.21}$$

where, for example, $\mathbf{R_{h^1h^\kappa}}$ is the covariance of the h data locations at time step 1 with h data locations at time step κ. The cross-time covariances required

in this cokriging formulation are readily calculated using (3.13) as mentioned previously at the end of section 3.2. Using equation (3.13) the cross-time head covariances $E\left[\mathbf{h}_{\Delta t}\,\mathbf{h}^{\mathrm{T}}\right]$ to $E\left[\mathbf{h}_{\kappa\Delta t}\,\mathbf{h}^{\mathrm{T}}\right]$ can be calculated and, from these, the cross-time head covariances for all data locations and all times can be obtained. The weights applied to all data through time are calculated simultaneously and the data vector includes all head data as $z = [\mathbf{f_d}, \mathbf{h_d}]$ where

$$\begin{aligned}\mathbf{h_d} = [&\phi(\mathbf{x_1}, t_0) - H(\mathbf{x_1}, t_0), \ldots, \phi(\mathbf{x_{n_h}}, t_0) - H(\mathbf{x_{n_h}}, t_0), \ldots, \\ &\phi(\mathbf{x_1}, t_\kappa) - H(\mathbf{x_1}, t_\kappa), \ldots, \phi(\mathbf{x_{n_h}}, t_\kappa) - H(\mathbf{x_{n_h}}, t_\kappa)]\end{aligned} \tag{3.22}$$

Note that the matrix and vector formulation presented here is the same as that given by *Myers* [1982] after some rearrangement. The implication of this system of equations is that the location covariances of h data across time as well as in space are required to condition a single f field on temporal head measurements.

The cokriging weights are then used in the iteration described in section 3.3.2 to improve the model match to the head data. The data cokriging matrices (3.21) and (3.19) can be nearly singular. Even using a singular value matrix decomposition method, small errors in the data covariance matrix are magnified and lead to erroneous cokriging results. As pointed out by *Dietrich and Newsam* [1989] the matrix becomes more ill-conditioned as more head data are included. A measurement error term added to the diagonal of (3.21) and (3.19) helps to stabilize the problem.

3.4 A block-centered finite difference algorithm

The mathematical development presented in section 3.1 is general and the discretized matrix equations in section 3.2 are not restricted to any particular solution method. In this section, a five-point stencil block-centered finite difference method is developed and described in detail for solving (3.10). A block-centered finite differencing discretization for solution of the groundwater flow model and first-order covariance approximation equations follows naturally from the method used to generate a correlated random field. In the random field generation algorithm of *Gutjahr et al.* [1996], field dimensions are specified such that the program returns a grid of uniform Δx and Δy spacing with a value of the field specified for each block and an option to integrate the point covariance process over the block area. The random field thus generated can be transferred block by block with no modification to a finite differencing discretization which has the same block size and field dimensions. If the random field were to be used in a grid with a variable block size or in a finite element grid with triangular or irregularly shaped elements, the block values would need to be averaged over each element. This averaging process could be problematic with regard to the concept of an effective transmissivity in transient flow [*Freeze*, 1975; *Gomez-Hernandez and Gorelick*, 1989]. As discussed earlier, the problem of averaging is avoided here by maintaining a block size approximately equivalent to the transmissivity sampling support volume.

3.4.1 Solving for head

Perturbations in the $ln(T(\mathbf{x}))$ field are generated on a uniform grid with n_x blocks along the x_1 axis and n_y blocks along the x_2 or y axis. The

field can be rectangular, that is n_x does not have to equal n_y, but both n_x and n_y must be powers of 2 since the algorithm uses the FFT to generate the field in the spectral domain [*Gutjahr et al.*, 1996]. A field where n_x and n_y are not powers of two can be obtained, however, by padding the grid with zeros between n_x and n_y and the next higher power of two. A finite difference approximation of (3.1) for constant Δx grid spacing in the x_1 direction , constant Δy grid spacing in the x_2 or y direction, and a fully implicit time step can be written as

$$S\frac{\phi_{i,j}^k - \phi_{i,j}^{k-1}}{\Delta t} - \left(\frac{T_r \frac{\phi_{i+1,j}^k - \phi_{i,j}^k}{\Delta x} - T_l \frac{\phi_{i,j}^k - \phi_{i-1,j}^k}{\Delta x}}{\Delta x} + \frac{T_u \frac{\phi_{i,j+1}^k - \phi_{i,j}^k}{\Delta y} - T_d \frac{\phi_{i,j}^k - \phi_{i,j-1}^k}{\Delta y}}{\Delta y} \right) = R_{i,j}^k \tag{3.23}$$

where $\phi_{i,j}^k$ is the head in block i,j at time step k and $R_{i,j}^k$ is the source or sink load on block i,j at time step k. The parameters T_l, T_r, T_d, and T_u are the harmonic averages of the block to block transmissivity values calculated as $T_l = \frac{2T_{i-1,j}T_{i,j}}{T_{i-1,j}+T_{i,j}}$, $T_r = \frac{2T_{i,j}T_{i+1,j}}{T_{i,j}+T_{i+1,j}}$, $T_d = \frac{2T_{i,j-1}T_{i,j}}{T_{i,j-1}+T_{i,j}}$, and $T_u = \frac{2T_{i,j}T_{i,j+1}}{T_{i,j}+T_{i,j+1}}$ where $T_{i-1,j}$, $T_{i+1,j}$, $T_{i,j-1}$, and $T_{i,j+1}$ are the transmissivity values to the left, right, below, and above the transmissivity value, $T_{i,j}$, of the block centered at location $x_{i,j}$.

Boundary conditions require some special treatment in order to maintain equal sizing of the grid blocks [1]. Blocks adjacent to the lower boundary are shown in Figure 3.2. For a type 1 boundary condition, the value of c, shown in Figure 3.2 at location x_b, represents a prescribed head, ϕ. Substituting the value of c into (3.23) for $\phi_{i,j-1}$ and Δy for uniform grid spacing gives

[1]Refer to Appendix A for further discussion

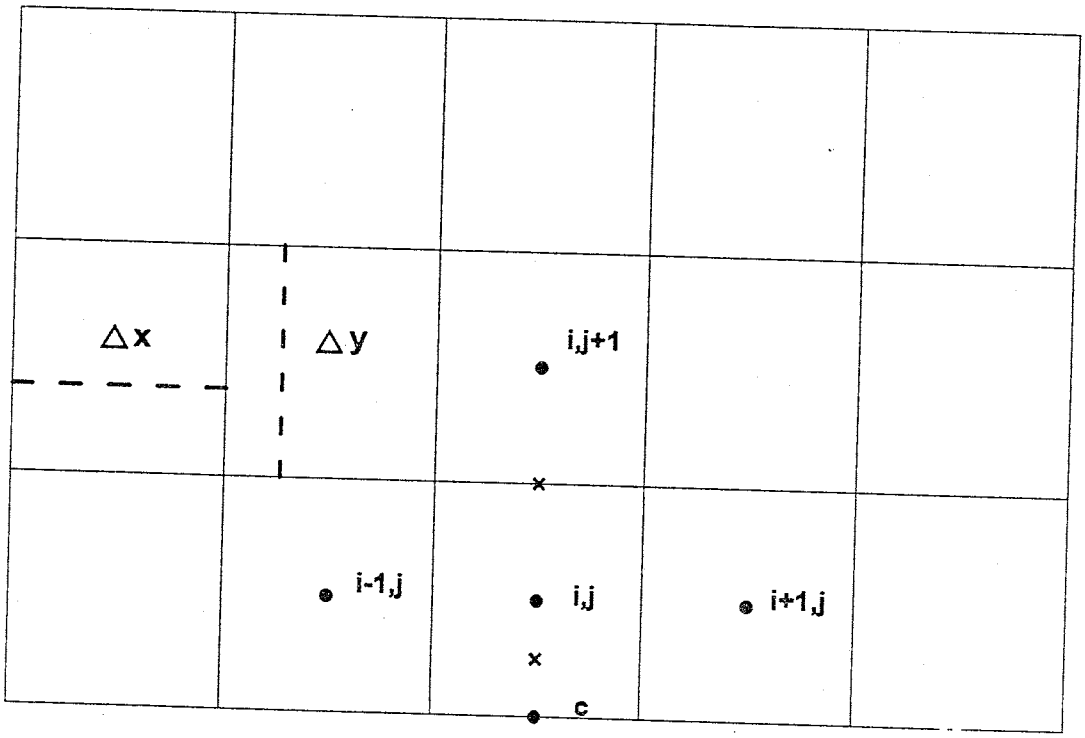


Figure 3.2: A portion of a block-centered finite difference grid showing a few blocks adjacent to the lower boundary. Refer to the text for a description.

$\frac{T_u \frac{\phi_{i,j+1}-\phi_{i,j}}{\Delta y} - T_{i,j} \frac{\phi_{i,j}-c}{\frac{\Delta y}{2}}}{\Delta y}$ for the second term in the parentheses of (3.23). For the case of a type 2 boundary condition, the c term represents the flux per unit length crossing crossing the boundary at x_b and the second term in the parentheses of (3.23) becomes $\frac{T_u \frac{\phi_{i,j+1}-\phi_{i,j}}{\Delta y} - c}{\Delta y}$. Treatment of the remaining boundary conditions follow from these two examples.

Rearranging (3.23) so that all of the unknown head values $\phi^k_{i,j}$, $\phi^k_{i-1,j}$, $\phi^k_{i+1,j}$, $\phi^k_{i,j-1}$, and $\phi^k_{i,j+1}$ are on the left hand side of the equation and the known values $\phi^{k-1}_{i,j}$ and $R_{i,j}$ are on the right hand side results in $N = n_x\, n_y$ equations in N unknown block-centered head values of the gridded domain. Arranging these equations in matrix form results in a pentadiagonal matrix for $\tilde{\mathrm{A}}$ in

equation (3.10). A row of $\tilde{\mathbf{A}}$ representing a block not adjacent to a boundary then contains the term $S+\Delta t\left(\frac{T_l+T_r}{\Delta x^2}+\frac{T_u+T_d}{\Delta y^2}\right)$ on the matrix diagonal, the term $-\Delta t\,\frac{T_r}{\Delta x^2}$ in the column on the right, the term $-\Delta t\,\frac{T_l}{\Delta x^2}$ in the column to the left, the term $-\Delta t\,\frac{T_u}{\Delta y^2}$ in the column n_x columns to the right, and the term $-\Delta t\,\frac{T_d}{\Delta y^2}$ in the column n_x columns to the left. For blocks adjacent to the boundaries these terms are modified according the example shown above in Figure 3.2. The vector $\mathbf{b}$ in equation (3.10) contains the known values of the source or sink terms, $R_{i,j}$, and the known prescribed head or flux quantities of the boundary conditions. The vector ϕ in equation (3.10) contains the head values calculated from the previous time step. For the initial condition, steady state is presumed and the heads are calculated from (3.10) with $S=0$. Calculation of the mean head for each of the N blocks in the gridded domain is accomplished simply by replacing the harmonic averaged transmissivities T_l, T_r, T_d, and T_u with the geometric mean transmissivity, T_G and, of course, replacing the vectors $\phi_{\Delta t}$ and ϕ with the mean head valued vectors. Matrix inversion was implemented with a preconditioned biconjugant gradient algorithm from Numerical Recipes [*Press et al.*, 1992]. This algorithm uses the diagonal part of the matrix $\tilde{\mathbf{A}}$ as the preconditioner.

3.4.2 Solving for covariance

In the previous section, the coefficient matrix $\tilde{\mathbf{A}}$ is of order N but has zero elements everywhere except for the main diagonal and four minor diagonals. Many numerical algorithms for solving sparse matrix systems require only storing non-zero matrix elements, which results in a tremendous savings of computer memory. On the other hand, the covariance and cross-covariance ma-

trices in equations (3.12), (3.13), and (3.14), are full. While some reduction may be possible by taking advantage of the symmetry of the head covariance matrix, $E\left[\mathbf{h}_{\Delta t}\,\mathbf{h}_{\Delta t}{}^{\mathrm{T}}\right]$, the cross-covariance matrices, $E\left[\mathbf{h}_{\Delta t}\,\mathbf{f}^{\mathrm{T}}\right]$ and $E\left[\mathbf{h}_{\Delta t}\,\mathbf{h}^{\mathrm{T}}\right]$, are asymmetric and must be stored entirely in computer memory. For example, a reasonably sized but still relatively small grid with $n_x = n_y = 64$ has 4096 blocks. Each covariance or cross-covariance matrix in equation (3.12), (3.13), or (3.14) thus has $16,777,216$ numbers to be stored. Just this simple problem would require over 400 Mb of computer memory in double precision. Fortunately, however, there is a way out of this dilemma for certain flow model conditions. Where the mean head field is changing slowly, i.e. small gradients, the covariance and cross-covariances are very smooth and can be solved on a coarser grid than the head or random field grids. The covariance and cross-covariances required for cokriging in (3.19) and (3.21) can be obtained then by interpolating between covariance locations on the coarser grid.

The technique implemented here is to solve the covariance and cross-covariance equations (3.12), (3.13), and (3.14) on a grid where each block of the coarser grid contains 16 blocks of the random f and head field grid. Thus in the example where $n_x = n_y = 64$, using a grid of $16x16$ blocks to solve equations (3.12), (3.13), and (3.14) the covariance and cross-covariance matrices are reduced to order 256. This example, which would have taken over 400 Mb of computer memory, can now be accomplished in a little over 1.5 Mb. A quadratic interpolation between coarse grid blocks is then used in the interior of the gridded domain in order to capture some of the curved shape of the covariance. At the corners of the grid, bilinear interpolation is used and, along the edges of the domain, a combination of quadratic and bilinear interpolation

is used. A schematic diagram of the grid is shown in Figure 3.3.

A quadratic interpolation of a polynomial, $f(\zeta)$, between the 3 points (ζ_1, f_1), (ζ_2, f_2), and (ζ_3, f_3) is given by the Lagrange formula

$$f(\zeta) = \frac{(\zeta-\zeta_2)(\zeta-\zeta_3)}{(\zeta_1-\zeta_2)(\zeta_1-\zeta_3)}f_1 + \frac{(\zeta-\zeta_1)(\zeta-\zeta_3)}{(\zeta_2-\zeta_1)(\zeta_2-\zeta_3)}f_2 + \frac{(\zeta-\zeta_1)(\zeta-\zeta_2)}{(\zeta_3-\zeta_1)(\zeta_3-\zeta_2)}f_3 \quad (3.24)$$

This formula is implemented in two-dimensions by repeating one-dimensional interpolations. For cokriging, equations (3.19) and (3.21), it is necessary to find the cross-covariance of all f data locations with all h data locations, the covariance of all h data with all h data locations, and the cross-covariance of f for every block on the fine grid with all h data. For example, to interpolate the cross-covariance between one f and one h datum, the algorithm first locates the coarse grid block containing the f datum and the coarse grid block containing the h datum and the eight other coarse grid blocks surrounding each. Suppose the f datum is surrounded by the coarse grid block centers 1, 2, 3, 4, 6, 7, 8, and 9 shown in Figure 3.3 and the h datum is located elsewhere in the field. There is a cross-covariance for f at block 1 with all nine blocks surrounding the h datum, a cross-covariance of f at block 2 with all nine blocks surrounding the h datum, and so on for a total of 81 cross-covariance block pairs. Each f block with nine h block set interpolation is done as a succession of one-dimensional interpolations, starting with three interpolations in the x_2 direction, to obtain values of the cross-covariance function of f at block 1 with rows of the set of blocks surrounding the h datum. One-dimensional interpolations in the x_1 direction are then used to obtain the cross-covariance function value of f at block 1 with the h datum. When a value of the cross-covariance for all nine f blocks with the h datum is obtained, a final set of one-dimensional

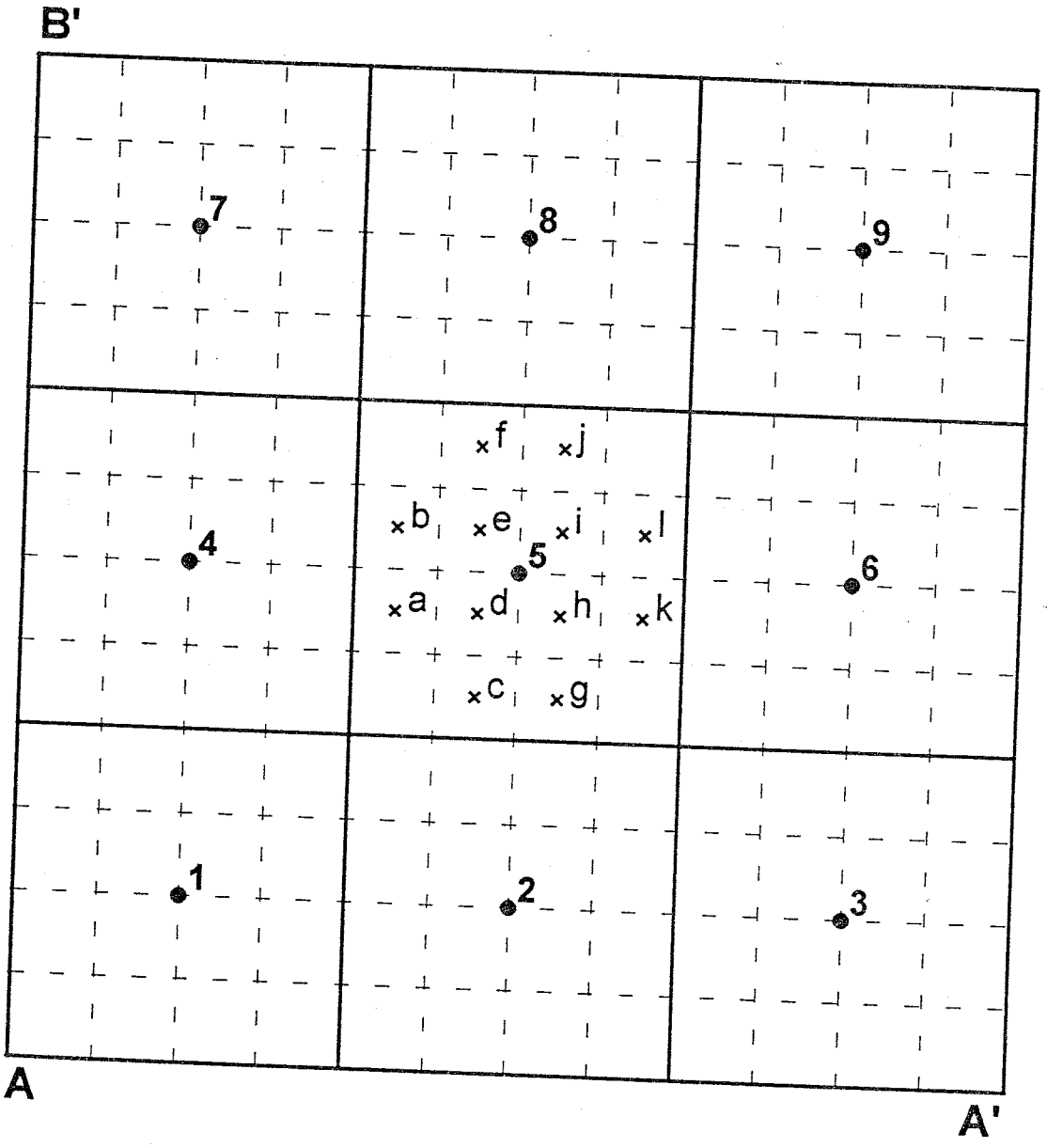


Figure 3.3: A portion of a block-centered finite difference grid showing the coarse grid used to compute covariances and the fine grid of the random f and head solution. The pattern of x symbols around block center 5 represents the fine grid blocks used to approximate the first and second derivatives of mean head.

interpolations is done to find the value of the cross-covariance for the f datum with the h datum. If the line $A-A'$ in Figure 3.3 is the lower domain boundary and the fine grid estimation block or datum is in block 2 or 3 then only blocks 1-6 are used in a combination bilinear quadratic interpolation. If the line $A-A'$ in Figure 3.3 is the lower domain boundary and the line $A-B'$ is the leftmost domain boundary and the datum or fine grid estimation block is in block 1 then only blocks 1,2,4, and 5 are used in a bilinear interpolation. For further discussion of the interpolation algorithm refer to Appendix A, section A.2.

The load term on the right hand side of the first-order covariance approximation equations (3.5) and (3.6) involves the divergence of the product of a covariance and ∇H. In this numerical scheme the derivatives of the product of covariance and ∇H are approximated by finite difference between blocks on the fine grid within each block on the coarse grid. The centers of the fine grid blocks used for this approximation are shown in Figure 3.3 with the small x symbols. For the cross-covariance load of equation (3.5) a derivative of mean head in the x_1 direction is calculated between blocks $\mathbf{x^b}$ and $\mathbf{x^e}$ and between blocks $\mathbf{x^a}$ and $\mathbf{x^d}$. The two finite difference approximations are then averaged and assigned to the corner location common to $\mathbf{x^b}$, $\mathbf{x^e}$, $\mathbf{x^a}$, and $\mathbf{x^d}$. Derivatives of H are also calculated between blocks $\mathbf{x^l}$ and $\mathbf{x^l}$ and blocks $\mathbf{x^h}$ and $\mathbf{x^k}$ with the average assigned to the block corner common to $\mathbf{x^l}$, $\mathbf{x^l}$, $\mathbf{x^h}$, and $\mathbf{x^k}$. Derivatives of H in the x_2 direction are obtained similarly by averaging the difference between blocks $\mathbf{x^f}$ and $\mathbf{x^e}$ with the difference between blocks $\mathbf{x^j}$ and $\mathbf{x^l}$ and the difference between $\mathbf{x^d}$ and $\mathbf{x^c}$ with the difference between $\mathbf{x^h}$ and $\mathbf{x^g}$. The C_{ff} covariance is then evaluated at the common corners of the blocks used for the H derivative approximations and the product $T_G\nabla\cdot(C_{ff}\nabla H)$ is

formed from the difference of $C_{ff}\nabla H$ across $2\Delta x$ and $2\Delta y$. The load term of equation (3.6) is obtained by differencing across coarse grid blocks since the cross-covariance C_{hf} has been evaluated previously at only at the centers of the coarse grid blocks. For the load on block 5, the derivative of H in the x_1 direction is approximated using the four blocks surrounding nodes 2, 4, 6, and 8 by averaging the difference between the upper right block and the upper left block with the difference between the lower right block and the lower left block. The derivative in the x_2 direction is approximated by the average of the difference between the upper right block and the lower right block with the difference between the upper left block and the lower left block. The product $T_G \cdot (C_{hf}\nabla H)$ is then formed by differencing $C_{hf}\nabla H$ between blocks 4 and 6 across $8\Delta x$ and between blocks 2 and 8 across $8\Delta y$. This coarse grid differencing is adequate for smoothly varying H fields due, perhaps, to temporal changes in uniform recharge rates. A source or sink located entirely within one coarse grid block would require some additional computations, however. One possibility is to evaluate C_{hf} at a few extra locations around the localized source or sink on the finer grid and use these to form the load for equation (3.6). Another possibility is to interpolate C_{hf} using the quadratic interpolation method described earlier. Neither one of these possibilities has yet been investigated and may form the subject of future research efforts.

3.4.3 Time integration

A fundamental assumption underlying the development of this transient analysis of head and log transmissivity cross-covariance behavior is that the initial condition is in a steady or quasi-steady state. The initial head in the

aquifer is assumed to be in equilibrium with boundary conditions, recharge, evapotranspiration, other sources or sinks, and the flux is not changing significantly with time. The transient condition is then assumed to be caused by a change in flux due to addition or removal of sources or sinks or changes in conditions at the model boundaries. The matrix of cross-covariance $E\left[\mathbf{h}_{t=0}\mathbf{f}^{\mathrm{T}}\right]$ at steady state is obtained from equation (3.12) with the storage term S set equal to zero. The transpose, $E\left[\mathbf{f}\mathbf{h}_{t=0}^{\mathrm{T}}\right]$ is input into equation (3.14) with $S = 0$ to give the head covariance at steady state. Time dependent covariance and cross-covariances are then arrived at by a combination of implicit and explicit time steps starting from these initial steady state matrices.

First the cross-time head covariance, $E\left[\mathbf{h}_{t=1}\mathbf{h}_{t=0}^{\mathrm{T}}\right]$, is obtained from $E\left[\mathbf{f}\mathbf{h}_{t=0}^{\mathrm{T}}\right]$ and the steady state head covariance using equation (3.13) with an explicit time step. Next the cross-time head covariances from $E\left[\mathbf{h}_{t=2}\mathbf{h}_{t=0}^{\mathrm{T}}\right]$ to $E\left[\mathbf{h}_{t=\kappa}\mathbf{h}_{t=0}^{\mathrm{T}}\right]$ are calculated from equation (3.13) using an explicit time step. This completes the calculation of all cross-time covariances of the initial state head with head at for κ time steps. Returning to equation (3.12), the cross-covariance $E\left[\mathbf{h}_{t=1}\mathbf{f}^{\mathrm{T}}\right]$ is obtained using a fully implicit time step with $E\left[\mathbf{h}_{t=0}\mathbf{f}^{\mathrm{T}}\right]$ as input. The transpose of this and the transpose of the first cross-time head covariance $E\left[\mathbf{h}_{t=0}\mathbf{h}_{t=1}^{\mathrm{T}}\right]$ are then used with a fully implicit time step in (3.14) to obtain the head covariance at time step one. Subsequent cross-time covariances $E\left[\mathbf{h}_{t=2}\mathbf{h}_{t=1}^{\mathrm{T}}\right]$ to $E\left[\mathbf{h}_{t=\kappa}\mathbf{h}_{t=1}^{\mathrm{T}}\right]$ and finally $E\left[\mathbf{h}_{t=\kappa}\mathbf{h}_{t=\kappa-1}^{\mathrm{T}}\right]$ are calculated using a fully explicit time step in equation (3.13). Cross-covariances $E\left[\mathbf{h}_{t=2}\mathbf{f}^{\mathrm{T}}\right]$ to $E\left[\mathbf{h}_{t=\kappa}\mathbf{f}^{\mathrm{T}}\right]$ and head covariances $E\left[\mathbf{h}_{t=2}\mathbf{h}_{t=2}^{\mathrm{T}}\right]$ to $E\left[\mathbf{h}_{t=\kappa}\mathbf{h}_{t=\kappa}^{\mathrm{T}}\right]$ are all calculated using a fully implicit time step with equations (3.12) and (3.14). Stability of the explicit solution, for uniform S, T_G and constant grid

spacing $\Delta x = \Delta y$ in two dimensions requires that the time step Δt must be less than $(S\Delta x^2)/(4T_G)$ [*Rushton and Redshaw*, 1979; *Wang and Anderson*, 1982]. This is a necessary but not necessarily sufficient condition. In practice the time step thus constrained must often be multiplied by a number smaller than one in order to ensure stability. A multiplication factor, determined by trial and error, which made the cross-covariance explicit time-stepping numerical solutions stable was .5. Time steps for both implicit and explicit solutions were set to the stable value and the necessary covariances saved to disk only for times at which head data was available for conditioning.

3.5 Conditional Monte Carlo simulations

The previous descriptions of methodology hold the philosophy that heterogeneity in the permeability parameter, transmissivity, can be modeled as a spatially random field, using a mean and covariance function, and that the spatial covariance structure of transmissivity can be related to covariance of hydraulic head through a groundwater flow model. Covariance of transmissivity can be considered as a description of the uncertainty in this parameter due to unknown spatial variability, and covariance of head as uncertainty in model prediction due to this same unknown spatial variability. That there are many more additional uncertainties in the conceptual groundwater flow model, boundary conditions, and other parameters is just cause for much more research and concern as to the true reliability of flow models. However, given a particular flow model, head data can augment transmissivity data in order to improve the characterization.

The methods presented here focus particularly on further augment-

ing the data set using head measurements through time in a non-steady system with the objective of more accurate characterization of transmissivity heterogeneity. The joint probability of transmissivity and head conditioned on data is constructed by repeated Monte Carlo simulations. Many simulations of the statistically stationary random transmissivity field, on the order of 100 or more, are generated with a specified covariance model using a random field generator [*Gutjahr*, 1989]. In practice the transmissivity covariance model must often be estimated from a limited set of transmissivity measurements which is an additional source of uncertainty. Each simulation is conditioned by cokriging iteratively until the flow model parameterized with the transmissivity simulation, conditioned on transmissivity data, results in head which matches the head data to within a specified criterion. The results of these Monte Carlo simulations conserve mass and momentum and are conditioned (i.e. match) the data to within known limits. They can include, or be presented as, spatial conditional mean and variance of head and transmissivity, conditional mean and variance of velocity, or as the full (empirical) conditional probability distribution at any location in the domain. The full distribution is especially important if the tails are of interest. In the following chapter the effect of conditioning on head data in time and spatial domains is explored through Monte Carlo experiments using a hypothetical model. Methods described in this chapter were implemented in Fortran and are included in Appendix A.

Chapter 4

Results

This chapter presents results of conditioning random simulations on transient head data in hypothetical groundwater flow models. For a hypothetical model all aspects of boundary conditions, sources and sinks, and parameters are known with certainty. Data sets were created by sampling a single random field so the true underlying heterogeneity is also known with certainty. This provides a basis of comparison in order to evaluate the effect of incorporating transient head measurements in the conditioning. Also presented in this chapter are some algorithm validation measures. Even though all aspects of the model, including details of parameter heterogeneity, are known with certainty, the true joint conditional probability distribution remains unknown. These questions of true joint conditional distribution and the verification of linearized covariance approximations were addressed by means of extensive Monte Carlo simulations.

Results presented in this chapter are organized in two main sections. The first section presents some comparisons of the algorithm described in Chapter 3 with analytical expressions and with independent results obtained by Monte Carlo simulations. The second section deals with a hypothetical groundwater flow model undergoing a transient response to a step change in uniform areal recharge with the objective of determining what information on

the conditional probability distribution of transmissivity can be gleaned from the transient state. The comparison with analytical solutions for steady state in an infinite domain reveals an important aspect of deterministic first-type boundary conditions. Head covariances are conditioned by the model boundaries which has the effect of reducing head variance or modifying the length over which heads appear to be correlated. Comparisons of the numerical covariance solutions with covariances estimated through Monte Carlo simulations show that the Monte Carlo estimated covariances are slow to converge and continue to fluctuate even after many thousands of realizations. Conditioning on data from a synthetic reality field of transmissivity perturbations reveals that more time intervals of head data do not necessarily produce a better estimation of head or the spatial detail of transmissivity heterogeneity. The hypothetical modeling exercise also reveals the sensitivity of advective transport to head data conditioning. Finally, the relationship between the head data sampling interval with respect to the duration of the transient state is investigated by conditioning simulations on various intervals of head data. The results show that conditioning on a single sampling interval of head data, well into the transient, produces a better estimation of parameters and velocities than does conditioning on many shorter head data intervals earlier in the transient state.

4.1 Verification of covariances and conditional probabilities

4.1.1 Comparisons with conditional probabilities obtained by another method

The method presented in the previous chapter produces simulations of random transmissivity fields with specified spatial covariance behavior which

match transmissivity data and, given a flow model, result in calculated hydraulic heads which match the head data. While conservation of mass and momentum is guaranteed by the flow model and the simulated fields match the data, there is no guarantee that the cumulative distribution produced by Monte Carlo simulations is in fact the true conditional joint probability distribution. To the author's knowledge, no relatively simple method for producing conditional joint probability distributions of head and transmissivity in systems of large state vectors is available for the purpose of comparison. One computationally expensive method of generating conditional simulations for comparison is to randomly generate simulations, accept only those which fortuitously match the head data, and reject the rest. The problem with this is that perhaps hundreds of thousands of simulations must be generated in order to obtain enough conditioned fields to form an empirical cumulative conditional distribution. The more head data there are, the less likely a random simulation is to match all of the data and, as the acceptance criterion becomes more stringent, the less likely a random simulation is to be accepted. This acceptance/rejection method for generating cumulative conditional distributions is thus limited to steady state analysis of very small sets of head data.

Simulations jointly conditioned on 5 head data and 5 transmissivity data under steady flow conditions were generated by the acceptance/rejection method. This was done by generating mean zero random simulations on a 64 x 64 grid of square blocks, each dimensioned length 100 x 100, with an f variance of 1.0 and a correlation scale of length 1000 using the modified Mizell-A covariance model (equation (3.9)). These random fields were conditioned on the 5 data, zero-mean log transmissivity perturbations, by standard kriging. The

conditioned log transmissivity fields were then used to parameterize a groundwater flow model with first-type boundary conditions of constant values 10 and 0 top and bottom of the square domain, no-flow boundaries left and right, and a geometric mean transmissivity of 8000 length squared per time. A simulation was accepted if the absolute maximum difference, AME, between the resulting head field and all of the 5 head data was less than 0.25. On average, approximately 1 out of 1400 simulations matched this criterion. Empirical cumulative distributions consisting of 105 conditioned simulations are compared to the iterative cokriging technique described in the previous chapter at six locations in the field shown by x symbols in Figure 4.1. Conditioning data are 5 head and

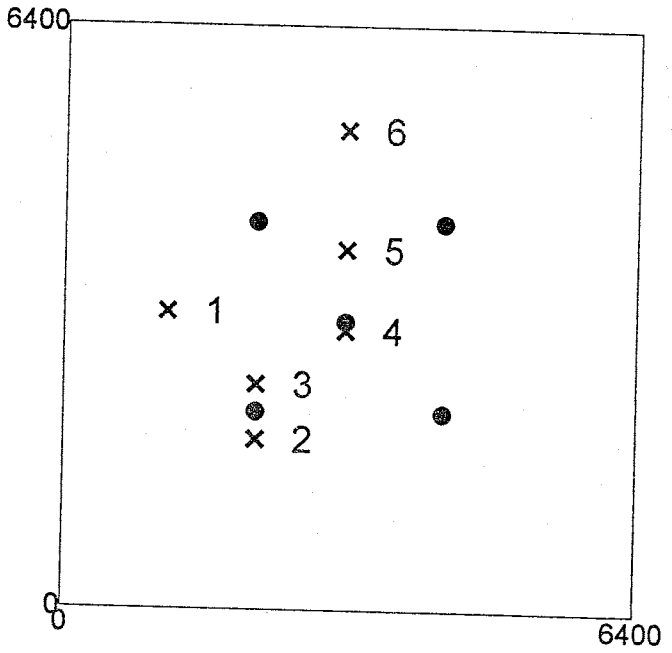


Figure 4.1: Cumulative probability distributions were collected at the locations indicated by x symbols and are conditioned on head and log transmissivity perturbation data locations shown as small filled circles

5 log transmissivity perturbations at the locations indicated in Figure 4.1 by small filled circles. Cumulative distributions at the six locations for f, log transmissivity perturbations, are shown in Figure 4.2 and cumulative distributions for head perturbations, h, are shown in Figure 4.3. Distributions generated by the acceptance/rejection method are plotted as solid lines and distributions from the iterative cokriging method as dotted lines. The Kolmogorov-Smirnov, KS, statistic and significance for each pair of distributions is given beneath the location plot. Sample means and variances of the 105 conditioned f field simulations, for both methods, along with the actual value at the six locations indicated by **x** symbols in Figure 4.1 are given in Table 4.1. The means and variances of the conditioned h fields at these locations are given in Table 4.2. Graphical comparisons of the means and variances of both h and f at these locations are shown in Figure 4.4.

	acceptance/rejection		iterative cokriging		
location	mean	var (std)	mean	var (std)	actual value
1	-.614	.720 (.849)	-.406	.698 (.835)	-.386
2	.906	.263 (.513)	.840	.233 (.483)	1.096
3	.751	.212 (.460)	.806	.256 (.506)	.366
4	.190	.039 (.197)	.204	.037 (.192)	-.070
5	-.521	.378 (.615)	-.771	.453 (.673)	-1.183
6	.524	.779 (.883)	.537	.766 (.875)	.836

Table 4.1: Sample mean and variance of f simulations at six field locations conditioned by the acceptance/rejection and iterative cokriging methods. Also shown is the actual value of the field at each location.

Qualitatively, the conditional distributions generated by both methods appear to be fairly similar. The KS statistics are larger, with smaller significance levels, for the h distributions than for the f distributions, which is

	acceptance/rejection		iterative cokriging		
location	mean	var (std)	mean	var (std)	actual value
1	.446	.528 (.727)	.605	.437 (.661)	-.533
2	.647	.042 (.205)	.625	.021 (.145)	.775
3	.196	.034 (.184)	.205	.012 (.110)	.082
4	.159	.025 (.158)	.061	.012 (.110)	.074
5	.725	.097 (.311)	.753	.061 (.247)	.521
6	.775	.071 (.266)	.726	.080 (.283)	1.102

Table 4.2: Sample mean and variance of h simulations at six field locations conditioned by the acceptance/rejection and iterative cokriging methods. Also shown is the actual value of the field at each location.

to be expected since the distributions are conditioned exactly on f data but match h data only to within an absolute maximum error AME of 0.25. Surprisingly the largest KS statistic of the h distributions occurs at location 4, which is the closest of the six distribution locations to a datum. The mean of 105 h simulations conditioned by the iterative cokriging technique at this location, .061, however, is closer to the actual value of h, .074, than the average of the acceptance/rejection simulations, .159. Also, in this case, the variance of the iterative cokriging simulations at this location is smaller than the variance of the acceptance/rejection simulations. Note that the scale of the h values at location 4 is much smaller than the other locations and the difference between the curves is less than the AME convergence criterion.

A comparison of distributions conditioned by two independent methods at only six locations gives little idea how the distributions differ block to block over the entire domain. For both methods a mean and variance at each of 4096 locations were calculated based on 105 conditioned simulations. One way to compare the variances is by the ratio. If the underlying random variable

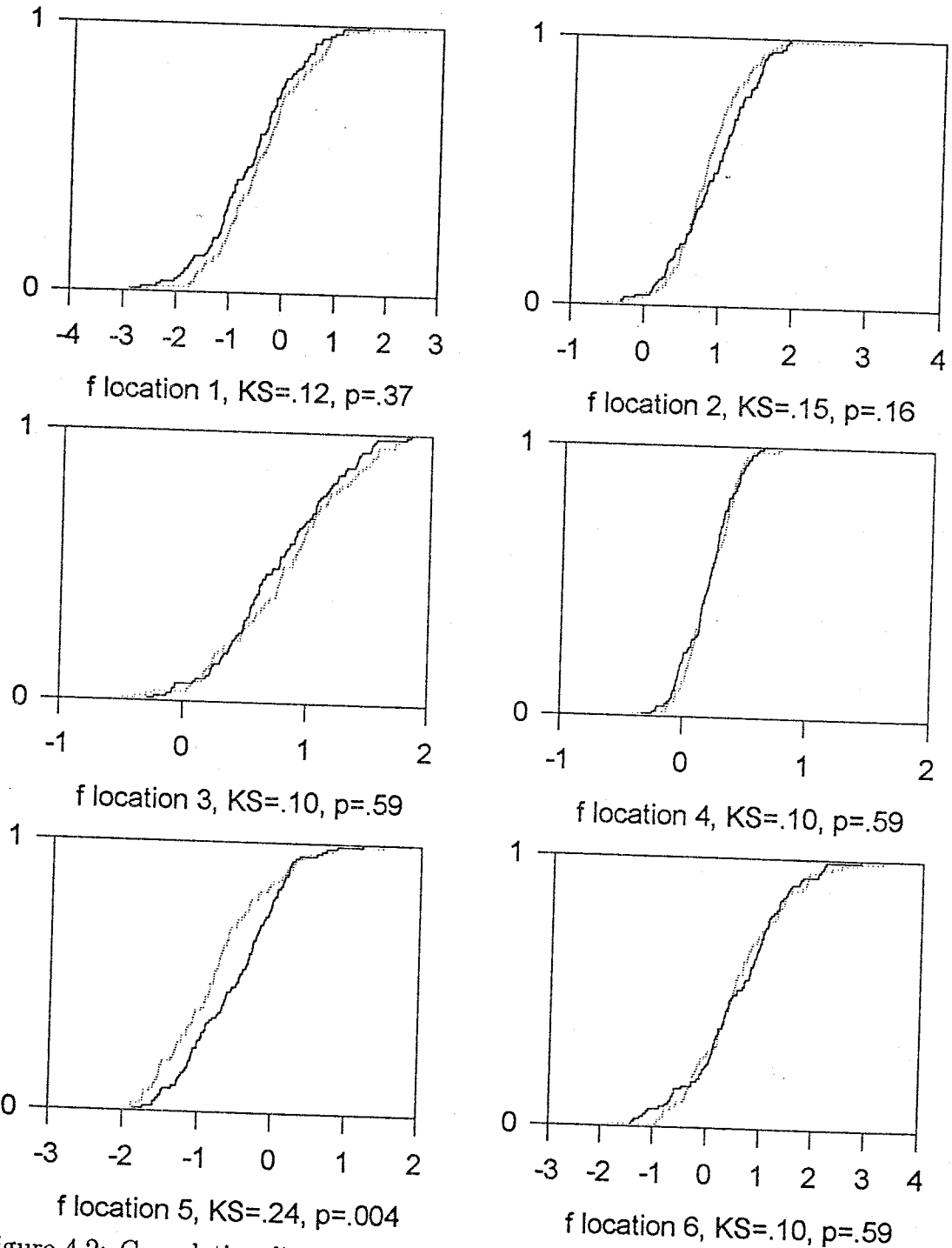


Figure 4.2: Cumulative distributions of 105 conditioned f fields are shown for six locations in the field. Simulations conditioned by acceptance/rejection are plotted as a solid line and simulations conditioned by iterative cokriging are plotted as a dotted line. KS is the Kolmogorov-Smirnov statistic and p is its significance

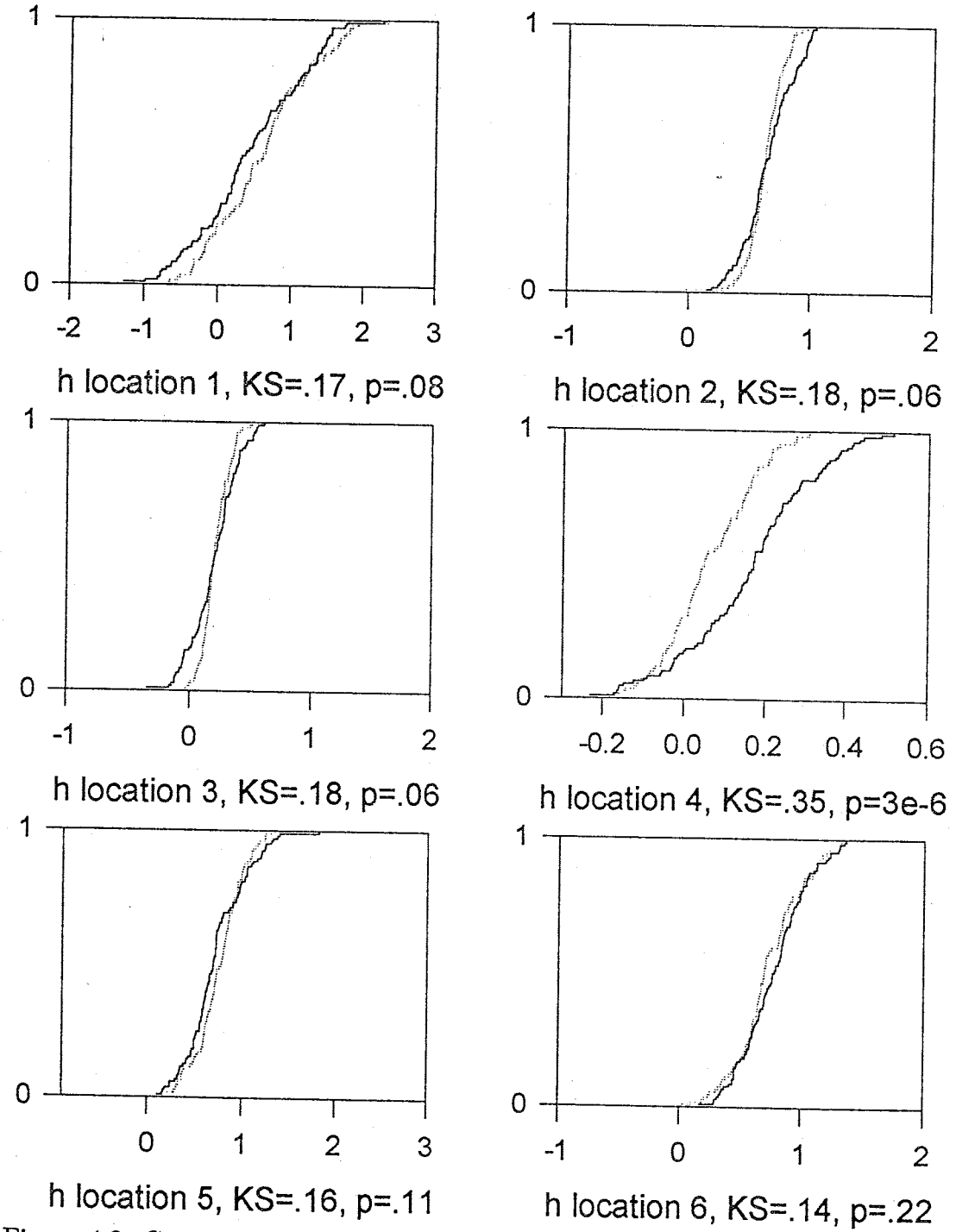


Figure 4.3: Cumulative distributions of 105 conditioned h fields are shown for six locations in the field. Simulations conditioned by acceptance/rejection are plotted as a solid line and simulations conditioned by iterative cokriging are plotted as a dotted line. KS is the Kolmogorov-Smirnov statistic and p is its significance

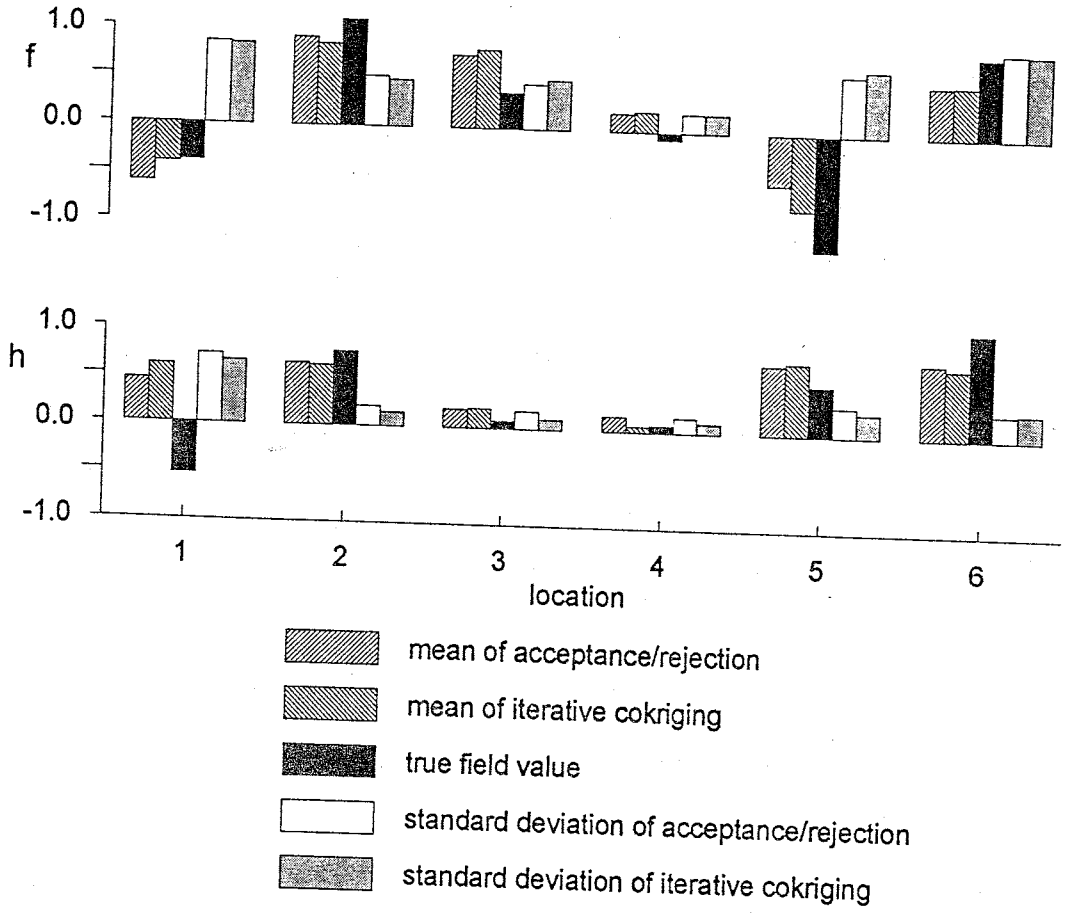


Figure 4.4: Comparison of the means and standard deviations of f and h, obtained from acceptance/rejection and iterative cokriging methods, and the actual field values at six locations (data from Tables 4.1 and 4.2).

is normal, the variance would have a chi-square distribution and the ratio of variances an F distribution. Random simulations of the unconditional f fields are multivariate normal and thus should appear gaussian with mean zero and variance one at a particular block but the conditioned fields are not necessarily normal. For the purpose of illustrating how the variance of f differs between the two conditioning methods this deviation from normality is assumed to be minor and the significance of the F statistic, or ratio of the two variances,

is plotted in Figure 4.5. At the 5 data locations, the conditional variance is zero. Of the remaining 4091 locations, 2077 have a variance conditioned by iterative cokriging which is smaller than the variance from acceptance/rejection conditioning, the largest ratio being 2.13.

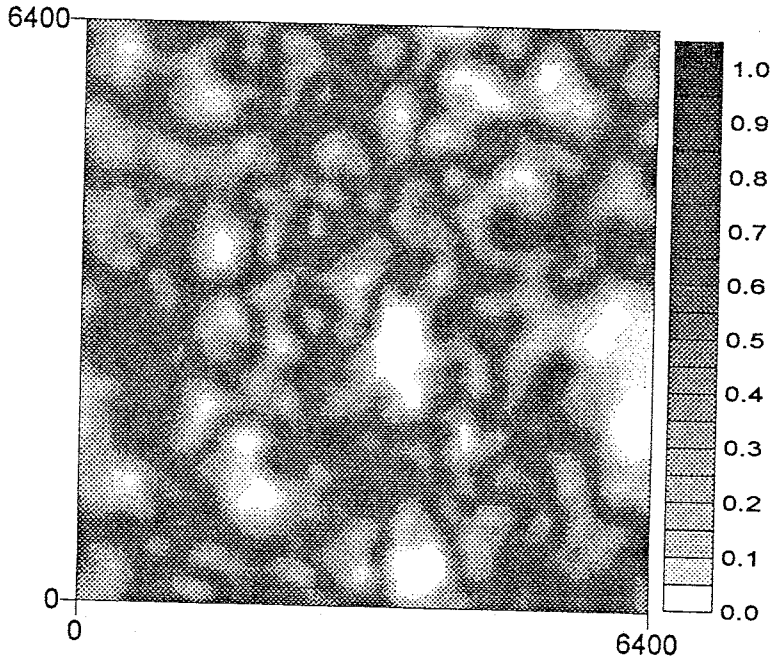


Figure 4.5: A plot of the F statistic significance level for f field variances conditioned by acceptance/rejection and iterative cokriging. Dark areas with a significance level near 1 indicate the two variances are nearly identical. Light areas mean the variances are significantly different. At the location with the greatest difference the variance from acceptance rejection was 2.13 times larger than the variance from iterative cokriging

Rather than compare the conditioned mean f fields to each other, both are compared to the known true field in Figure 4.6 using a mean squared difference between the conditioned simulations and the true field defined as:

$$MSE = f_v + (f_m - f_r)^2 \tag{4.1}$$

where $f_v = \sum_{i=1}^{M} (f_{c_i} - f_m)^2 / M$ is the sample variance of the conditioned f field, f_c from M simulations at a single block location, $f_m = \sum_{i=1}^{M} f_{c_i} / M$ is the sample mean of the conditioned f field at that location, and f_r is the actual value of the known true field. Note that this measure is merely the mean of the squared differences between the conditioned simulations and the true field. In the case of the acceptance/rejection conditioning, the average of MSE

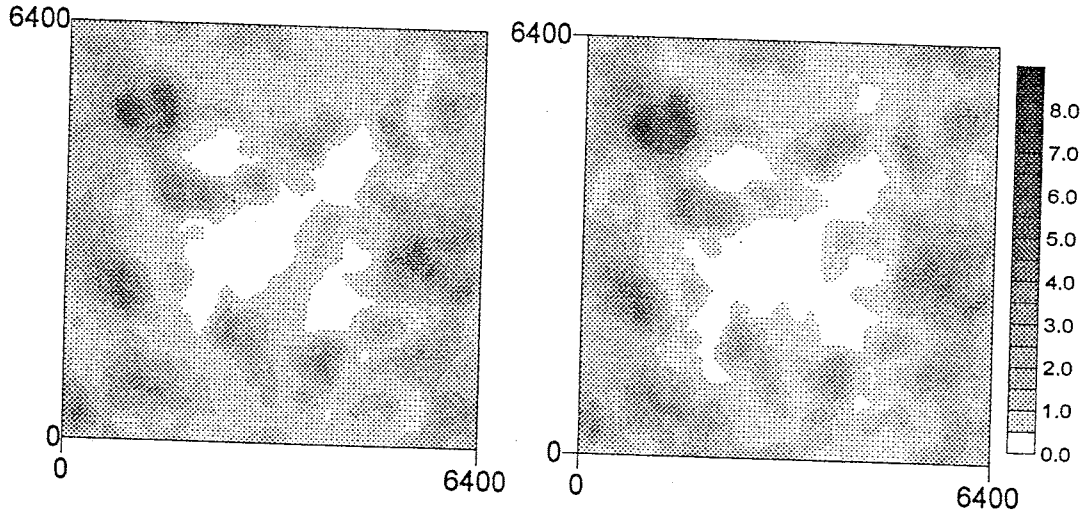


Figure 4.6: Comparison of the mean squared error between the f field conditioned by acceptance/rejection and the actual true field, shown on the left, and the mean squared error between the f field conditioned by iterative cokriging and the true field, shown on the right

over all 4096 blocks is 1.32 and, for iterative cokriging, the average MSE is 1.38. Not only are these average MSE measures very similar but the fields of MSE measures for each block f in the domain (Figure 4.6) are nearly identical indicating that the iterative cokriging method results in the same conditional distribution as does the acceptance/rejection method.

Again making an assumption that the non-normality of the conditioned h distributions is not an important consideration in comparing variances obtained by acceptance/rejection and iterative cokriging, the significance level of the F statistic (variance ratio) is shown in Figure 4.7. Dark areas signify regions where the variances are nearly identical. At 2520 of 4096 blocks the variance conditioned by iterative cokriging was smaller than variance conditioned by acceptance/rejection, with the largest ratio 5.33. The smoother pattern of high significance levels, compared to the f distribution, probably reflects the longer correlation length of the h variable.

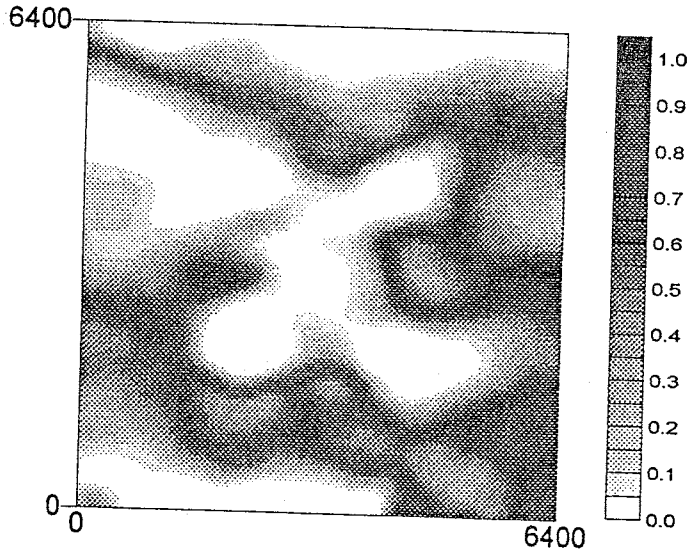


Figure 4.7: A plot of the F statistic significance level for h field variances conditioned by acceptance/rejection and iterative cokriging. Dark areas with a significance level near 1 indicate the two variances are nearly identical. Light areas mean the variances are significantly different.

Since the true h field is known from the synthetic reality true f field and the prescribed deterministic boundary conditions, the simulated and con-

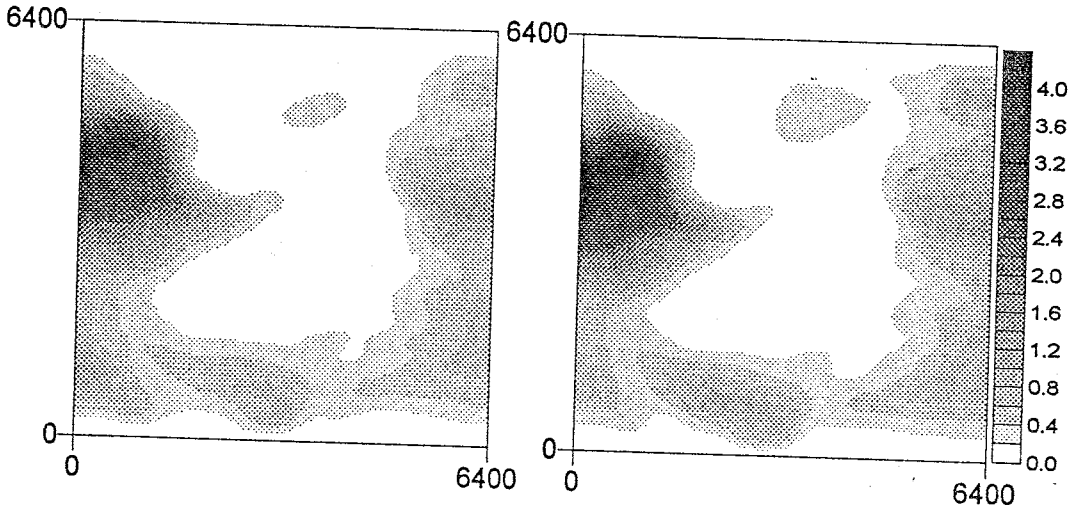


Figure 4.8: Comparison of the mean squared error between the h field conditioned by acceptance/rejection and the actual true field, shown on the left, and the mean squared error between the h field conditioned by iterative cokriging and the true field, shown on the right

ditioned h fields can be compared to the true h field by mean squared error where f_m and f_v in equation 4.1 are replaced by the mean and variance of the conditioned h simulations at a block location and f_r is replaced by the value of the true h field. A plot of the mean squared error for the 105 conditioned h field simulations is shown in Figure 4.8. The average MSE over all 4096 blocks is .454 for the acceptance/rejection conditioned h simulations and is .487 for the iterative cokriging conditioned h simulations. As with the MSE for the conditioned f fields, the patterns of MSE measures for each block location in the domain are nearly identical from both the iterative cokriging and acceptance/rejection conditioning methods (Figure 4.8).

This comparison of iterative cokriging with an acceptance/rejection

method is presented only as a qualitative verification of the iterative cokriging method. It is not presumed that the iterative cokriging method is proven or that the two methods will converge in the limit. It does, however, indicate that the two methods give very similar results where both distributions match data on head and transmissivity.

4.1.2 Covariance comparisons with analytical solutions

Cokriging matrices, equations (3.19) and (3.21) consisting of the covariances and cross-covariances between f and h data, are formulated by a combination of small perturbation approximation, finite-difference numerical solution, and a quadratic polynomial interpolation. Error due to perturbation linearization is not significant when the variance of f is less than one. Errors due to finite-difference approximation and interpolation become more significant as the grid discretization becomes coarser, and error due to interpolation also depends on how well the intermediate, or fine block, covariances fit a local second order polynomial. Using the finer grid blocks to approximate the load term $T_G \nabla \cdot (C_{ff} \nabla H)$ in equation (3.5) may minimize effect of scale caused by interpolation from larger blocks to smaller blocks. Scale effect may be more pronounced in the head covariance approximation since the load term derivative $T_G \nabla \cdot (C_{hf} \nabla H)$ in equation (3.6) is approximated using the coarse grid blocks. Both the load term derivative approximations and the interpolation are affected by the smoothness and shape of the covariance.

In this section covariances calculated from a numerical solution of the linearized equations 3.12, 3.13, and 3.14 are compared with analytical expressions for cross-covariance and head covariance at steady state for constant

mean head gradient in an infinite domain [*Gutjahr and Wilson*, 1989; *Mizell et al.*, 1982] where the Mizell-A covariance function characterizes uncertainty in f. Graphs comparing analytical expressions in the infinite domain with numerical approximations for the bounded domain are shown in Figure 4.9 for 3 modifications of the basic flow geometry. Basic flow geometry is that described in section 4.1.1 with first type boundaries at the top and bottom and no-flow on the sides. First type boundary conditions were prescribed so that the mean head gradient is the same constant 1/640 for all 3 flow geometries. Variance of f for all plots in Figure 4.9 is 1.0 and the correlation length is 1000. Each row of two graphs in Figure 4.9 represents a slightly different geometry. Plots on the left are cross-covariance, as a function of separation length, and plots on the right are of head covariance. The origin of the plots is h at the node of a coarse grid block very near the center of the domain and the plots show covariance with h or cross-covariance with f up-gradient along the center of the domain to the top boundary. Analytical expressions are shown as solid lines and the numerical approximations as dashed lines with $\diamond$ or $+$ symbols.

The cross-covariance and head covariance in Figures 4.9a, 4.9b were calculated using a 64 by 64 fine grid of blocks, each dimensioned length 100 by 100. Thus there are 10 blocks of the fine grid per f correlation length and 6.4 correlation lengths across the domain. The Δx and Δy coarse grid spacing used for covariance approximations is 4 times the fine grid spacing in all cases. The numerical approximation of head covariance is forced to zero by the non-random boundary condition at separation distance 3400, which has the effect of shifting the approximation downward from the analytical solution. In Figures 4.9c, and 4.9d the fine grid block size was reduced by half to a

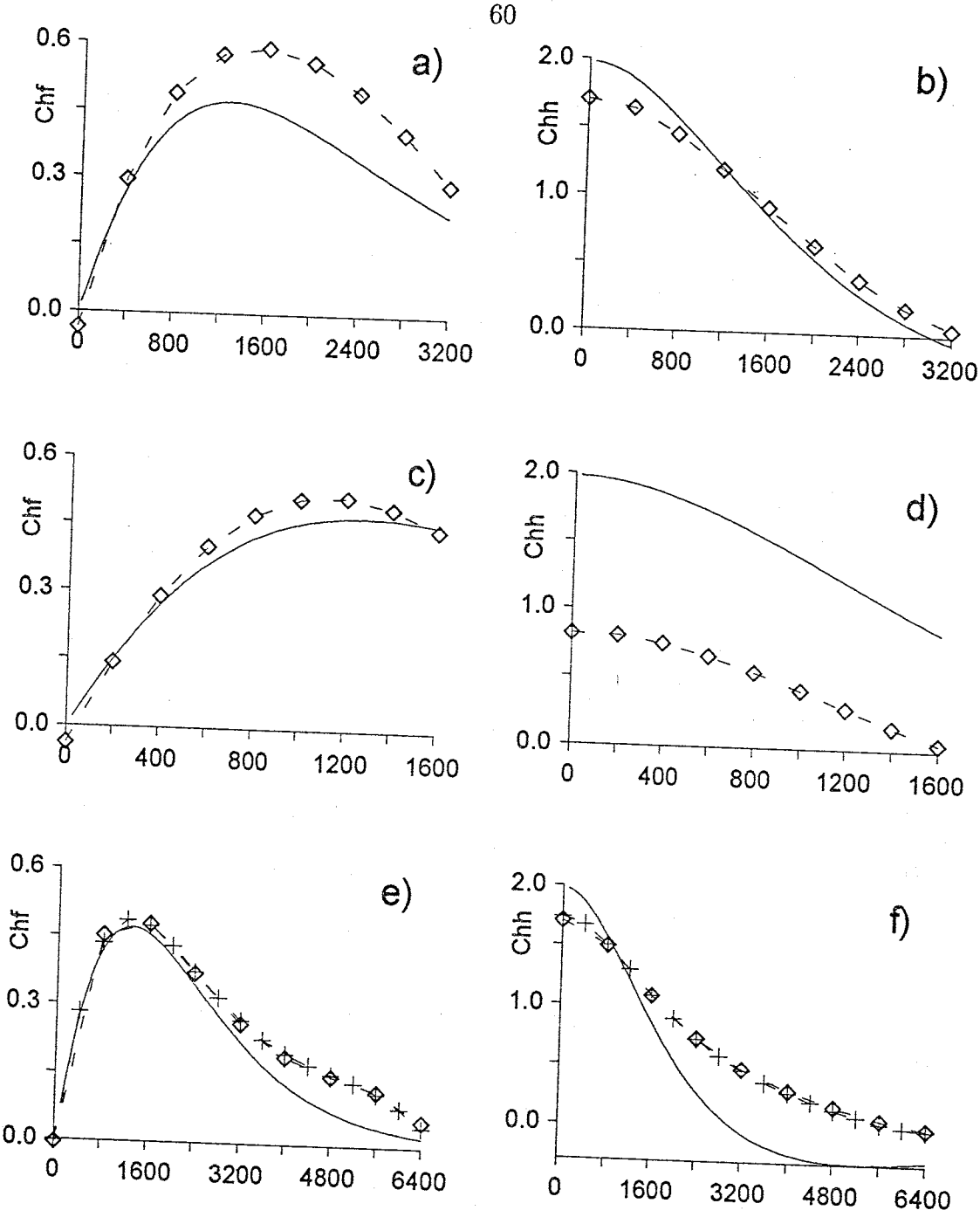


Figure 4.9: Graphs in the left column are C_{hf} cross-covariance plotted versus separation distance. Graphs in the right column are C_{hh} head covariance plotted versus separation distance. Covariance in f is characterized by a Mizell-A covariance function. Analytical expressions are shown as solid lines and the numerical approximations as dashed lines with $\diamond$ or $+$ symbols. Refer to the text for more explanation.

Δx spacing of length 50 so that there are 20 fine grid blocks per correlation length and only 3.2 correlation lengths across the domain region. This block size reduction has a relatively minor impact on the cross-covariance within a f correlation length of the origin (Figure 4.9c). Beyond a correlation length the cross-covariance is influenced by the boundary condition. However, for the same block size reduction (Figure 4.9d), a significant difference results between the head covariance analytical expression and the numerical approximation. The head covariance analytical solution predicts a much higher head variance than the numerical approximation since the numerical approximation is being conditioned by a deterministic boundary condition which is 1.6 f correlation lengths closer to the origin in Figure 4.9d than in Figure 4.9b. Conditioning on the assumed known information contained in the boundary condition results in a significantly smaller head variance in the bounded domain than in an infinite domain.

In Figures 4.9e and 4.9f the effect of doubling the fine grid block size from length 100 to 200, so that there are 12.8 f correlation lengths across the domain and 5 fine grid blocks per correlation length, is shown. Again the effect within a f correlation length on the cross-covariance, Figure 4.9e, is relatively small. The numerical approximation for head covariance is similar at the origin to the analytical expression for the infinite domain (Figure 4.9f) with slightly smaller variance. As the lag exceeds a few correlation lengths, however, the analytical solution for head covariance becomes negative, implying a negative correlation with head perturbations more than a few correlation lengths up-gradient, while the numerical approximation is conditioned to zero at the boundary which results in all positively correlated head perturbations

over a longer scale. Another calculation of covariance was done by doubling the number of grid blocks from 64 by 64 to 128 by 128 while keeping the block size at 100 by 100. The results, shown in Figures 4.9e and 4.9f as + symbols connected by a dashed line, indicate that the longer correlation length is due to conditioning on the boundaries and not to a scale effect due to larger block dimensions. Boundary conditions are thus seen to exert a significant influence on the head covariance structure. Boundaries close to data have the effect of reducing the variance as a result of conditioning on presumed certainty. Moving the boundaries farther away avoids this artificial variance reduction but causes head perturbations to be correlated over a longer scale.

This comparison between analytical and numerical solutions to a first-order approximation of the covariance equations involves deterministic first type boundary conditions only for the numerical solution. Deterministic (non-random) boundary conditions are seen to have the effect of conditioning the numerical solution of covariance by reducing the head variance, if the domain is small compared to the f correlation length, or stretching the head correlation, if the domain is large compared to the f correlation length. This effect is different than propagation of uncertainty due to boundaries into the domain, which was shown by *Rubin and Dagan* [1988, 1989] to be negligible farther than a few f correlation lengths away from the boundaries. In this case the effect is either to reduce apparent uncertainty in head or to increase the apparent correlation in head; the latter occurring if the boundaries are several f correlation lengths away. The implication is that deterministic boundaries impose certainty on the model as if head data with no measurement error were available everywhere along the extent of the boundary. Including randomness in first type bound-

aries involves additional parameters characterizing this uncertainty with the choice of those parameters being somewhat arbitrary. Second type boundary conditions, which force the derivative of covariance to be zero at the boundary, may reduce the conditioning effect of the boundaries allowing the conditioning to be dominated by the head data; however, investigating the effect of second type boundary conditions on conditioning and uncertainty is left for future work.

4.1.3 Comparison with Monte Carlo generated covariances

In section 4.1.1 Monte Carlo simulations with an acceptance/rejection conditioning technique were used as reference for comparing the linearized, numerically approximated and iterated conditional probabilities calculated by the method described in Chapter 3. This linearized iterative method relies on cokriging matrices of covariance between head data locations and cross-covariance between head and transmissivity data locations for the geostatistical inversion. In this section Monte Carlo simulations are used to generate these cokriging covariances between data locations. Covariances generated by Monte Carlo simulations do not suffer from a small perturbation linearization approximation and should, in the limit, approach the true expectation. So then, one might wonder, why use the linearized approximation at all? Why not construct the cokriging matrices via Monte Carlo simulations then condition by inversion and iteration? One reason is computational efficiency. In order to calculate an accurate covariance by the Monte Carlo method, a few thousand simulations may be needed. Generating a few thousand simulations requires a few hours of processor time for a problem that the linearized approximation

algorithm would complete in a few minutes. Another reason is that the cokriging vector in equations (3.19) and (3.21) requires the covariance of every block in the fine grid with all of the data. The estimation of this covariance vector would require saving the result of several thousand simulations for every fine grid block in the domain. Another problem is with the inversion of the cokriging matrix. Covariances generated by Monte Carlo simulations are noisy even after a few thousand simulations. A plot of the covariance from simulation appears bumpy and irregular while the linearized approximation is smooth. These small irregularities are magnified by the matrix inversion into very large errors in the cokriged fields.

Despite the problems of using Monte Carlo simulations for geostatistical inversion, they can provide a basis for verifying the results of the linearized approximation and evaluating the error. Values of the random f simulations and resulting h perturbations, given a flow model, are collected at the data locations and the covariances are calculated with the non-ergodic estimator

$$\hat{C}_{VW} = \frac{\sum_{i=1}^{M} \left(V_i - \bar{V}\right)\left(W_i - \bar{W}\right)}{M-1} \tag{4.2}$$

where $\bar{V} = \sum_{i=1}^{M} V_i/M$ and $\bar{W} = \sum_{i=1}^{M} W_i/M$ for M simulations. If V and W are both h perturbations at the same location, this is the h variance estimator. If V and W are h perturbations at different locations, then this is the h covariance estimator, and if V and W are h and f perturbations the cross-covariance is estimated.

The cokriging covariance matrix calculated by linearized numerical approximation for the data locations shown in Figure 4.1, assuming the flow model described in section 4.1.1, is displayed graphically in Figure 4.10a.

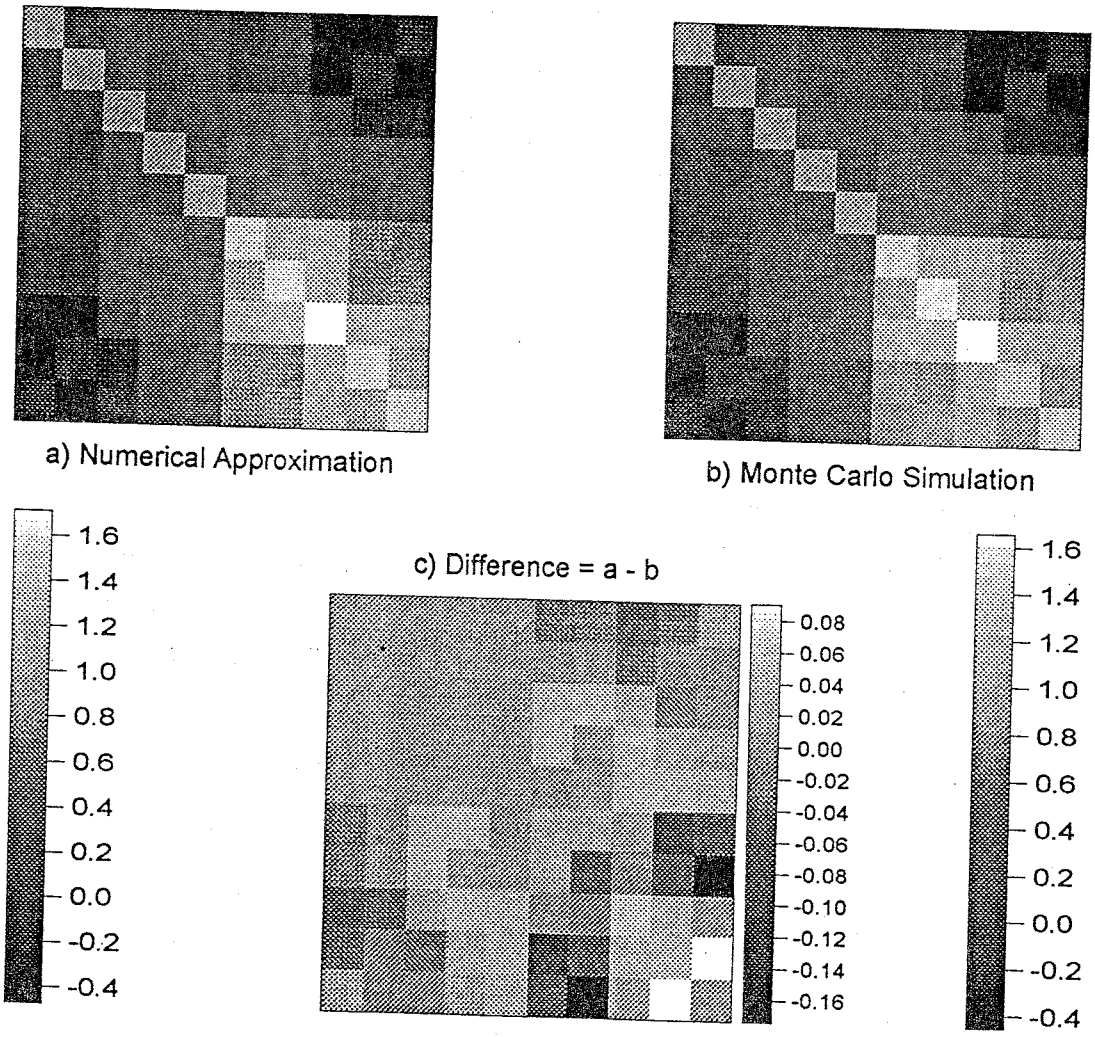


Figure 4.10: Graphical display of the cokriging matrix from the 5 f and 5 h data locations of Figure 4.1 calculated by linearized approximation and 2000 Monte Carlo simulations.

The upper left $5x5$ block area of Figure 4.10a represents the matrix $\mathbf{R_{ff}}$ so the first 5 diagonal elements are $C_{ff}(0)$ of value 1.0. The lower right $5x5$ block area represents the matrix $\mathbf{R_{hh}}$, the upper right is matrix $\mathbf{R_{fh}}$, and the lower left its transpose $\mathbf{R_{hf}}$. Figure 4.10b shows the cokriging matrix computed with the non-ergodic covariance estimator, equation (4.2), from 2000 simulations. The difference between the linearized and simulated values is then displayed in Figure 4.10c. There is no difference between the $\mathbf{R_{ff}}$ matrices in Figures 4.10a and 4.10b as they are both constructed by the same C_{ff} function, so this area in Figure 4.10c is uniform gray of value 0.0. The average of the remaining 75 non-zero differences is -.0071 with the largest difference being -.1742 for covariance of h at data location 2 with h at data location 5. An average difference of zero between the linearized and simulated covariances would imply that the Monte Carlo simulation had converged and that the linearized approximation was exact. The small average difference of -.0071 indicates that the two covariance approximation methods are producing similar results where, at the very worst, the Monte Carlo simulation result at data location 2 with data location 5 is 23% larger than the linearized approximation.

When the number of simulations used to estimate the covariances was doubled from 2000 to 4000, the average of the differences between the 75 C_{fh} and C_{hh} matrix elements increased to -.0115 with a maximum difference of -.2019, both significantly different from the results of 2000 simulations. Here, at the largest difference, the result from the Monte Carlo simulations is 26% larger than the linearized approximation. If the Monte Carlo simulations had converged then this would indicate the error involved in the linearized approximation; however, doubling the number of simulations again, from 4000 to 8000,

the average became -.0077 and the maximum -.1790, closer to the values from 2000 simulations. The problem can be seen that, even at 8000 simulations, convergence of the covariance estimator had not been reached yet the execution time was approaching 9 hours on an IBM RS/6000 66 Hz processor capable of 125 MFLOPS.

In the following section, 16 f data and 16 h data at 6 times are used in the construction of the cokriging matrix. The 12288 C_{fh} and C_{hh} elements of the cokriging matrix were estimated by 1000 Monte Carlo simulations. The average of the difference between the linearized numerical approximation and the Monte Carlo estimate was .0380 with a maximum difference of 2.47 located on the diagonal at row and column 97. This element represents the variance of head at the lower left data location shown in Figure 4.11 at a time of 75 years after the step change in recharge. Estimates of the covariances obtained from 2000 simulations had an average difference from the linearized approximation of .0362 with the maximum of 2.34 at the same location. The linearized approximation of the head variance at this location and time is 11.06 while the variance estimated from 2000 simulations is 8.72. Going from 1000 to 2000 Monte Carlo simulations in this case shows a slight improvement in that the average difference becomes slightly smaller and, at the worst point, the Monte Carlo results goes from 29% larger than the linearized approximation to 27% larger.

This section has presented some comparisons with other methods of the small-perturbation numerical solution approach to head covariance and head-log transmissivity cross-covariance for the purpose of verifying the algo-

rithm. A brute-force method of creating conditional simulations by randomly generating simulations, accepting those which coincidentally match the data, and rejecting the rest was seen to produce conditional simulation results remarkably similar to the iterative cokriging method developed here. The acceptance/rejection method is so computationally intensive though, it is limited to very small data sets and steady state conditions only. In this case iterative cokriging using the numerically approximated covariances has a tremendous computational advantage. A comparison of analytical solutions for steady state in an infinite domain with the numerically approximated covariances revealed the strong influence that deterministic first-type boundary conditions have on the computed covariances. The boundaries were seen to reduce the calculated head variance or increase the correlation length of the head covariance depending on the relationship between the size of the domain and the correlation length of the log transmissivity perturbations. Finally, a comparison between the numerically approximated covariances and covariances calculated via Monte Carlo simulations indicated that, while the two methods produce similar results, covariances calculated from Monte Carlo simulations are noisy and slow to converge. Again the numerical approach has a distinct computational advantage of a factor of about 10 in computation run-time depending, of course, on how many Monte Carlo simulations are generated before the covariances are assumed to have converged. One conclusion I'll draw from this section is that iterative cokriging using numerical solutions to the linearized covariance equations is a computationally feasible method for producing simulations of head and transmissivity co-conditioned on both head and transmissivity data.

4.2 Application to a hypothetical model

In this section the geostatistical inverse model presented in Chapter 3 is applied to a hypothetical groundwater flow model. A rationale for applying the method to a hypothetical model is that all aspects of a hypothetical model; boundary conditions, recharge, storativity, and the actual heterogeneity of transmissivity as well as its statistical properties are known. Thus various changes or additions to the inversion methodology can be compared against a "truth" in order to evaluate their effect. Matrices calculated by the linearization approximation in equations (3.12), (3.13), and (3.14) are sufficient to find the unconditional joint probability distribution of head and transmissivity. Actually, these joint distributions are unconditional with respect to the data but are conditioned by the model and boundary conditions as illustrated in Figure 4.9.

Simply kriging the transmissivity measurements provides the first level of conditioning on data. Probability distributions jointly conditioned on both f and h data can be obtained through the inversion of the cokriging matrix (3.19) and iteration at initial steady state conditions. If the groundwater flow system is in a transient state and h data were available through time, this process could be repeated for a series of time steps, sequentially conditioning on data by inverting the cokriging matrix in equation (3.19) at each time step. Even though all data are honored by this process, the time steps are effectively decoupled so the system is treated as a series of quasi-steady states. Accounting for interdependence of time steps through the h cross-time covariance, conditioning on all time dependent head data, can be accomplished

at once using the cokriging matrix in equation (3.21). Finally, a probability distribution of f and h jointly conditioned on all f and time dependent h data to within a specified criterion is obtained.

Results of each of these levels of conditioning are presented in the following sections with the objective of evaluating the information gained from conditioning on head data in a transient state. Simulations conditioned using the collateral method are presented first followed by results obtained from sequential conditioning. The final set of simulations investigates the effect of the head data sampling interval on conditioning results versus the number of sampling intervals. Computations were performed on nodes of the SP1 at the Albuquerque Resource Center, Albuquerque, New Mexico and the SP2 at the Maui High Performance Computing Center in Hawaii. These are equivalent to an IBM RS/6000 66 Hz processor capable of 125 and 266 MFLOPS respectively. At each stage the value of additional conditioning data is compared in terms of estimation variance of f and h fields and in how well the conditioned mean fields match the true underlying hypothetical reality. I'll look at three performance measures specifically, and label them

AMSE – Mean-Squared-Error (equation (4.1)) averaged over all simulations,

MMSE – Maximum Mean-Squared-Error over all simulations, and

MVAR – Maximum Variance of simulated fields.

The effect of f and h data conditioning on advective pathlines and travel times is also examined graphically. Each set of simulations was started from the same

random seed so that the comparison is between conditioning methods and data and not between different realizations.

4.2.1 A hypothetical groundwater flow model

The square bounded domain of the hypothetical groundwater flow model with 16 data locations is shown in Figure 4.11. Boundary conditions are prescribed first type head of 10 m at the top, 0 m at the bottom, and prescribed second type, no-flow, on the sides. The statistically stationary transmissivity field has a geometric mean of 8000 $m^2/year$, a variance of 1.0, and a correlation length of 1000 m described by the Mizell-A covariance function. For the purpose of random field simulation and numerical solution of the groundwater flow equations, the domain is discretized into a grid of 64 by 64 uniform blocks each dimensioned 100 m on a side. Transient flow was induced by a step change in uniform recharge over the entire domain from 0 to .025 $m/year$ at time t=0 and head was calculated for a period of 80 years using a time step of 0.25 years and a storativity S=0.1. A synthetic data set was created by sampling the hypothetical reality transmissivity field at the 16 locations shown in Figure 4.11. Head, in equilibrium with the boundary conditions, was sampled at the same 16 locations at t=0 then again in transient at the same 16 locations for times t=1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 35, 40 45, 50, 60, and 75 years. The time constant or response time of the system, where about 63% of the transient state has occurred, for a groundwater flow model with this set of parameters can be estimated as $\frac{SL^2}{\beta T_G}$. With $S = 0.1$, $L = 3200m$, $\beta = 3$, and $T_G = 8000m^2/year$ the time constant for this system is estimated to be about 43 years. For each simulation, a particle was tracked for 80 years starting from

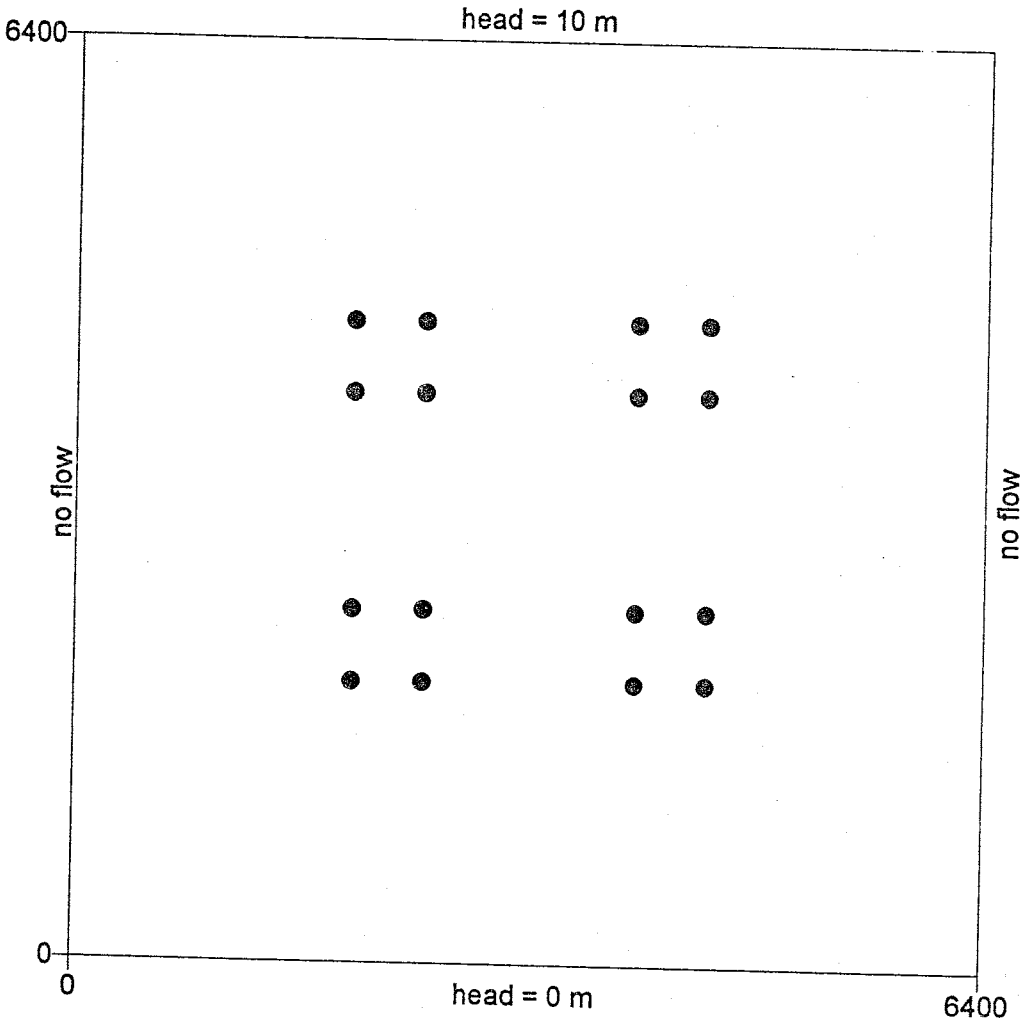


Figure 4.11: Bounded flow domain of the hypothetical flow model described in section 4.2.1. The 16 • symbols indicate sample locations of both f and h data.

the central location at x=3250, y=3250 at time t=0. A porosity of 0.2 and an aquifer thickness of 5 m was assumed. While this thickness may be somewhat unrealistic in relation to the high storage coefficient and the assumption of no change in saturated thickness, it was chosen so that most of the particles released near the center of the domain would reach the boundary within the period of simulation.

In Figure 4.12a the hypothetical true field of mean zero perturbations in log transformed transmissivity is shown. Figure 4.12b is the head field resulting from this transmissivity field and the assumed model. Shown as stacked maps are head at an initial steady state in equilibrium with boundaries and head in the aquifer increasing, 80 years after a step change in uniform recharge.

4.2.2 Simulations conditioned only on f data

Random simulations of the perturbations in log transmissivity can be conditioned on f data very quickly by kriging. Since no head data is incorporated, cross-covariances and head covariances need not be computed, as the kriging covariances are obtained directly from the f covariance function. Conditioning only on f data was done here as a baseline for evaluating the improvement obtained by incorporating head data. Aquifer heads in the hypothetical flow model resulting from simulations of log transmissivity conditioned on the f data were examined after 80 years of uniform recharge. The ensemble averages of 100 simulations of f fields and resulting head, conditioned only on f data are shown in Figure 4.13. Some of the major features of the true f field, such as the low permeability zone at about x=1800 m y=3800 m and the

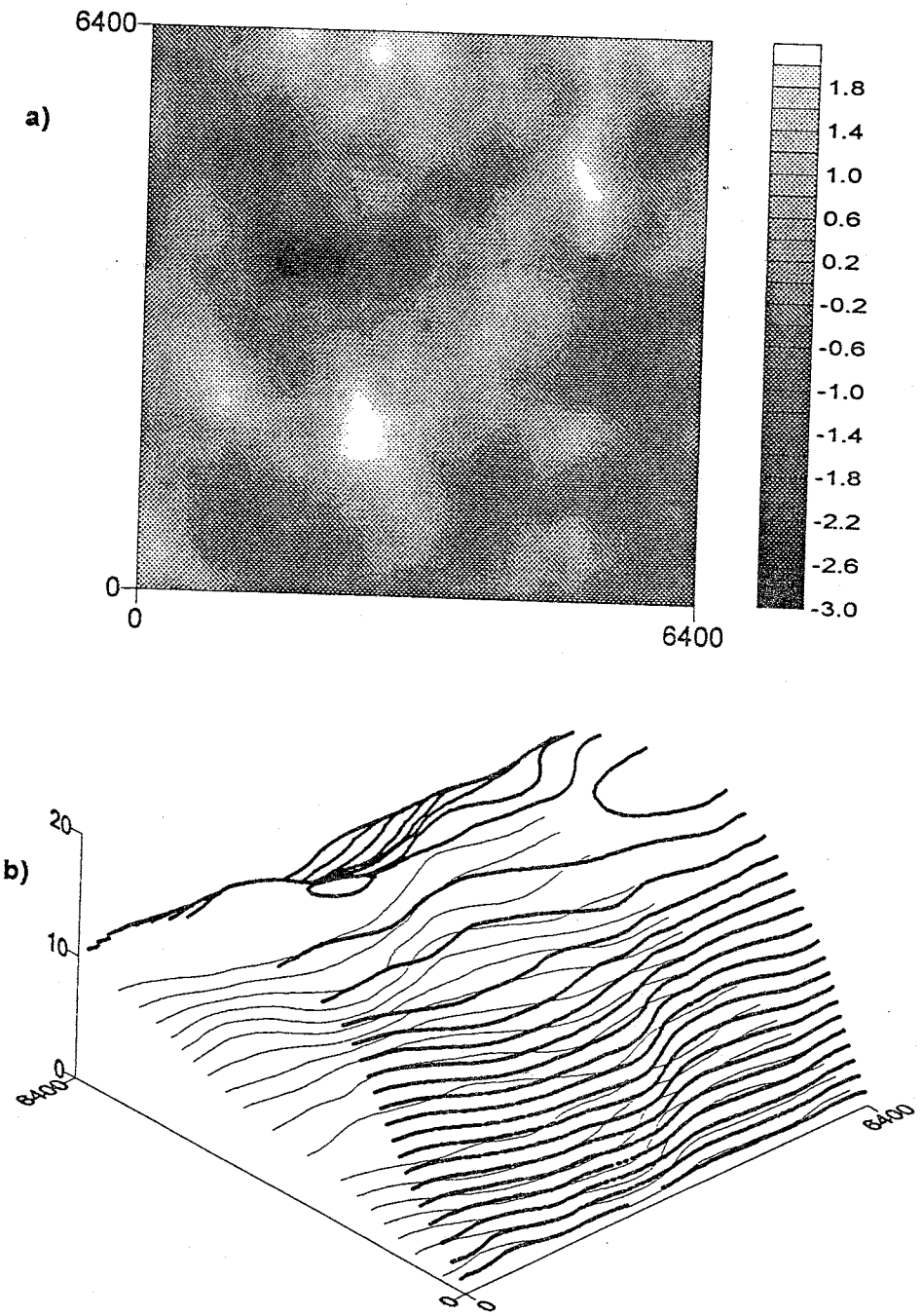


Figure 4.12: a) Simulated mean zero perturbations of ln(T), with variance 1.0 and correlation length 1000 m, designated as the hypothetical true field. b) Head given this transmissivity field and the model described in section 4.2.1 at initial steady state t=0 (thin lines) and after 80 years of uniform recharge (thick lines).

high permeability zone around x=2550 m y=1800 m, are reproduced in the f data conditioned mean f field (Figure 4.13a). Both the conditional mean f and head fields are very smooth, however, and represent very little of the heterogeneity seen in the true fields (Figure 4.12). The mean squared error between the simulations and the true field, calculated using equation (4.1), is shown in Figure 4.14a for conditional mean f and in Figure 4.14b for mean head. The average MSE of the f simulations is 1.47 with a maximum MSE of 6.28 and the average MSE of the simulated heads is 15.86 with a maximum of 56.0. The ensemble variance of the 100 simulations is shown in Figure 4.15a for conditioned f and in Figure 4.15b for head fields. The maximum ensemble variance of the 100 conditioned f simulations is 1.47 and the maximum head variance is 13.74. This shows that conditioning only on f data does not provide much information concerning the spatial detail of the log transmissivity perturbations except in the immediate proximity of the f data locations. Simulated head fields resulting from transmissivities conditioned only on f data vary widely from the actual true head, as shown by the MSE measure and variance.

4.2.3 Simulations conditioned on f data and initial steady state h data

Incorporating steady state head data in a geostatistical estimate of the conditional mean and variance of the transmissivity parameter has been presented in several of the publications discussed previously in the literature review section of this thesis [e.g. *Hoeksema and Kitanidis* 1984; *Gutjahr, et al.*, 1994]. The method presented here does differ slightly from other geostatistical inverse methods in that various, albeit deterministic, types of boundary condi-

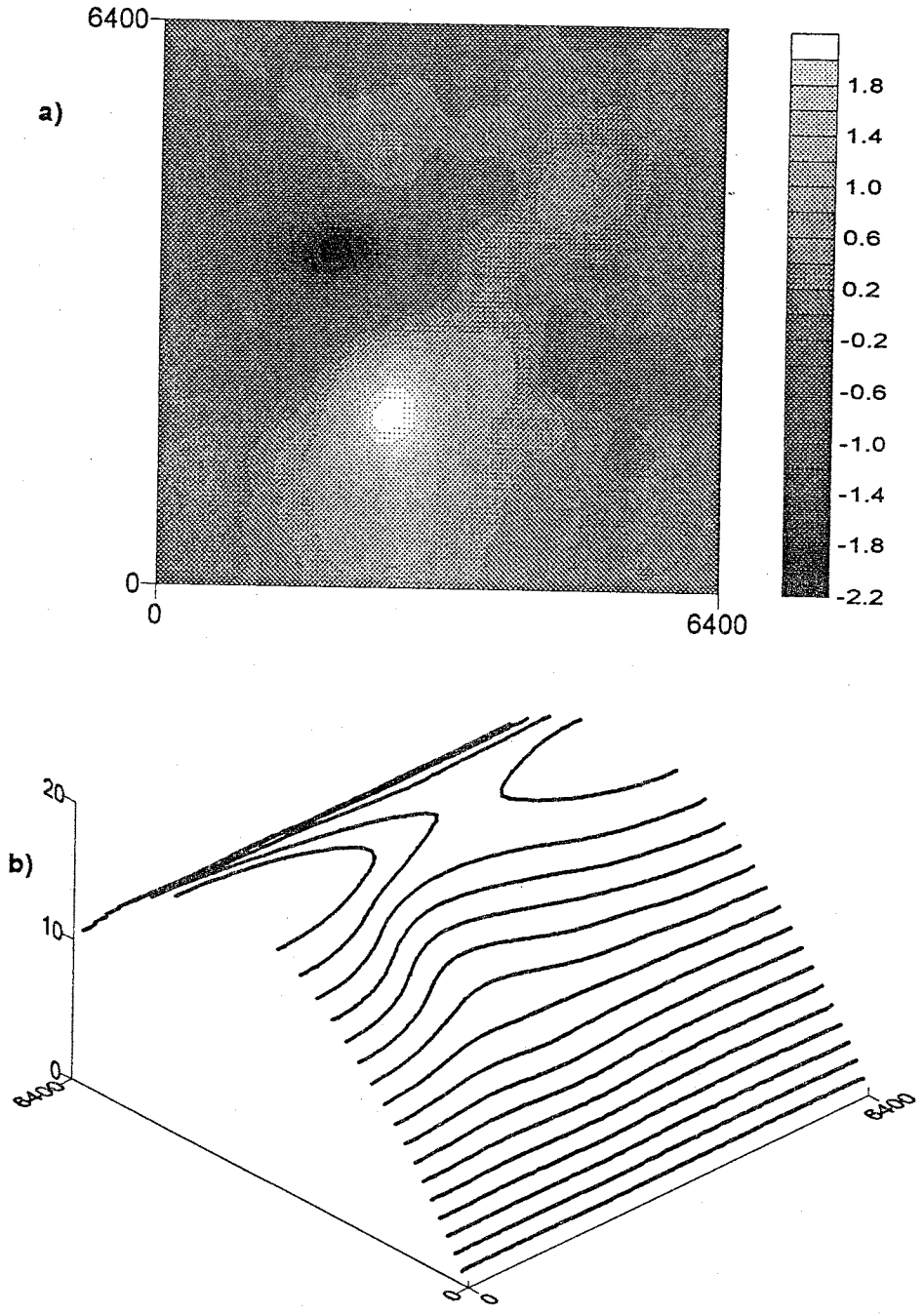


Figure 4.13: a) Ensemble mean of 100 f simulations conditioned on only f data. b) Ensemble mean of resulting head fields after 80 years of uniform recharge.

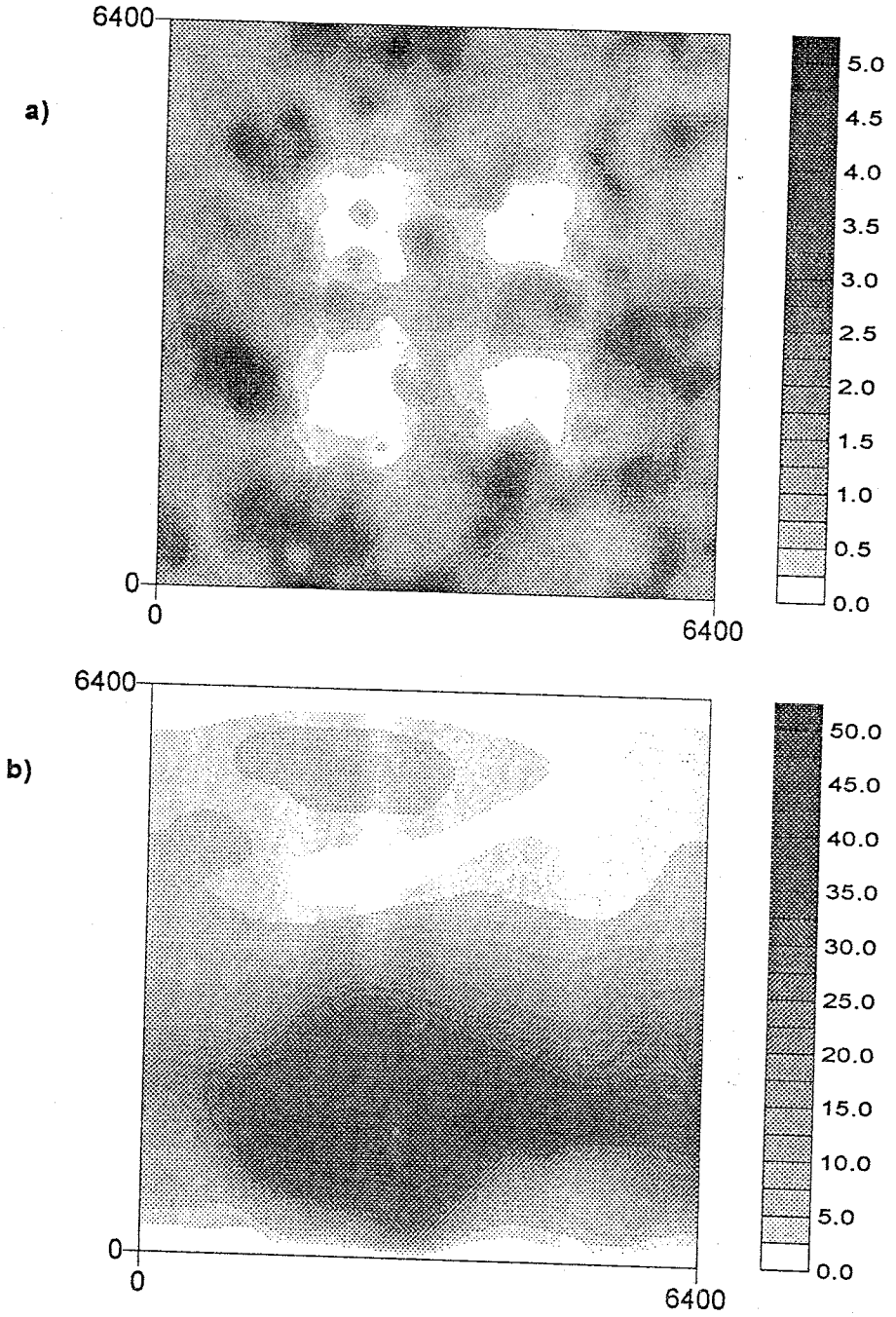


Figure 4.14: a) Mean squared error between 100 f simulations conditioned on f data and the true field. b) Mean squared error between 100 resulting head fields and the true head at 80 years.

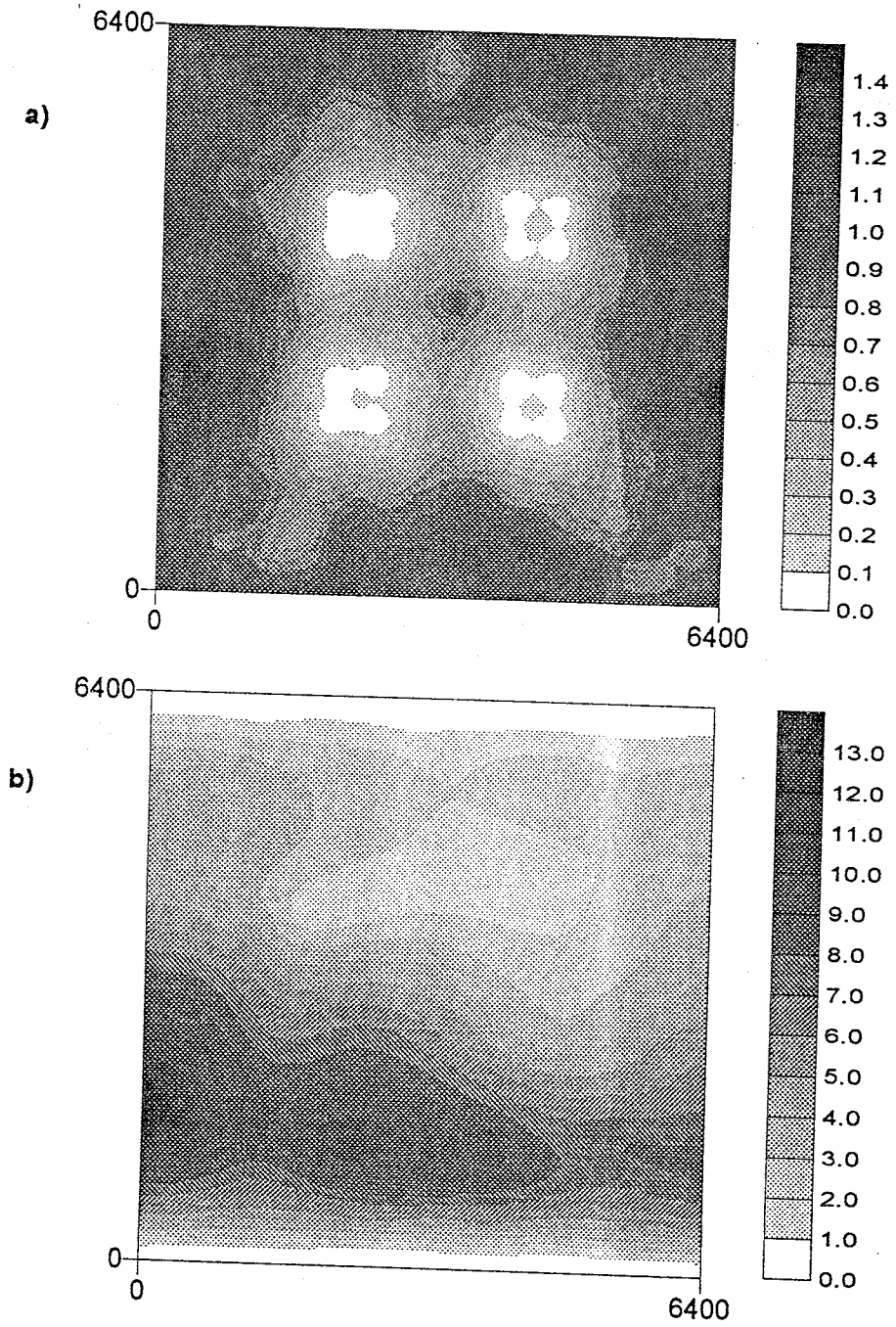


Figure 4.15: a) Ensemble variance of 100 simulated f fields conditioned on f data and, b), variance of the 100 resulting heads.

tions and spatially distributed recharge are allowed and head data is matched fairly closely through the flow model via iteration. For the hypothetical flow model described above in section 4.2.1, 99 simulations of log transmissivity perturbations were conditioned on 16 f data and 16 h data at the locations shown in Figure 4.11 at initial steady state. The conditional mean f field resulting from these 99 simulations is shown in Figure 4.16a. Mean head field, conditioned on steady state data, is shown in Figure 4.16b in a transient state after 80 years of uniform recharge. Conditioning on steady state head data results in more detail of the true f field represented by the conditional mean f (Figure 4.16a), which can be seen by comparing it with the true field in Figure 4.12a, while the mean head (Figure 4.16b) still appears much smoother than the true head (Figure 4.12b) field after 80 years of uniform recharge. Mean squared difference (MSE) between the conditioned f simulations and the true field is shown in Figure 4.17a and MSE between the conditioned head fields, at t=80 years, and the true head field is shown in Figure 4.17b. The average MSE between the true and conditioned f fields is 1.02 with a maximum of 5.20 and the average MSE between the true and conditioned head fields is 3.38 with a maximum of 19.32. Shown in Figure 4.18a is variance of the 99 conditioned f simulations. The variance of the head fields at t=80 years conditioned on data at t=0 years is shown in Figure 4.18b. Maximum conditional f variance is 1.11 and maximum variance of head at t=80 years is 9.13. The improvement obtained by conditioning on h data in addition to f data is quite noticeable and significant in all reported measures. Incorporating a single time (steady state) of head data is seen to reduce the average MSE between the conditioned f simulations and the true f field from 1.47 to 1.02, between the conditioned

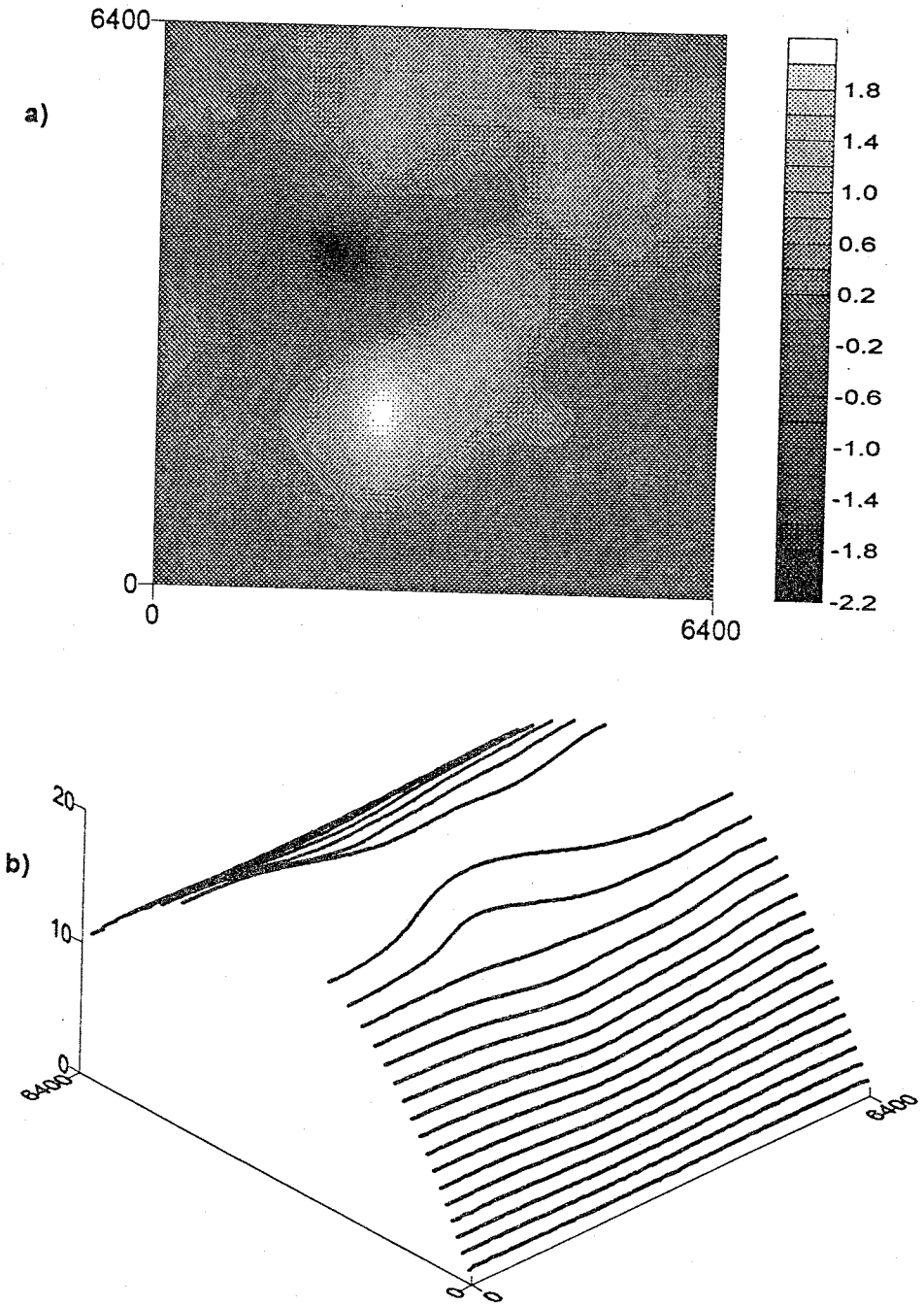


Figure 4.16: a) Ensemble mean of 99 f simulations conditioned on f data and steady state h data. b) Ensemble mean of resulting head fields, conditioned on initial steady state h data, after 80 years of uniform recharge.

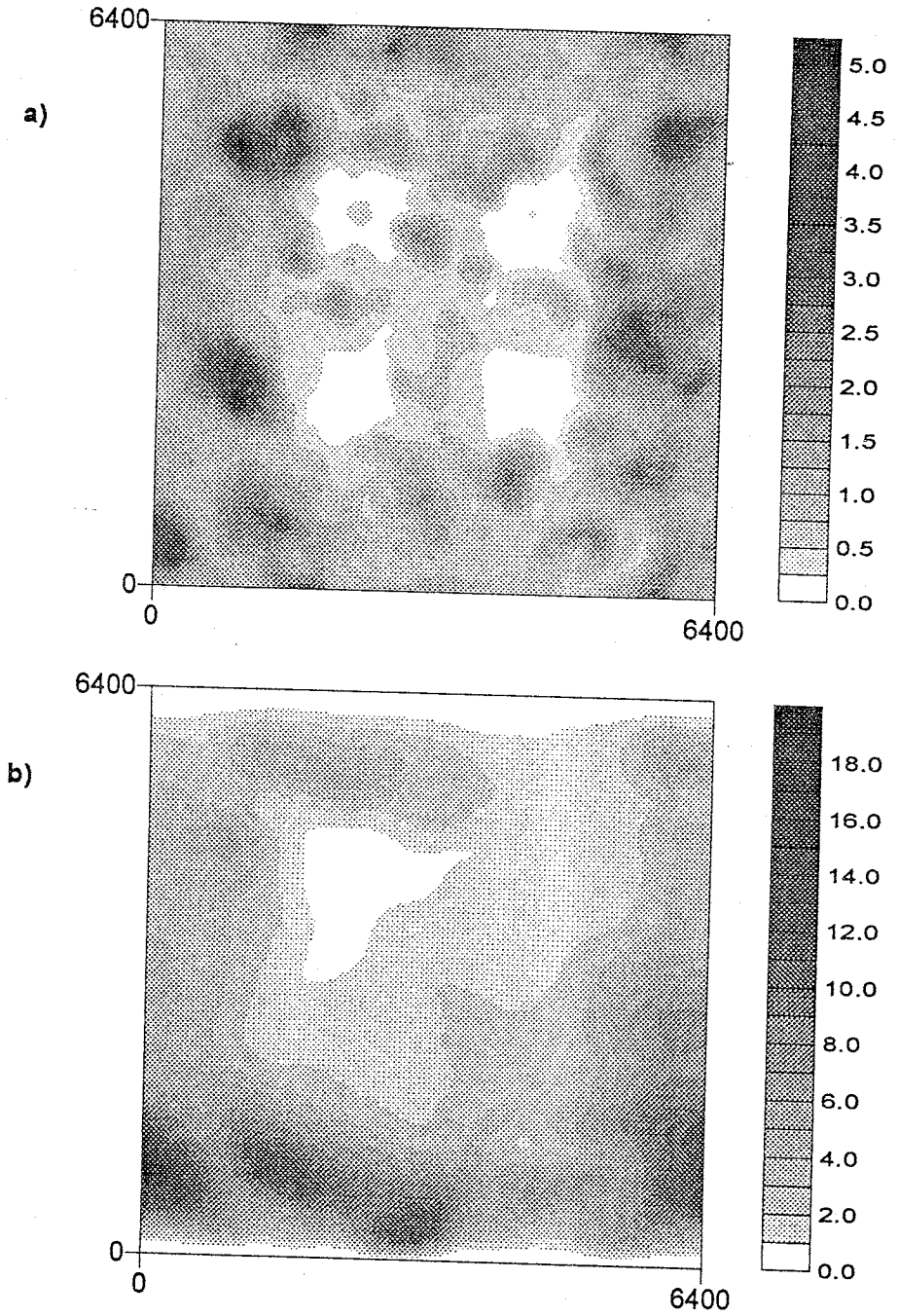


Figure 4.17: a) Mean squared error between 99 f simulations, conditioned on f data and steady state h data, and the true field. b) Mean squared error between 99 resulting head fields, conditioned on initial steady state h data, and the true head at 80 years.

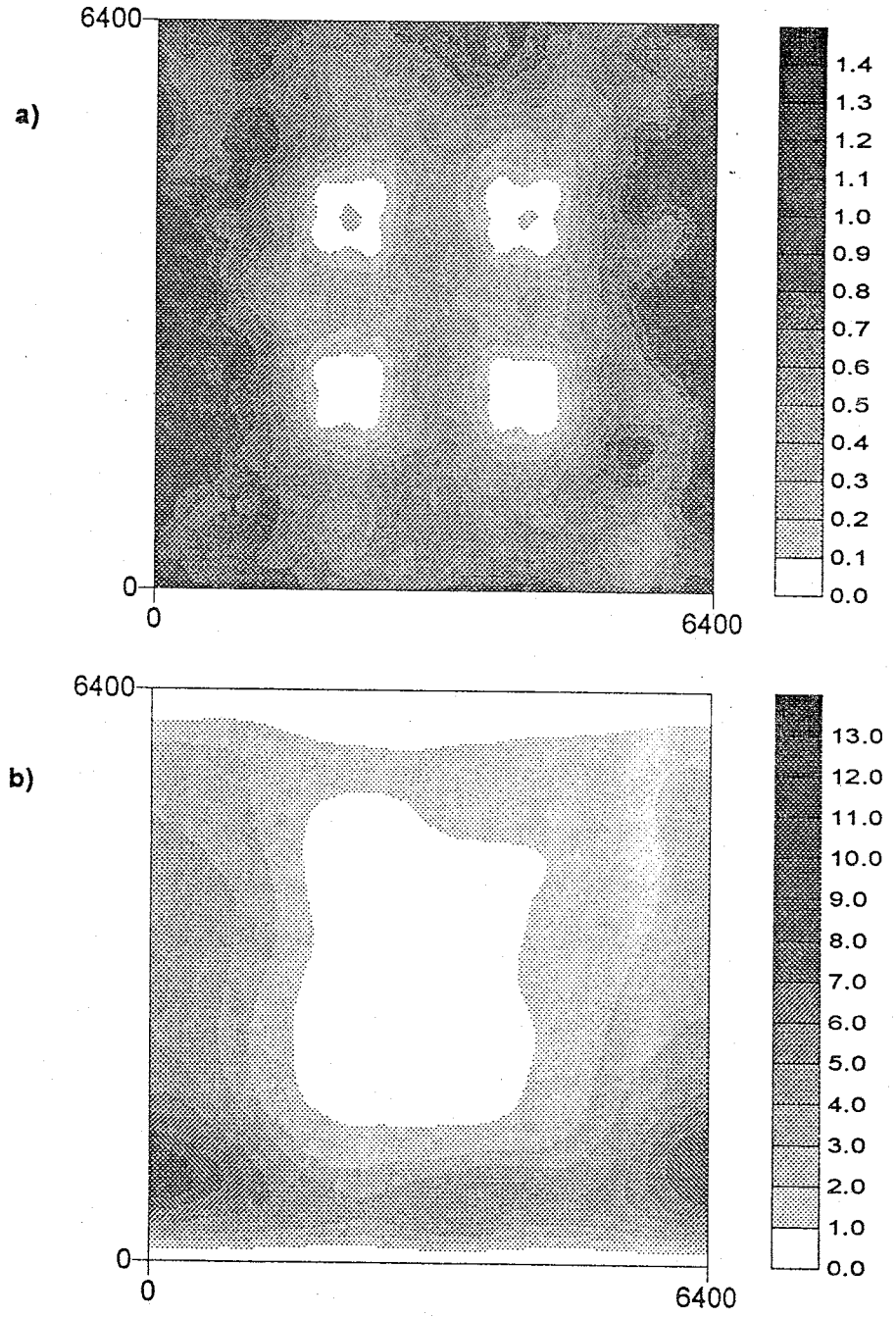


Figure 4.18: a) Ensemble variance of 99 simulated f fields, conditioned on f data and steady state h data. b) Variance of the 99 head fields, conditioned on initial steady state h data, after 80 years of uniform recharge.

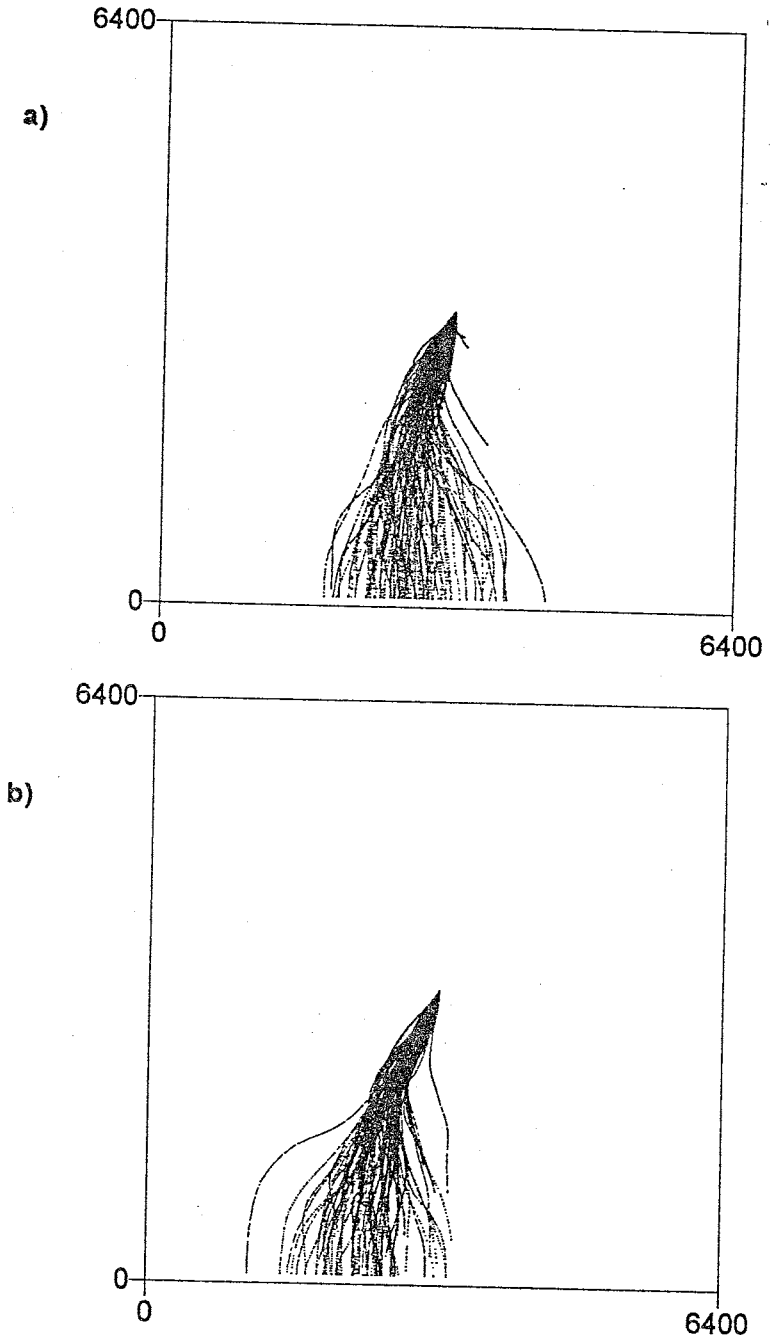


Figure 4.25: a) Advective 80 year particle pathlines from 100 simulations conditioned only on f data. b) Advective 80 year particle pathlines from 99 simulations conditioned on f data and h data at time t=0 years.

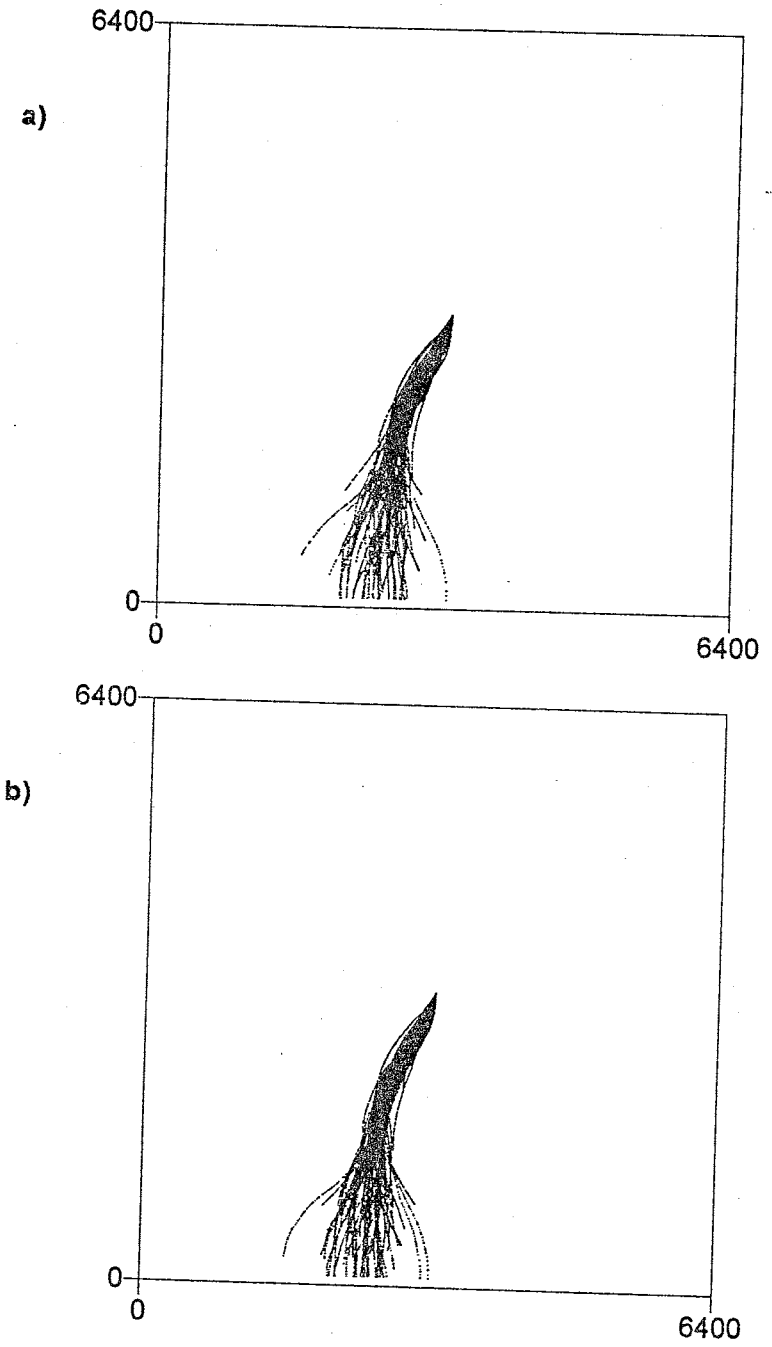
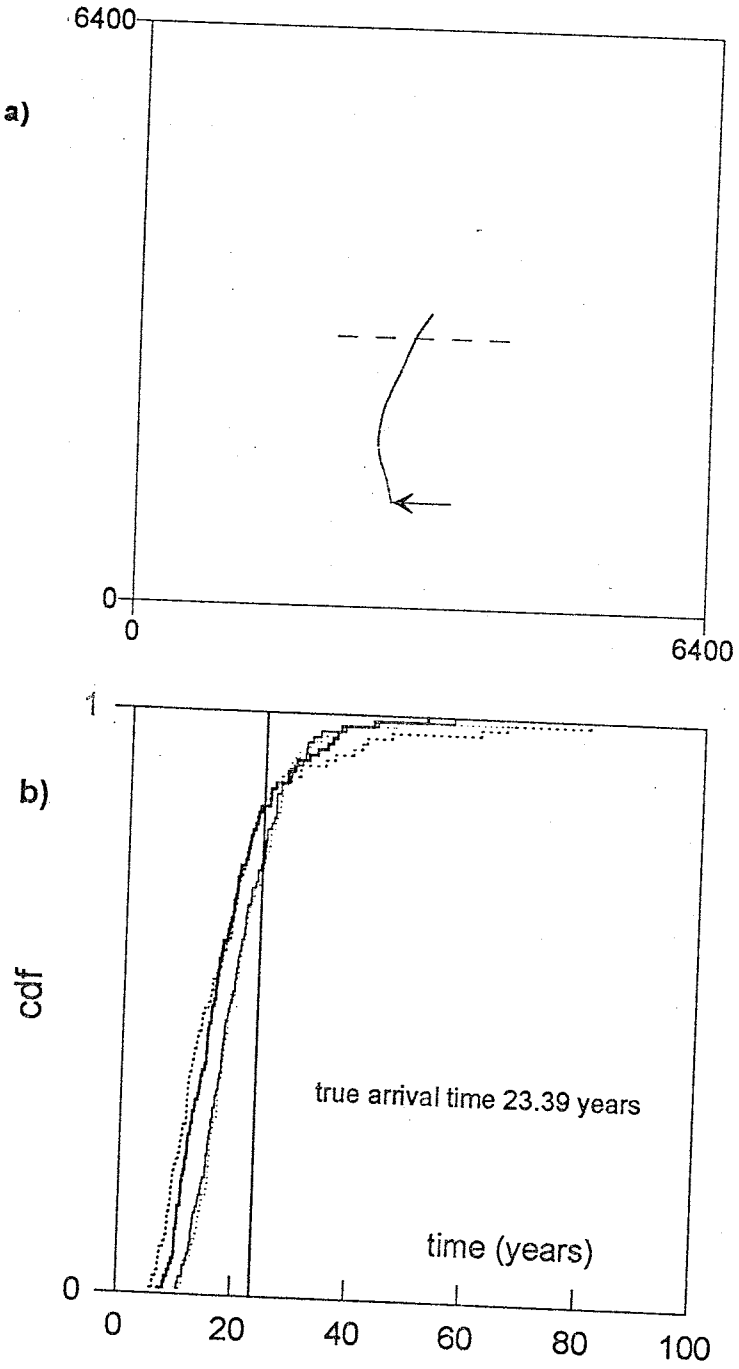


Figure 4.26: a) Advective 80 year particle pathlines from 100 simulations conditioned on f data and h data at times t=0, 15, 30, and 45 years. b) Advective 80 year particle pathlines from 105 simulations conditioned on f data and h data at times t=0, 15, 30, 45, 60, and 75 years.



gure 4.27: a) Advective 80 year particle pathline obtained from the true f d h fields. Arrow points to particle location at 80 years and dashed line y=2969. b) Cumulative distributions of particle arrival times at location =2969 for simulations conditioned on f data (thick dotted line), and on h ta at t=0 years (thick solid line), h data at t=0, 15, 30, and 45 years (thin id line), and h data at t=0, 15, 30, 45, 60, and 75 years (thin dotted line).

Not all of the released particles reached the model boundary. In Figure 4.25a some of the particles appear to get trapped in a low permeability zone and travel only a very short distance during the 80 years. The slowest particle, from the simulations conditioned only on f data (Figure 4.25a), arrived at location x=3355.05 y=2968.82 after 80 years of travel time. Rather than exclude some realizations, the location y=2969 was chosen to formulate cumulative arrival time distributions. This location y=2969 is indicated in Figure 4.27a by a horizontal dashed line. While this location choice has the advantage of using all simulations to formulate the cumulative arrival time distributions, it has a disadvantage in that the travel distance is only about one fifth of a f correlation scale. Even with such a relatively short travel distance, though, the effects of head data conditioning can be distinguished. Shown in Figure 4.27b are the cumulative arrival times at y=2969 of particles from simulations which are conditioned *1)* only on f data, *2)* f data and h data at t=0, *3)* f data and h data at t=0, 15, 30, and 45 years, and *4)* f data and h data at t=0, 15, 30, 45, 60, and 75 years. All 4 distributions have tails of long arrival times. The mean arrival time of *1)*, conditioning only on f data, is 17.86 with a variance of 147.5. Conditioning on h data at initial state, *2)*, reduces the variance of arrival times to 61.51 with a mean of 17.55. Conditioning on h data at t=0, 15, 30 and 45 years, distribution *3)*, reduced the variance to 49.59 with a mean of 20.06. Additional conditioning data at times 60 and 75 years, distribution *4)*, results in an arrival time variance increase to 56.18 with a mean of 20.43. The true particle pathline, shown in Figure 4.27a arrived at y=2969 at time t=23.39 years. Mean arrival times and standard deviations for each of the four conditioning data sets, along with the true arrival time, are

displayed in a bar-graph format in Figure 4.28. Even though conditioning on

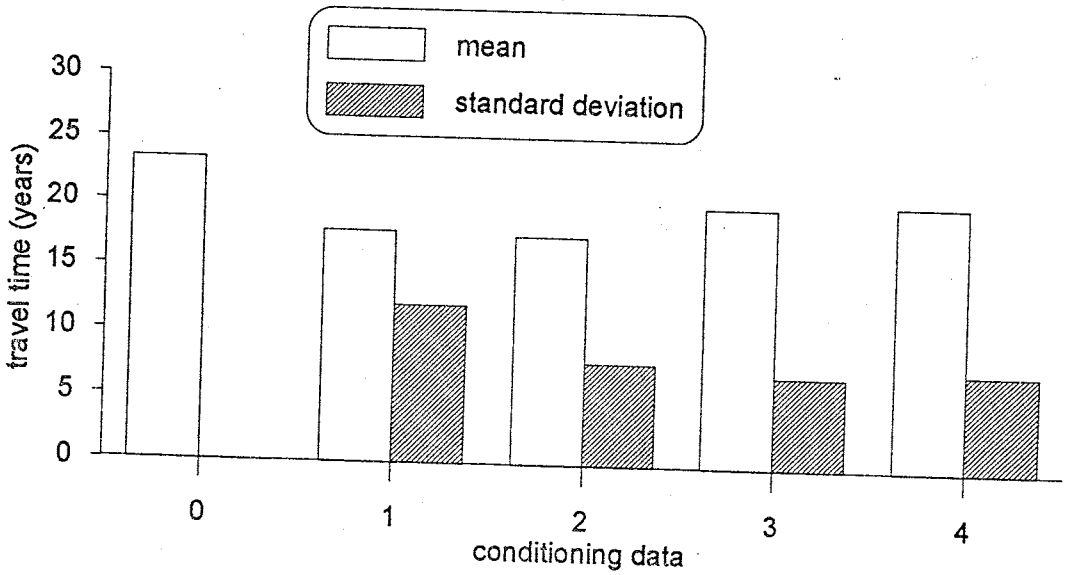


Figure 4.28: Mean arrival time and standard deviation of advective transport for: *1)* conditioning on f data only, *2)* conditioning on f and h at steady state, *3)* conditioning on f and h at t=0, 15, 30, and 45 years, and *4)* conditioning on f and h at t=0, 15, 30, 45, 60, and 75 years. The bar at *0* is the true arrival time of 23.39 years.

head data at all the times of t=0, 15, 30, 45, 60, and 75 years, distribution *4)*, does not result in the smallest arrival time variance, the mean arrival time is closest to the truth. Thus the time dependent head data are seen to both constrain the particle pathlines and improve advective travel time prediction. It is interesting to note that, of all the measures used to compare the effect of conditioning on time dependent head data, the particle pathlines appear to be most sensitive as clearly shown in the sequence of plots in Figures 4.25 and 4.26. Conditioning on head data has the noticeable effect of focusing particle trajectories into a narrow region. While some reduction in arrival time variances can be attributed to the head data conditioning, the effect is less noticeable; and

the mean arrival times are nearly the same for all sets of conditioning data. The same general trend in improvement due to head data conditioning can also be seen in the MSE, maximum MSE, and maximum variances where most of the improvement occurs for the first time of head data and there is subsequent diminishing improvement with additional times. But the improvement is more noticeable in the focusing of particle pathlines while the improvement trend is actually reversed for maximum MSE in head as the last two times of head data are added. These conditioning effects of the time dependent data, however, depend on matching the model predictions to the head data through iteration as is shown in the next section.

4.2.7 Effect of iterating to match the head data

A significant amount of extra computational effort is expended in iteratively solving the cokriging equation and flow model in order to obtain a better match of the model results to the head data (equation (3.20)). In this section the results of simulations conditioned on f data and h data at times t=0, 15, 30, 45, 60, and 75 years, iteratively so that the maximum difference between any head datum and the model results is less than 0.25 m, are compared to simulations conditioned on the same data without iteration to see if actually matching the head data is important. Conditioning on head data without iteration involves only solving the cokriging equation (3.21) to obtain a transmissivity field then solving the flow equation to obtain heads. Without iteration, the solution obtained from the flow model matches the head data only approximately. The average of the maximum absolute differences (AME) between all the head data and the model solution for 100 simulations was

1.62 m. Maximum difference occurred in all simulations at the last time step, 300, of an 80 year simulation. The largest absolute maximum difference (AME) from a single simulation was 3.36 m. With iteration, the maximum absolute difference becomes less than 0.25 m for all head data over all simulations. Shown in Figure 4.30a is the mean f field and in Figure 4.30b the mean head field at t=80 years conditioned without iteration on f data and h data at times t=0, 15, 30, 45, 60, and 75 years. While many of the major features of the true fields (Figure 4.12) are observed, a comparison of the mean squared error between the simulations and the true fields (Figure 4.31) shows that iteration moves the conditioned fields closer to the true fields. The average MSE of the 100 f fields conditioned without iteration is 1.03 and has a maximum of 7.57 compared with iteration, which reduces the average MSE to .76 with a maximum of 4.65. The performance measures, AMSE, MMSE, and MVAR for f and h are compared in the bar plot (Figure 4.29). This comparison appears to show that the iteration effort improves estimation by a factor of between about 1.2 to 1.8 or about 1.5 on average. Contrast this with the computational cost. Running on the SP1, the simulation without iteration took about 14 hours of CPU time while the iteration used another 28 hours for a total of 42 hours of CPU time. The estimation improvement of a factor of about 1.5 was obtained at 3 times the cost.

The effect of iterating to match the head data is more clearly illustrated by comparing the pathlines without iteration (Figure 4.33a) to pathlines with iteration (Figure 4.26b). Iteration is seen to tightly constrain the particle paths, especially in the vicinity of head data. Iteration also constrains the particle arrival times. In Figure 4.33b are shown the cumulative distributions

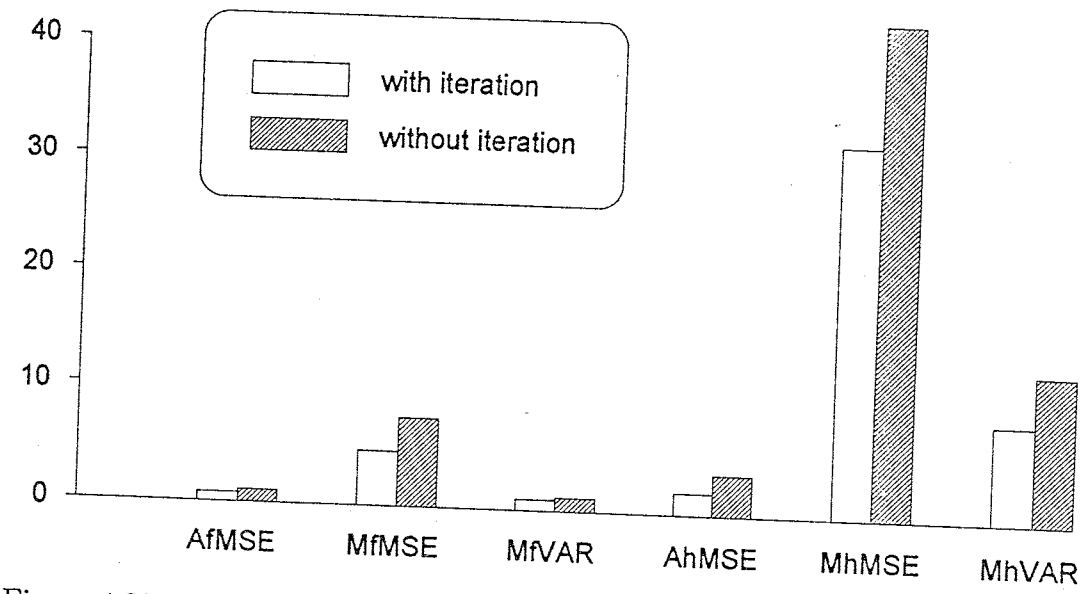


Figure 4.29: Comparison of six performance measures for conditioning on data with and without iteration.

of particle arrival times at y=2969 for conditioning with iteration (dotted line) and conditioning without iteration (solid line). With iteration, as mentioned previously, the mean arrival of 105 conditioned simulations was 20.43 years with a variance 56.18. Without iteration, the mean arrival time of 99 conditioned simulations was 25.23 but the variance increased to 143.0. For one of the simulations conditioned without iteration, the particle did not even arrive at y=2960 during the 80 year simulation period.

Evaluation of the worth of iteration must be made against the cost of the effort. Iteration definitely does focus the particle trajectories into a more narrow zone and improve recognition of the detail in spatial heterogeneity. But uncertainties inherent in the conceptual modeling may be of greater influence and first-cut simple cokriging simulations may be sufficient.

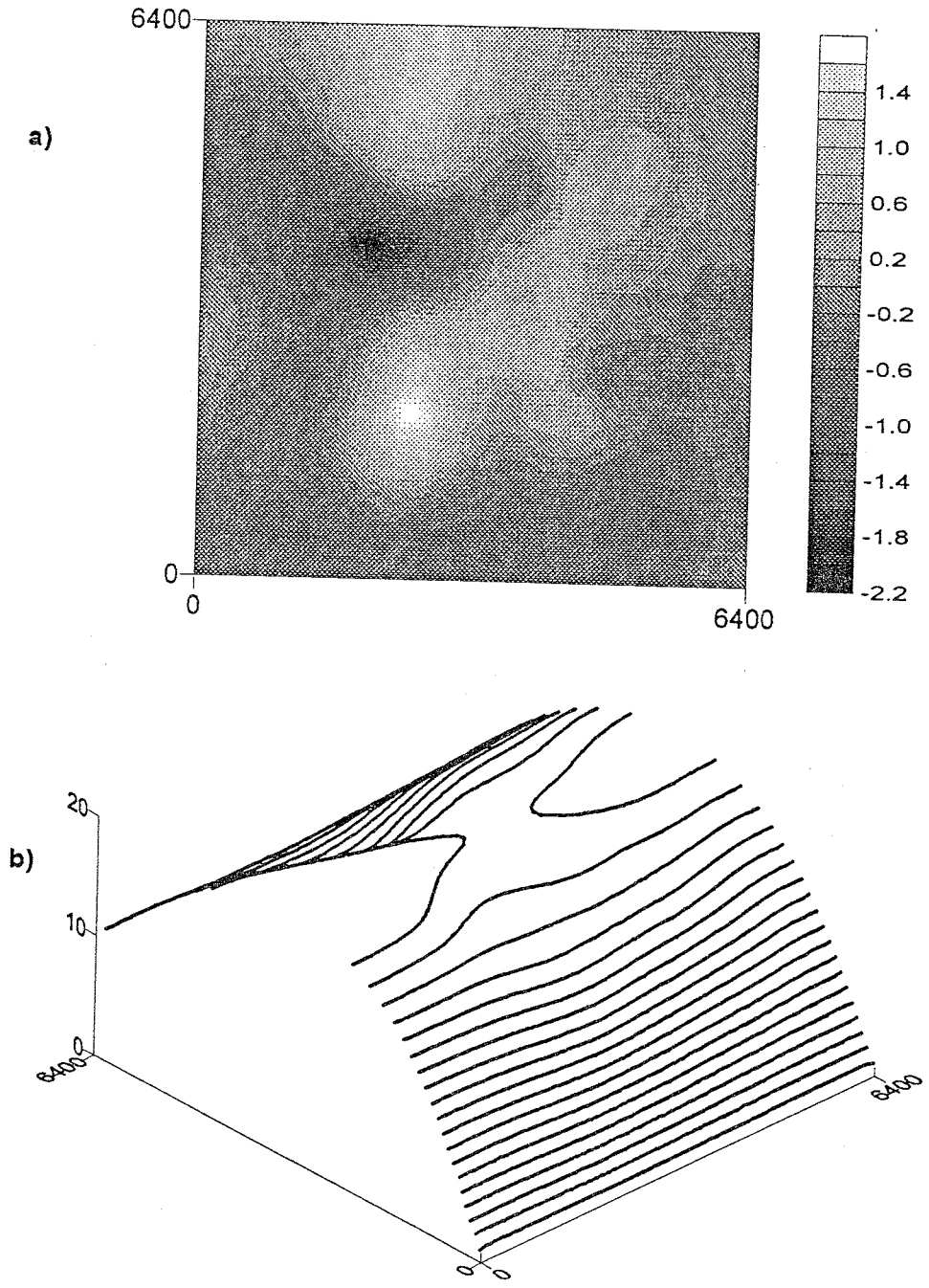


Figure 4.30: a) Ensemble mean of 100 f simulations conditioned, without iteration, on f data and h data at times 0, 15, 30, 45, 60 and 75 years. b) Ensemble mean of resulting head fields, conditioned, without iteration, on h data, after 80 years of uniform recharge.

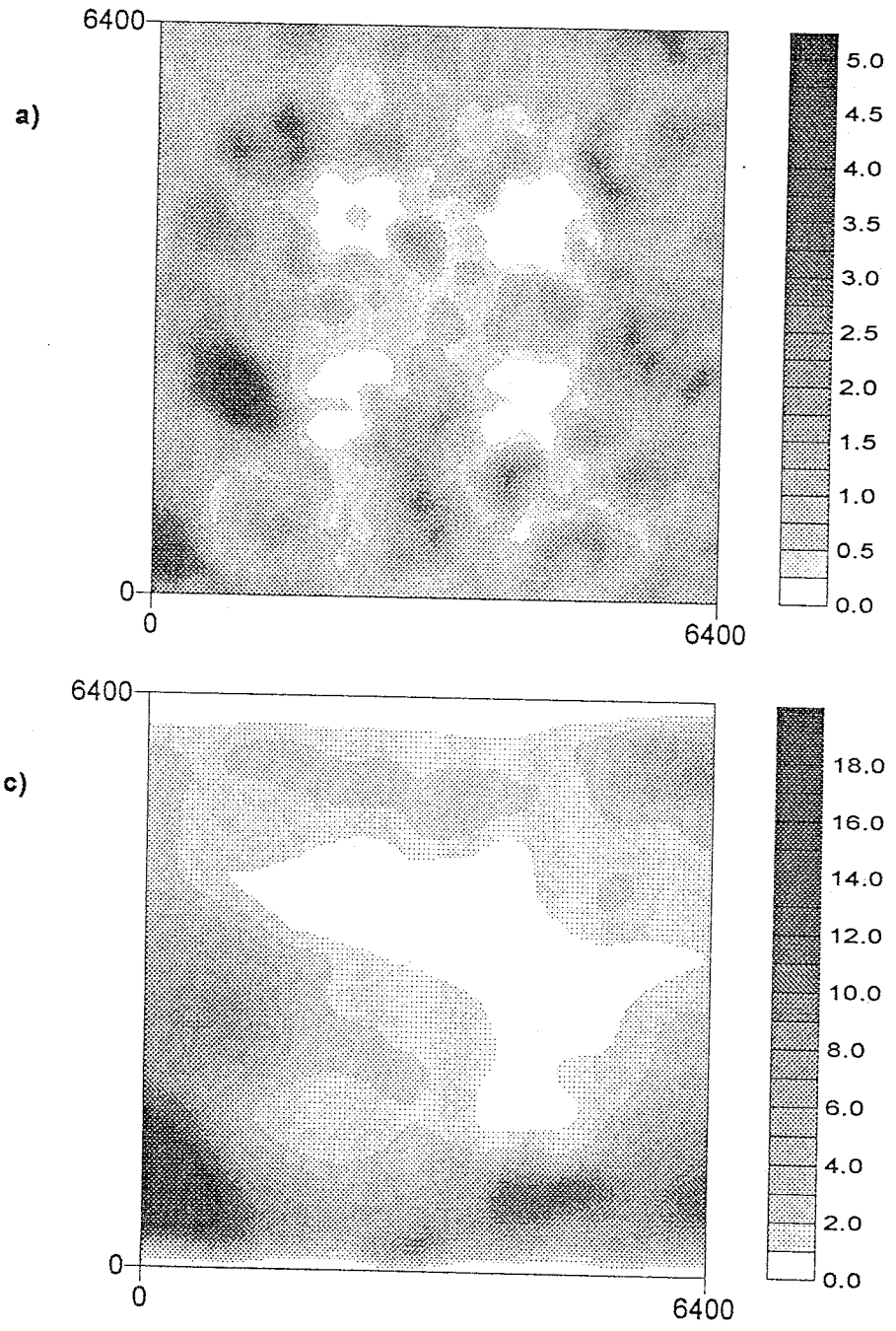


Figure 4.31: a) Mean squared error between the true field and 100 f simulations conditioned, without iteration, on f data and h data at times 0, 15, 30, 45, 60 and 75 years. b) Mean squared error between 100 resulting head fields, conditioned without iteration, and the true head at 80 years.

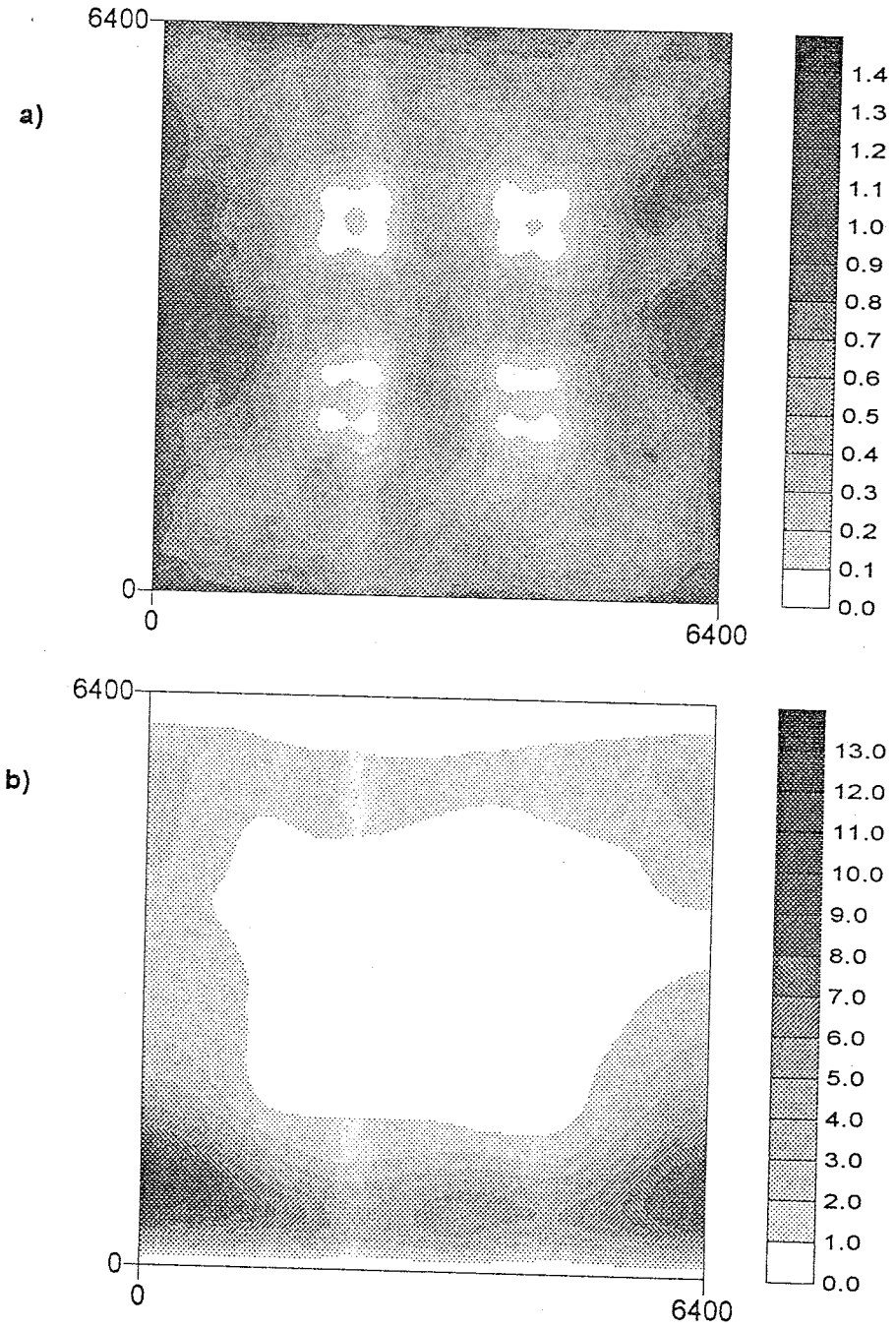


Figure 4.32: a) Ensemble variance of 100 simulated f fields, conditioned, without iteration, on f data and h data at times 0, 15, 30, 45, 60 and 75 years. b) Variance of the 100 head fields, conditioned without iteration, after 80 years of uniform recharge.

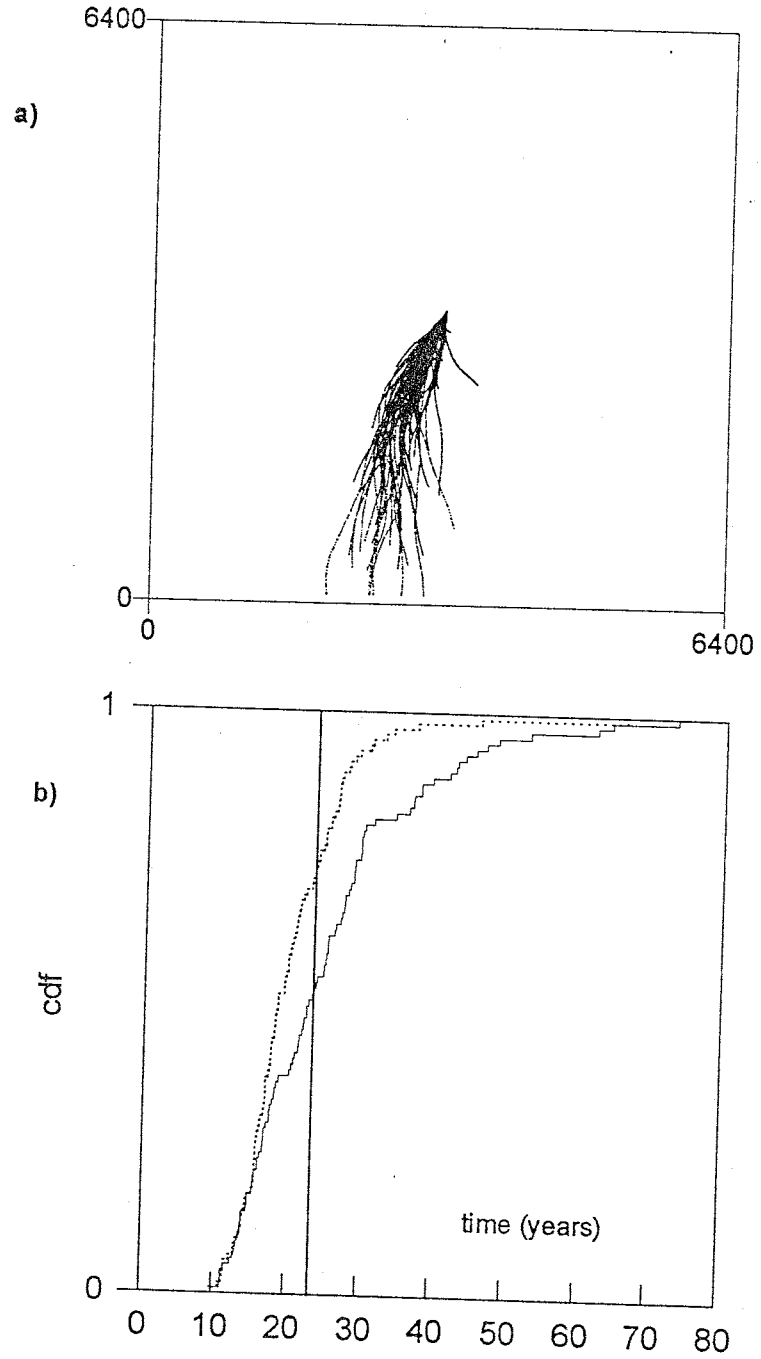


Figure 4.33: a) Advective 80 year particle pathlines from 100 simulations, conditioned without iteration, on f data and h data at times t=0, 15, 30, 45, 60, and 75 years. b) Cumulative distributions of particle arrival times at location y=2969 for simulations conditioned on f data and h data at t=0, 15, 30, 45, 60, and 75 years with iteration (dotted line) and without iteration (solid line).

4.2.8 The sequential conditioning approach

In section 3.3.1 a sequential (quasi-steady state) approach to iteratively conditioning on h data is described which uses only the head covariance and cross-covariance in space at a given time (equation (3.19)). Effectively, this treats each time step as independent by ignoring the cross-correlation of heads in time. At each time step where head data is available for conditioning, this method proceeds by iteratively modifying the f field and re-solving the flow equation from the previous time step where head data was conditioned on until a sufficiently close match is obtained. The iteratively modified f and data matching head fields are then used to proceed on to the next time step where more head data are available for conditioning. Thus the conditioned f field changes each time the algorithm encounters more head data. No global convergence behavior was observed in this algorithm, i.e. the changes in the conditioned f fields do not become progressively smaller nor does the final conditioned f field adequately represent any head data at earlier times. The merit of this approach, however, may be found in a situation where many time steps of head data over a long period are to be conditioned on and the cokriging matrix containing all cross-time covariances (equation (3.21)) becomes too unwieldy.

In this section conditioning on f data and h data at times t=0, 15, 30, 45, 60, and 75 years using the sequential conditioning approach is compared to results obtained by conditioning on the same data collaterally using all cross-time covariances (section 4.2.5). The mean of 94 f field simulations conditioned at the final adaptation to h data at time t=75 years is shown in Figure 4.34a

and the mean head field at time t=80 years is shown in Figure 4.34b. The mean squared errors between the conditioned f simulations, shown in Figure 4.35a, have an average of .89 and a maximum of 5.0 which is better than the results obtained by conditioning collaterally on the same data without iteration as described in section 4.2.7. The same is true for the MSE between the true head field at time t=80 years (Figure 4.35b) and the conditioned head simulations which have an average of 1.96 and a maximum of 27.12. A slightly different result is seen in the variance of the 94 conditioned f field simulations shown in Figure 4.36a, however. The maximum variance in Figure 4.36a is 1.63 which is significantly higher than the maximum variance obtained by conditioning collaterally without iteration, 1.14 (Figure 4.32a), or with iteration, .94 (Figure 4.24a). The maximum head variance in Figure 4.36b of 9.27, however, is smaller than maximum head variance of 12.74 obtained by collateral conditioning without iteration but larger than the maximum head variance of 8.34 from collateral conditioning with iteration. The larger variance of f in the sequential method is probably due to the fact that the f simulations change each time they are conditioned on head data. The head field at 80 years has been conditioned on data at 75 years and iterated to match within a maximum error of 0.25 m while the collateral conditioning without iteration may result in a head field which misses the data by more than 3 m. This may account for the smaller sequential method maximum head variance than that obtained by collaterally conditioning without iteration. This nearly exact, within 0.25 m, conditioning effect is also seen in the advective particle pathlines shown in Figure 4.37a. The pathlines are constrained to a narrow region due to the conditioning effect of the head data despite the changing, or adapting, f field. A higher variance

in the f field may be partially responsible for the earlier mean arrival time of the particles at y=2969 of 16.61 years (Figure 4.37b solid line) compared to the mean arrival time of 20.43 years for particles traveling in fields conditioned collaterally with iteration. Recall that a particle advecting through the true field arrives at y=2969 in 23.39 years. The variance of particle arrival times at y=2969 in Figure 4.37b is 33.16 for the sequential conditioning method and 56.18 for the collateral conditioning with iteration.

4.2.9 Effect of the sampling interval vs. the number of sampling intervals

In the previous sections it is seen that collaterally conditioning on head data in more times steps reduces the estimation variance of log transmissivity and head perturbations, provides a closer match between the conditional mean and true fields, and constrains the pathlines and travel times of advecting particles. In these examples, the sampling interval was always 15 years and the time constant of the hypothetical flow model system about 43 years. The question remains, is the improvement in the performance measures due to the increased number of samples or due to the longer proportion of the system time constant sampled? For example, would 2 or 3 sampling intervals spaced over 75 years provide as much information as 5 fifteen year sampling intervals over that 75 year period?

In this section the results of simulations conditioned on 6 different sets of h data are presented which may provide some small insight into this question. These sets, numbered for reference, are f data and: *7)* h data at times t=0, 1, 2, 3, 4, and 5 years, *8)* h data at times t=0, 2, 4, 6, 8, and 10

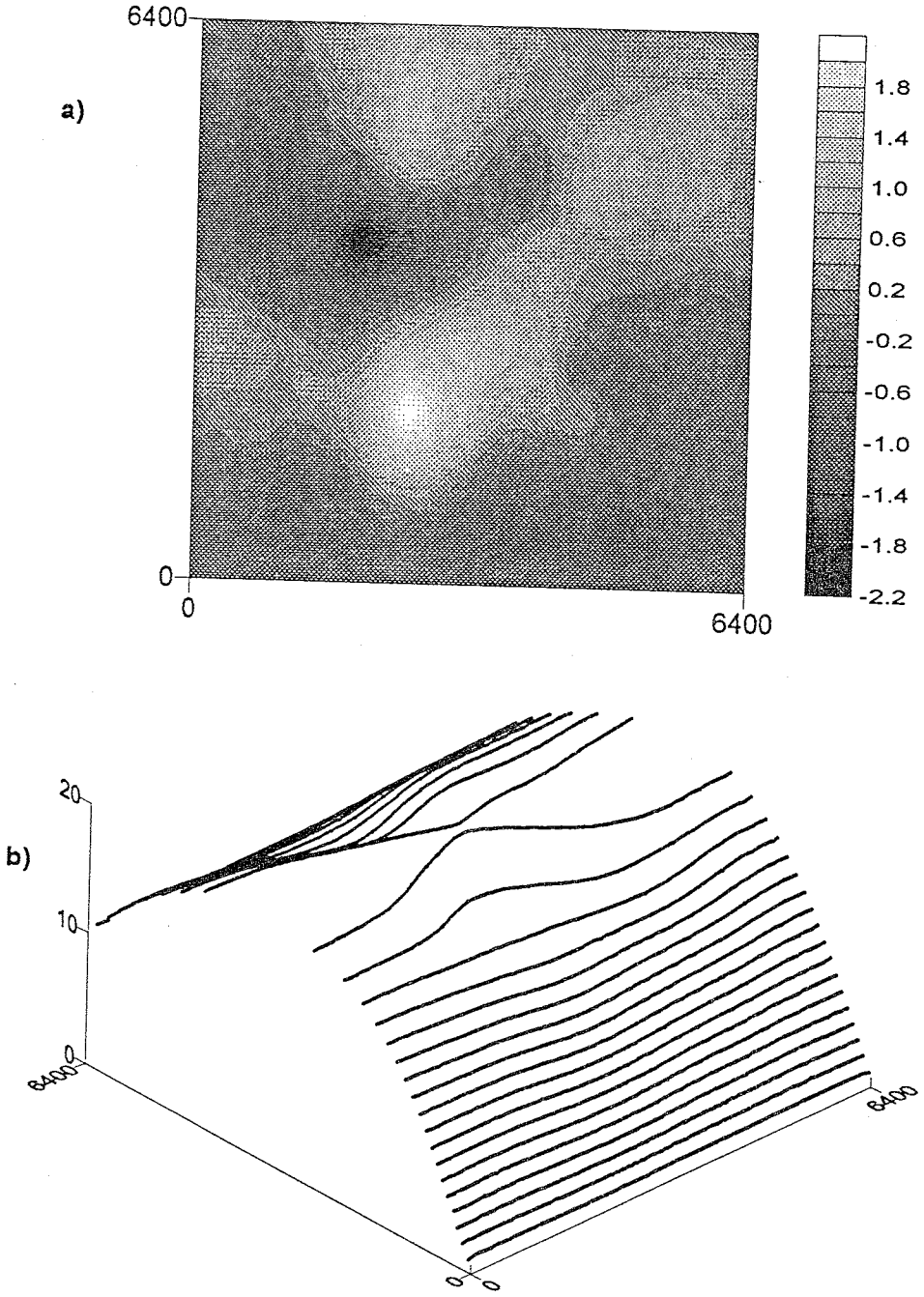


Figure 4.34: a) Ensemble mean of 94 f simulations conditioned by the sequential method on f data and h data at times 0, 15, 30, 45, 60 and 75 years. b) Ensemble mean of resulting head fields, conditioned by the sequential method on h data, after 80 years of uniform recharge.

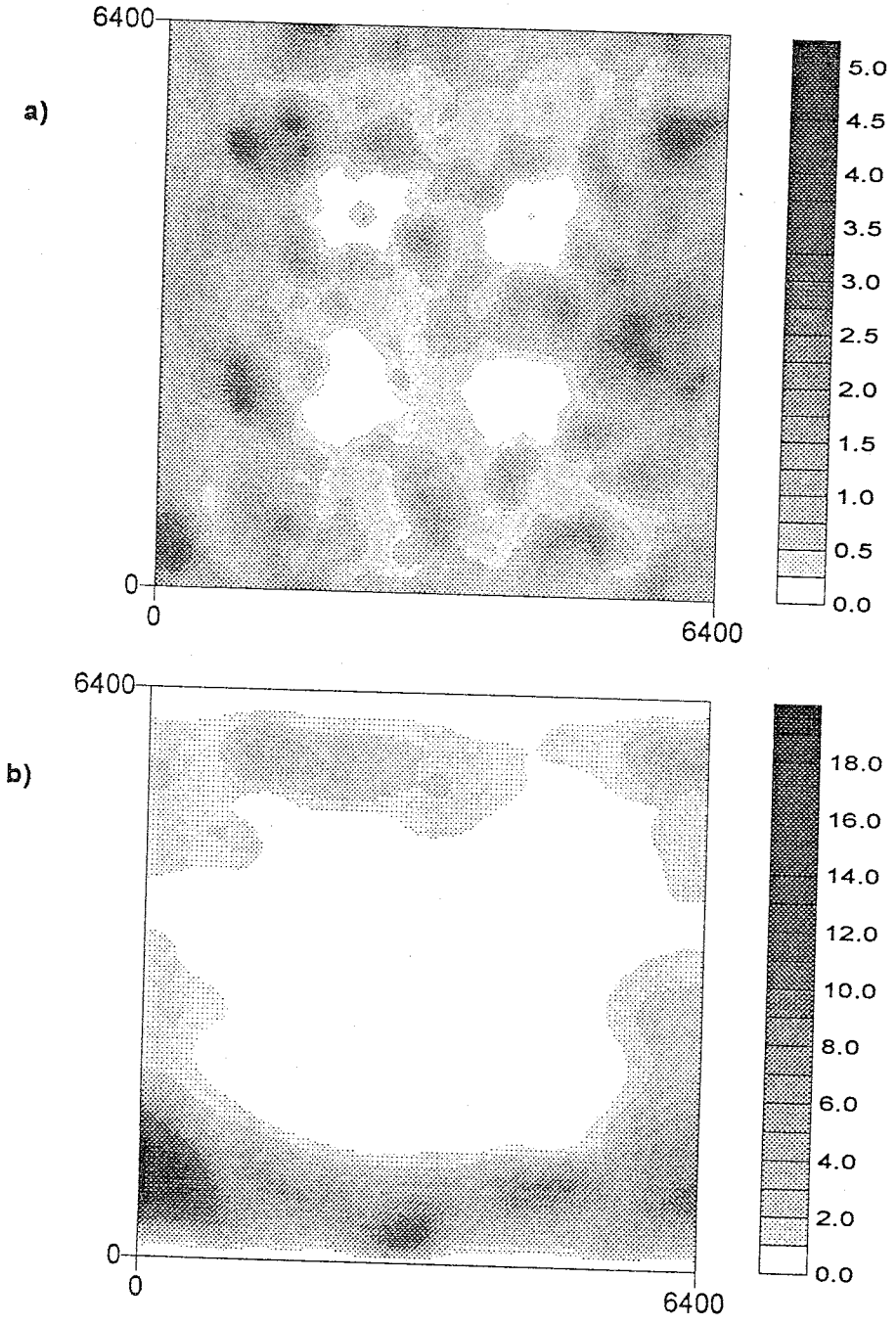


Figure 4.35: a) Mean squared error between the true field and 94 f simulations conditioned by the sequential method on f data and h data at times 0, 15, 30, 45, 60 and 75 years. b) Mean squared error between 94 resulting head fields conditioned by the sequential method and the true head at 80 years.

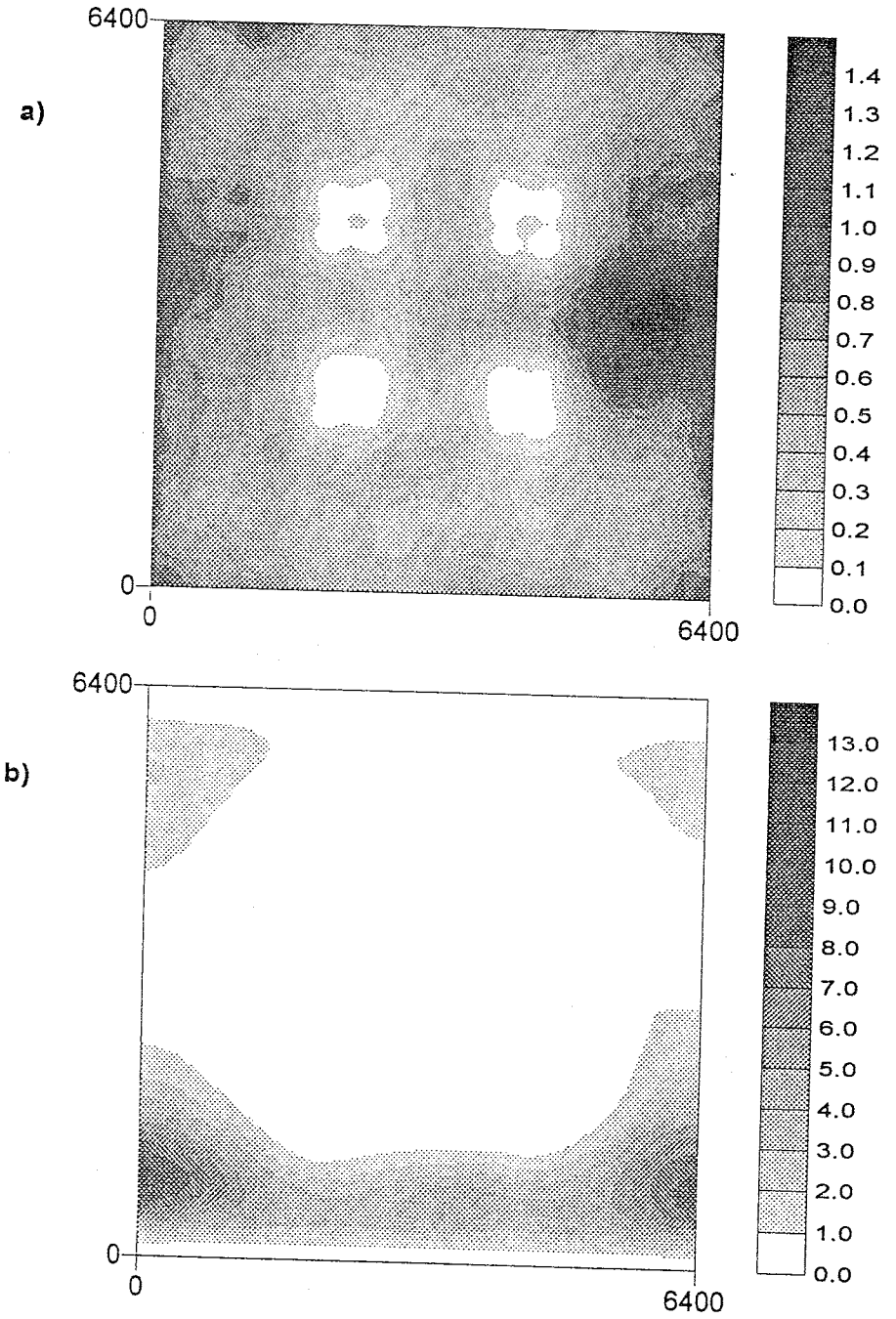


Figure 4.36: a) Ensemble variance of 94 simulated f fields, conditioned by the sequential method on f data and h data at times 0, 15, 30, 45, 60 and 75 years. b) Variance of the 94 head fields conditioned by the sequential method after 80 years of uniform recharge.

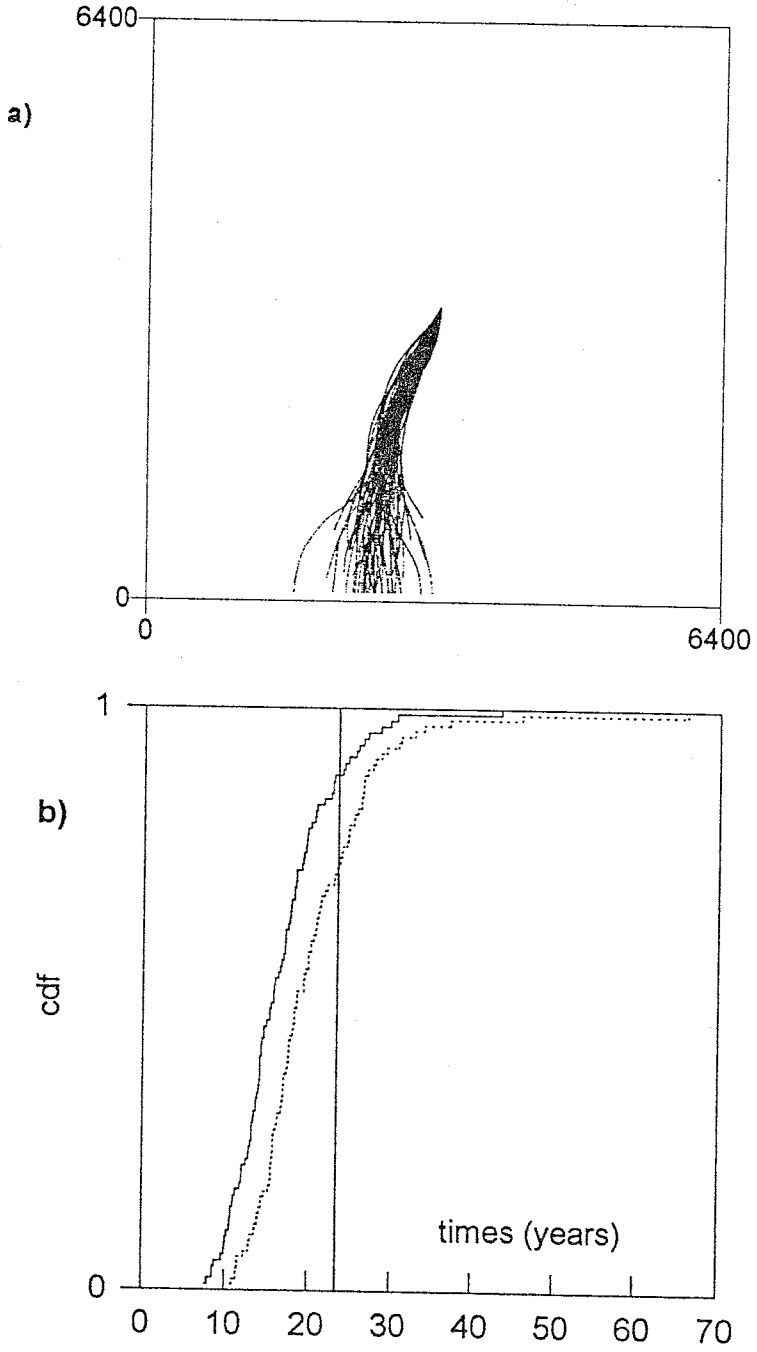


Figure 4.37: a) Advective 80 year particle pathlines from 94 simulations conditioned by the sequential method on f data and h data at times t=0, 15, 30, 45, 60, and 75 years. b) Cumulative distributions of particle arrival times at location y=2969 for simulations conditioned on f data and h data at t=0, 15, 30, 45, 60, and 75 years by the sequential method (solid line) and collaterally with iteration (dotted line).

years, *9)* h data at times t=0, 10, 20, 30, 40, and 50 years, *10)* h data at times t=0, 25, 50, and 75 years, *11)* h data at times t=0, 35, and 75 years and *12)* h data at times t=0, and 75 years. In the data set of 5-one year intervals, reference number *7)*, h data are sampled at 5 times plus the initial state but only over less than 10% of the time constant of the transient state. The next set, reference *8)*, about 25% of the transient time constant is sampled with 5 time intervals. In set reference *9)* head data are also sampled for 5 times but at 10 year intervals so that more than one time constant of the transient state is observed. Simulation sets reference *10)*, *11)*, *12)* progressively reduce the number of sample intervals from 3 to 1 of a transient interval about 1.75 time constants long. Six performances are graphically compared in Figure 4.38 and also summarized in Table 4.3. It's apparent from Figure 4.38 that the

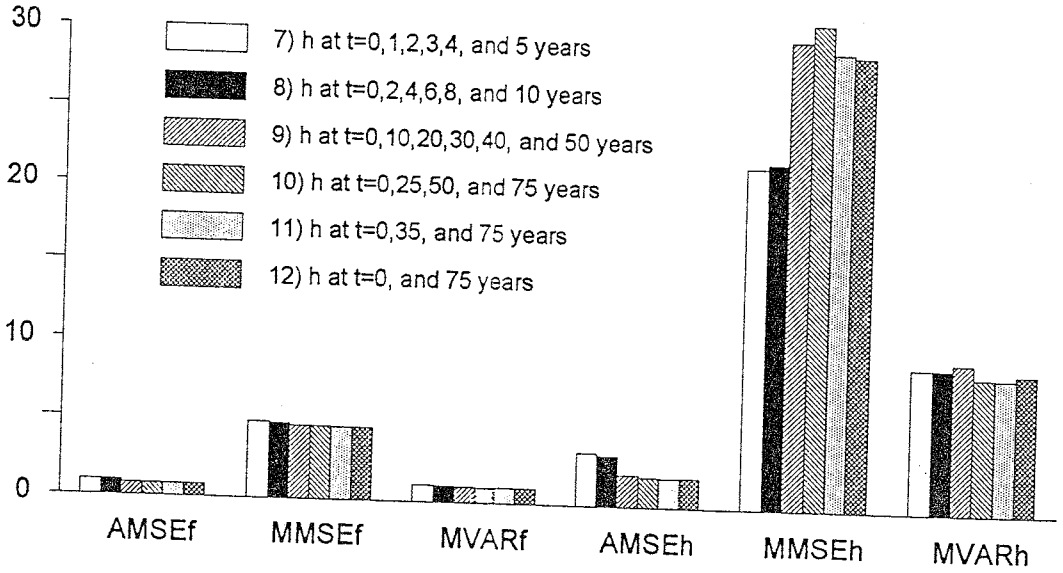


Figure 4.38: Six performances measures of the effect of various sampling time intervals on conditioning results.

performance measures do not change drastically over the six different time in-

tervals of head data conditioning. The measure of how well the conditioned f fields match the true field, AMSEf, ranges from a maximum of .936 at *7)*, the 5-one year intervals, to a minimum of .767 at *10)*, which is three 25 year intervals. Similar patterns are also seen in the maximum MSE, MMSEf, and the maximum variance, MVARf. The average MSE for head fields, AMSEh, which indicates how well the conditioned head field simulations match the true head shows, however, a more substantial improvement as conditioning is done on wider intervals over the transient time constant. This measure improves from 3.31 in reference set *7)* to 1.84 in reference set *11)*, almost a factor of 2. The improvement in the measure, AMSEh, is offset by a comparable worsening in the maximum MSE, MMSEh, which has a minimum of 21.65 at reference set *7)* and a maximum of 30.88 at reference set *10)*. With the exception of MMSEh the performance measures all show fairly similar behavior for the sampling intervals which span the 80 year simulation period and show a definite trend towards poorer performance as less of the transient state is observed. Comparing 5-one year intervals (reference *7)*) with one 75 year interval (reference *12)*), particularly using the AMSEh, seems to indicate that the proportion of the time constant sampled is more important than the number of sampling intervals in that proportion of the time constant. A similar, but not as dramatic, improvement in AMSEf from reference *7)* to *12)* says that to obtain the most information on heterogeneity of transmissivity from a head data in a transient state, data covering a longer span of the transient condition are more important than the quantity of data.

The effect of conditioning on head data spanning a larger portion of the transient state is illustrated by the pathlines of advective particles in the

conditioned simulations. Shown in Figure 4.39a are the pathlines in simulations conditioned on 5 h data intervals of a one year spacing and in Figure 4.39b the pathlines are conditioned on 5 h data intervals with a two year spacing. In both cases the pathlines are as nearly unconstrained as the simulations conditioned only on initial state h data. Conditioning on 5 intervals of 10 years (Figure 4.40a), 3 intervals of 25 years (Figure 4.40b), 2 intervals at 35 and 75 years (Figure 4.41a) or one interval only at 75 years (Figure 4.41b) however, produce nearly indistinguishable constraints on the particle pathlines. Again it is interesting to note that particle trajectories are more sensitive to head data conditioning than any of the other performance measures examined.

4.2.10 Summary of the hypothetical model simulation results

The hypothetical model described in section 4.2.1 has been used to condition simulations on 12 different sets of time dependent h data sampled at the locations shown in Figure 4.11. Besides the visual comparison provided by particle pathlines, 6 performance measures have been used to compare the conditioning effects of the various sets of h data. The 6 performance measures are average MSE between the true f field and the conditioned simulations, AMSEf, the maximum MSE of the f simulations, MMSEf, the average MSE between the conditioned head simulations and the true head at 80 years, AMSEh, the maximum MSE for the head simulations, MMSEh, the maximum ensemble variance of the conditioned f simulations, MVARf, and the maximum ensemble variance of the conditioned head simulations, MVARh. All simulations are conditioned on f data. The 12 sets of h data at the locations shown in Figure 4.11 are given reference numbers as follows:

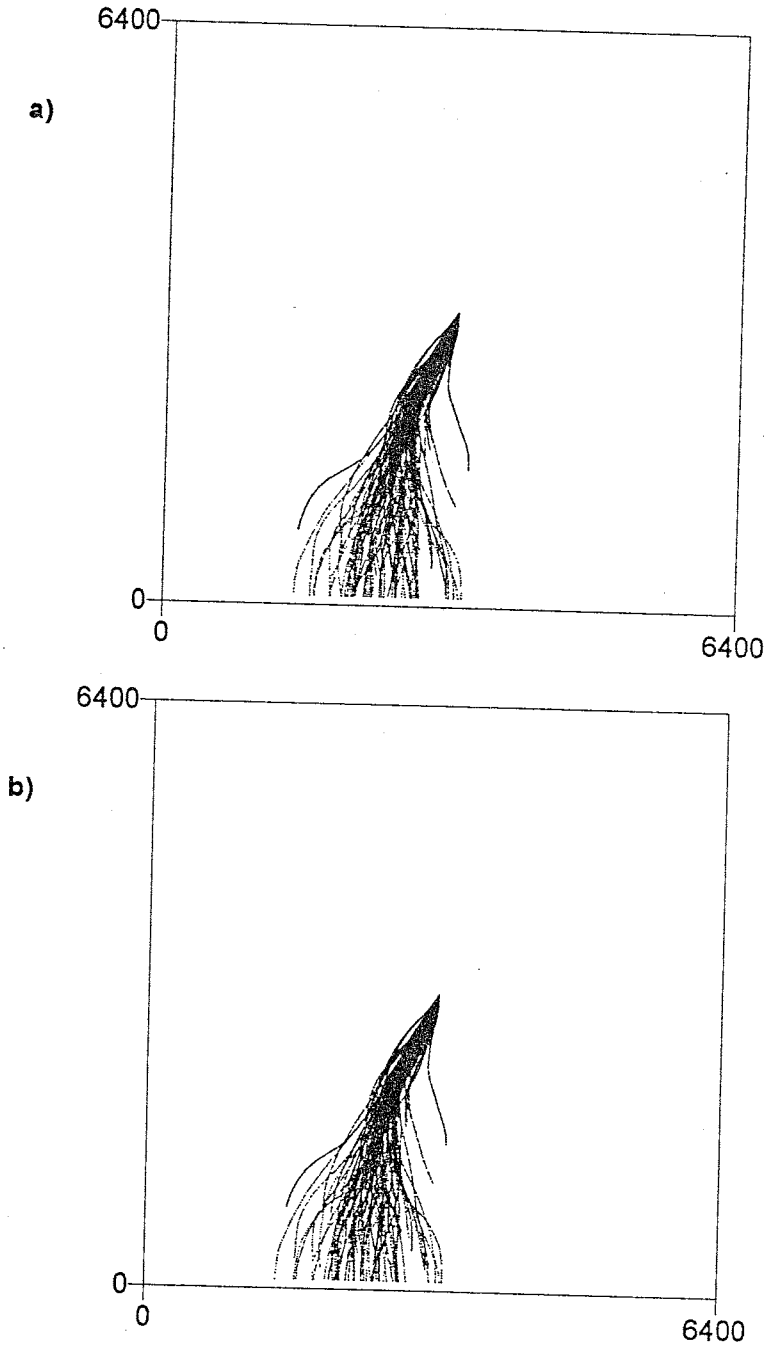


Figure 4.39: a) Advective 80 year particle pathlines from 113 simulations conditioned on *7) f* data and *h* data at t=0, 1, 2, 3, 4, and 5 years. b) Advective 80 year particle pathlines from 113 simulations conditioned on *8) f* data and *h* data at t=0, 2, 4, 6, 8, and 10 years.

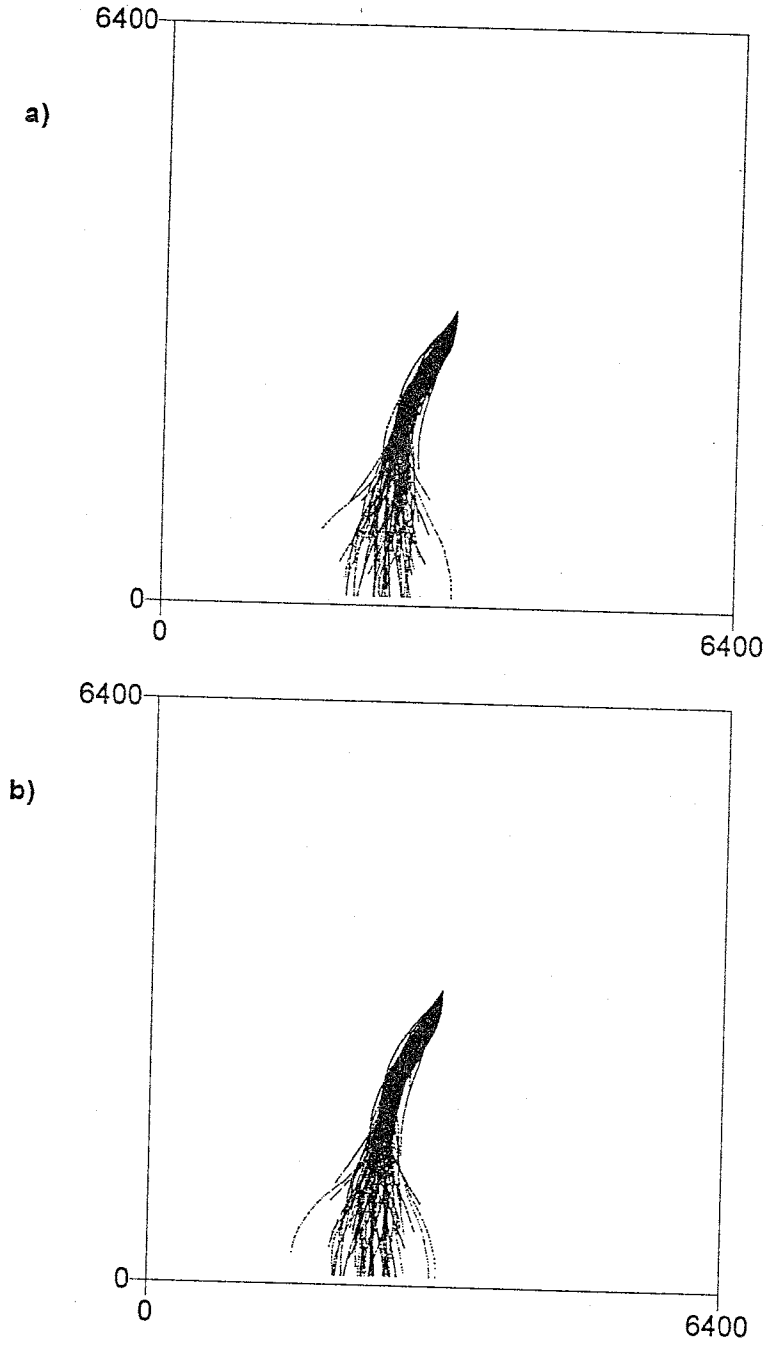


Figure 4.40: a) Advective 80 year particle pathlines from 105 simulations conditioned on *9) f* data and h data at t=0, 10, 20, 30, 40, and 50 years. b) Advective 80 year particle pathlines from 113 simulations conditioned on *10) f* data and h data at times t=0, 25, 50, and 75 years.

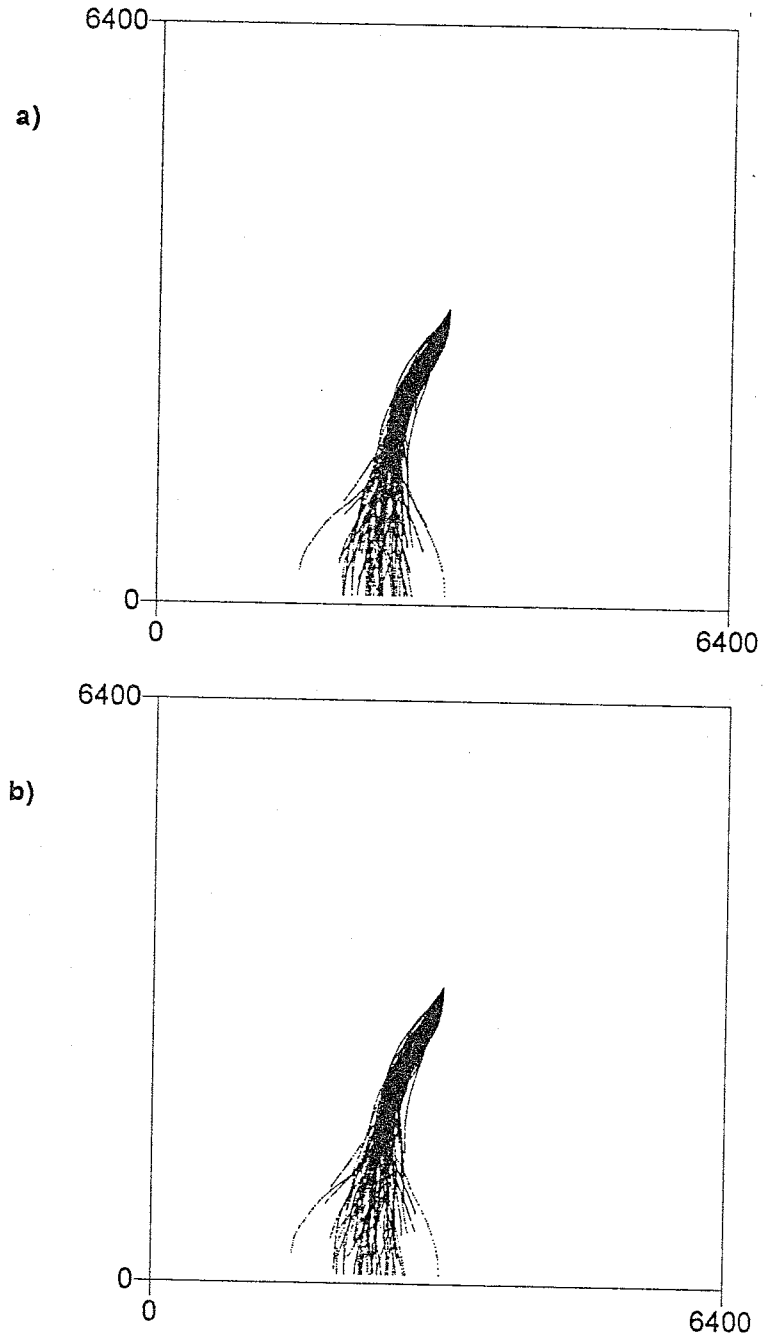


Figure 4.41: a) Advective 80 year particle pathlines from 108 simulations conditioned on *11)* f data and h data at t=0, 35, and 75 years. b) Advective 80 year particle pathlines from 104 simulations conditioned on *12)* f data and h data at times t=0 and 75 years.

1. no h data, simulations conditioned on f data only,
2. h data at the initial steady state at time t=0,
3. h data at times t=0, 14, 30, and 45 years,
4. h data at times t=0, 15, 30, 45, 60, and 75 years,
5. h data at times t=0, 15, 30, 45, 60, and 75 years where conditioning does not include iteration to improve the match between model and data,
6. h data at times t=0, 15, 30, 45, 60, and 75 years using the sequential method,
7. h data at times t=0, 1, 2, 3, 4, and 5 years,
8. h data at times t=0, 2, 4, 6, 8, and 10 years,
9. h data at times t=0, 10, 20, 30, 40, and 50 years,
10. h data at times t=0, 25, 50, and 75 years,
11. h data at times t=0, 35, and 75 years,
12. and h data at times t=0 and 75 years.

The 6 performance measures for the twelve sets of conditioning data are summarized in Table 4.3 and displayed graphically in Figure 4.42. Approximate CPU time (single node of the SP1) for simulations conditioned on each data set is displayed graphically in Figure 4.43.

Simulations of head and log transmissivity, using the flow model described in section 4.2.1, were conditioned on twelve different sets of data for

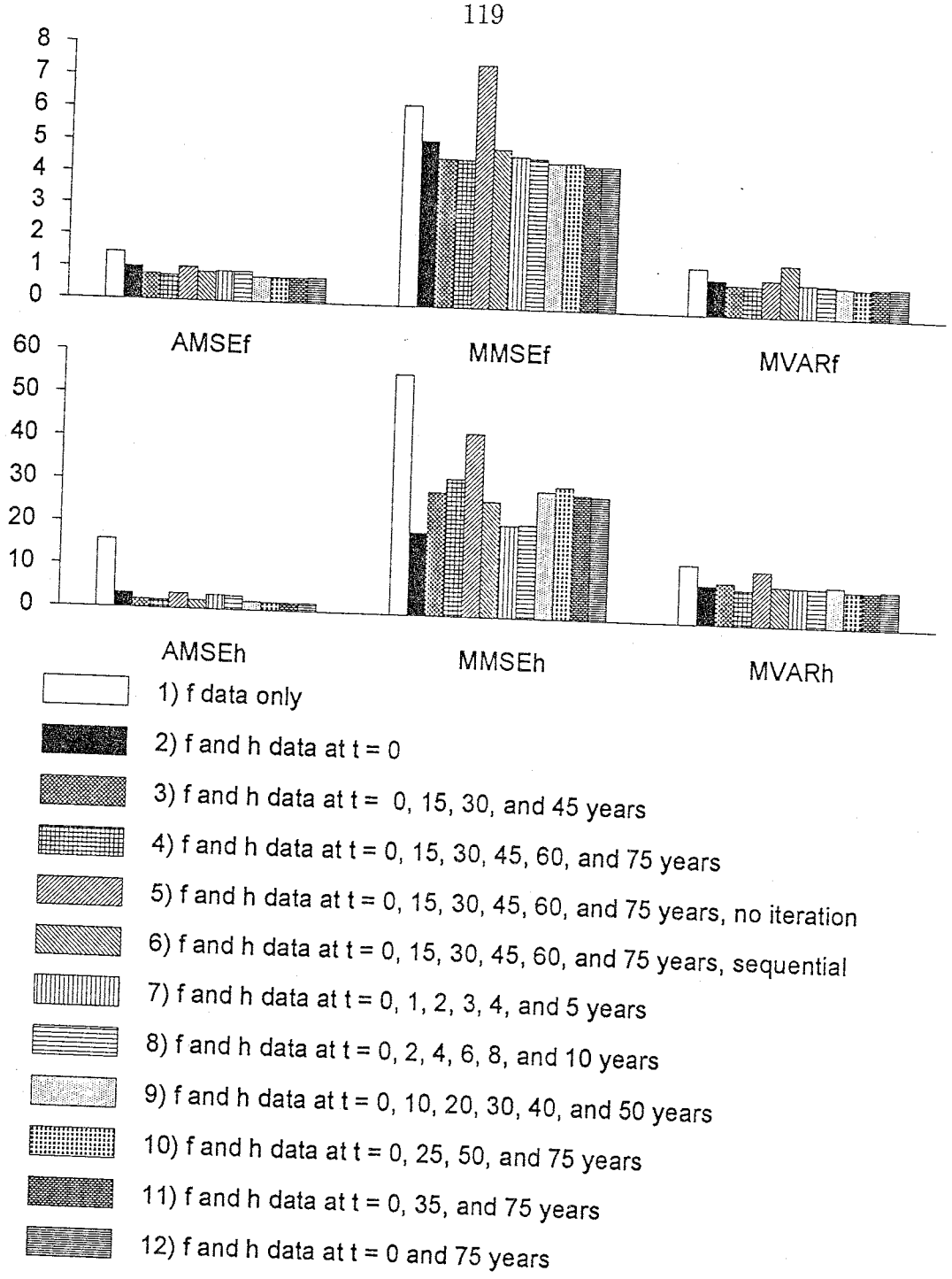


Figure 4.42: Graphical comparison of the performance measures given in Table 4.3 for simulations using 12 different sets of conditioning data.

		f fields			head fields		
ref.	M	AMSE	MMSE	MVAR	AMSE	MMSE	MVAR
1	100	1.47	6.28	1.47	15.86	56.0	13.74
2	99	1.02	5.20	1.11	3.38	19.32	9.13
3	100	.798	4.67	.960	2.05	28.94	9.79
4	105	.761	4.65	.943	1.88	32.17	8.34
5	100	1.03	7.57	1.14	3.51	42.68	12.74
6	94	.886	5.0	1.63	1.96	27.12	9.27
7	113	.936	4.79	1.04	3.31	21.65	9.06
8	113	.924	4.74	1.02	3.17	21.95	9.08
9	105	.773	4.60	.970	1.99	29.76	9.44
10	113	.767	4.62	.935	1.87	30.88	8.59
11	108	.769	4.54	.955	1.84	29.07	8.52
12	104	.781	4.54	.985	1.89	28.86	8.83

Table 4.3: Performance measures indicating the effect of conditioning on 12 different sets of head data with the hypothetical flow model described in section 4.2.1. M = number of simulations.

the purpose of evaluating the effect of data conditioning. The baseline comparison case was conditioned only on f data (reference *1)*). The next three sets progressively add head data, first at the initial state, then for 3 and 5 intervals of 15 years. The last of these, which has 5 intervals of head data 15 years apart, (reference *4)*) was then used to compare the effect of conditioning with and without iteration and conditioning using the sequential rather than the collateral method. The remaining six sets compare the effect of spacing the head data intervals very closely in time (with respect to the aquifer time constant of the transient state) to head data at intervals spanning more of the transient time. From these simulations some conclusions were drawn.

- The first use of head data for conditioning, which was at the initial time where the aquifer was in steady state, had the most noticeable effect on

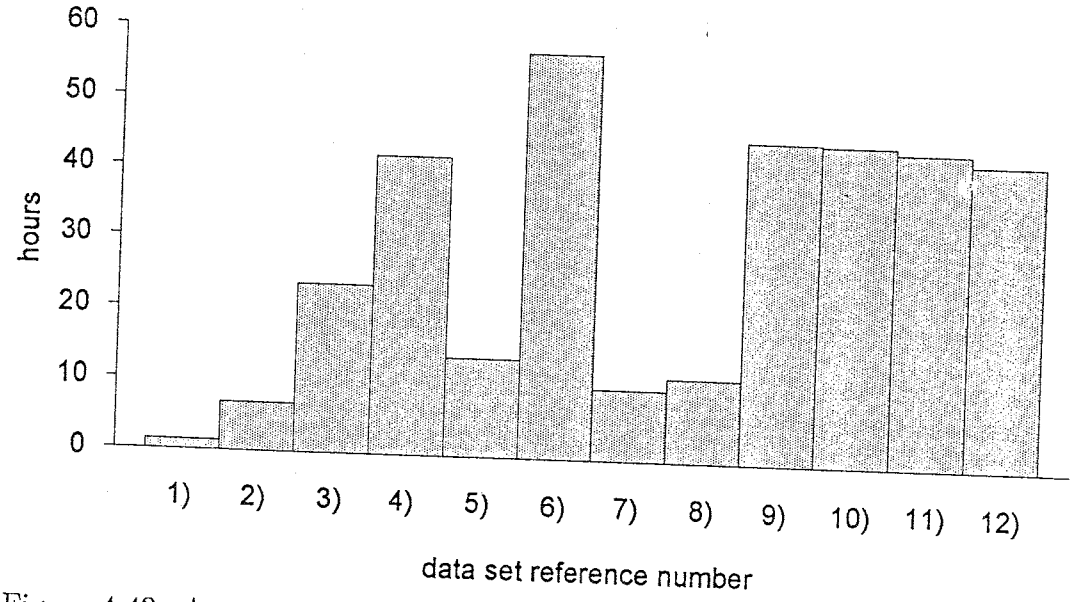


Figure 4.43: Approximate CPU time in hours for simulations conditioned on twelve different sets of data. The actual number of simulations is reported in Table 4.3

improvement in the performance measures. Using more head data caused little, if any, additional improvement and any additional improvement seen was not as dramatic in effect as this first set of head data. The cost in CPU time of conditioning on additional head data intervals, however, grew exponentially.

- Iterating to match the head data is important in maximizing the benefit obtained from this head data. Again, however, this gain in estimation improvement was made at the cost of about a factor of 3 in CPU time.

- While the sequential method has the dissatisfying consequence of modifying the f field at every head data interval, it's performance measures are not significantly worse than those for the collateral method. For few head data intervals the cross-time head covariance helps to maximize in-

formation obtained from head data but its effect was not very significant. Even though the order of the cokriging matrices incorporating cross-time head covariances is proportional to the number of head data intervals, for the data sets examined here the sequential method did not have any significant computational advantage. The large computational cost of sequential conditioning (reference *6)*) is most likely the result of iterating at each head data time interval.

- Conditioning on the same number of head data intervals spanning a greater portion of the transient time resulted in better performance measures than when the head data intervals sampled only a small part of the transient. Conditioning on a single interval spanning the transient time gave nearly the same results as 5 sample intervals over the same time. From this I conclude that the sampling time interval, with respect to the transient state and the aquifer time constant, is more important than the number of sampling intervals for maximizing the benefit obtained from conditioning on time dependent head data. Computational run-times are nearly comparable for simulations from data sets reference *9)* to *12)*. This is because most of the CPU time is consumed in solving for heads at each iteration.

- Particle trajectories are more sensitive to conditioning on head data than the other performance measures examined. Conditioning on head data has the effect of focusing particle trajectories into a narrow path. While some effect on particle travel times was observed, it was much less noticeable than the effect on the pathlines.

Chapter 5

Case Study

The usefulness of any model, stochastic inverse or otherwise, can best be determined by application to the real world. Where there are no known absolute truths, the performance of the model can be gauged in terms of what new understanding of the natural system it provides. As mentioned earlier, a stochastic framework poses parameters and system state as regionalized random variables described by a joint multivariate probability distribution. The geostatistical stochastic inverse model described herein conditions the multivariate probability distribution on data of state (i.e. head) as well as parameters (e.g. transmissivity) seeking primarily to evaluate the effect, or worth, of head data in estimating unknown parameters and in constraining model predictions. A greater problem of evaluating overall prediction uncertainty is beyond the scope of this stochastic model as the uncertainties inherent in the underlying assumptions remain unquantified. Relaxation of certain assumptions, primarily the steady state assumption, allows the modeler more freedom in choosing a conceptual groundwater flow model. Incorporating time dependent head data however, while adding some detail to the parameter estimation, illustrates the difficulty in determining an appropriate conceptual model. The choice of conceptual model has as great or a greater influence, and hence uncertainty, on model predictions as parameter spatial heterogeneity. This observation has

consequences of significance to other stochastic models as well as to the one presented here and may partially explain why practical applications of stochastic concepts are so much rarer than theoretical developments.

In this chapter the geostatistical inverse method for conditioning randomly heterogeneous transmissivity fields on transient head data is applied to real data from the Kirtland Air Force Base located in Central New Mexico. Kirtland Air Force Base is the site of landfills which have leaked solvents, and other toxic substances, contaminating an aquifer which is the main water supply for the City of Albuquerque. Pumping from the City's water supply wells has caused a transient condition to develop in the aquifer as evidenced by declining water tables over the region [*Kernodle et al.*, 1995]. Thus transport and fate of the contaminant plumes at nearby Kirtland Air Force Base is of concern. In applying this geostatistical inverse model to prediction of contaminant transport at the Kirtland Air Force Base site I hope to detect any effect the transient might have on plume advection and evaluate the influence that conditioning on head data may have on the prediction uncertainty. The first part of this chapter is devoted to analyzing head and transmissivity data from the site and developing a conceptual model of groundwater flow. Next the model is used to condition simulations on head data created from a hypothetical reality. Results from conditioning on hypothetical data can be compared directly to a known reality and thus form a basis for interpreting the results obtained from conditioning on the real data. While conditioning on head data through time during transient flow is seen to constrain the direction of advective transport, conditioning on more than a single head data interval does not seem to provide any additional prediction improvement.

5.1 Kirtland Air Force Base, New Mexico

Existence anywhere in the world of an aquifer which satisfies all of the assumptions of this geostatistical inverse model is extremely unlikely. The location and conditions along the boundaries of the aquifer would have to be known with certainty, flow would have to be essentially horizontal with no point sources or sinks, and the storage coefficient known and homogeneous. Applying the model to a complicated aquifer system rife with uncertainty may in fact provide more educational return on expended effort than searching for that aquifer which most closely resembles the model's assumptions. With this philosophy the model is applied to the aquifer system underlying the Kirtland Air Force Base in Central New Mexico.

The Kirtland Air Force Base, referred to hereafter as KAFB, lies on the eastern edge of the Albuquerque Basin, a prominent structural feature of the central Rio Grande rift. The site is bisected by the north to northwesterly trending westerly dipping Tijeras and Sandia faults which bound the rift graben on the easterly margin. Immediately to the north and west of KAFB is the City of Albuquerque, a population center of approximately a half million people. The City of Albuquerque withdraws on the order of 120,000 acre-feet of groundwater per year, with slightly over half coming from aquifer storage depletion [*Kernodle et al.*, 1995]. Some of the City's municipal supply wells are located along the northerly boundary of the site and KAFB also has its own on-site supply wells. The easterly portion of KAFB extends into the foothills and canyons of the northern Manzano Mountains. Approximately 5 miles west of the site lies the course of the Rio Grande River and adjacent to the southern

boundary is the Isleta Pueblo. Shown in Figure 5.1 is the regional tectonic setting of Kirtland Air Force Base and Sandia National Laboratories (SNL) taken from the Sandia National Laboratories Site-Wide Hydrogeologic Characterization Project 1993 Annual Report [*McCord et al.*, 1993].

5.2 Preliminary data analysis

Upon request, Sandia National Laboratories presented me with a data set consisting of water level measurements at 95 locations and 35 transmissivities obtained from pumping tests. The complete data set is included in Appendix B. Of the 95 head measurement locations, 40 are piezometers with water levels recorded on a monthly basis. The remaining 55 are simply a location and water level with no associated time at which the measurement was made. The earliest recorded water level measurement was taken in January, 1986, at a piezometer labeled CWL-BW-2, where CWL is an acronym for Chemical Waste Landfill. The time series ends in September, 1994. Figure 5.2 shows the locations of water level measurements as + symbols, overlain with an **x** symbol where measurements are available through time, and transmissivity measurements locations as small circles. The dashed lines indicate the piezometric surface obtained by contouring the 95 water level measurements by ordinary kriging with a linear variogram model using measurements made in March, 1992, where a time series of measurements was available at a piezometer, and single water level measurements at unknown times otherwise. The axes (Figure 5.2) are in New Mexico State Plane Coordinates units of feet. The domain modeled in this study is bounded by solid lines connecting the points labeled c1, c2, c3, and c4.

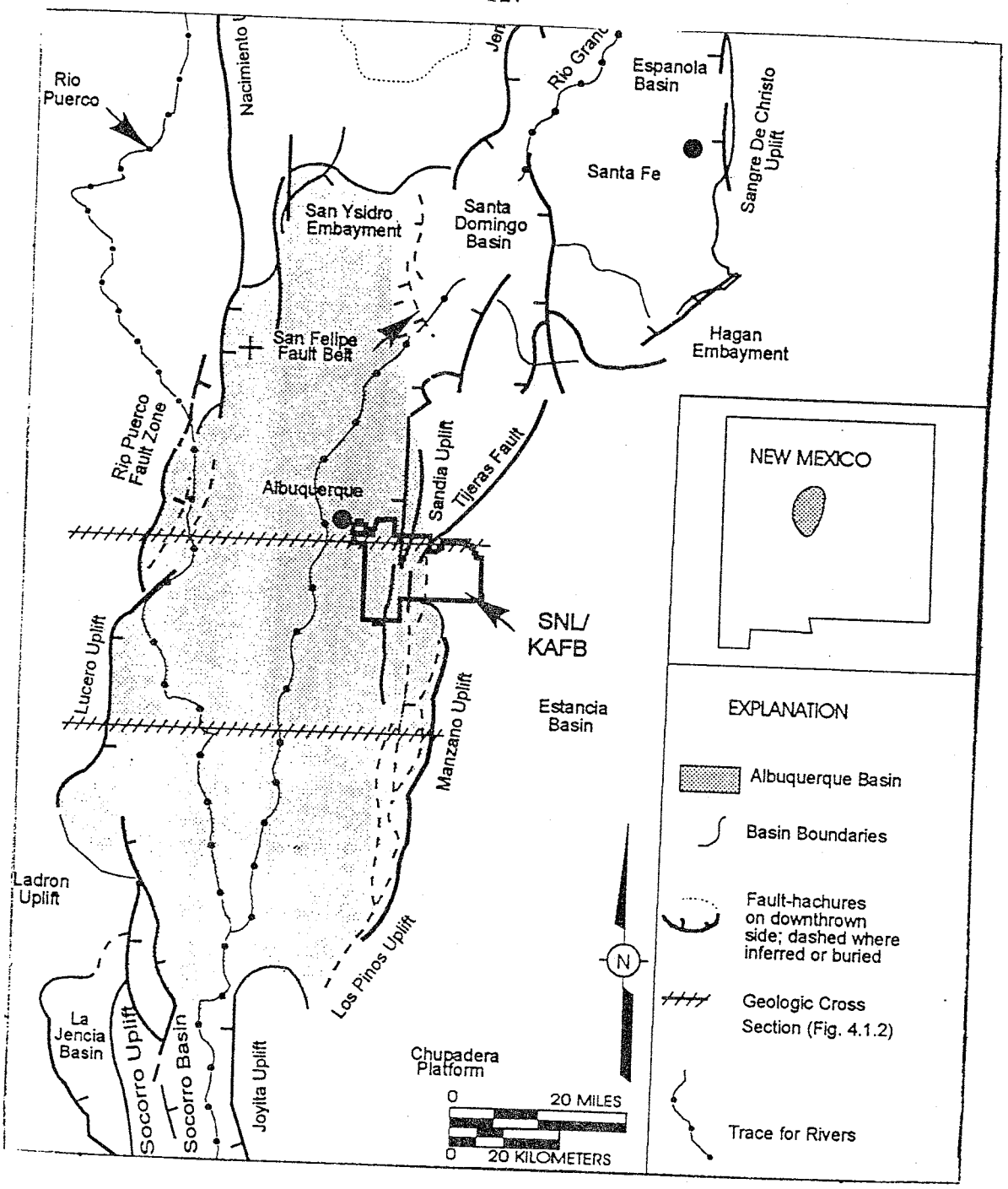


Figure 5.1: Regional tectonic setting of the Kirtland Air Force Base site in central New Mexico (plate taken from the Sandia National Laboratories Sitewide Hydrogeologic Characterization Project 1993 Annual Report [*McCord et al.*, 1993]).

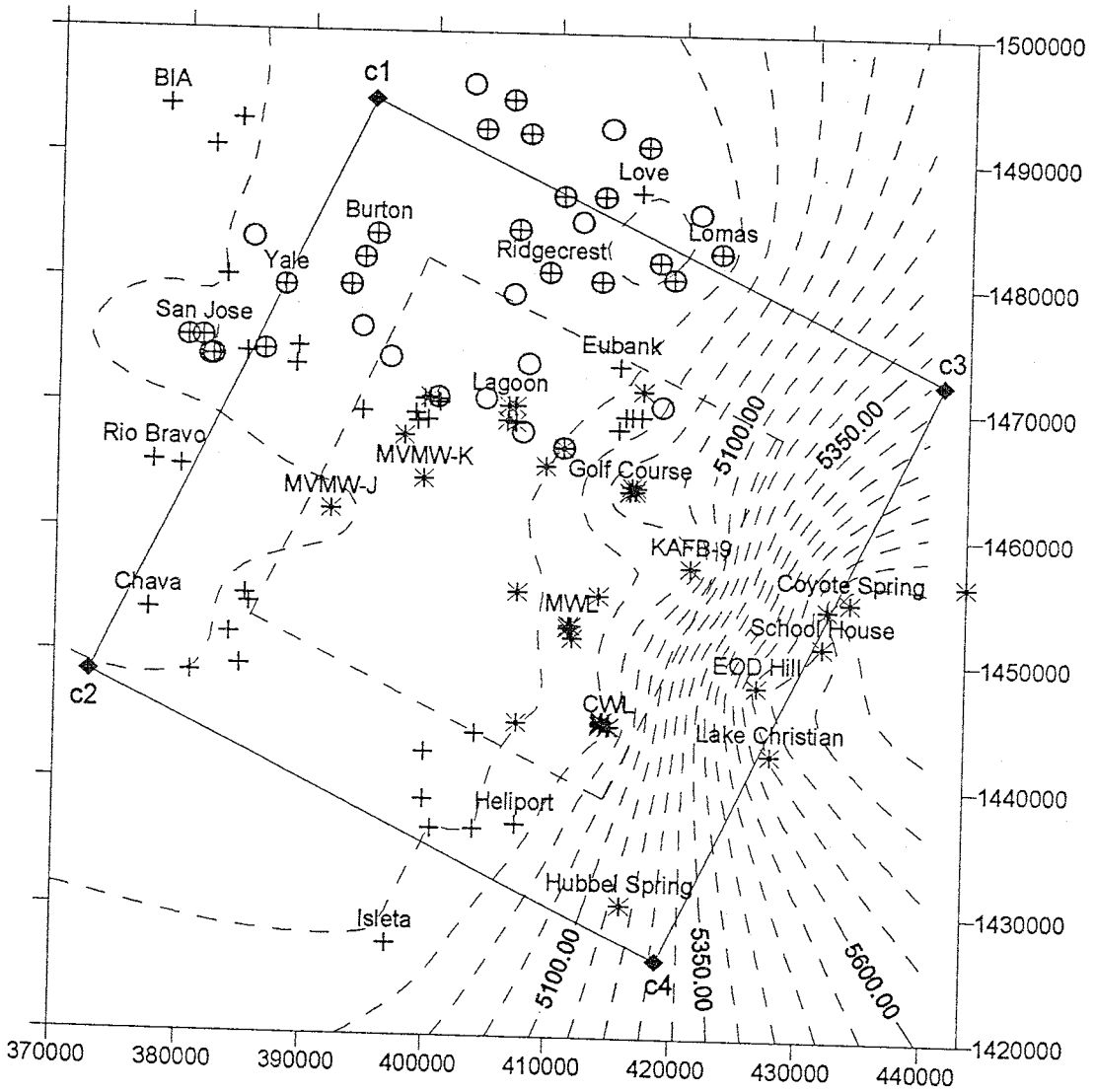


Figure 5.2: Location map of 95 head measurements shown with + symbols, overlain with x symbols where measurements are available through time, and 35 transmissivity data shown as small circles. Dashed contour lines show water levels contoured by ordinary kriging with a linear variogram model. The solid line square with corners labeled c1, c2, c3, and c4 is the boundary of the domain modeled in this study and the dashed line smaller square is the region where the model is presumed valid. The axes are in State Plane coordinates (ft) and the vertical axis is aligned with true north.

5.2.1 Transmissivity data

Transmissivity data, reportedly all from long term pumping tests, were given originally in units of gallons/day/ft. Since water levels are reported on a monthly basis, months are a convenient time unit for this analysis. The 35 transmissivity data in units of $ft^2/month$ have a mean in the natural log transformed value of 13.3, or 597000 $ft^2/month$, and a variance in the natural log transformed value of 0.6. The sample variogram of log transmissivity data is shown in Figure 5.3. A lag interval somewhat greater than 2000 ft gives a significant number of pairs of data for each interval, despite the small size of the data set. The first two points have 21 and 31 pairs respectively and the remaining sample variogram points have 50 or greater pairs of data. The dashed line in Figure 5.3 is an exponential variogram model fitted by eye with a variance, or sill, of 0.6 and a correlation scale of 6000 ft. The solid line is the Mizell-A model also with a variance of 0.6 and a correlation parameter of 6000 ft. Based on this rather limited analysis, a Mizell-A input covariance model is chosen with a sill of 0.6, correlation scale of 6000 ft, and a nugget effect of zero. Using data from slug tests and core analysis, as well as pump tests, *McCord et al.* [1993] estimated a geometric mean hydraulic conductivity of 19 ft/day with a variance of 1.2 for the log transformed values and a correlation scale of 3000 ft. This mean hydraulic conductivity corresponds to an aquifer transmissivity of approximately 87,000 $ft^2/month$ assuming an aquifer thickness of 150 ft. Scaling effects caused by larger sample support volumes probably result in the smaller variance and longer correlation scale estimated here using 35 pump test transmissivity data.

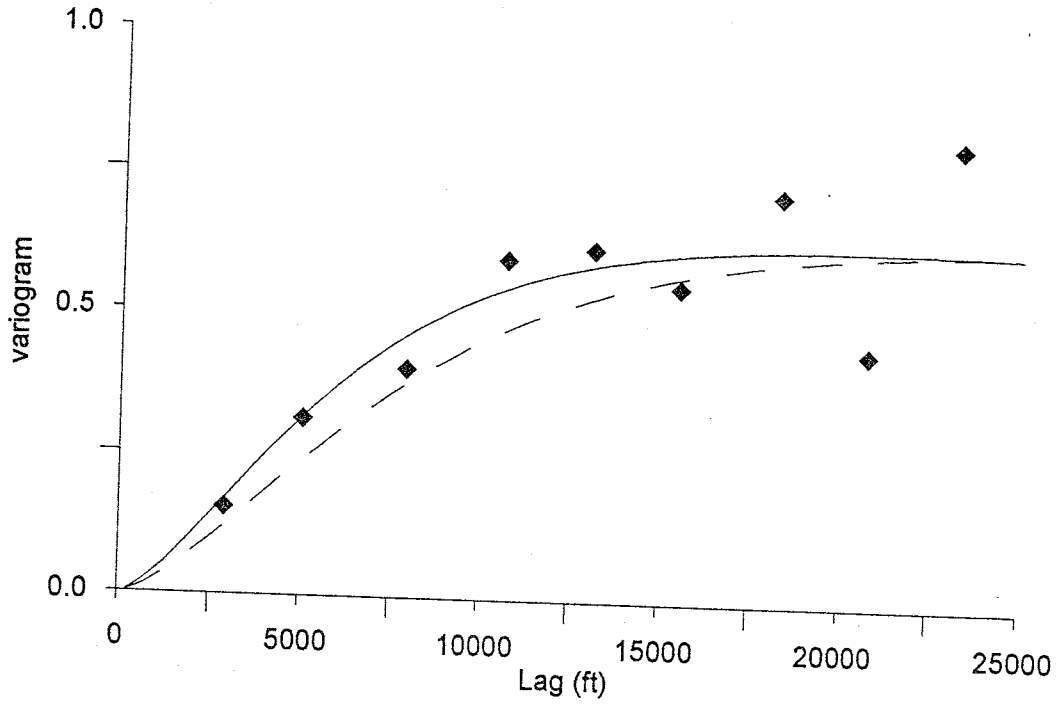


Figure 5.3: Sample variogram estimated from 35 transmissivity data. Dashed line is an exponential model and solid line is the Mizell-A model with a variance (sill) of 0.6 and a correlation scale (range) of 6000 ft.

Additional parameters required for the tracking of particles transported by advection are aquifer thickness and porosity. A simple constant value of .2 was chosen for porosity. Well completion information given in the Sandia National Labs Sitewide Hydrogeologic Characterization Project Report [*McCord et al.*, 1993] shows considerable variability in the thickness of the gravel-packed intervals, ranging from approximately 50 ft to 190 ft. The value of saturated thickness chosen for this study was 150 ft. This thickness and the geometric mean transmissivity estimated from the pump test data give a mean hydraulic conductivity of 130 ft/day. Estimates of hydraulic conductivity in this area range from 2 ft/day to 171 ft/day [*McCord et al.*, 1993]. Estimates

based only on the pump test data from 35 wells are thus on the high end of the estimated range of conductivities and the aquifer saturated thickness estimate may also be closer to a maximum rather than a mean.

5.2.2 Head data

The stochastic method implemented in this case study is concerned primarily with transient flow conditions using head data recorded through time. For this reason the 55 wells with only one recorded water level measurement at an unknown time are used only for kriging water level contours in Figure 5.2. A very steep gradient is seen in these contours from approximately 5800 ft MSL around the Coyote Spring and School House wells (Figure 5.2) to approximately 4950 ft MSL at KAFB 9 and the CWL site. The existence of this steep gradient is postulated to be a result of low hydraulic conductivity in the Sandia and Tijeras fault zones and is indicative of nonstationary transmissivity. Also, water levels recorded on the east side (right side of Figure 5.2) of the fault zone do not change significantly over the recorded time period. For instance, water level at the Coyote Spring well was recorded at 5845.06 ft MSL in November, 1990, and 5844.96 ft MSL in November, 1993. Presence of the Sandia and Tijeras faults effectively separate the domain into two hydrogeologic zones. Of the 40 wells at which head data are recorded through time, 7 lie in a region east of the faults and will be neglected for the purpose of this study. The remaining 33 wells at which water levels are recorded through time are all inside the dashed line square area shown in Figure 5.2.

A significantly anomalous piezometric surface exists in the area around the Golf Course in Figure 5.2. The 4 Golf Course monitoring wells have heads

of approximately 5040 ft MSL in January, 1991, increasing to approximately 5050 ft MSL by December, 1993. This is about 90 ft more head than the next highest level of approximately 4960 found at the CWL site. *McCord et al.* [1993] devoted a section of the Sitewide Hydrogeologic Characterization Project Report to this anomaly. They postulated the existence of perched water tables recharged from the Tijeras Arroyo and artificial impoundments at the Golf Course. The model developed here is exclusively for saturated two-dimensional flow and cannot cope with the added complexity of perched aquifers and/or vertical flow. Therefore the groundwater mound in the Golf Course vicinity is neglected and the four Golf Course monitoring wells are excluded from the data set. Another monitoring well, labeled KAFB-8, was omitted from the data set since there were only 5 measurements over a short period and one of these has a value approximately 1000 ft less than the other 4.

Data from the remaining 28 monitoring wells are plotted versus time in Figure 5.4. There are 4 wells in the Lagoon group, 4 wells in the MWL group, and 9 wells in the CWL group plus CWL-BW-1 which shows the highest water level. A change in water level from about 4958 ft at CWL-BW-1 to approximately 4938 ft at CWL-BW-2 occurs over a horizontal distance of about 100 ft. The remaining 8 wells at the CWL site have only a few feet difference in head even though they are located several hundred feet east of CWL-BW-2. This steep gradient between CWL-BW-1 and CWL-BW-2 may be due to one of the fault zones or due to different well completion depths tapping different aquifers. Even if the additional information needed to determine the source of this head difference were available, this two-dimensional flow model could not deal with the extra complexity. Since observations from monitoring well CWL-

BW-1 cannot be incorporated into the limitations of the model assumptions, this data is also neglected.

5.3 Model grid and boundary locations

The region for which there are head data measured through time and where the model assumptions are not seriously violated is shown in Figure 5.2 as a square with the dashed line border. Within this region there are 7 transmissivity measurements and 27 monitoring wells which are deemed to reasonably satisfy the model assumptions. Recall the effect of deterministic boundary conditions from Figure 4.9 in section 4.1.2. A datum located very close to a boundary will be forced by conditioning to match that boundary value, which may be different than the datum value. To avoid undue influence of the boundaries over conditioning on data, the model boundaries are located approximately 2 correlation lengths away from the closest datum. Model boundaries are shown in Figure 5.2 as solid lines with corners labeled c1, c2, c3, and c4. The square domain of dimension 51200 ft by 51200 ft is then divided into 4096 equally sized blocks of dimension 800 ft by 800 ft to produce a finite difference grid of 64 by 64. A block size of dimension 800 ft by 800 ft is presumed to not be significantly larger than the sample support volume of the pumping tests so that the scaling effect introduced by the grid discretization is presumed minimal. This, of course, is an assumption since the actual sample support volume of the transmissivity data is unknown.

Wells in the CWL, MWL, and Lagoon groups are clustered such that wells in each group essentially occupy the same grid block. Wells in each group also show nearly identical behavior through time with differences at

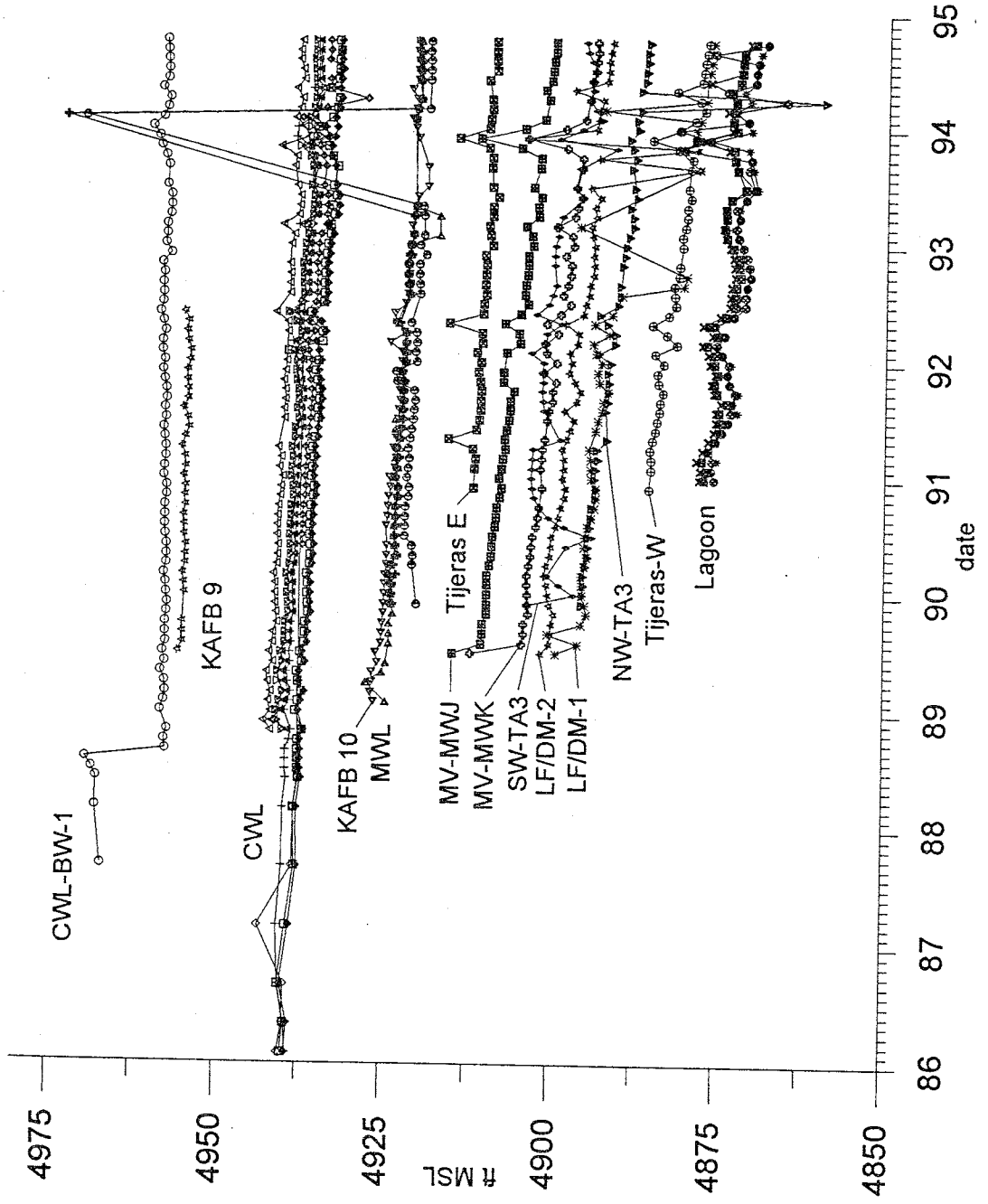


Figure 5.4: Water level measurements in 28 wells are plotted as feet above Mean Sea Level (MSL) from January, 1986 to September 1994. There are 9 wells in the CWL group, 4 wells in the MWL group, and 4 wells in the Lagoon group.

most of a few feet. For the purpose of simplifying the analysis, monitoring well CWL-BW-2 was chosen as representative of the CWL group. Data from well CWL-BW-2 is marked in Figure 5.4 with + symbols. Monitoring well MWL-MW-1, shown in Figure 5.4 with right pointing triangles, was chosen to represent the MWL group and Lagoon SE, indicated as small squares, was chosen as the Lagoon group representative. The final 12 head data locations and 7 transmissivity locations are plotted on the discretized model domain shown in Figure 5.5. The origin at the lower left corner of the square region in Figure 5.5 corresponds to the corner labeled c2 in Figure 5.2. A formula for converting from the New Mexico State Plane coordinates of Figure 5.2 to the model coordinates in Figure 5.5 is given as

$$\begin{aligned} x' &= r\,cos\left(sin^{-1}\left(\frac{y-y_0}{r}\right)+\theta\right) \\ y' &= r\,cos\left(\frac{\pi}{2}-\theta-sin^{-1}\left(\frac{y-y_0}{r}\right)\right) \\ r &= \sqrt{(x-x_0)^2+(y-y_0)^2} \end{aligned} \tag{5.1}$$

where x' and y' are model domain coordinates, x and y are New Mexico State Plane coordinates, $x_0 = 385553.69$ and $y_0 = 1452989.39$ is the location of the lower left corner of the model in state plane coordinates, and $\theta = .4497$ radians is the rotation angle of the model domain from state plane coordinate axes.

5.4 Boundary conditions

Boundaries for this model were located so that they are at least 2 correlation lengths away from data locations. Locating boundaries farther away would involve including more features which violate model assumptions, such as point sources and sinks and nonstationarity. Also, a larger modeled region

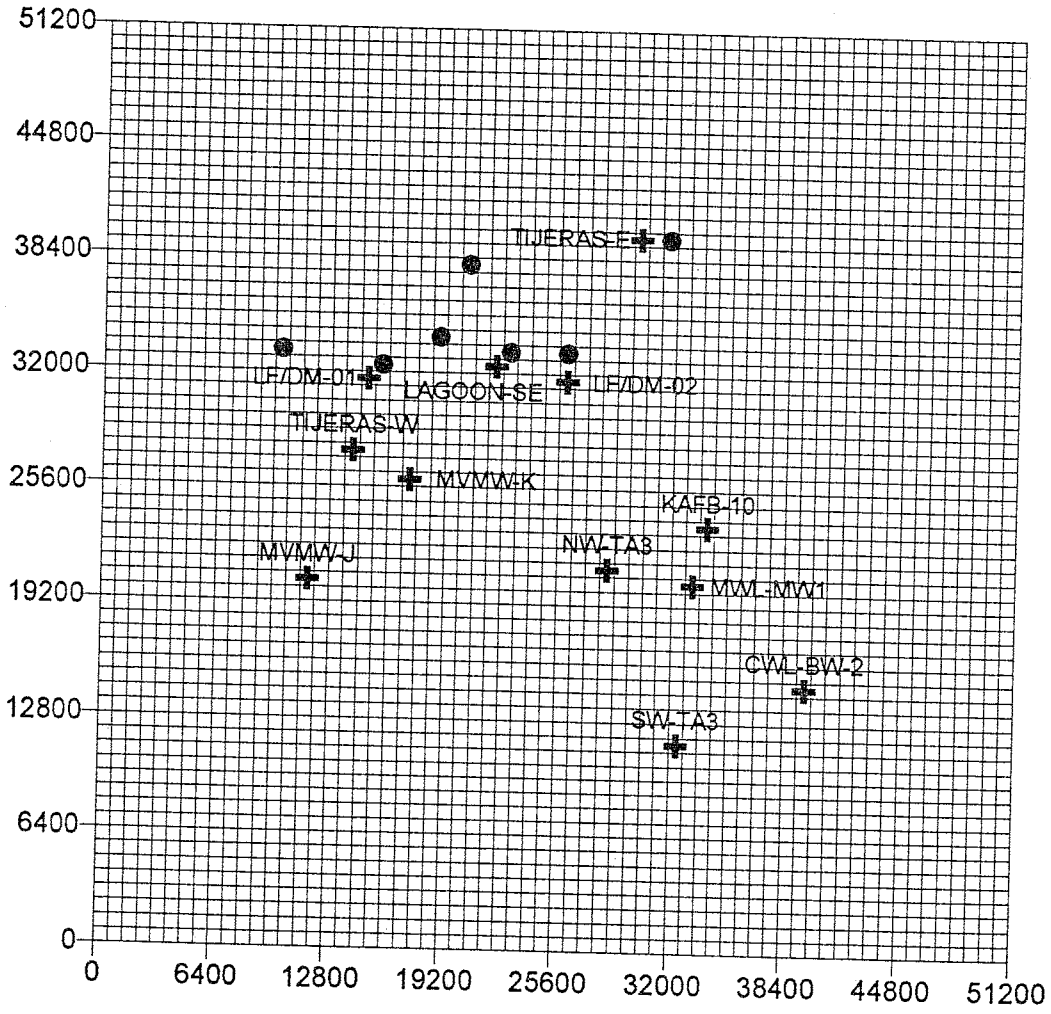


Figure 5.5: Discretized model domain with head data locations shown as + symbols and transmissivity data locations shown as filled circles. Axis units are in feet and the vertical axis is bearing 25.8 degrees from true north.

would necessitate more grid blocks, which would add computational burden, or larger grid blocks which may produce some scaling effect on the transmissivity covariance structure. Boundaries thus located obviously have no relationship to any natural hydraulic boundaries–they merely mark the extent of the modeled region. Hydrologic conditions occurring in the greater Albuquerque basin are communicated to this small modeled portion of the KAFB site through these mathematical boundaries.

One approach to determining the stresses imposed on the small modeled region by conditions in the larger basin is by nesting models. In the nested approach the smaller region is incorporated as a subdomain in a model of the entire basin and the relationship between the basin and smaller region is determined by solving the larger model on a coarser scale. Of course, boundary conditions for the basin-scale model must still be determined and sufficient data must be collected in order to adequately model the entire basin. Determining covariance and cross-covariance relationships between the smaller region and basin-scale models across the boundaries of the smaller region, however, involves additional complications. Nested or hierarchical stochastic inverse models may become the subject of future research efforts but are beyond the scope of the present method.

For this case study, boundary conditions were estimated using a nonlinear least-squares optimization on head observations all located inside the dashed line square region depicted in Figure 5.2. Recall that the only piezometers outside of this dashed line region at which time series measurements are available are east of the Tijeras and Sandia fault zones. As explained be-

low, optimization was performed using a free version of the software package PEST [*Doherty et al.*, 1994], a flexible implementation of the Gauss-Marquardt-Levenberg algorithm [*Marquardt*, 1963]. The boundaries were assumed to be first-type, or prescribed head.

5.4.1 Quasi-steady state assumption

Response of the square modeled region, depicted in Figure 5.2 by solid lines, to basin-scale influences is governed by the region's dimensions, L = 51200 ft, transmissivity, $T_G = 597000 ft^2/month$, and storage coefficient. *Kernodle et al.* [1995] estimated a specific storage coefficient for the aquifer matrix in this area of $2 \times 10^{-6}/ft$ and a specific yield of between .1 and .25. *McCord et al.* [1993] presumed the aquifer underlying the KAFB site is phreatic but then found artesian conditions in a few boreholes. There may be some evidence of a fairly rapid aquifer response, on the order of a few months, in the head data shown in Figure 5.4. Towards the end of 1993 there appears to have been a positive perturbation in water levels over most of the region. High water levels are seen to decay away in 2 or 3 months. The response is seen clearly in wells MV-MWJ, MV-MWK and Tijeras-W and as far away as CWL-BW-1. A response time of 3 months for this region would correspond to a storativity of approximately .002 and imply that the region is essentially in equilibrium with conditions on the boundaries.

5.4.2 Estimated boundary conditions

Prescribed head boundary conditions for the model were estimated by optimizing the 4 boundary head values at each of the corners labeled c1,

c2, c3, and c4 in Figure 5.2, assuming steady state equilibrium flow conditions, no recharge, and a uniform transmissivity of 597000 $ft^2/month$. Boundary values for nodes between corners were obtained by straight-line interpolation. The software package PEST adjusted head values at locations c1, c2, c3, and c4 until the sum of squared differences between head data at the 12 locations shown in Figure 5.5 and head calculated by a steady state flow model was minimized. Water level data at the 12 locations shown in Figure 5.5 were collected at monthly intervals but wells came on-line at different times and data is missing for many of the monthly intervals. Water level data exists for all 12 wells for the months of January, February, March, July, August, September, October, and November in 1991, January, February, March, April, July, and August in 1992, and January, March, May, June, July, August, and September in 1994. January, 1991, is taken as the initial condition t=0 and thus the monthly head data are recorded at times 0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 38, 40, 41, 42, 43, and 44 months with reference to this initial time. The minimum sum of squared residuals from the optimization routine was between 1591 ft^2 and 1845 ft^2 at all times except for time t=38, where the sum of squared residuals was 3483 ft^2, due to an anomalously low measurement at well NW-TA3. A plot of the head values estimated by this optimization procedure for each of the corners is shown in Figure 5.6. An equation of the form

$$y = a + bt^c \tag{5.2}$$

was then fit to the 21 points in each plot, again by nonlinear least-squares. The model resulting from this exercise represents head at the corners of the square domain as a function of time with a linear interpolation of first type boundaries

between these corners. Equations of the form in (5.2) were fitted to points representing a 44 month period then extrapolated beyond the data an additional 44 months to April, 1998. Power function, equation (5.2), parameters a, b, and c for the 4 corners are recorded in Table 5.1 and the functions are graphed in Figure 5.6.

corner	a	b	c
c1	4819.91	-.07716	1.63857
c2	4927.56	7.5939E-5	3.0
c3	4957.04	.06688	1.57425
c4	4918.50	-.09922	1.37253

Table 5.1: Parameters a, b, and c of equation (5.2) representing boundary heads as a function of time for the 4 corners c1, c2, c3, and c4 of the modeled region depicted in Figure 5.2

5.5 Summary of the KAFB site conceptual model

A conceptual model has been developed for the KAFB site based upon assumptions and limitations inherent in this stochastic model, a limited amount of head and transmissivity data, and previous investigations [*McCord et al.*, 1993; *Kernodle et al.*, 1995]. Analysis of 35 transmissivity data from pumping tests gave a mean transmissivity of $T_G = 597000 ft^2/month$ and a covariance structure with a variance in the log transformed values of 0.6 with a correlation length of 6000 ft. A Mizell-A functional form for the covariance structure is assumed. Model boundaries were located rather arbitrarily to be approximately 2 correlation lengths distant from data locations and assumed to be first type. Though model boundaries were extended a distance away from data to minimize their influence, the entire domain encompassed by these

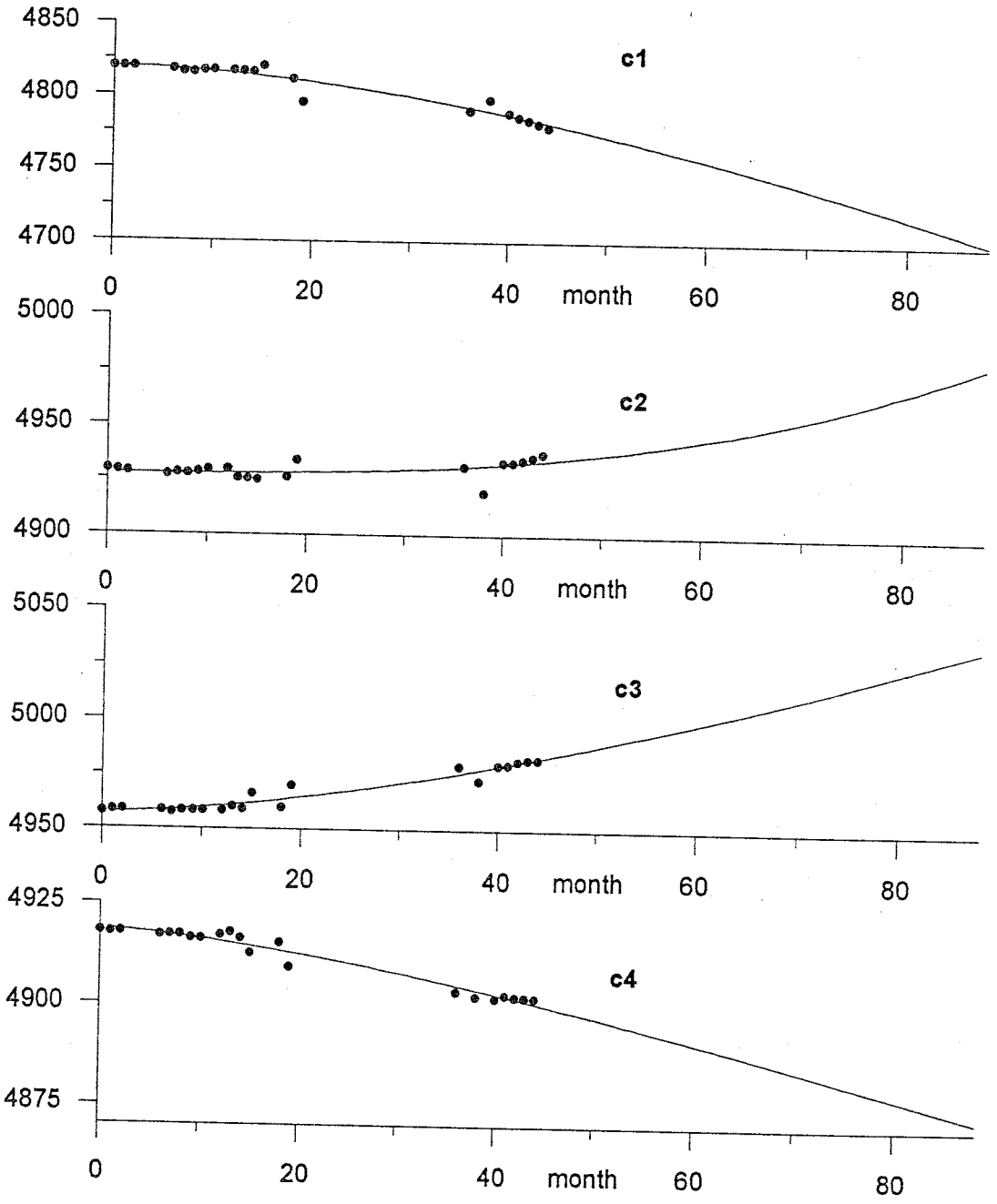


Figure 5.6: Boundary heads at the four corners of the modeled region, c1, c2, c3, and c4, obtained by nonlinear least squares optimization for 21 times fitted with a power function between 0 and 44 months and extrapolated to 88 months.

boundaries obviously does not satisfy some requirements of the model including statistical stationarity and absence of point sources and sinks. The area for which the model is presumed to be valid is delineated by the dashed line square shown in Figure 5.2. Conditions on the boundaries were determined through nonlinear least squares optimization by assuming a quasi-steady state. A functional form for the behavior of the corners of the modeled region through time is shown in Figure 5.6 with linear interpolation between corners along the sides of the modeled region. The square bounded region was discretized into a 64 x 64 grid of uniformly sized blocks with dimensions 800 by 800 ft. For the purpose of modeling transient flow conditions inside this bounded region, a storativity value of .05 is assumed. This storativity value, one which represents neither a truly phreatic nor confined aquifer, was chosen for two reasons. First the aquifer under investigation is reportedly phreatic but is also artesian at some piezometers [*McCord et al.*, 1993] and, secondly, it results in an aquifer time constant on the order of a year and a half in keeping with the quasi-equilibrium assumption stated above. Covariance calculations for this transient state use an explicitly stable time step of .1 months. For simulations, where no explicit time stepping is performed, the time step was increased to 1.0 month in order to decrease computational run-times. Simulations using a time step of 1.0 month were compared to simulations using a time step of .1 month which produced nearly identical results. Simulations were performed for a period of 88 months from January, 1991 to April, 1998. Advective velocity particle tracking assumes a saturated thickness of 150 ft and a porosity of 0.2. Particles were released from the locations of the CWL (Chemical Waste Landfill) and MWL (Mixed Waste Landfill) at time t=0.

5.6 Simulation Results

Monte Carlo simulations were performed using the conceptual model summarized in section 5.5 and various time combinations of head data using both the sequential and collateral approaches to conditioning on data. There is no way of knowing how well or how poorly this conceptual model represents the actual groundwater flow regime at the KAFB site nor is there any known true field against which to compare the results as there is with a hypothetical model. Thus comparison between results using sequential state conditioning or collateral conditioning, or head data at several times versus head data at only a few times is only qualitative. Quantitative comparisons would be possible if the truth were known. For this purpose, another suite of simulations was performed for a hypothetical transmissivity field with an identical conceptual model so that effects due to conditioning on various times of head data by the two different conditioning methods could be quantified. A transmissivity field was generated on a 64 x 64 block grid and designated as the one true field (Figure 5.7). This transmissivity field was sampled at the locations shown as filled circles in Figure 5.5 then combined with the conceptual model developed for the KAFB site to produce a transient head field. This head field was sampled at the locations shown with + symbols in Figure 5.5 at the times of 0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 38, 40, 41, 42, 43, and 44 months to create a hypothetical head data set. Each set of Monte Carlo simulations using the true KAFB data was also done using the hypothetical data. Comparing the results of simulations using the true KAFB data against simulation results using the hypothetical data helps to separate the effects of varying conditioning data and conditioning methods from effects due to an inadequate conceptual

model.

Simulations for both the true KAFB data and the hypothetical data were conditioned on 4 different time combinations by both the collateral and sequential methods. These time interval sets of head data are numbered for ease of reference as follows.

1. The first set of conditioning data used is just the f data with no head data in order to establish a baseline for evaluating the influence of head data.

2. Next, simulations are conditioned on head data at the initial time t=0 and the final time t=44 months where head data are available. Conditioning only on head data at initial and final times was seen from the hypothetical model presented earlier to provide good results with minimal computational effort, although the cause of that transient condition was different than the change in "regional gradient" dealt with here.

3. Conditioning is then done on head data at times 6, 9, 36, and 41 months in addition to head data at times t=0 and 44 months.

4. Lastly, conditioning is done on head data for all times except time t=38 months. Month 38 was omitted due to the anomalous datum recorded at observation location NW-TA3.

Simulation results are presented first for the hypothetical data from these 4 subsets, where each realization of the random f field is the only unknown. Results obtained from the KAFB data can then be seen in light of what happens with the hypothetical data.

5.6.1 Conditioning using hypothetical data

The simulated f field designated as hypothetical reality and head at time t=88 months, given the KAFB conceptual model, are shown in Figure 5.7. Paths of a particle released from the locations of the CWL and MWL sites are plotted on the head field in Figure 5.7b and indicated with an arrow as they are rather hard to see. These fields and particle paths represent the truth against which the simulations are compared. Performance measures used to compare simulations to the true fields are the average MSE ($AMSE$), obtained by averaging the MSE from equation 4.1 over all grid blocks in the field, and the maximum MSE ($MMSE$). Particle tracks from simulations are compared to the true paths (Figure 5.7b) by calculating the distance, r, between a simulated particle at t=88 months and the location of the true particle at t=88 months. Pathlines from conditioned simulations having the smallest of the average μ_r, standard deviation σ_r, and maximum max_r of these distances most closely resemble the true particle path. As conditioning on head data at more times appears to produce larger variation in the simulated f fields, the extrema of the ensemble averaged f fields from the simulations, the maximum and minimum mean f are examined. These measures are the maximum value ($Maxf$) and minimum value ($Minf$) that occur in the mean of the conditioned f field simulations. Maximum ensemble variance from the conditioned f and head fields are also reported.

Performance measures comparing the results of simulations conditioned on the hypothetical data for various times and collateral versus sequential methods are summarized in Table 5.2. Statistics of the distance between

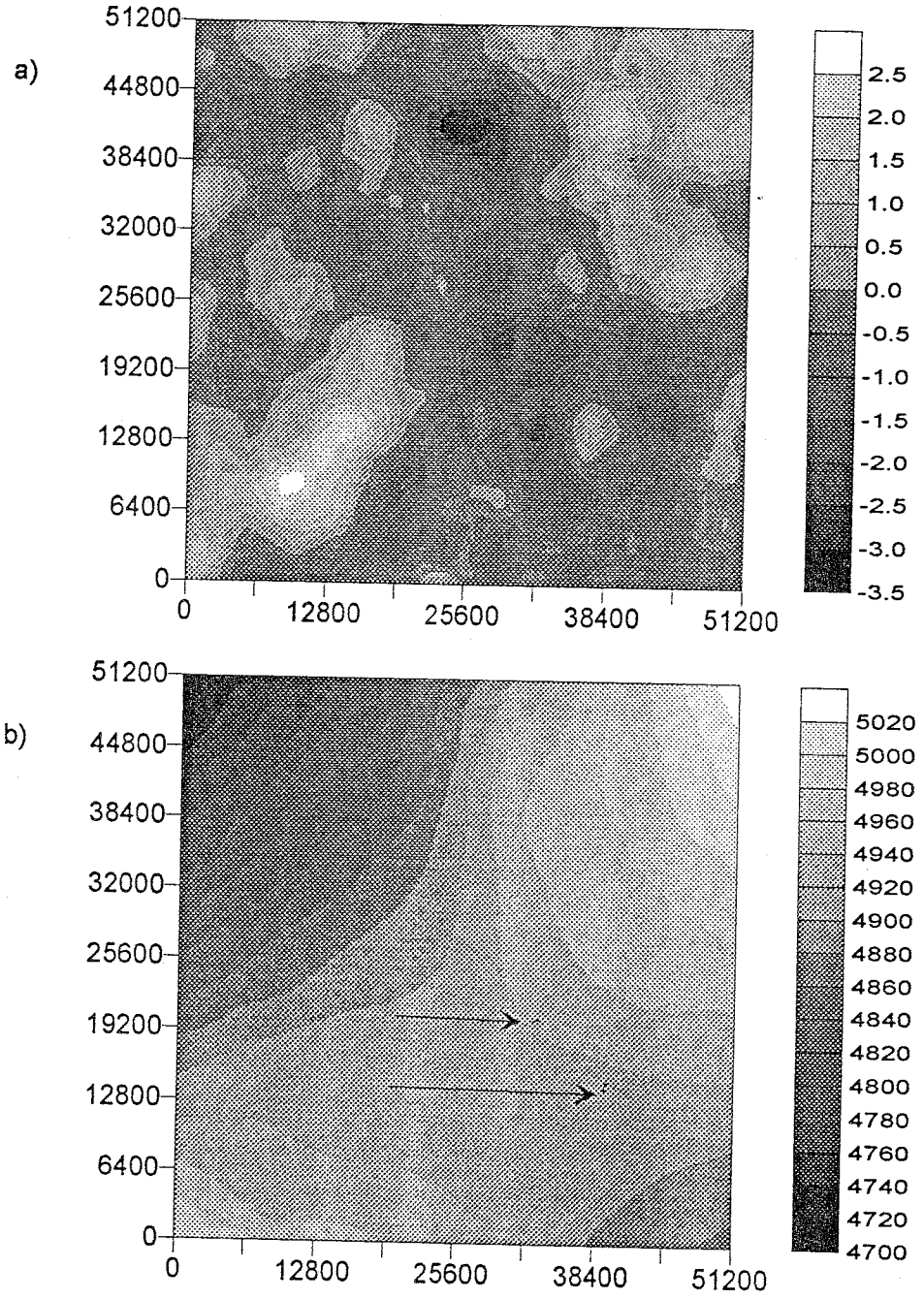


Figure 5.7: a) Simulated f field designated as the hypothetical reality. b) Head resulting from the hypothetical reality and the KAFB conceptual model at t=88 months. Two small lines in the lower right quadrant (indicated with arrows) are particle paths released from the CWL and MWL sites.

particles tracked in the simulations and the true particle location at t=88 months are summarized in Table 5.3. I'm concerned with comparing three major aspects of the conditioning technique. First of all there is the effect of conditioning on more head data through time. Is this an effective and useful means of gaining information about the co-conditional probability distribution of head and transmissivity and detail of spatial heterogeneity? Secondly there is the issue of using the cross-covariances of head in time. Certainly incorporating these cross-time covariances is more costly, but does it actually improve the estimation significantly? Finally there is the question of iteration. Iteration was seen to be important in maximizing the benefit obtained from head data in chapter 4. These questions are addressed by comparing the different conditioning techniques, which are labeled:

C for collateral conditioning,

S for sequential conditioning, and

I for iteration

using the different data sets numbered 1-4 above.

Iteration did not always converge reliably to the prescribed criterion, absolute maximum difference between head data and conditioned simulation (AME), as was the case with the hypothetical model presented in chapter 4. Whether this was due to the nature of the transient, i.e. uniform step change in recharge or changing boundary conditions with time, or if it may be an effect of the irregularly spaced data, is unknown. What I did was to adjust this AME criterion by trial and error so that the majority of simulations converged.

The convergence criterion (AME) for each simulation employing the iteration technique is given in Table 5.3.

	f fields					h fields		
set	AMSE	MMSE	Minf	Maxf	MVAR	AMSE	MMSE	MVAR
1	1.34	9.14	-1.11	.333	.849	111	1103	162
2CI	1.08	6.65	-2.63	1.19	1.25	50.7	258	160
2C	1.12	6.58	-1.17	.59	.871	69.2	412	188
2SI	1.13	6.38	-2.07	1.11	.986	41.6	265	136
2S	1.20	8.38	-.725	.365	.855	93.9	611	163
3CI	2.45	29.2	-4.57	2.40	22.1	93.8	757	259
3C	1.23	6.63	-2.06	.996	1.66	88.9	754	232
3SI	1.09	6.73	-1.56	.693	.865	51.7	253	161
3S	1.18	8.16	-.771	.374	.853	81.8	557	173
4CI	2.08	22.2	-3.93	2.17	4.05	119	1275	253
4C	2.01	21.4	-3.89	2.12	3.65	124	1292	312
4SI	1.10	6.88	-1.51	.684	.863	53.4	251	157
4S	1.18	8.09	-.772	.370	.852	80.9	558	176

Table 5.2: Summary of performance measures for simulations conditioned on a hypothetical data set. Measures are labeled as AMSE for Average Mean Squared Error, MMSE for Maximum Mean Squared Error, Minf for minimum mean f, Maxf for maximum mean f, and MVAR for maximum simulated variance.

The performance measure numbers presented in Tables 5.2 and 5.3 are plotted in a series of bar graphs so as to make direct comparisons between the methods and data sets easier. In Figures 5.8 and 5.9 the performance measures for collateral conditioning on data sets 1-4 are plotted. Performance measures from collateral conditioning without iteration are shown in Figures 5.8 and 5.9 as white bars and collateral conditioning with iteration as filled bars. The most noticeable improvement obtained with this method is from the first addition of head data at times 0 and 44 months (reference set *1*) but even this

	CWL site			MWL site				
set	μ_r	σ_r	max_r	μ_r	σ_r	max_r	M	AME
1	987	734	4700	1699	1066	5841	100	
2CI	908	705	3357	2191	1651	8457	110	3.0
2C	916	652	3316	1928	1289	6871	100	∞
2SI	1425	1090	5861	2357	1699	8434	110	3.0
2S	899	667	3515	1774	1141	5294	100	∞
3CI	881	708	3208	2004	1637	8864	97	8.0
3C	946	668	3349	2044	1407	7525	100	∞
3SI	1081	821	4230	2068	1448	7745	110	8.0
3S	813	596	2930	1749	1141	5212	100	∞
4CI	1020	773	4097	2024	1466	7700	129	20.0
4C	1078	784	4097	2140	1480	7700	100	∞
4SI	1000	761	3949	1998	1408	7448	110	8.0
4S	781	558	2693	1723	1136	5162	100	∞

Table 5.3: Summary of statistics of the distance between each simulated particle at 88 months and the true particle at 88 months. The symbol μ_r references the average distance, σ_r the standard deviation, max_r the maximum distance between a simulated particle and the true particle location, M is the number of simulations, and AME is the iteration convergence criterion. Units are feet.

is equivocal. While the MMSE and AMSE for h show a marked improvement going from set *1* to set *2* (Figure 5.8), and the maximum r shows a slight improvement, (Figure 5.9) nearly all of the measures for collateral conditioning without iteration show a steady decline in performance after a slight improvement from *1* to *2*. This suggests that the collateral conditioning method operates best with a single head data time interval. While some measures, particularly the AMSE and MMSE for head fields, are improved through iteration most measures show little, if any, improvement and a few, such as the AMSE, MMSE, MVAR, and Maxf for the f fields get dramatically worse.

Performance measures for the sequential conditioning method, Fig-

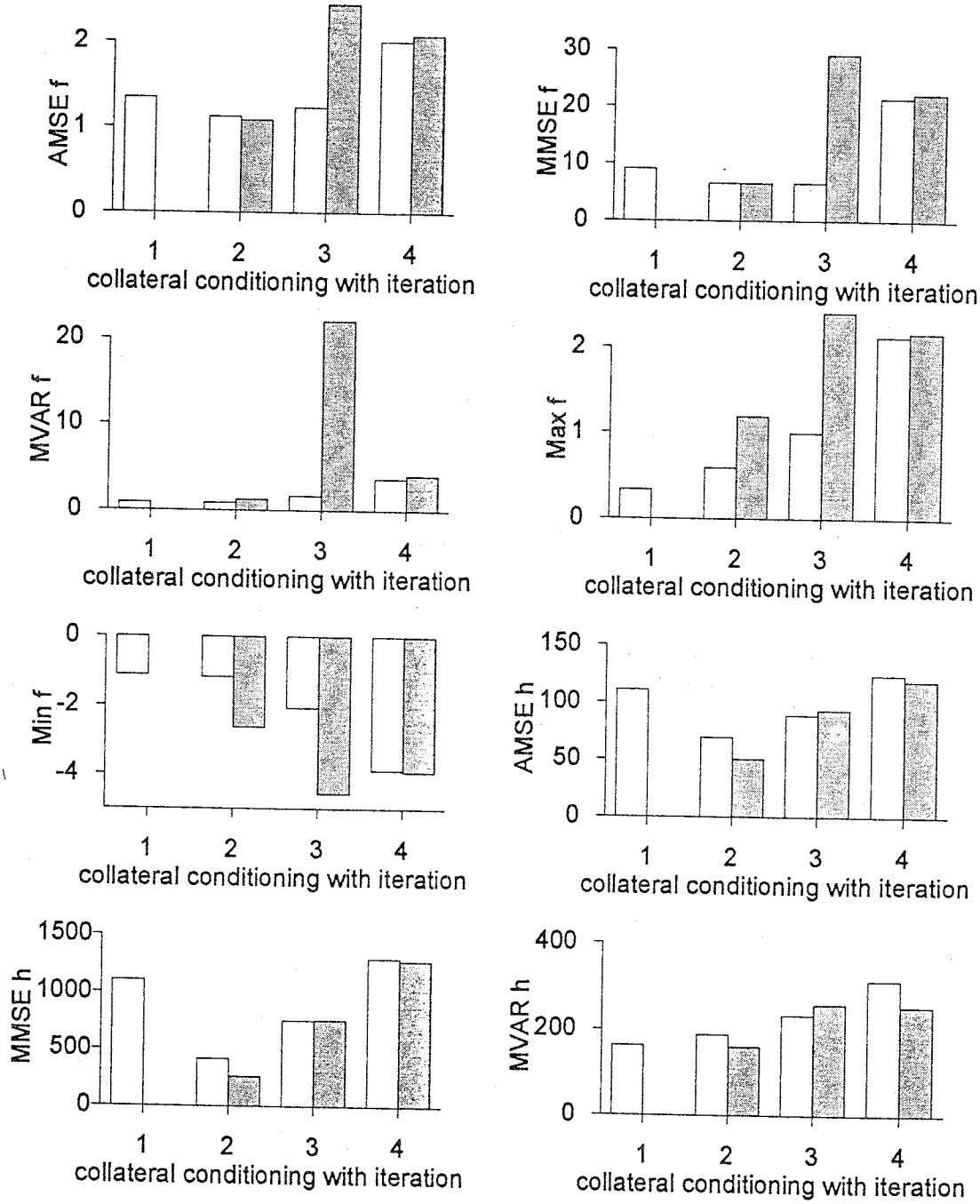


Figure 5.8: Comparison of performance measures AMSE, MMSE, MVAR, Maxf, and Minf for collateral conditioning. Measures from conditioning with iteration are shown as filled bars and from conditioning without iteration as white bars. Numbers 1, 2, 3, and 4 refer to the data sets described above.

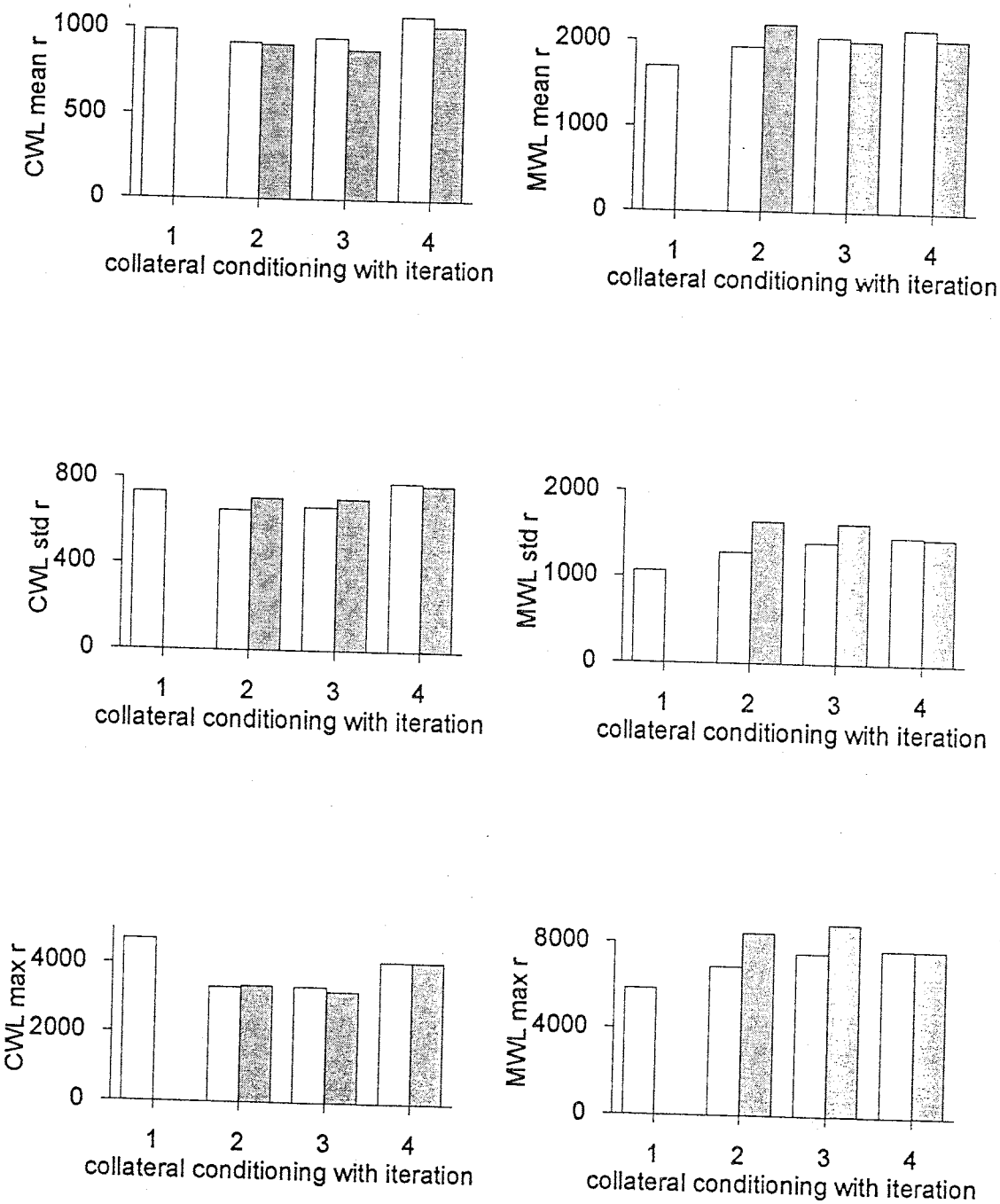


Figure 5.9: Comparison of performance measures μ_r, σ_r, and max_r for particles released from the CWL and MWL sites obtained using collateral conditioning. Measures from conditioning with iteration are shown as filled bars and from conditioning without iteration as white bars. Numbers 1, 2, 3, and 4 refer to the data sets described above.

ures 5.10 and 5.11, show either a steady, albeit slight, improvement with the addition of more time intervals of head data or no change at all. The most noticeable improvements occur in AMSE and MMSE for head fields and in μ_r, σ_r, and max_r for particles released from the CWL site. However the improvements seen going from 1 to 5 to 19 head data time intervals diminish quite rapidly. This suggests that conditioning on more than the single head data interval is not contributing any new information to the estimation.

Iteration using the sequential conditioning method appears to have contradictory results. On the one hand iterating results in consistent improvements in the AMSE, MMSE, and MVAR measures for both f and h fields for all sets of head data (Figure 5.10). On the other hand it produces consistently worse statistics for particle locations and the Maxf and Minf in the conditioned mean f field (Figure 5.11). It is possible that the poor particle location statistics at t=88 months and the extreme highs and lows of the conditioned mean f field are related. If the particle travel path were in a zone where conditioning was causing the f field to be higher, the resulting longer travel paths would account for the larger μ_r, σ_r, and max_r statistics seen in Figure 5.11 after iteration.

In Figures 5.12 and 5.13 the performance measures for collateral and sequential conditioning without iteration are compared directly. Measures from collateral conditioning are the darker bars, plotted on the left. The first dark bar at location *1* represents the performance measure obtained from conditioning of f data only. Measures from sequential conditioning have a lighter colored fill and are plotted on the right for comparison at each set, 2-4, of head

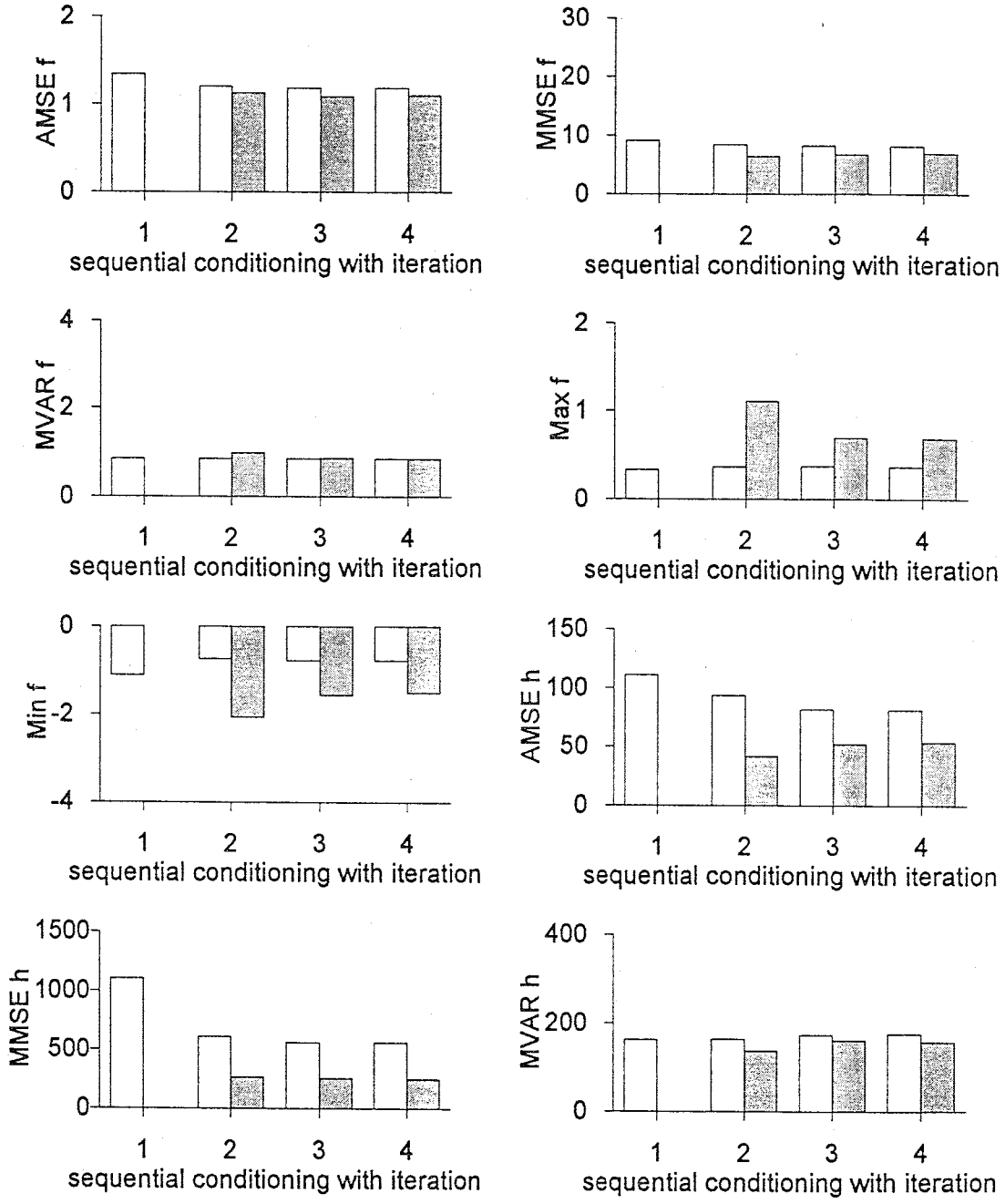


Figure 5.10: Comparison of performance measures AMSE, MMSE, MVAR, Maxf, and Minf for sequential conditioning. Measures from conditioning with iteration are shown as filled bars and from conditioning without iteration as white bars. Numbers 1, 2, 3, and 4 refer to the data sets described above.

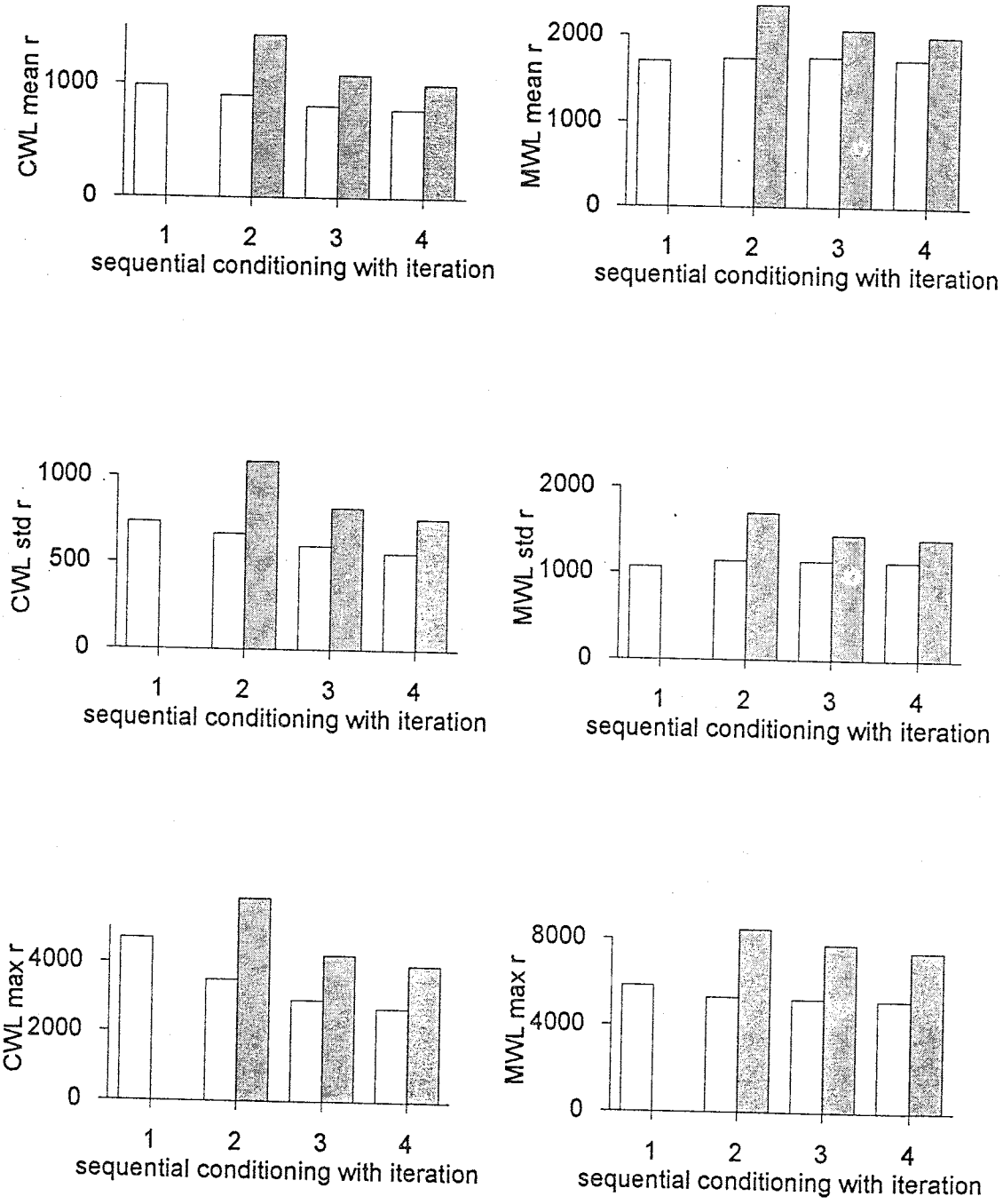


Figure 5.11: Comparison of performance measures μ_r, σ_r, and max_r for particles released from the CWL and MWL sites obtained using sequential conditioning. Measures from conditioning with iteration are shown as filled bars and from conditioning without iteration as white bars. Numbers 1, 2, 3, and 4 refer to the data sets described above.

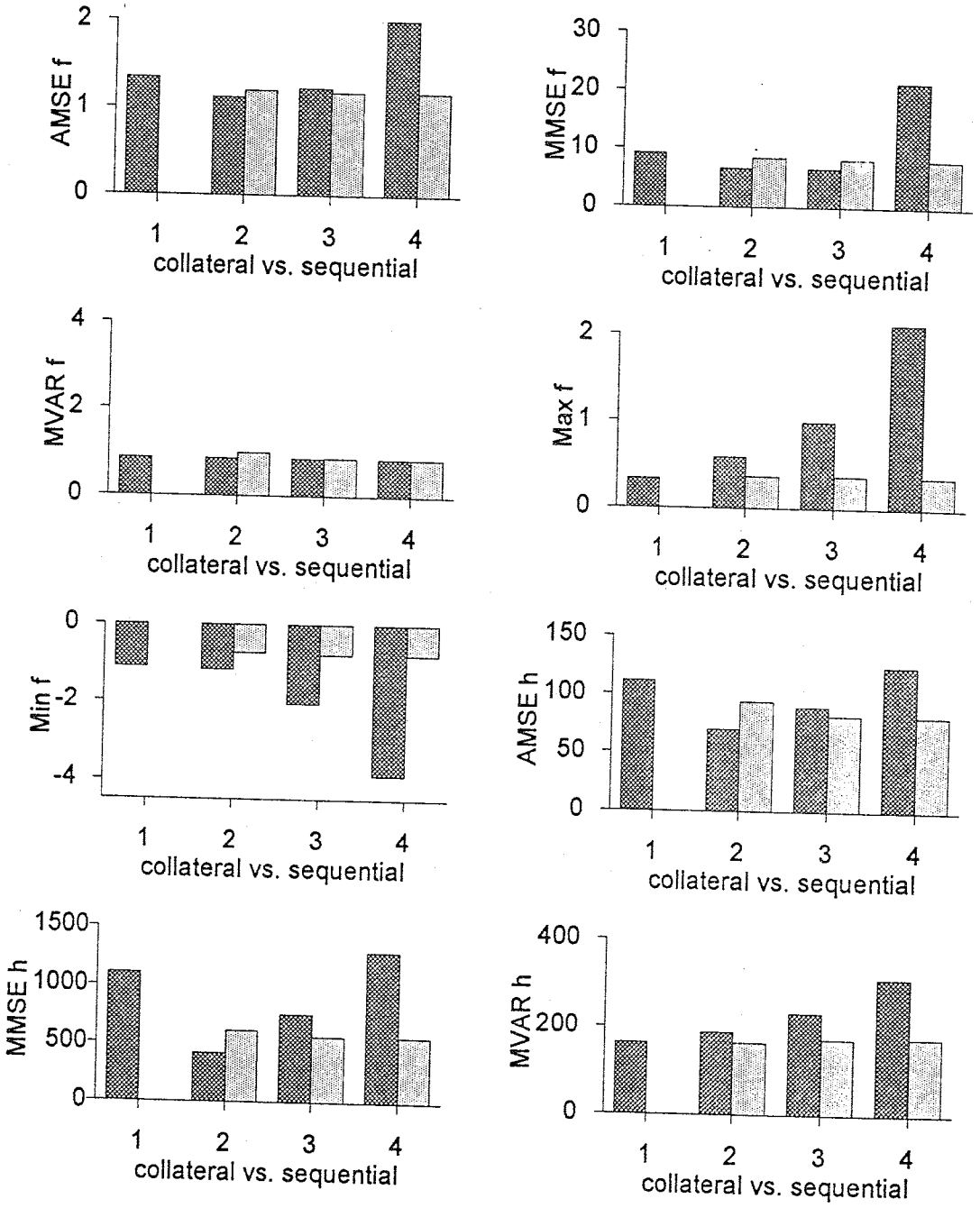


Figure 5.12: Performance measures AMSE, MMSE, MVAR, Maxf, and Minf comparing collateral to sequential conditioning without iteration. Bars with the darker fill are performance measures from collateral conditioning and bars with the lighter fill, on the right, are from sequential conditioning. Numbers 1, 2, 3, and 4 refer to the data sets described above.

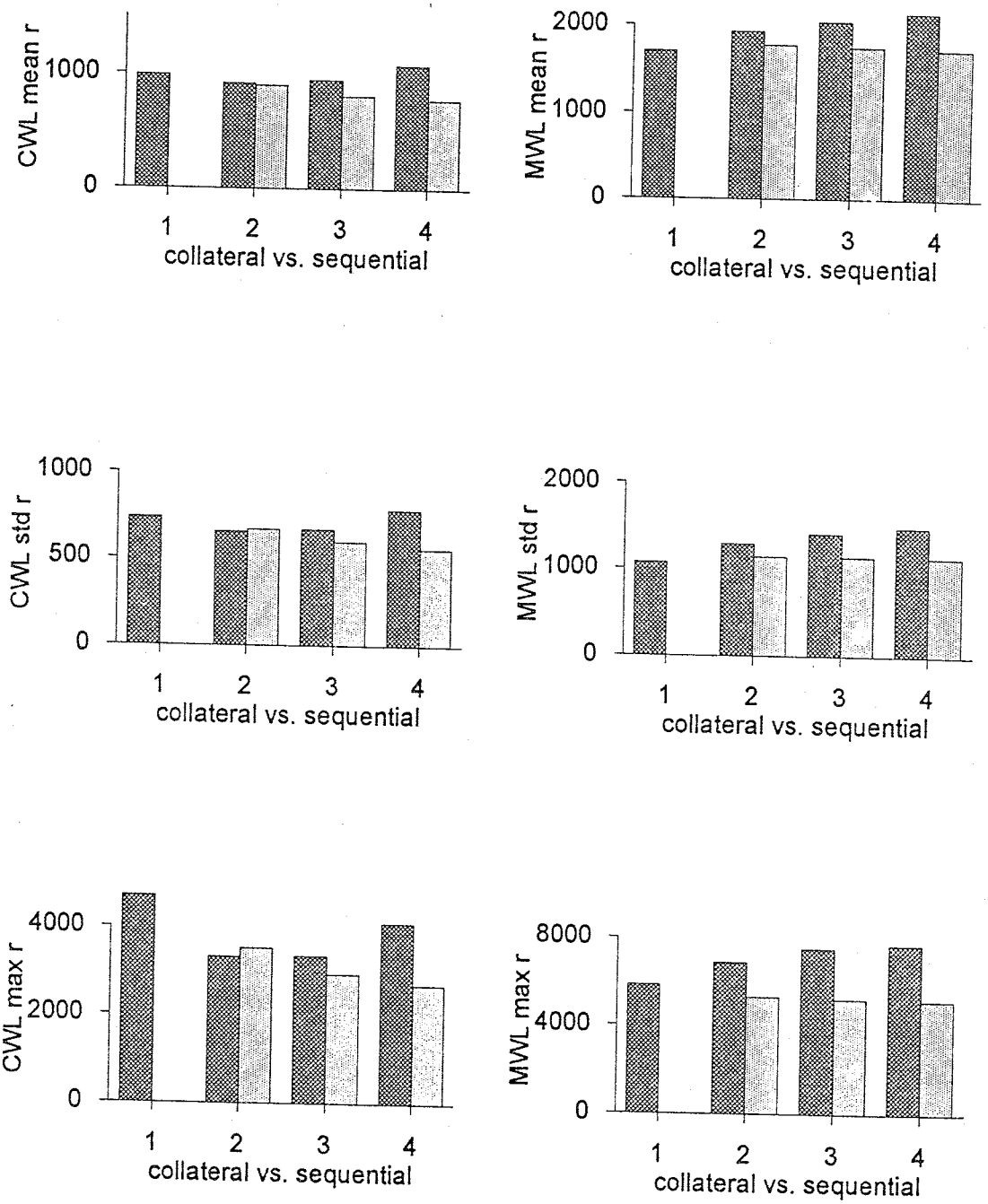


Figure 5.13: Performance measures μ_r, σ_r, and max_r for particles released from the CWL and MWL sites comparing collateral to sequential conditioning without iteration. Bars with the darker fill are performance measures from collateral conditioning and bars with the lighter fill, on the right, are from sequential conditioning. Numbers 1, 2, 3, and 4 refer to the data sets described above.

conditioning data. Consistently, the collateral conditioning shows decreasing performance as more intervals of head data are added while measures of the sequential method remain about the same, as was seen previously. It is primarily at the first set, conditioning on a single interval of head data, where the two methods are most comparable. For example, the best average fit (AMSE) of conditioned f fields to the true field occurred using collateral conditioning with iteration on set *2*. However the smallest maximum MSE (MMSE) occurred using sequential conditioning with iteration on the same data set with an AMSE nearly the same as for collateral conditioning. For head fields the situation was reversed, with sequential iterative conditioning having the smallest AMSE and collateral iterative conditioning having a smaller MMSE. Sequential conditioning with iteration had the smallest MMSE for head fields. Particle location statistics show even different results; the best from the CWL site occurring using collateral conditioning with no iteration on set *4* and the best from the MWL site may very well be set *1*, which uses no head data at all. Possibly the statistics, μ_r, σ_r, and max_r, for particle locations are not the appropriate measures since they take into account only the distance from the true particle location and not the spreading. Spreading, or uncertainty in the direction of advective transport can be qualitatively evaluated by maps of the particle paths for each simulation.

Shown in Figure 5.14 is the average head at t=88 months resulting from 100 simulations conditioned only on 7 f data. Particle tracks from these simulations at t= 88 months are an average distance $\mu_r = 987$ ft away from the true particle released from the CWL site with a standard deviation $\sigma_r = 734$ ft and maximum distance, max_r, of 4700 ft. Particles released from the MWL

site have $\mu_r = 1699$, $\sigma_r = 1066$ ft, and $max_r = 5841$ ft. While these measures are comparable to, or better, than, measures obtained from the simulations conditioned on head data, the map of particle trajectories (Figure 5.14) shows a great deal of uncertainty in the actual direction of advective transport.

The average of 110 head fields at time t=88 months conditioned collaterally with iteration on head data at the initial time t=0 and at the time t=44 months is shown in Figure 5.15. While there is still much uncertainty in the travel distance, the map of particle trajectories (Figure 5.15) reveals that conditioning on head data significantly constrains the direction of transport.

The average head at t=88 months obtained by conditioning the same 110 simulations again on head data at times t=0 and 44 months using the sequential conditioning method with iteration to an AME of less than 3.0 ft is shown in Figure 5.16. Compared with the collateral conditioning (Figure 5.15) the sequential method appears to result in slightly more uncertainty in both the direction and travel distance (note the two long paths from the CWL site in Figure 5.16).

Conditioning collaterally on head data at times 6, 9, 36, and 41 months in addition to the times of t=0 and 44 months requires more computational effort and the iteration becomes more likely to diverge. In order to obtain consistent convergence, the iteration convergence criterion was relaxed from a maximum error of 3.0 ft to a maximum error of 8.0 ft between the head data and model simulations for collateral conditioning on data set *3*. An ensemble average head field at t=88 months from 97 simulations conditioned on head data at times t=0, 6, 9, 36, 41, and 44 months (set *3*) is shown in

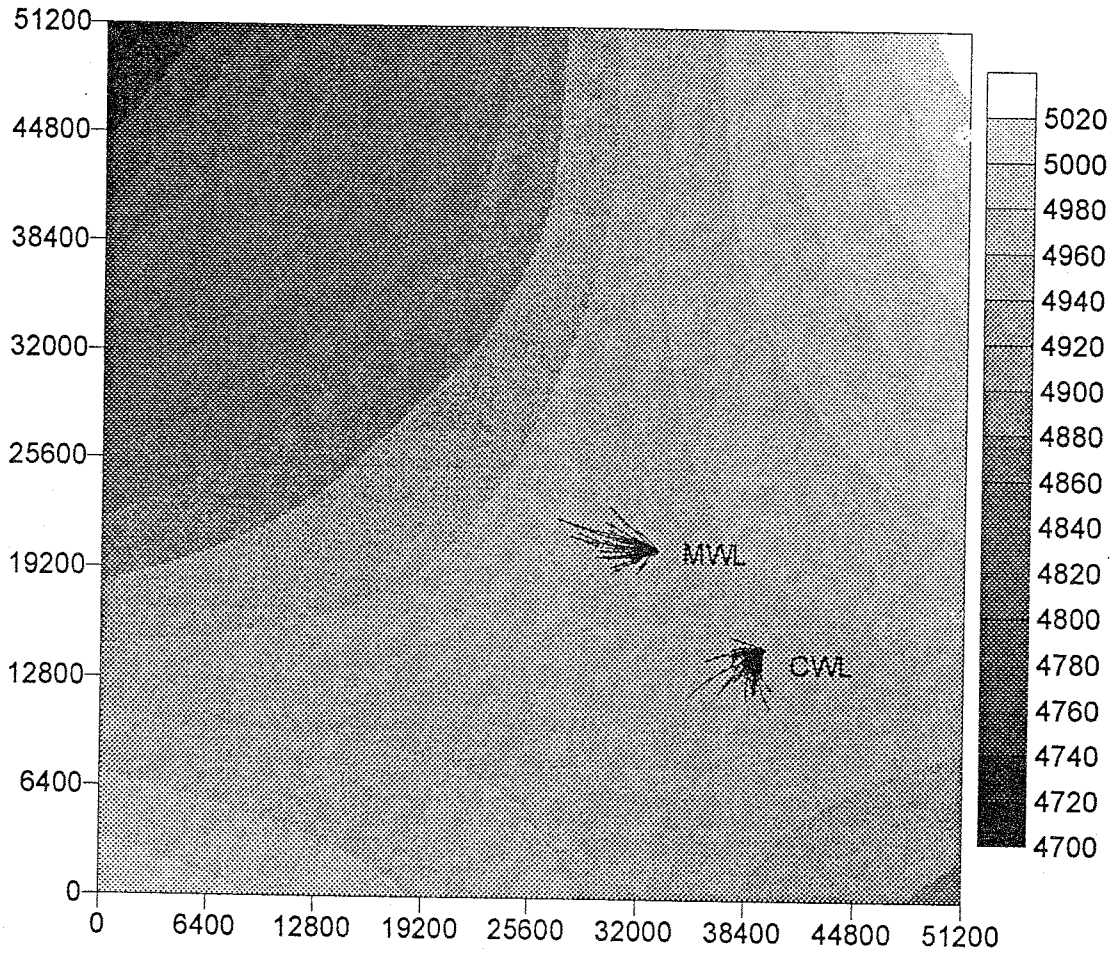


Figure 5.14: Ensemble mean head at t=88 months obtained from 100 simulations conditioned only on *f* data (set *1*). Pathlines are of particles released from the CWL and MWL sites.

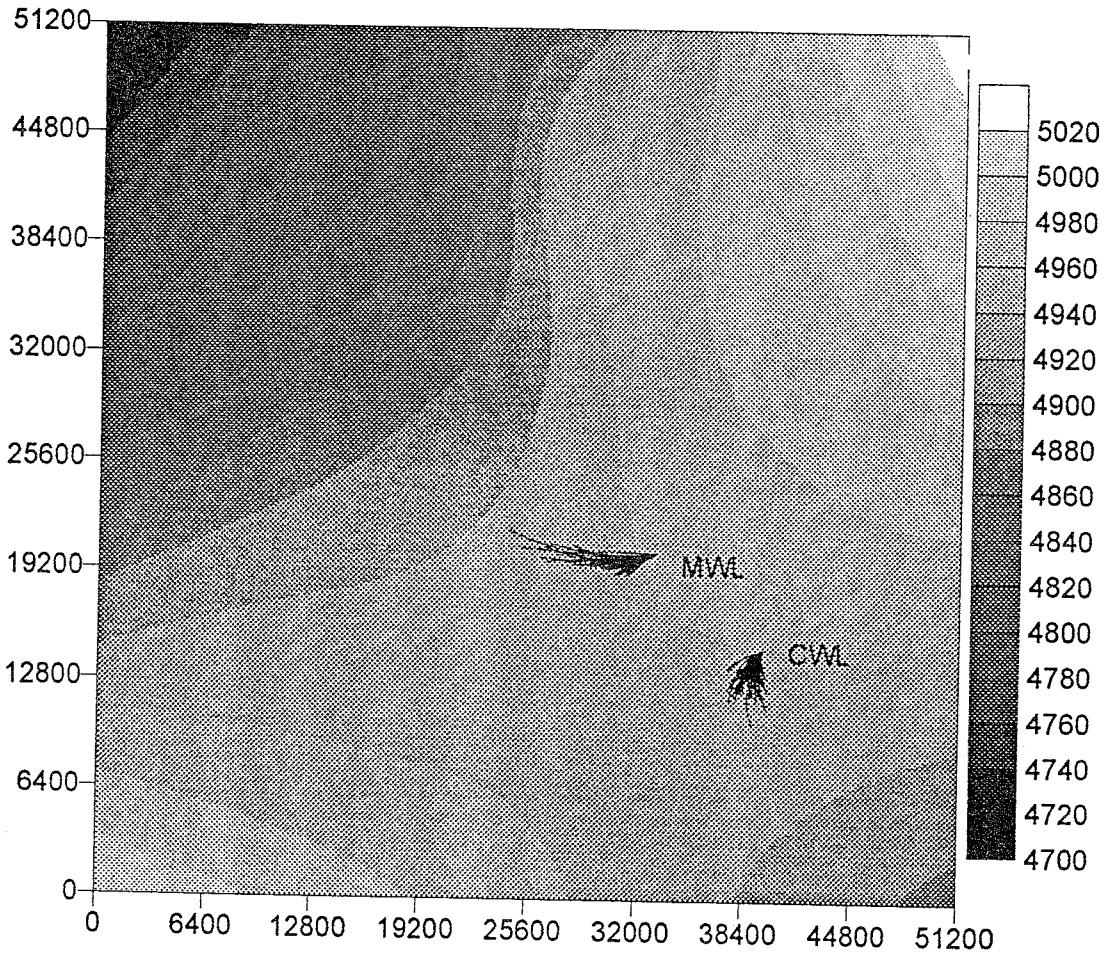


Figure 5.15: Ensemble mean head at t=88 months obtained from 110 simulations conditioned collaterally with iteration, to an AME criterion of 3.0 ft, on f data and head data at t=0 and 44 months (set *2*). Pathlines are of particles released from the CWL and MWL sites.

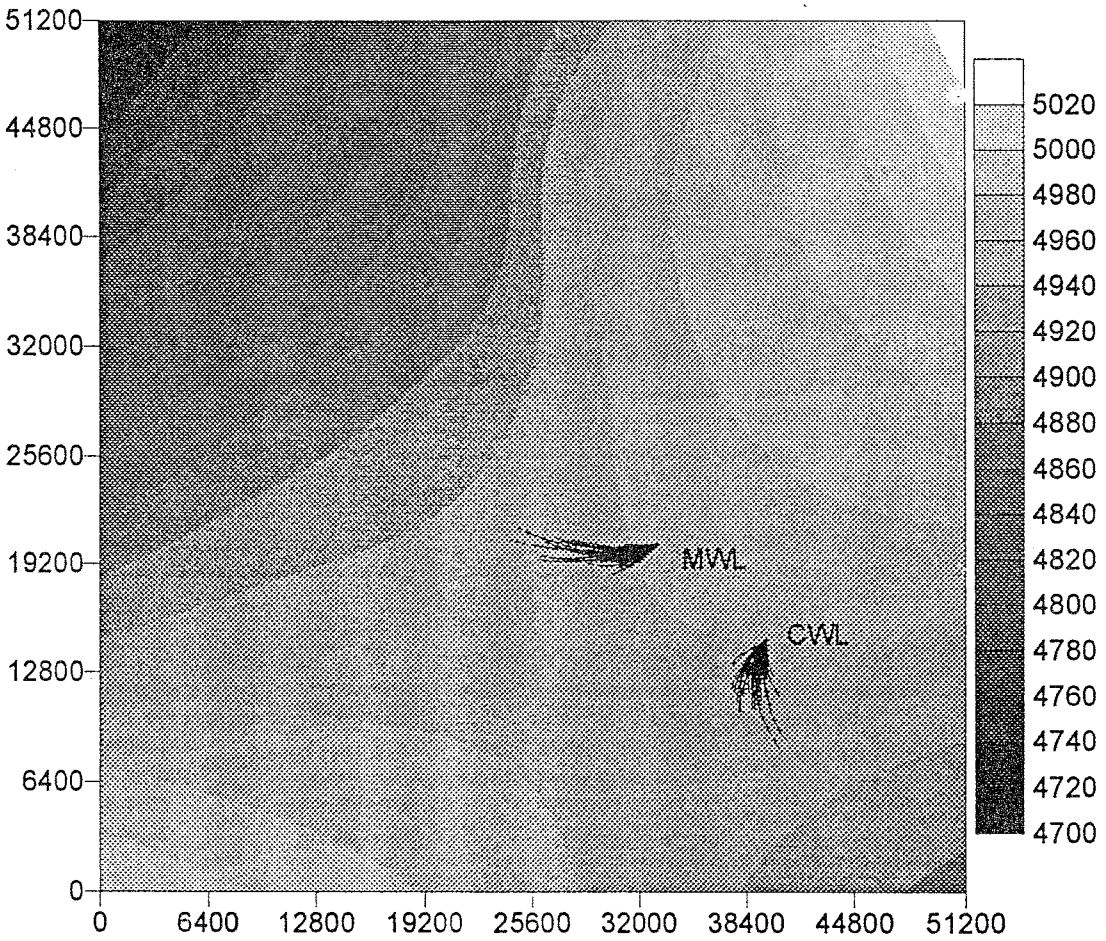


Figure 5.16: Ensemble mean head at t=88 months obtained from 110 simulations conditioned sequentially with iteration, to an AME criterion of 3.0 ft, on f data and head data at t=0 and 44 months (set *2*). Pathlines are of particles released from the CWL and MWL sites.

Figure 5.17 along with particle tracks from the CWL and MWL sites. While these indicators of performance in matching the true fields using collateral conditioning with iteration (Figures 5.8 and 5.9) become worse with the addition of more times of head data, the map of particle trajectories (Figure 5.17) is nearly identical to that obtained by collateral conditioning with iteration on only one head data time interval (set *2*). Again this suggests that collateral conditioning on more than a single interval of head data is wasted effort.

Conditioning on the same head data at times t=0, 6, 9, 36, 41, and 44 months (set *3*) by the sequential method was also done using an iteration convergence criterion of AME less than 8.0 ft. The average of 110 conditioned head fields at t=88 months is shown in Figure 5.18. The map of particle trajectories (Figure 5.18) appears very similar to the results of conditioning sequentially with iteration on a single head data interval (Figure 5.16) with possibly a slight constraint on travel distance and slightly more spreading. The slight spreading may be due to the less stringent convergence criterion used in conditioning simulations on set *3* data.

Computations became even more time consuming as the cokriging covariance matrix, equation (3.21) was expanded to incorporate head data at times 0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44 months, set *4*, and the iteration became more prone to divergence. As no more than one or two iterations were computationally practical and the likelihood of a divergence step became more probable after several iterations steps, the convergence criterion was relaxed even further to a maximum error of less than 20 ft between model simulated heads and head data. Shown in Figure 5.19 is the

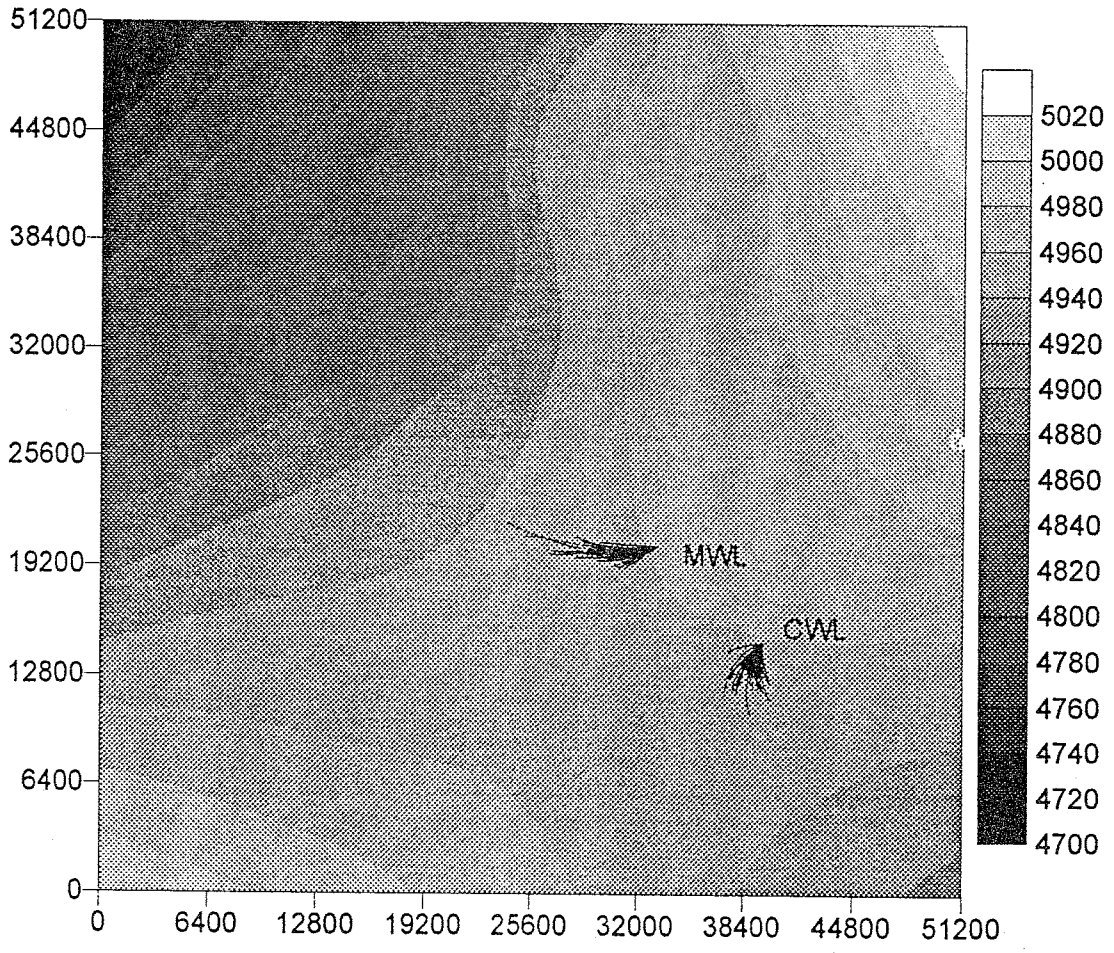


Figure 5.17: Ensemble average head at t=88 months obtained from 97 simulations conditioned collaterally with iteration, to an AME criterion of 8.0 ft, on f data and head data at t=0, 6, 9, 36, 41, and 44 months (set *3*). Pathlines are of particles released from the CWL and MWL sites.

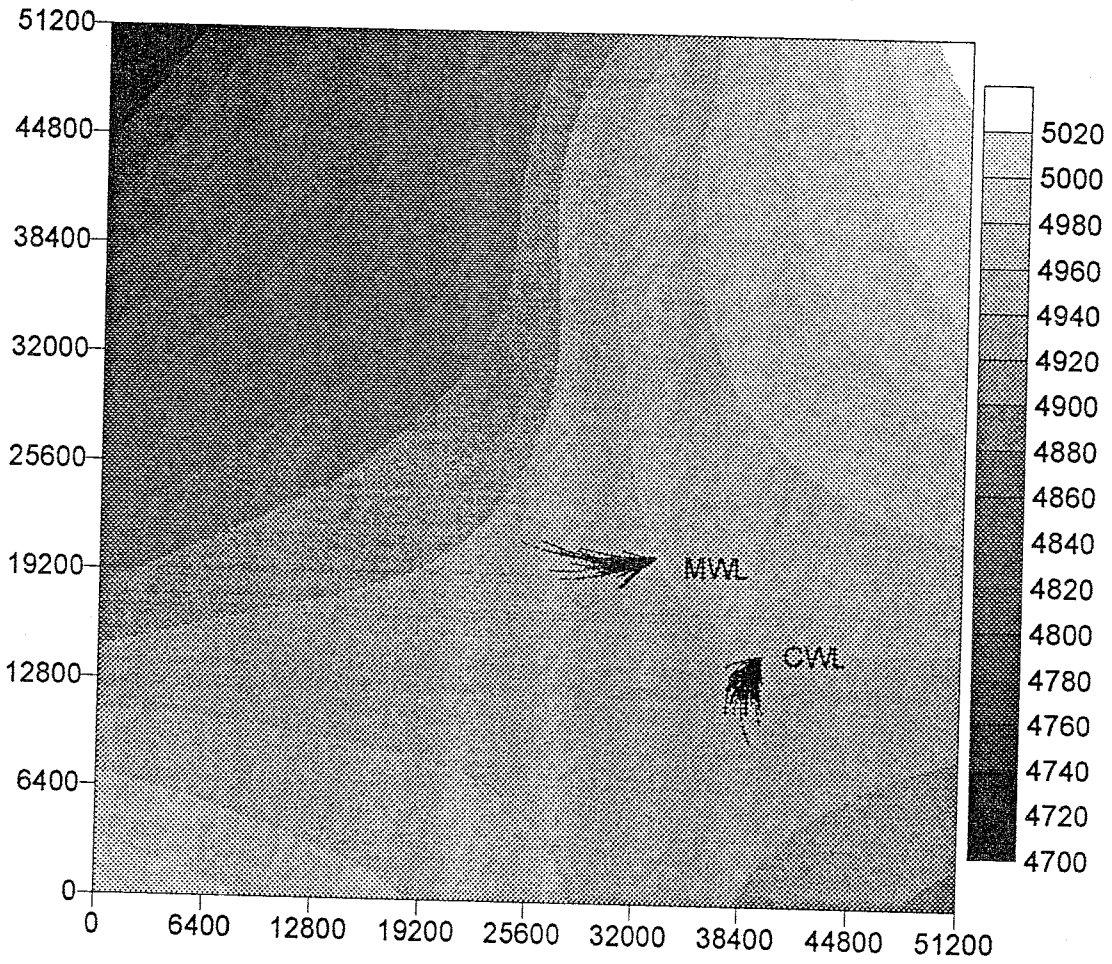


Figure 5.18: Ensemble average head at t=88 months obtained from 110 simulations conditioned sequentially with iteration, to an AME criterion of 8.0 ft, on f data and head data at t=0, 6, 9, 36, 41, and 44 months (set *3*). Pathlines are of particles released from the CWL and MWL sites.

mean of 129 simulated head fields conditioned on head data at all times except time t=38 months. The data at time t=38 months was excluded for reasons elucidated earlier. Again the map of particle trajectories (Figure 5.19) shows no improvement in constraining either particle travel distance or direction over conditioning on a single time head data time interval (Figure 5.15), reinforcing the conclusion stated earlier that conditioning collaterally on more than a single head data interval is counter-productive.

Conditioning all head data, excluding t=38 months, using the sequential method is somewhat less computationally demanding so it was possible to keep the iteration convergence criterion at an absolute maximum error (AME) of less than 8.0 ft. Figure 5.20 shows the average head at 88 months from 110 simulations conditioned sequentially on head data at times t=0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44 months (set *4*). Any improvement over sequential conditioning on data set *3* (Figure 5.18) is not obvious, but there does appear to be some slight constraining of the longer trajectories seen in sequential conditioning on data set *2* (Figure 5.16). What is noticeable, however, is that there is slightly more uncertainty in travel direction with sequential conditioning than is seen using collateral conditioning with the same data (Figure 5.19).

Results presented in chapter 4 indicated that iterating to match the head data is important in maximizing the benefits obtained from conditioning on head data. Conditioning without iteration on a single head data interval (set *2*) using the collateral method is shown in Figure 5.21. Figure 5.22 displays the conditioned mean head field and particle tracks at t=88 months obtained

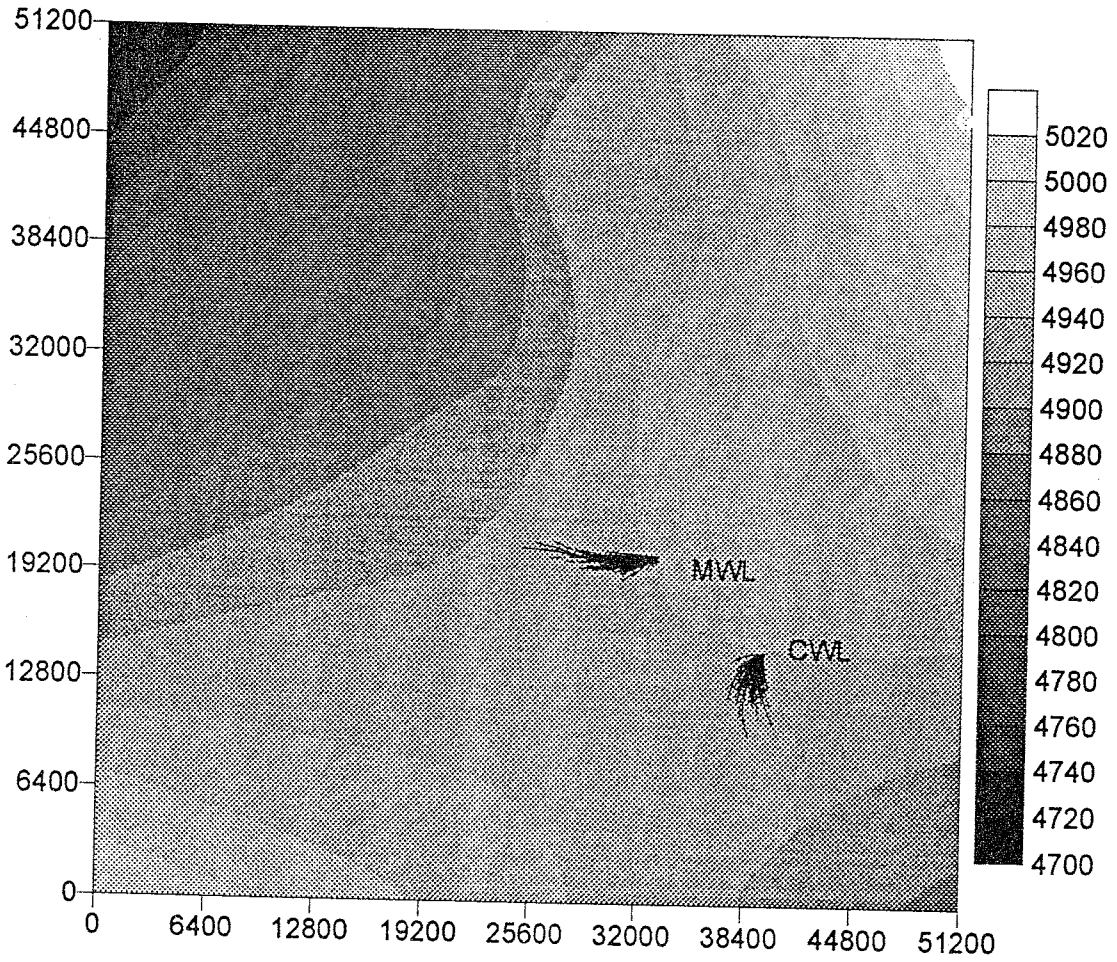


Figure 5.19: Ensemble average head at t=88 months obtained from 129 simulations conditioned collaterally with iteration, to an AME criterion of 20.0 ft, on f data and head data at t=0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44 months (set 4). Pathlines are of particles released from the CWL and MWL sites.

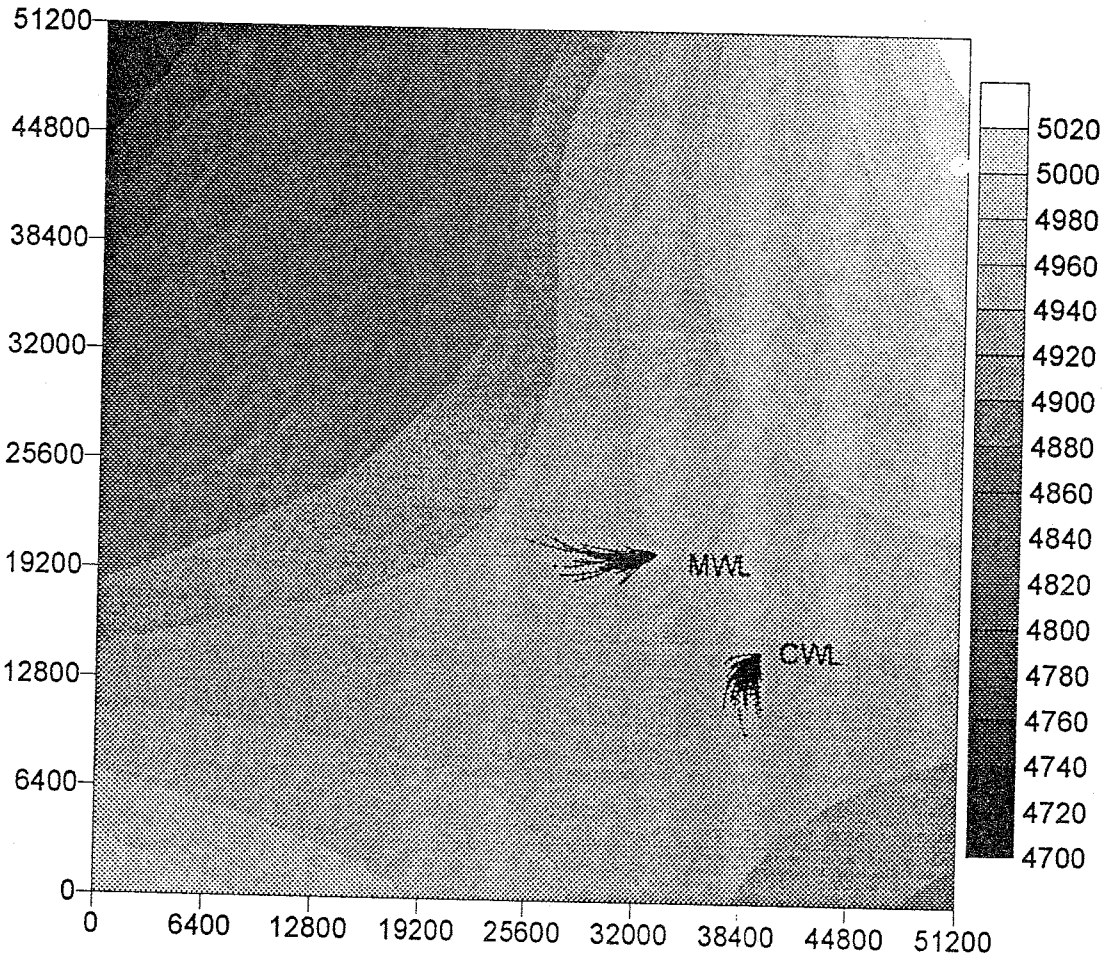


Figure 5.20: Ensemble average head at t=88 months obtained from 110 simulations conditioned sequentially with iteration, to an AME criterion of 8.0 ft, on f data and head data at t=0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44 months (set *4*). Pathlines are of particles released from the CWL and MWL sites..

using the sequential conditioning method without iteration. These plots again indicate that iterating to match the head data has the effect of constraining both the travel direction and distance of advecting particles. One interesting thing to note is that the iteration seems to have more effect on constraining the direction particles advect with the sequential method than it does with the collateral method. Using the sequential method without iteration (Figure 5.22) results in an uncertainty in travel direction nearly the same as that obtained without conditioning on any head data at all (Figure 5.14).

5.6.2 Summary of results from conditioning using hypothetical data

Results obtained from this set of conditional simulations are generally similar to the results from chapter 4, even though the model, the distribution of data, and the nature of the transient state are different. In chapter 4 the data was rather uniformly distributed with the same number of f and h data at the same locations. Here there are more h than f data and no f datum exists at the same location as an h datum. In chapter 4 the boundaries remained constant while the transient state was induced by a step change in uniform recharge. Here there is no recharge and the transient was induced by changing the boundary conditions through time. Despite these dissimilarities in the underlying model, conditioning simulations on head data was seen to have some consistent effects.

- The first use of head data for conditioning was seen to have the most noticeable effect on improving parameter estimation. Conditioning on more head data intervals was seen to produce either diminishing returns

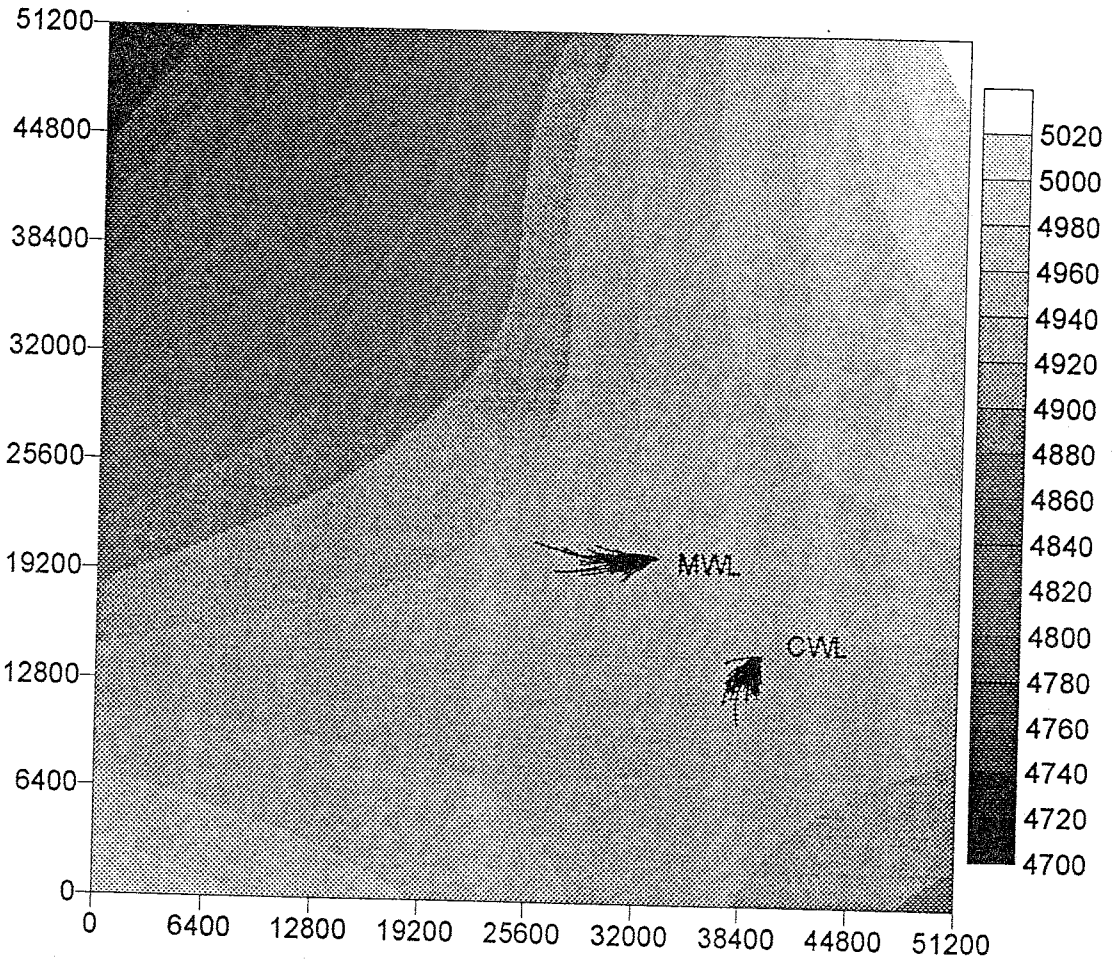


Figure 5.21: Ensemble average head at t=88 months obtained from 100 simulations conditioned collaterally without iteration on f data and head data at t=0 and 44 months (set *2*). Pathlines are of particles released from the CWL and MWL sites..

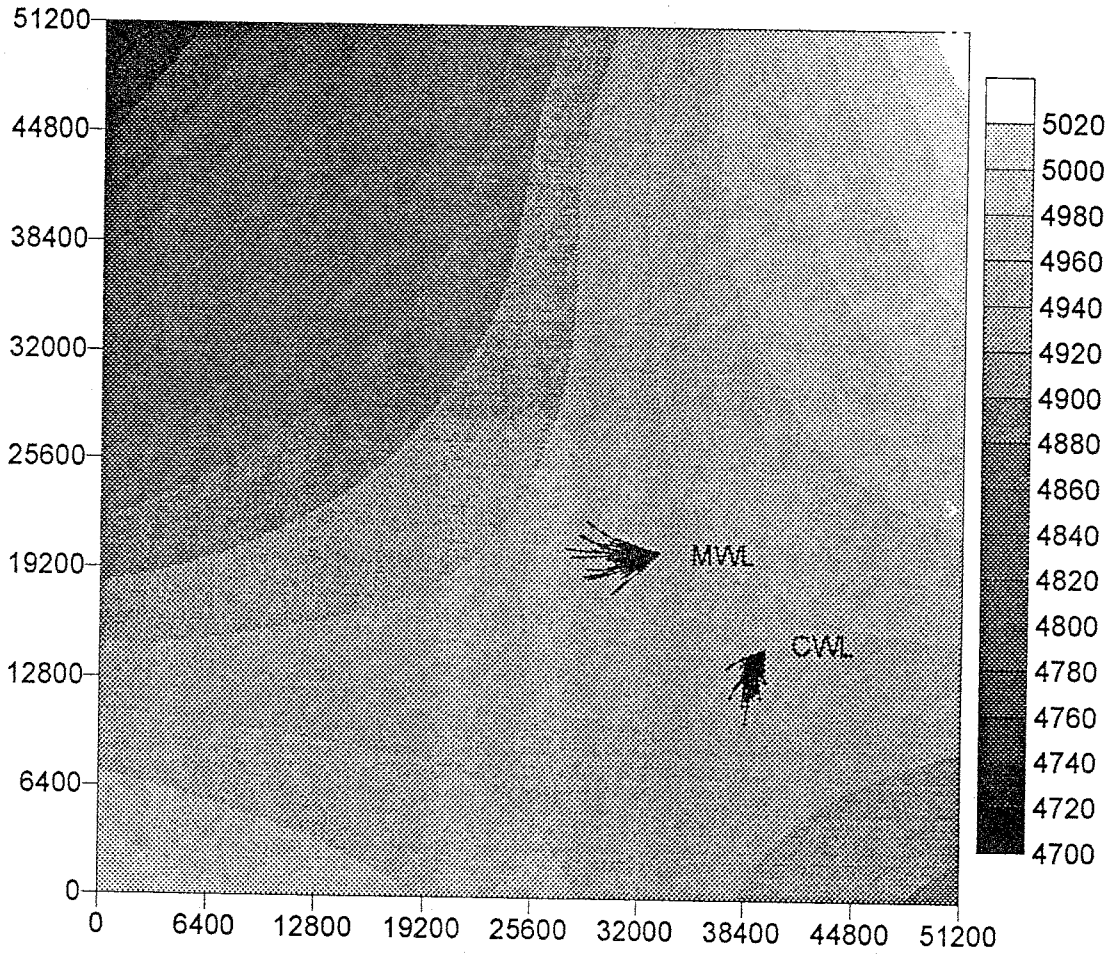


Figure 5.22: Ensemble average head at t=88 months obtained from 100 simulations conditioned sequentially without iteration on f data and head data at t=0 and 44 months (set *2*). Pathlines are of particles released from the CWL and MWL sites..

in improvement or else actually poorer results. This may be due in part to difficulties encountered inverting the nearly singular data covariance matrices. As these matrices get larger through the use of more head data, small errors in the covariances may become magnified by the matrix decomposition negating any improvements which would otherwise be obtained by using the extra head data.

- The direction of advective transport appears to be the most sensitive indicator of the effect of conditioning simulations on head data. The effect is to narrow the focus of particle trajectories, constraining the direction of transport to a narrow path. Collateral conditioning appears to have a stronger influence on the focusing of particle trajectories than does sequential conditioning.

- Iterating to match the head data is again seen to have a noticeable effect. The effect, however, is not consistent across all of the performance measures examined. Iterating caused some of the performance measures to move in a direction different than would be expected if the estimation of parameters were actually improving. The most obvious effect of iterating to match the head data was seen in narrowing the direction of advective transport and this effect was more noticeable with the sequential conditioning method than with the collateral method. Without iteration collateral conditioning tends to focus particle trajectories more than does sequential conditioning, but iteration appears to affect sequential conditioning more by the same measure.

- Nearly all of the performance measures as well as the computational cost indicate that using the collateral conditioning method with more than a single head data interval is disadvantageous.

- While conditioning on more than a single interval using the collateral method resulted in poorer performance, conditioning on more head data using the sequential method seemed to give either a slight improvement or very little change at all.

Some general conclusions can be drawn from this exercise regarding the interpretation of results presented in the following section of simulations conditioned on actual KAFB data.

- First, conditioning on all of the available head data through time adds to the computational burden while providing very little, if any, improvement in the results. The reason for this may be due to a combination of problems with the algorithm and with its implementation. The first-order perturbation approximation of covariances has error which increases with the variance of f. Matrices of covariances at the data locations are ill-conditioned and often nearly singular. As these matrices expand with additional head data intervals, errors in the covariances may propagate and be magnified by the singular value matrix decomposition into large errors in the cokriged fields.

- Secondly, conditioning collaterally on more head data intervals appears to cause the highs and lows of the conditioned mean f field to become

larger. This effect is probably also related to errors in the covariance matrices and the singular value decomposition algorithm.

- Thirdly, collateral conditioning constrains the particles to travel a more narrow pathway than does conditioning by the sequential method. This last item may be of some interest to those who wish to predict which direction the contaminant plume will migrate.

A final note on this section is that the bilinear interpolation scheme used to estimate the boundary conditions resulted in a saddle-point in the mean head field very near the CWL site. The presence of this saddle-point has the effect of increasing uncertainty in travel direction and ultimate destination of contaminants released from the CWL site. For this hypothetical reality exercise, transport from the CWL site appears to be predominantly towards the bottom boundary and transport from the MWL site predominantly towards the left and upper boundaries. For the real KAFB data, as will be seen in the next section, this saddle-point becomes a more dominant factor in transport direction uncertainty from the CWL site.

5.6.3 Conditioning using KAFB data

In this section results of simulations conditioned on real data from Kirtland Air Force Base are presented. The same sequence of conditioning data sets and methods as was used for the hypothetical data is maintained here. Along with maps of the average conditioned head fields at t=88 months and particle tracks from the CWL and MWL release points, plots of the average conditioned f fields and simulated variance are shown. The true field of

heterogeneous permeability underlying the KAFB site is unknown hence there are no AMSE or MMSE comparison performance measures nor statistics comparing particle tracking results. However, the extrema of the mean conditional f field, and the maximum variance of the conditioned f and head fields are given.

Shown in Figure 5.23a is the mean f of 100 simulations conditioned on 7 f data only. The minimum is -1.55 and the maximum is 1.30. Note that this variation occurs near the cluster of f data locations while the remainder of the field is essentially uniform. Variance of the 100 simulations is given in Figure 5.23b which has a maximum of .849 at location x=6800 ft y=48400 ft. Tracks of particles released from the CWL and MWL sites and average head from the 100 simulations conditioned only on f data are shown in Figure 5.24. Due the saddle-point, particles released from the CWL site may possibly travel towards the bottom boundary or towards the left boundary. The maximum variance of head resulting from these simulations is 156 ft^2 occurring at location x=14800 ft y=39600 ft.

Mean f field and f field variance resulting from 110 simulations conditioned collaterally, with iteration, on head data at times t=0 and 44 months are shown in Figures 5.25a and 5.25b. Iteration to match the head data was stopped at a criterion of AME less than 14.0 ft between head data and model results. Convergence criterion for the real KAFB data is greater than for the hypothetical data due to the higher variance of the real data and uncertainty in the conceptual model. Conditioning collaterally on head data results in much more variability in the mean f field, especially in the vicinity of head data

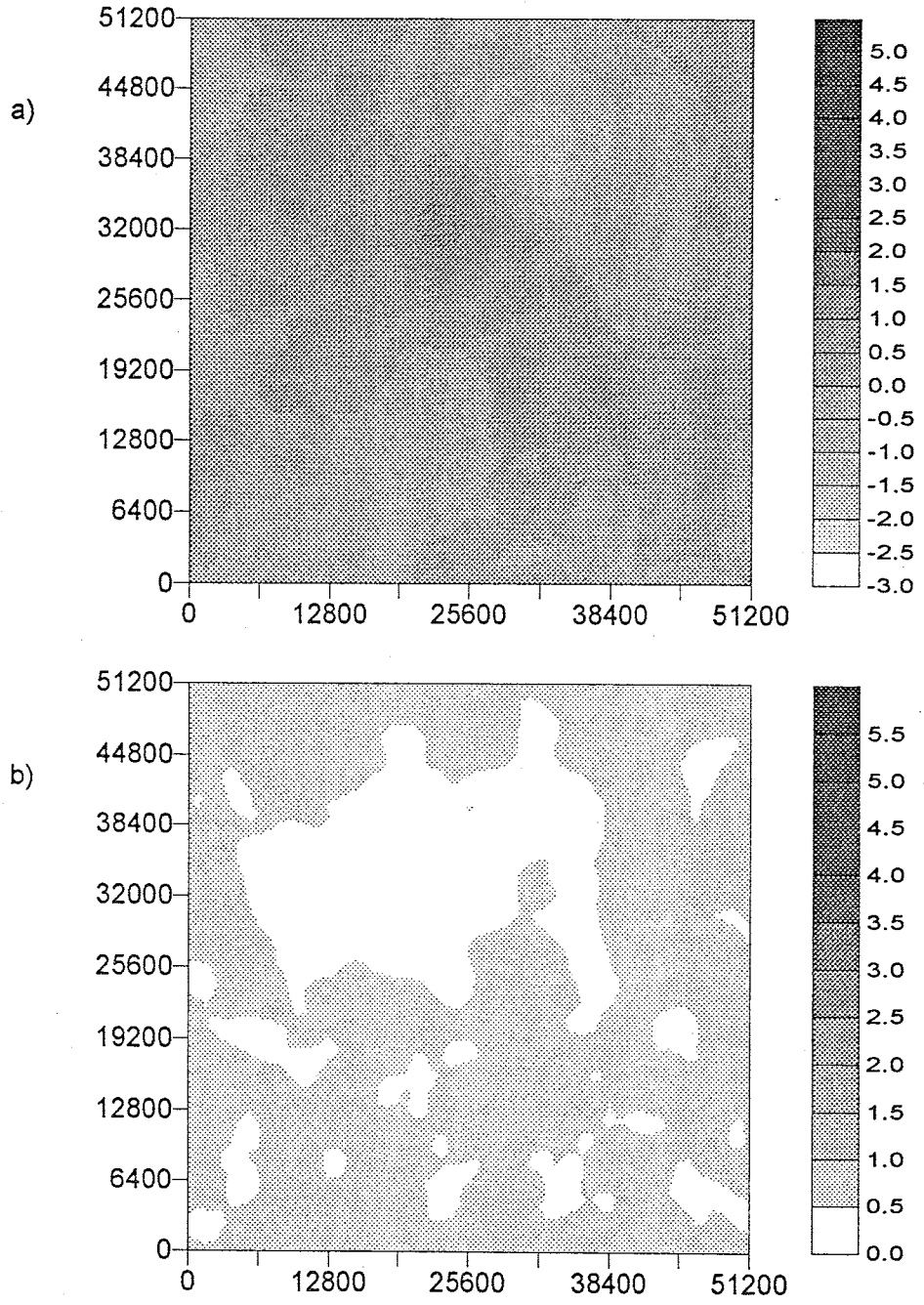


Figure 5.23: a) Ensemble average of 100 simulated f fields conditioned only on f data. b) Ensemble variance of 100 f fields conditioned only on f data.

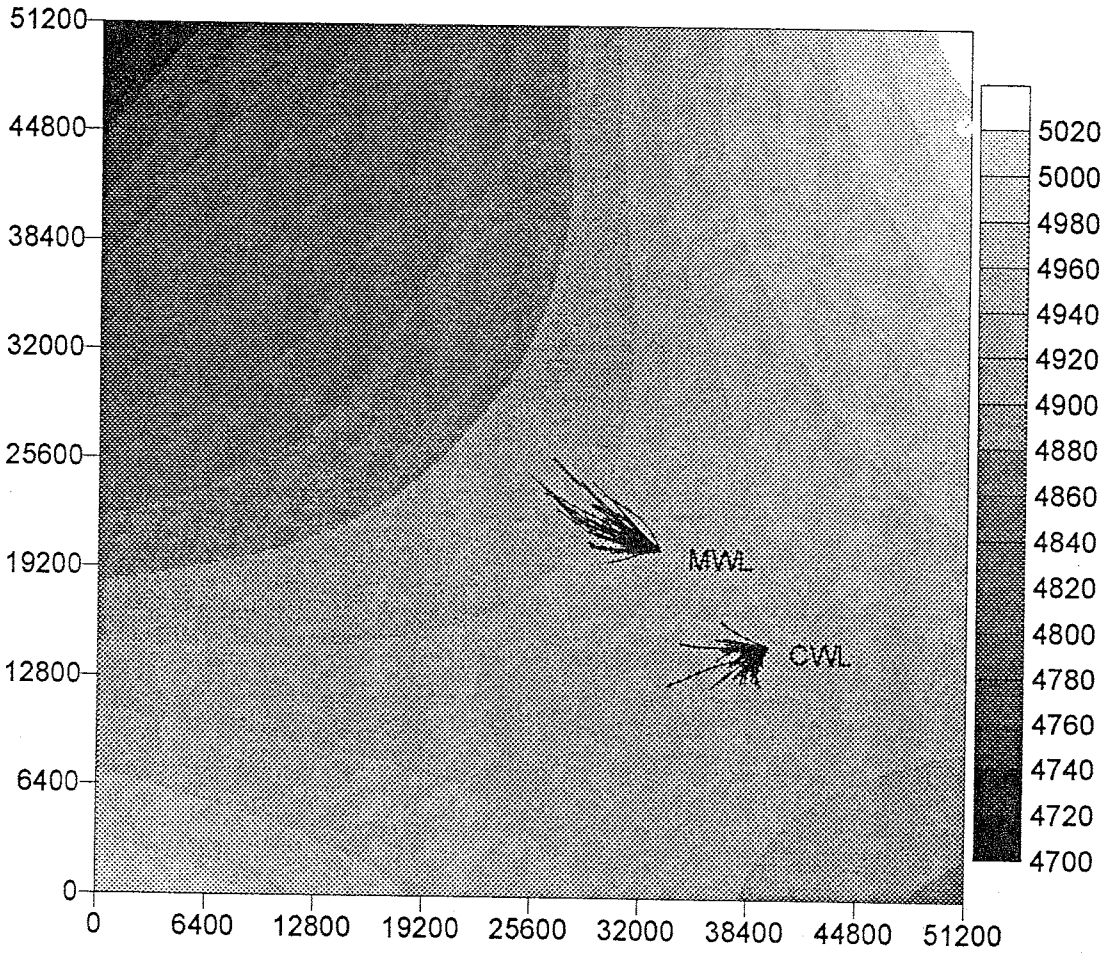


Figure 5.24: Average head field resulting from 100 simulations conditioned only on f data. Particle tracks shown were released from the CWL and MWL sites.

locations, with a minimum and maximum mean f of -2.56 and 5.09. Maximum f variance of 5.90 occurs at location x=20400 y=19600. Conditioning collaterally on head data at times t=0 and 44 months has the most distinct effect of moving the saddle-point and groundwater divide down and to the right. This is seen in the average of 110 head fields shown in Figure 5.26. Thus particles released from the CWL site appear more likely to travel towards the left boundary even though many may still eventually be bound for the south boundary. Conditioning collaterally on head data also is seen to constrain particles released from the MWL site to a more narrow pathway. The maximum head variance resulting from 110 simulations conditioned collaterally on head data at times t=0 and 44 months is 163 ft^2 at location x=7600 y=33200.

Collateral conditioning without iteration on f data and h data at times t=0 and 44 months was also done. The conditioned mean head field and particle tracks released from the CWL and MWL sites are shown in Figure 5.27. The effect of iterating to meet the AME criterion of 14 ft on head data at t=0 and 44 months has the barely perceptible effect of slightly constraining the direction of transport. The extremes of the conditioned mean f field, however, are reduced to 3.18 and -1.93 and the maximum variance of f is reduced from the high of 5.90 with iteration down to only .850. The difference in head variance obtained from collateral conditioning with iteration to collateral conditioning without iteration, though, is a mere 163 ft^2 versus 161 ft^2. A table summary and graphical comparisons of the $maxf$, $minf$, and MVAR measures for simulations conditioned on the real KAFB data are included at the end of this section.

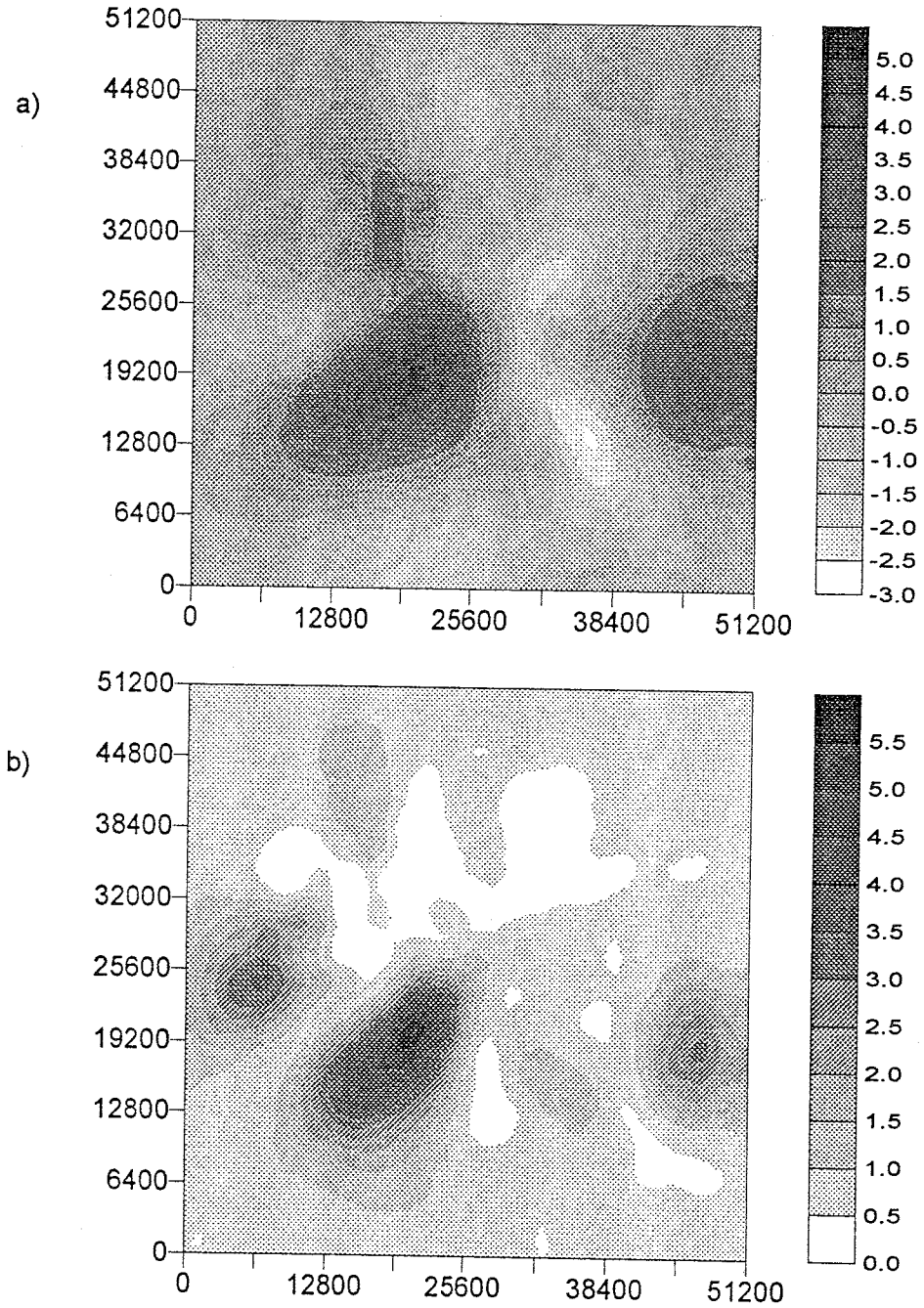


Figure 5.25: a) Ensemble average of 110 simulated f fields conditioned collaterally on f data and head data at times t=0 and 44 months to an iteration AME criterion of less than 14.0 ft. b) Ensemble variance of 100 f fields conditioned collaterally on f data and head data at times t=0 and 44 months.

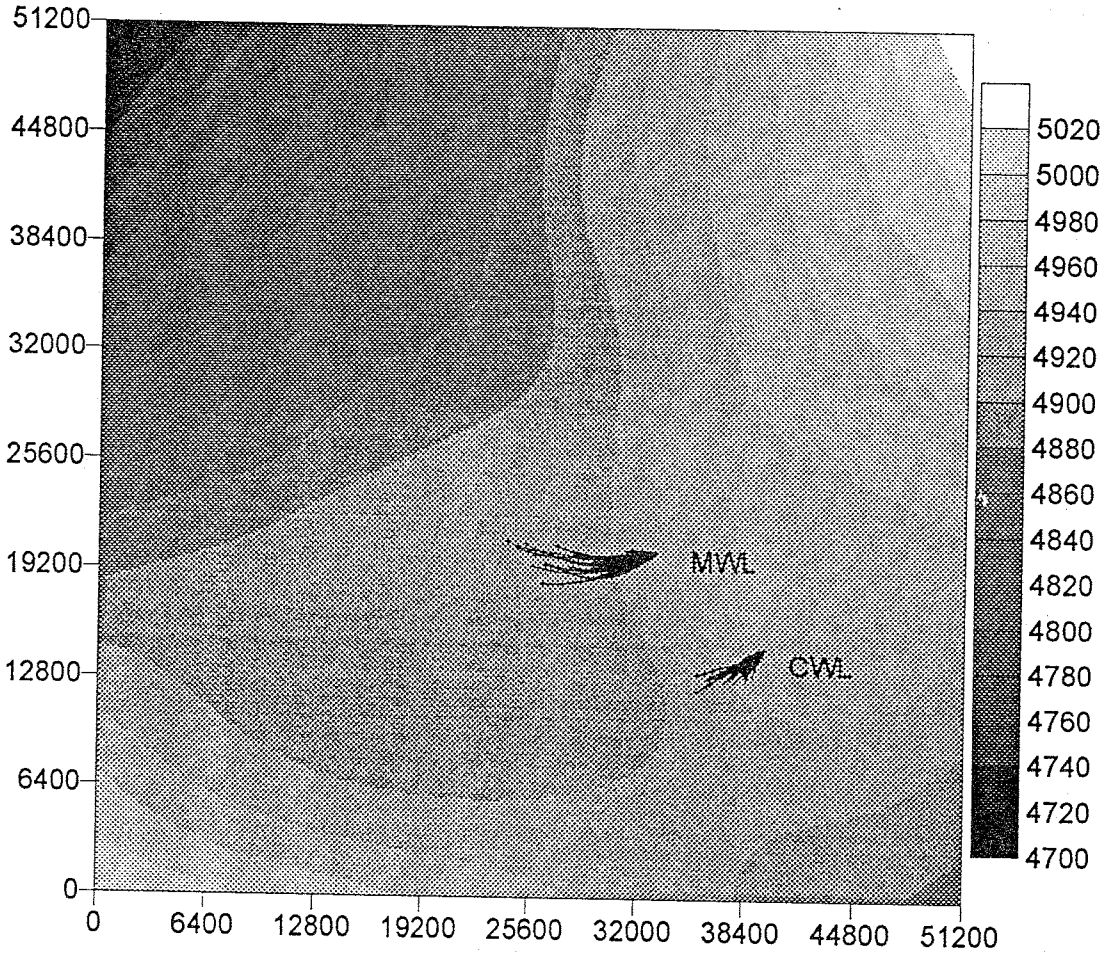


Figure 5.26: Average head field resulting from 110 simulations conditioned collaterally on f data and head data at times t=0 and 44 months to an iteration AME criterion of less than 14.0 ft. Particle tracks shown were released from the CWL and MWL sites.

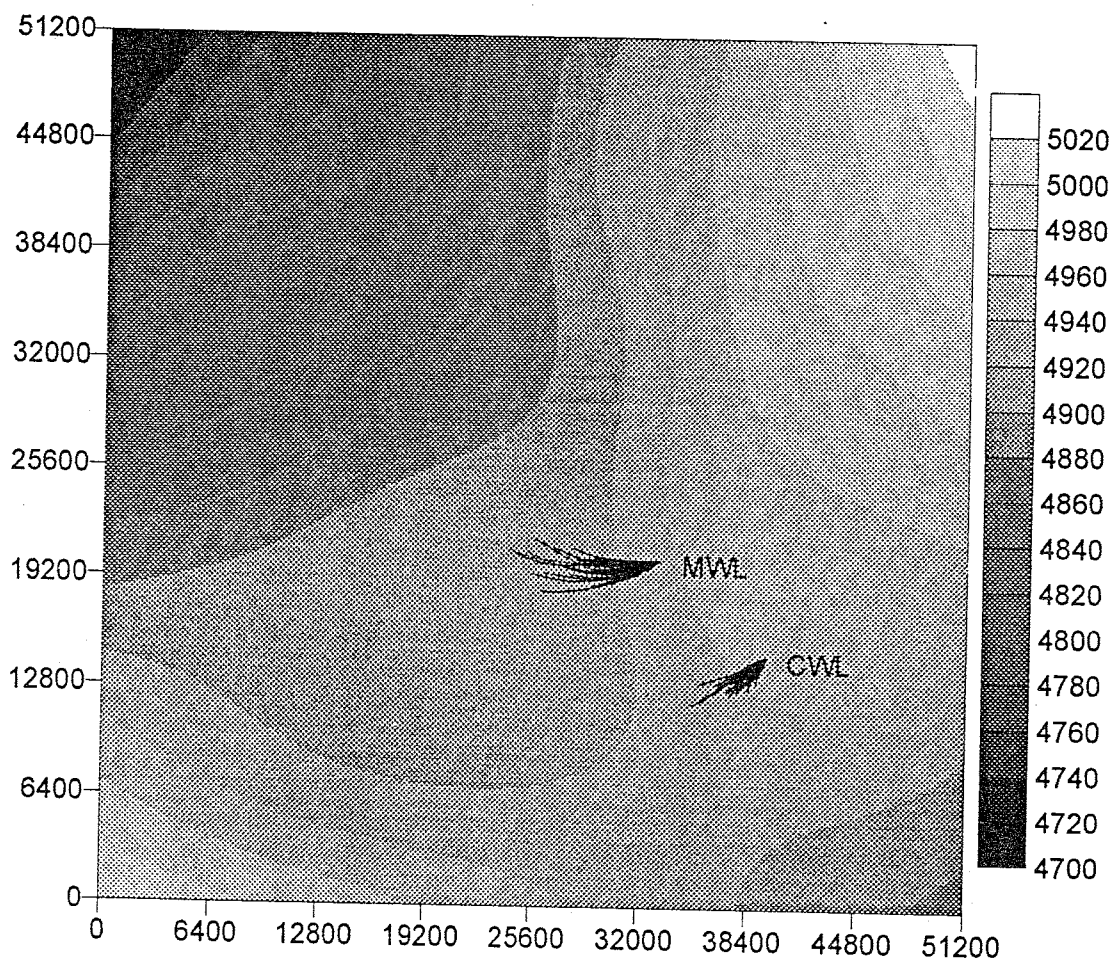


Figure 5.27: Average head field resulting from 100 simulations conditioned collaterally, with no iteration, on f data and head data at times t=0 and 44 months. Particle tracks shown were released from the CWL and MWL sites.

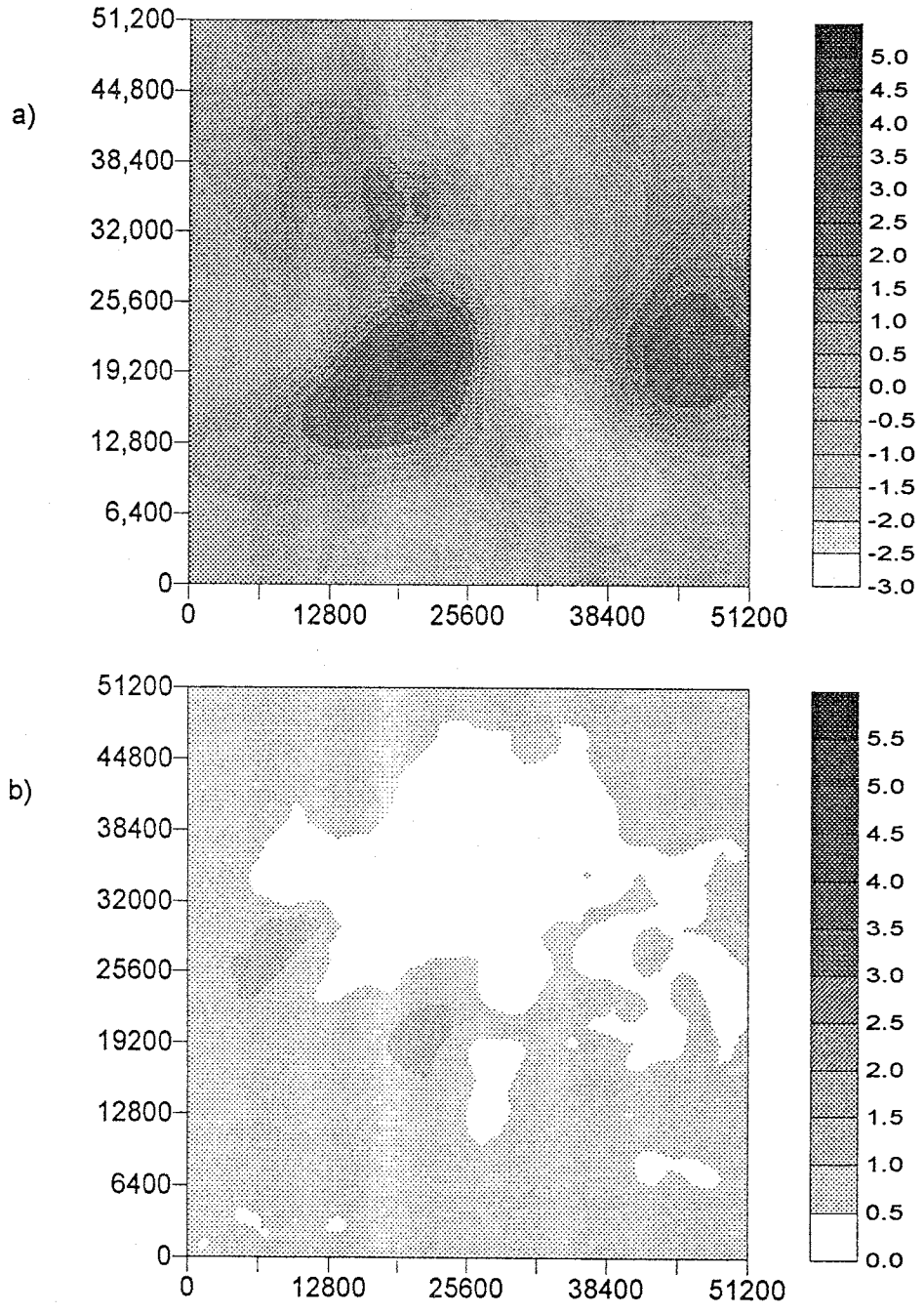


Figure 5.28: a) Ensemble average of 110 simulated f fields conditioned on f data and head data at times t=0 and 44 months to an AME criterion of less than 14.0 ft using the sequential method. b) Ensemble variance of 110 f fields conditioned, using the sequential method, on f data and head data at times t=0 and 44 months.

Conditioning on head data using the sequential method, with iteration to an AME criterion of 14.0 ft., at times t=0 and 44 months results in a nearly identical appearance in the mean f field, as seen in the collateral conditioning case, but with less extreme difference between the highs and lows. The mean f field from 110 simulations conditioned by the sequential method on head data at times t=0 and 44 months is shown in Figure 5.28a. The maximum mean f is 3.69 and the minimum -1.88. Variance of the 110 conditioned f fields is shown in Figure 5.28b with a maximum variance of 1.42 at location x=10000 ft y=29200 ft.

Particle tracks in fields conditioned by the sequential method on head data at times t=0 and 44 months are fairly similar to collateral conditioning except that the pathlines show slightly wider spread and particles released from the CWL site appear more likely to travel towards the bottom boundary. The mean of 110 head fields conditioned using the sequential method with iteration on head data at times t=0 and 44 months is shown in Figure 5.29. Maximum head variance from these simulations is 164 ft^2 at location x=9200 ft y=34800 ft. Conditioning on head data at times t=0 and 44 months using the sequential method without iteration is also seen to lessen the highs and lows of the conditioned mean f field from the extremes which appear with iteration. This is reflected in the f field variance, which decreases from a maximum of 1.42 with iteration to a maximum of .855 without iteration, using the sequential conditioning method. Variance of the head fields is also less without iteration, going from a maximum of 164 ft^2 with iteration to a maximum of 151 ft^2 without iteration. The mean head field conditioned on head data at t=0 and 44 months from 100 simulations using the sequential method without iteration,

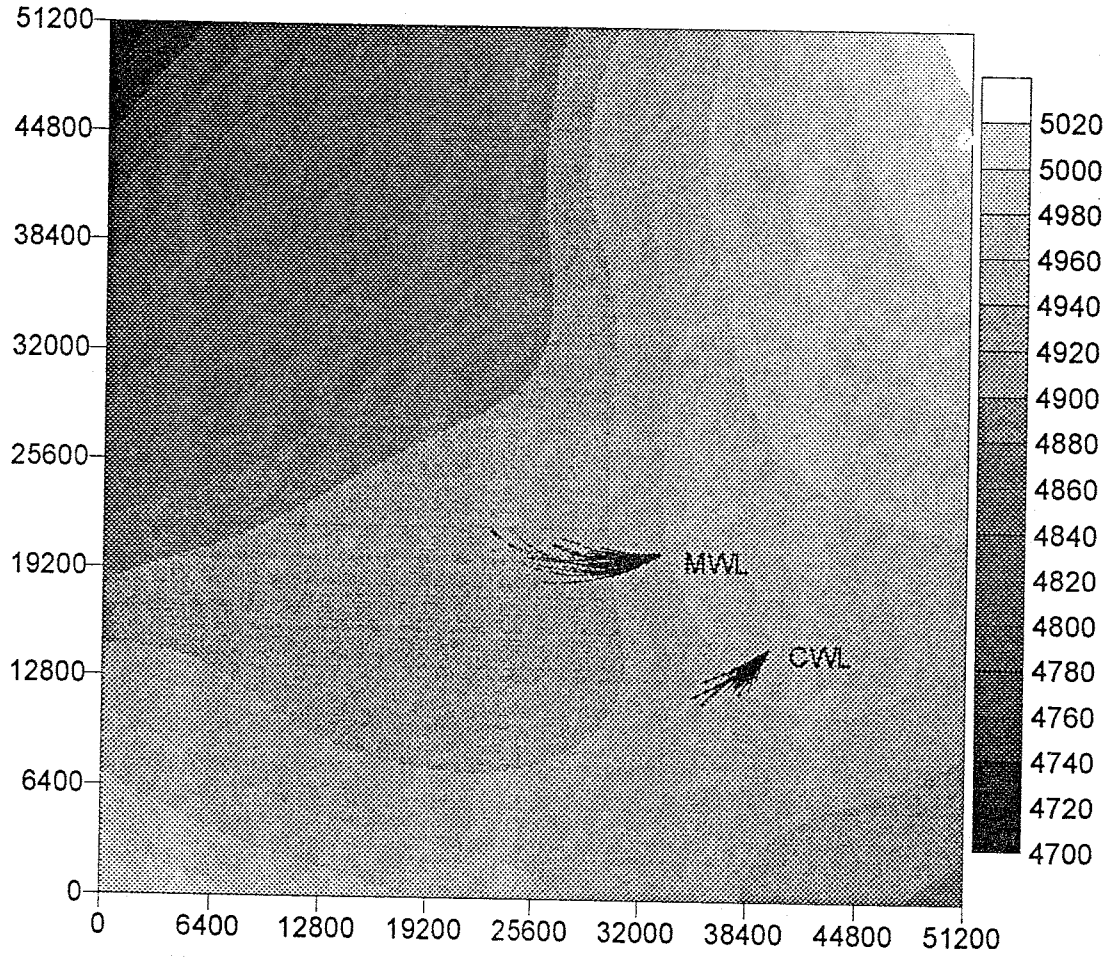


Figure 5.29: Average head field resulting from 110 simulations conditioned on f data and head data at times t=0 and 44 months to an AME criterion of less than 14.0 ft by the sequential method. Particle tracks shown were released from the CWL and MWL sites.

along with particle tracks from the CWL and MWL release points, is shown in Figure 5.30. The influence of iteration on particle travel direction appears to be greater with the sequential method than with the collateral method. Particle tracks seem to have a wider spread using sequential conditioning without iteration (Figure 5.30) compared to sequential conditioning with iteration (Figure 5.29) than using collateral conditioning with and without iteration (Figures 5.26 and 5.27).

Collateral conditioning on head data at times t=0, 6, 9, 36, 41, and 44 months results in more extreme highs and lows in the mean f field, shown in Figure 5.31a, but with the same basic pattern as seen in the conditioning on t=0 and 44 months head data. The maximum mean f is 7.68 and the minimum is -4.05. Mean and variance of f shown in Figures 5.31a and 5.31b result from 107 simulations conditioned collaterally on head data at times t=0, 6, 9, 36, 41, and 44 months to an AME criterion of less than 14.0 ft. The maximum f variance in Figure 5.31b is 2.96 at location x=10000 ft y=29200 ft.

Collateral conditioning on head data at the additional times of 6, 9, 36, and 41 months has the most obvious effect of predicting shorter paths that the particles travel in 88 months. Also the groundwater divide appears to be moved farther towards the bottom boundary by collateral conditioning on the additional times resulting in more likelihood for particles from the CWL site to travel towards the left boundary. Shown in Figure 5.32 is the mean of 107 head simulations conditioned by the collateral method, with iteration to an AME of 14.0 ft., on head data at times t=0, 6, 9, 36, 41, and 44 months. The maximum head variance from these simulations is 183 ft^2 located at x=7600 ft y=43600

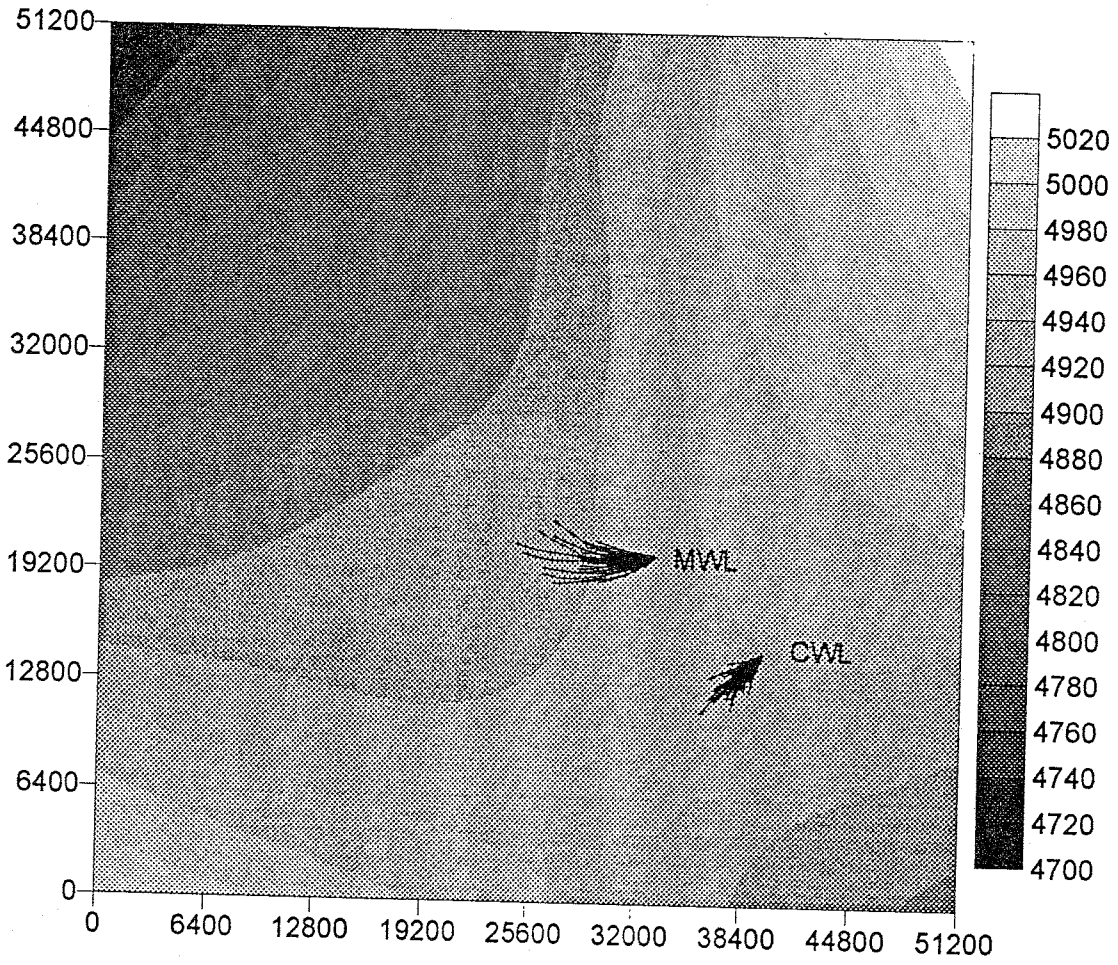


Figure 5.30: Average head field resulting from 100 simulations conditioned on f data and head data at times t=0 and 44 months using the sequential method without iteration. Particle tracks shown were released from the CWL and MWL sites.

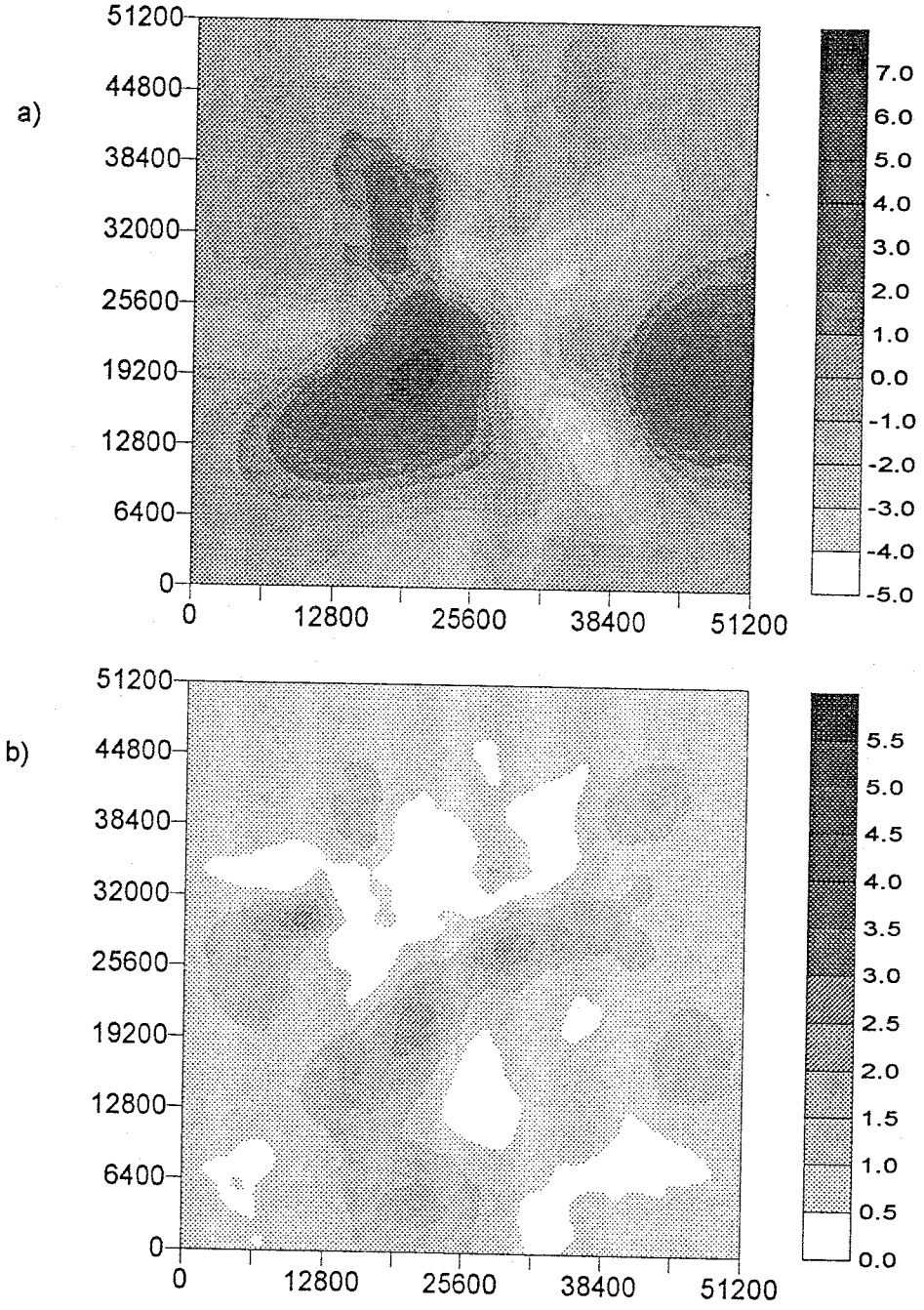


Figure 5.31: a) Ensemble average of 107 simulated f fields conditioned collaterally on f data and head data at times t=0, 6, 9, 36, 41, and 44 months to an AME criterion of less than 14.0 ft. b) Ensemble variance of 107 f fields conditioned collaterally on f data and head data at times t=0, 6, 9, 36, and 44 months.

ft. The mean head field conditioned collaterally on head data at times t=0, 6, 9, 36, 41, and 44 months without iteration is shown in Figure 5.33. Surprisingly particle tracks in the fields conditioned without iteration appear to be slightly more focused than those in the fields conditioned with iteration, counter to the effect of iteration seen previously. The same trend in highs and lows of the conditioned mean f field and lower maximum variances of both f and head fields, though, is again repeated.

Conditioning by the sequential method on head data at times t=0, 6, 9, 36, 41, and 44 months results again in a similar pattern for the mean f field, shown in Figure 5.34a, but less extreme highs and lows than the collateral conditioning method. From 105 simulations conditioned on head data at times t=0, 6, 9, 36, 41, and 44 months using the sequential method the minimum mean f is -1.67 and the maximum is 3.43. The maximum f variance resulting from these simulations, shown in Figure 5.34b, is 1.18 at location x=22800 y=20400. The mean head field resulting from sequential conditioning on head data at times t=0, 6, 9, 36, 41, and 44 months appears very similar to the sequential conditioning results using only head data at times t=0 and 44 months, with the possibility that the length of the travel paths becomes somewhat shorter with the additional conditioning at more times. The mean of 105 head field simulations conditioned using the sequential method on head data at times t=0, 6, 9, 36, 41, and 44 months is shown in Figure 5.35. The maximum head variance for these simulations of 164 ft^2 occurs at x=9200 ft y=35600 ft. Particle tracks and mean head obtained from 100 simulations conditioned, using the sequential method without iteration, on f data and h data at times t=0, 6, 9, 36, 41, and 44 months are shown in Figure 5.36. Compared

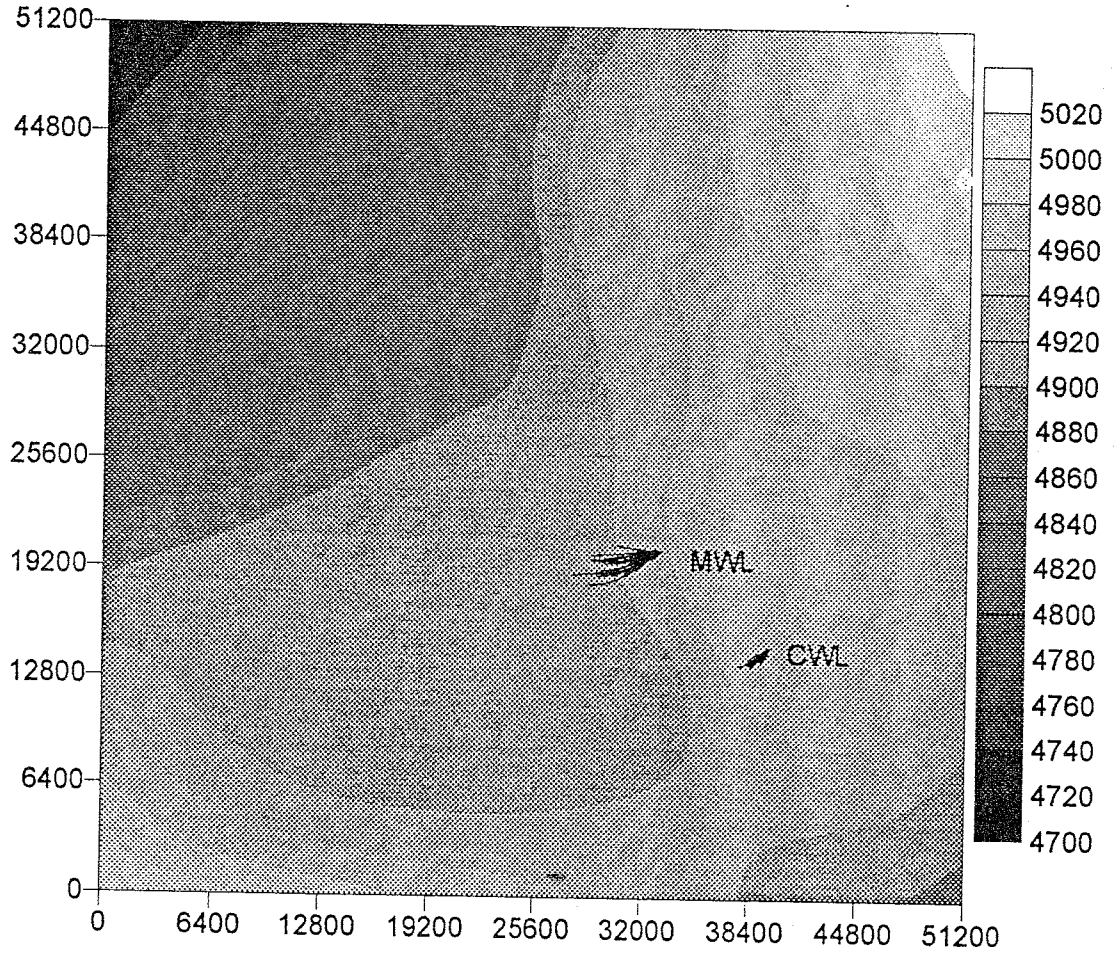


Figure 5.32: Average head field resulting from 107 simulations conditioned collaterally on f data and head data at times t=0, 6, 9, 36, 41, and 44 months to an AME criterion of less than 14.0 ft. Particle tracks shown were released from the CWL and MWL sites.

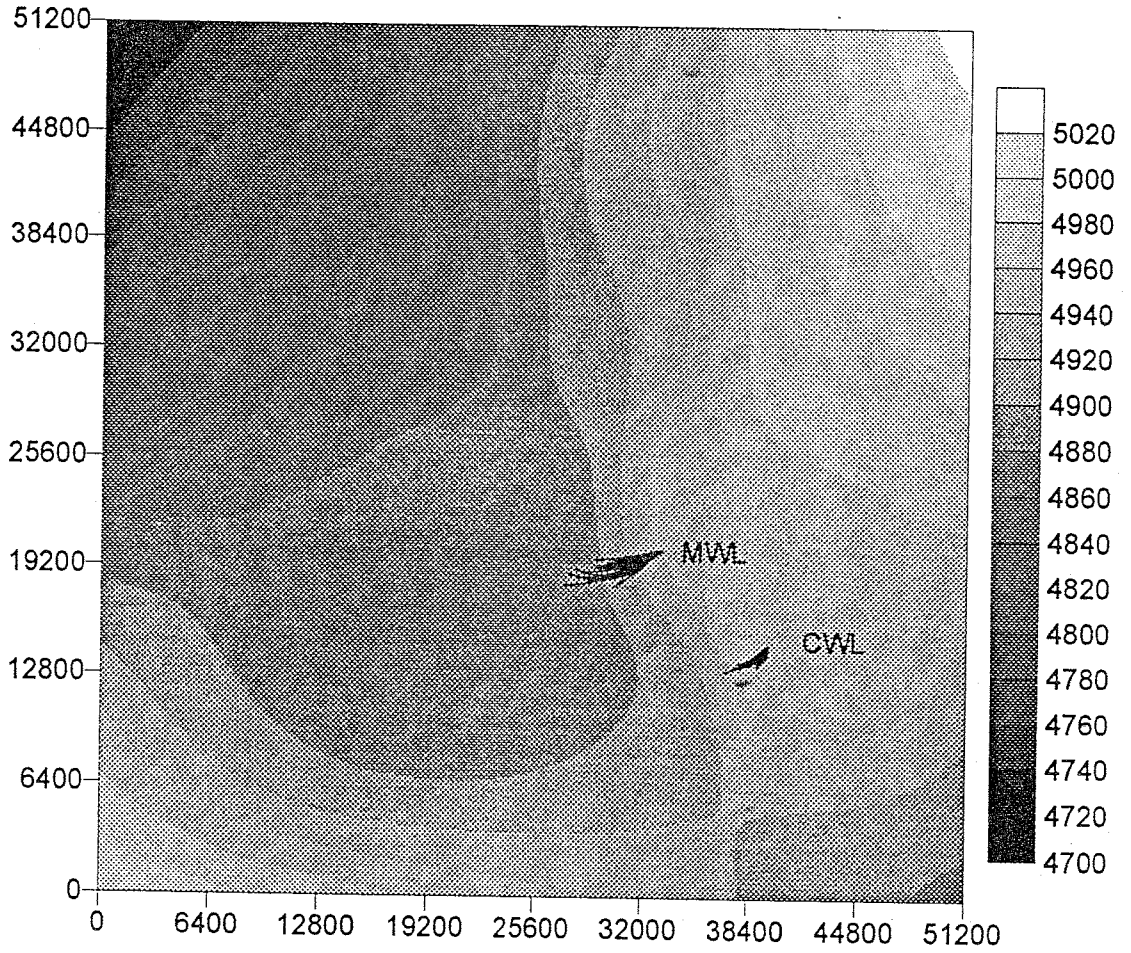


Figure 5.33: Average head field resulting from 100 simulations conditioned collaterally without iteration on f data and head data at times t=0, 6, 9, 36, 41, and 44 months. Particle tracks shown were released from the CWL and MWL sites.

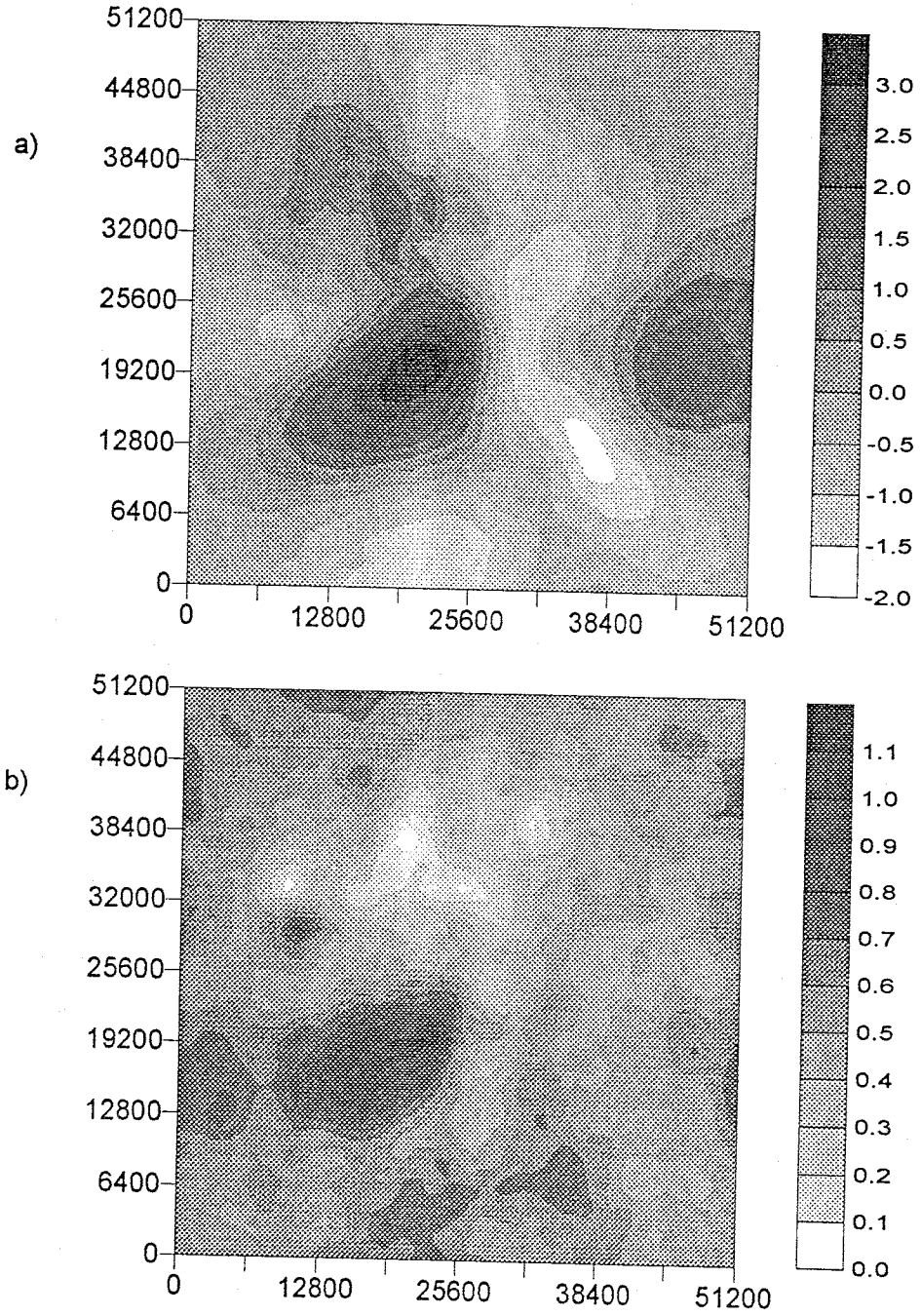


Figure 5.34: a) Ensemble average of 105 simulated f fields conditioned on f data and head data at times t=0, 6, 9, 36, 41, and 44 months to an AME criterion of less than 14.0 ft using the sequential method. b) Ensemble variance of 105 f fields conditioned by the sequential method on f data and head data at times t=0, 6, 9, 36, and 44 months.

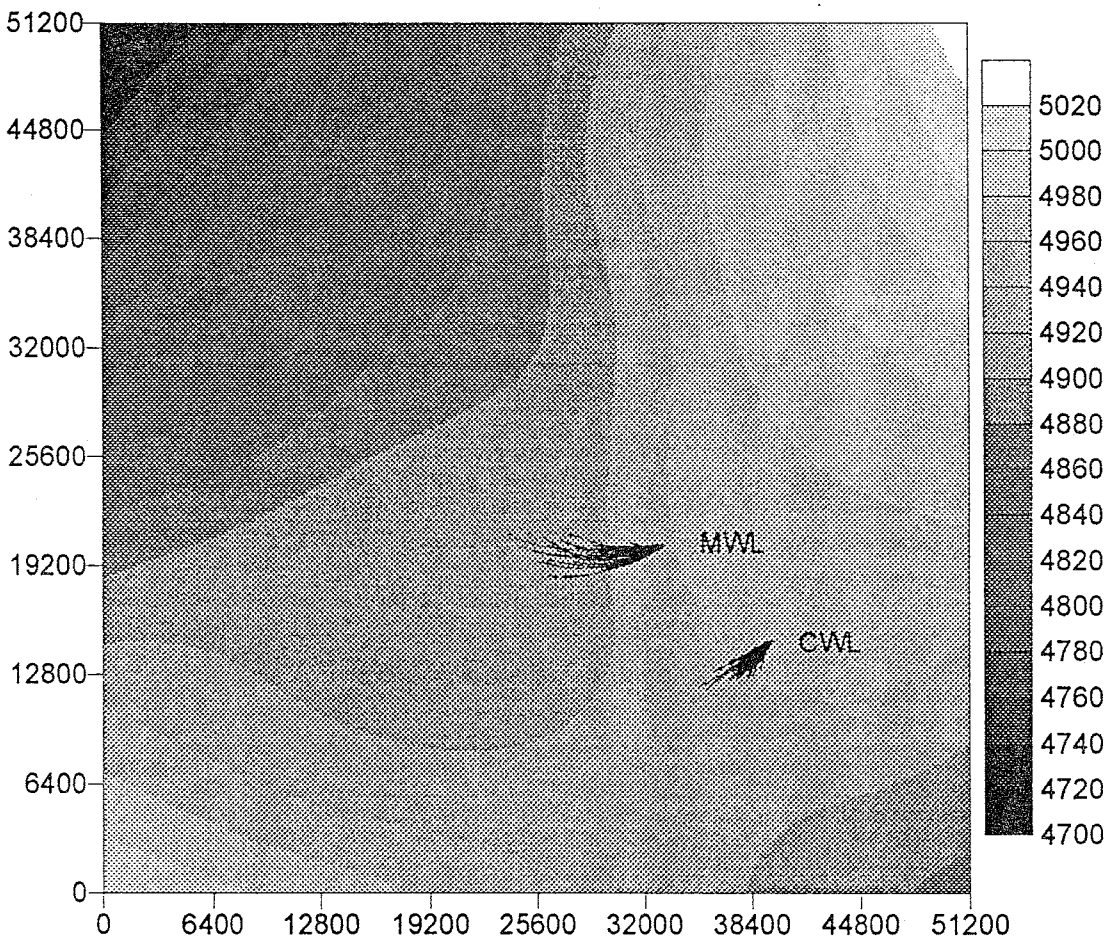


Figure 5.35: Average head field resulting from 105 simulations conditioned by the sequential method, with iteration to an AME criterion of 14.0 ft., on f data and head data at times t=0, 6, 9, 36, 41, and 44 months. Particle tracks shown were released from the CWL and MWL sites.

to particle tracks in the simulations conditioned sequentially without iteration (Figure 5.35), iteration again is seen to focus the particle trajectories. The overall appearance of the particle tracks from conditioning sequentially without iteration on h data at times t=0, 6, 9, 36, 41, and 44 months, however, is very similar to conditioning sequentially without iteration only on h data at times t=0 and 44 months (Figure 5.30) which indicates that conditioning on h data at the intervening times of t=6, 9, 36, and 41 months has very little effect on transport predictions.

Computational problems with the larger matrix incorporating collateral conditioning on head data at times t=0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44 months forced the increase of the convergence criterion to an AME of 26.0 ft. Mean and variance of 67 simulations conditioned collaterally on head data at these times are shown in Figure 5.37a and 5.37b. The maximum mean f is 8.41 and the minimum is -5.18. Maximum f variance is 3.22 at location x=10800 ft y=30000 ft.

The most striking result of collateral conditioning on head data at all times of 0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44 months is the shortening of the distance the particles travel in 88 months. The lower mean f field in the vicinity of the MWL site also created a much steeper head gradient. Mean head from 67 simulations conditioned collaterally on head data at times t=0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44 months is shown in Figure 5.38. Maximum head variance from these simulations is 251 ft^2 at location x=7600 ft y=43600 ft. The mean head field obtain by collaterally conditioning 100 simulations, without iteration, on

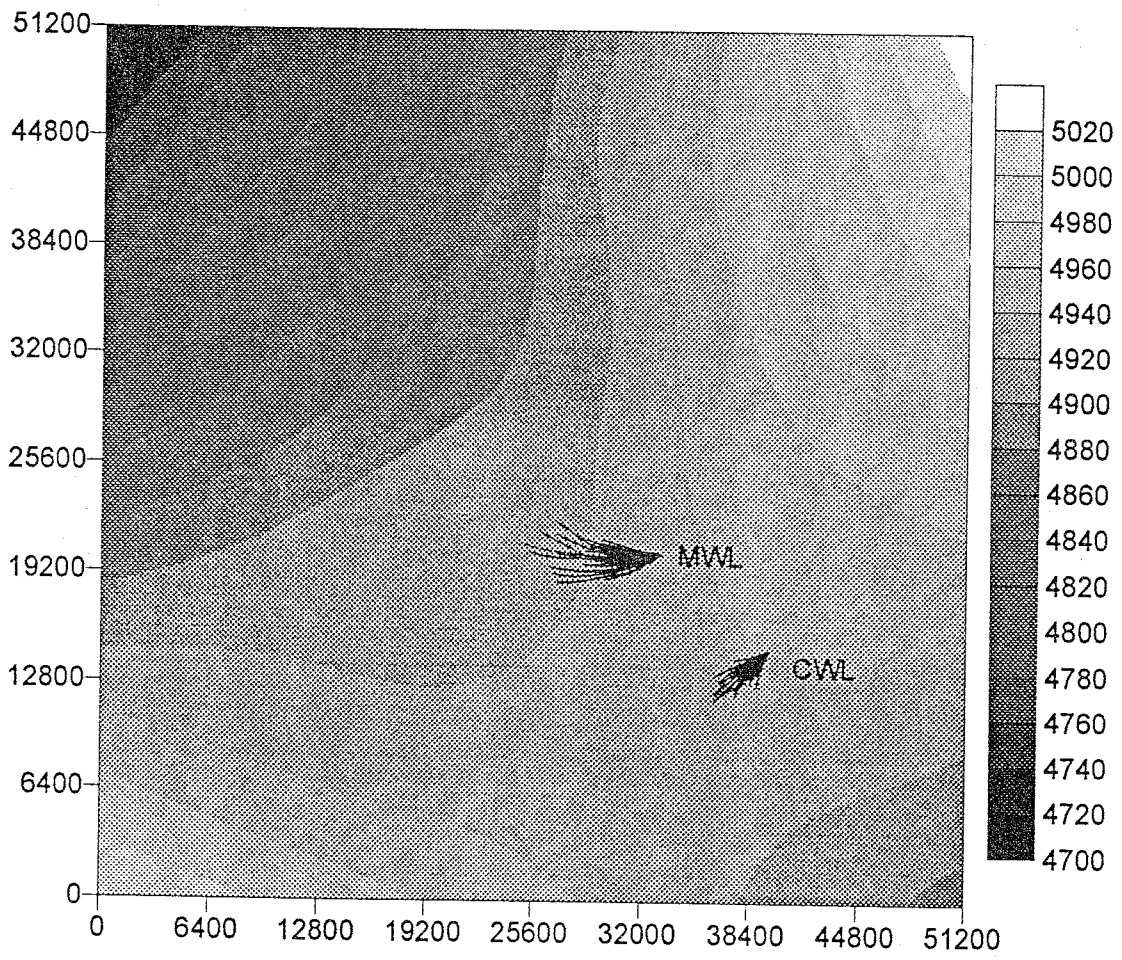


Figure 5.36: Average head field resulting from 100 simulations conditioned by the sequential method, without iteration, on f data and head data at times t=0, 6, 9, 36, 41, and 44 months. Particle tracks shown were released from the CWL and MWL sites.

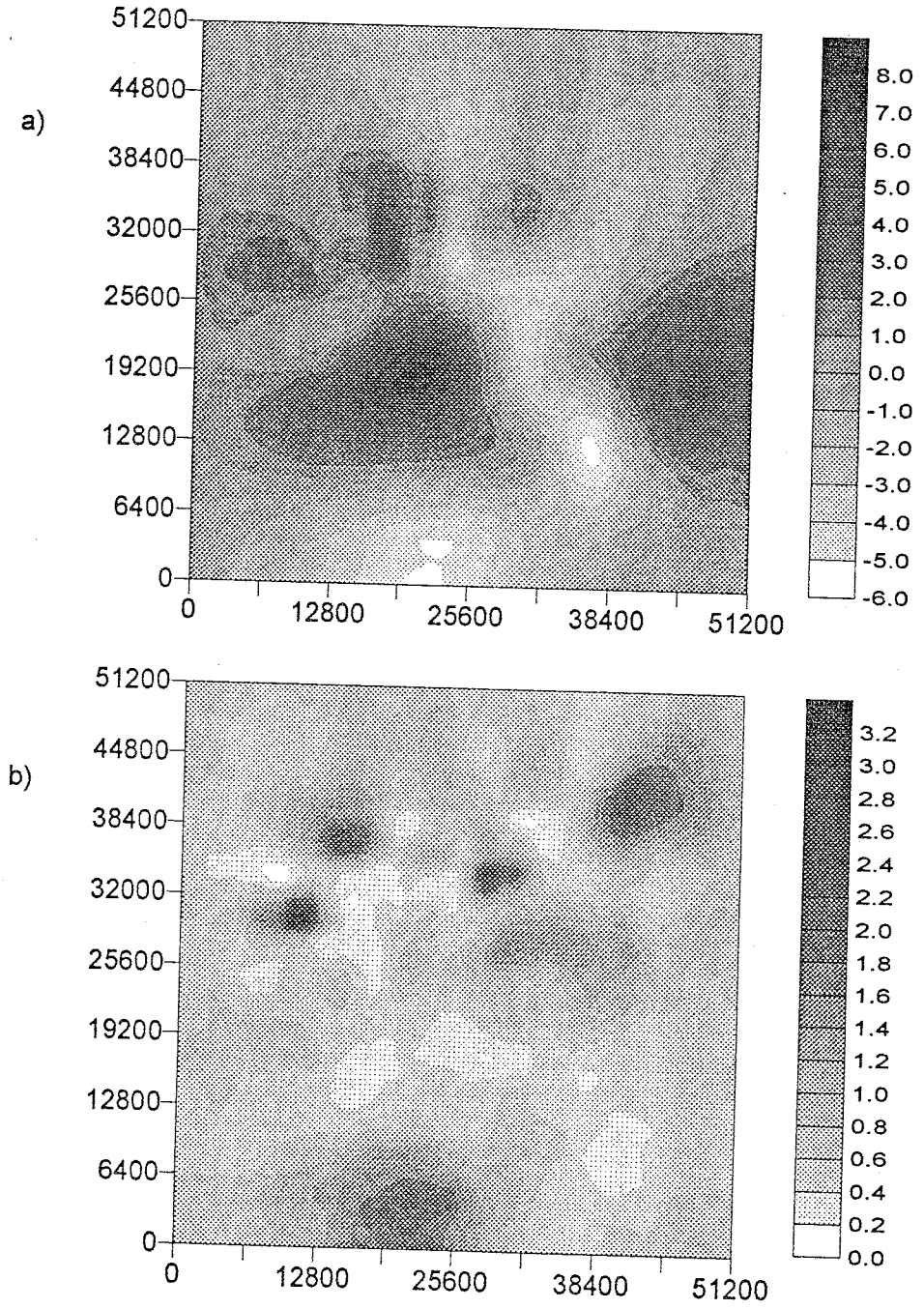


Figure 5.37: a) Ensemble average of 67 simulated f fields conditioned collaterally on f data and head data at times t=0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44 months to an AME criterion of less than 26.0 ft. b) Ensemble variance of 67 f fields conditioned collaterally on f data and head data at times t=0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44 months.

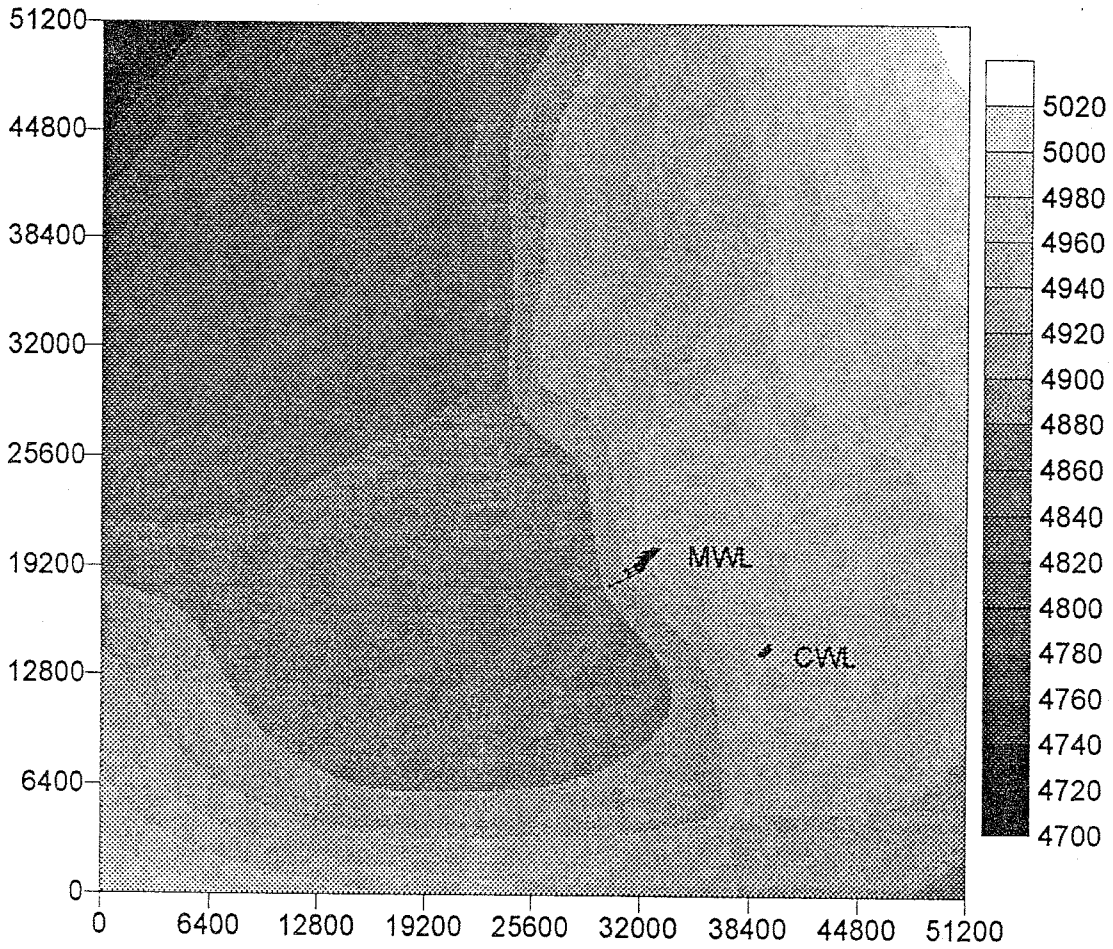


Figure 5.38: Average head field resulting from 67 simulations conditioned collaterally, with iteration to an AME criterion of 26.0 ft., on f data and head data at times t=0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44 months. Particle tracks shown were released from the CWL and MWL sites.

h data at times t=0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44 months is shown in Figure 5.39 along with tracks of particles released from the MWL and CWL sites. Comparing this to Figure 5.38, obtained using collateral conditioning on the same data with iteration, shows very little effect on transport predictions with iteration.

Shown in Figure 5.40a is the mean f field and in Figure 5.40b the f field variance resulting from 110 simulations conditioned on head data at times t=0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44 months by the sequential method. Maximum mean f is 3.56 and the minimum is -1.80. The maximum f variance is 1.48 at location x=22800 ft y=20400 ft. The smaller iteration convergence criterion of an AME of 14.0 ft was used.

Mean head resulting from 110 simulations conditioned by the sequential method on head data at times t=0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44 months is shown in Figure 5.41. The criterion for iteration convergence used was an AME less than 14.0 ft. A maximum head variance of 149.90 ft^2 occurs at location x=9200 ft y=36400 ft. Shown in Figure 5.42 are particle tracks and mean head from 100 simulations conditioned sequentially, with no iteration, using h data at times t=0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44 months. The map resulting from this effort is essentially indistinguishable from that obtained by sequentially conditioning without iteration on h data at times t=0, 6, 9, 36, 41, and 44 months (Figure 5.36). The obvious conclusion, repeated again, is that conditioning on the extra head data time intervals has provides no gain of information.

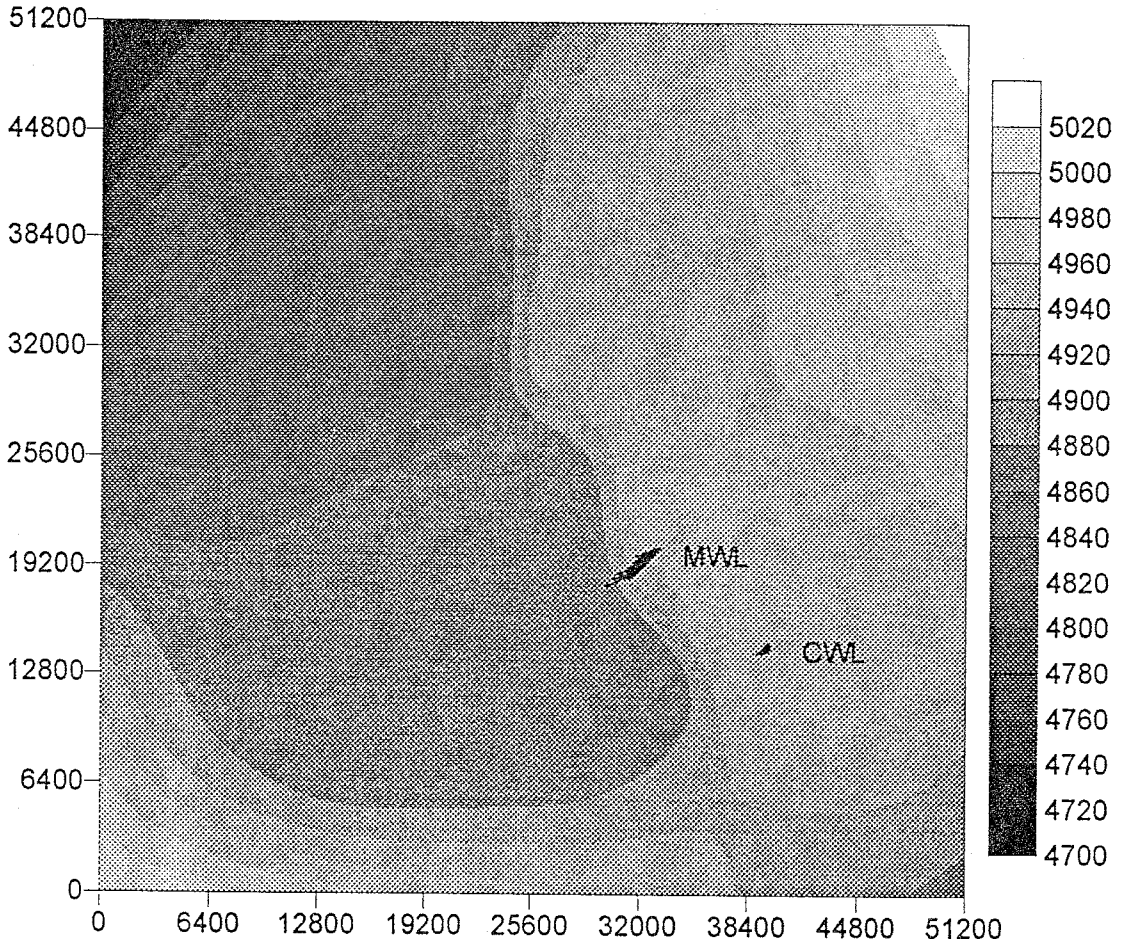


Figure 5.39: Average head field resulting from 100 simulations conditioned collaterally, without iteration, on f data and head data at times t=0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44 months. Particle tracks shown were released from the CWL and MWL sites.

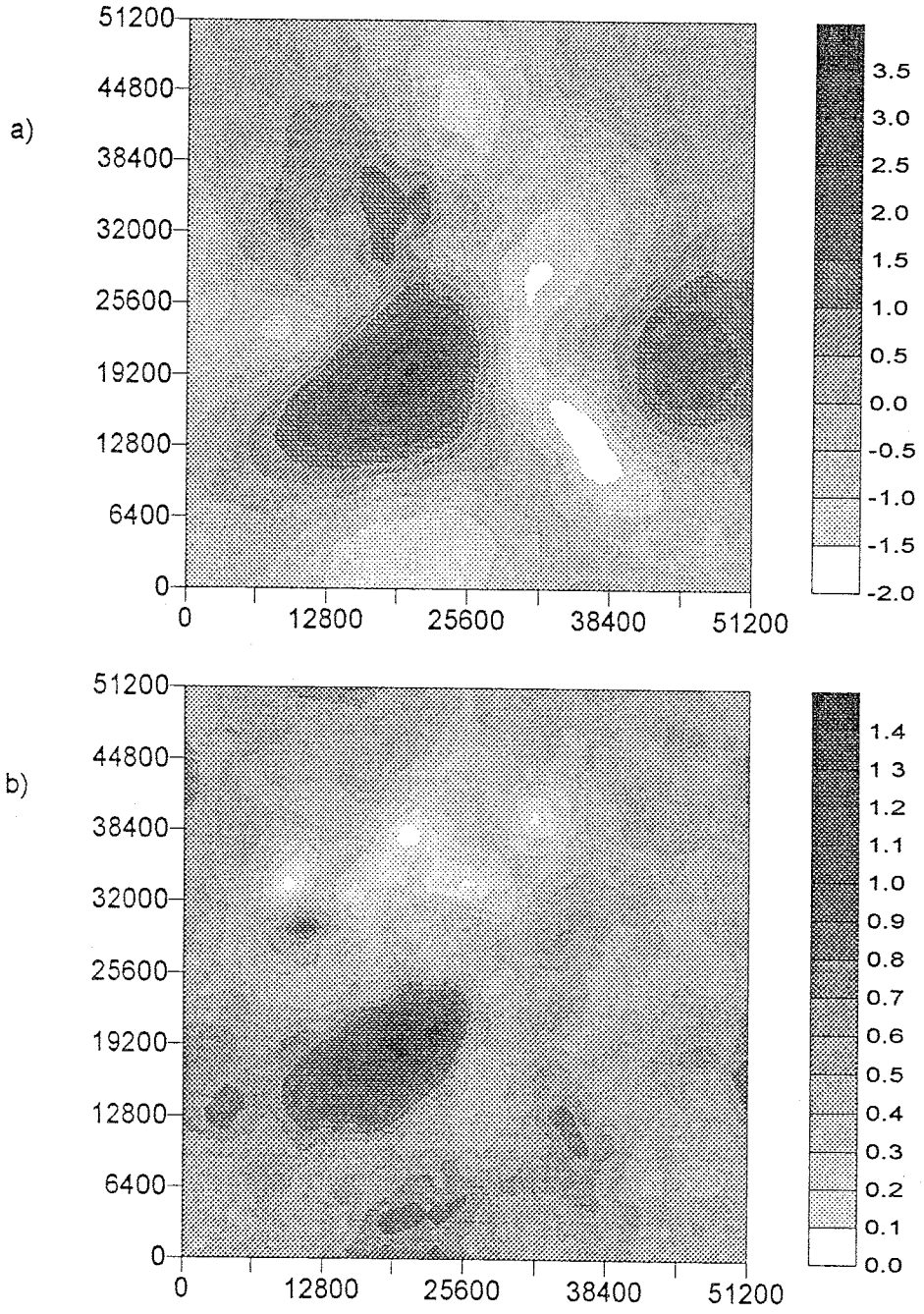


Figure 5.40: a) Ensemble average of 110 simulated f fields conditioned on f data and head data at times t=0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44 months to an AME criterion of 14.0 ft by the sequential method. b) Ensemble variance of 110 f fields conditioned by the sequential method on f data and head data at times t=0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44 months.

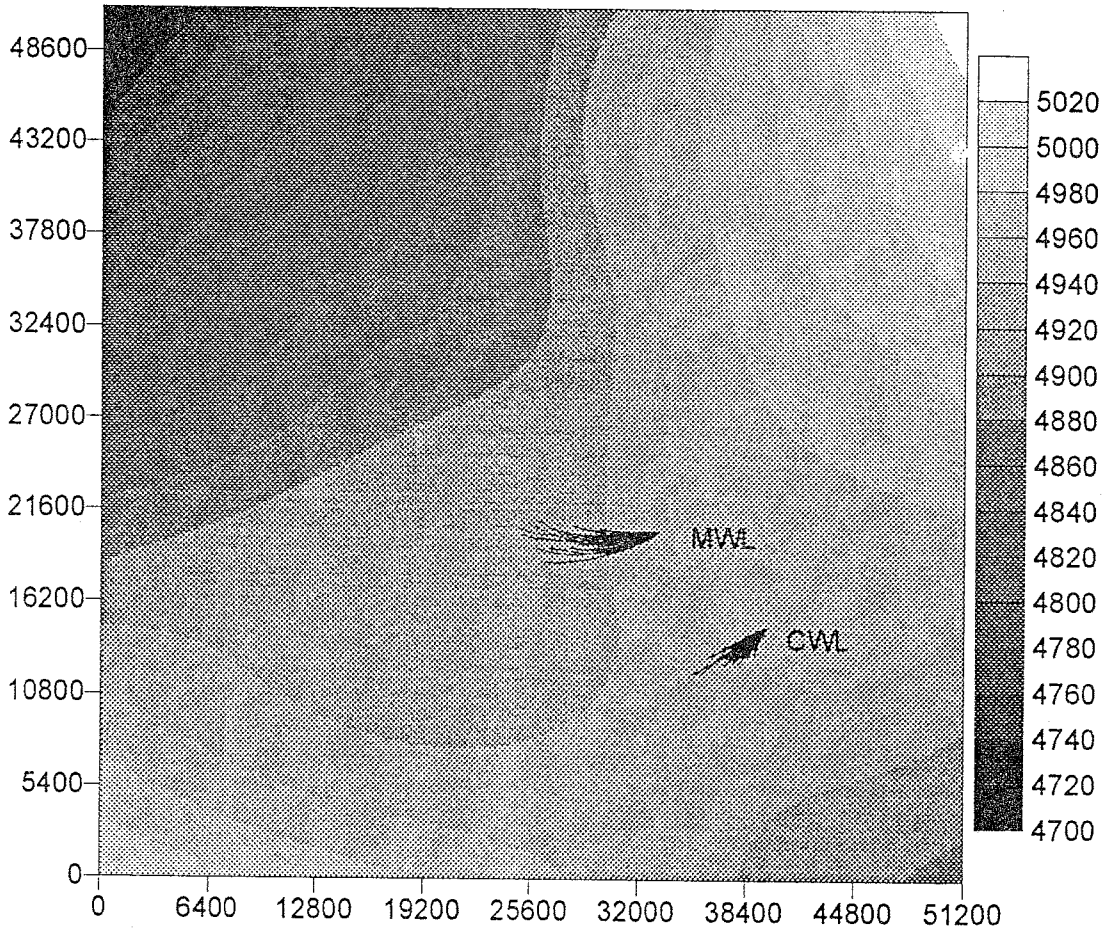


Figure 5.41: Average head field resulting from 110 simulations conditioned on f data and head data at times t=0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44 months to an AME criterion of less than 14.0 ft using the sequential method. Particle tracks shown were released from the CWL and MWL sites.

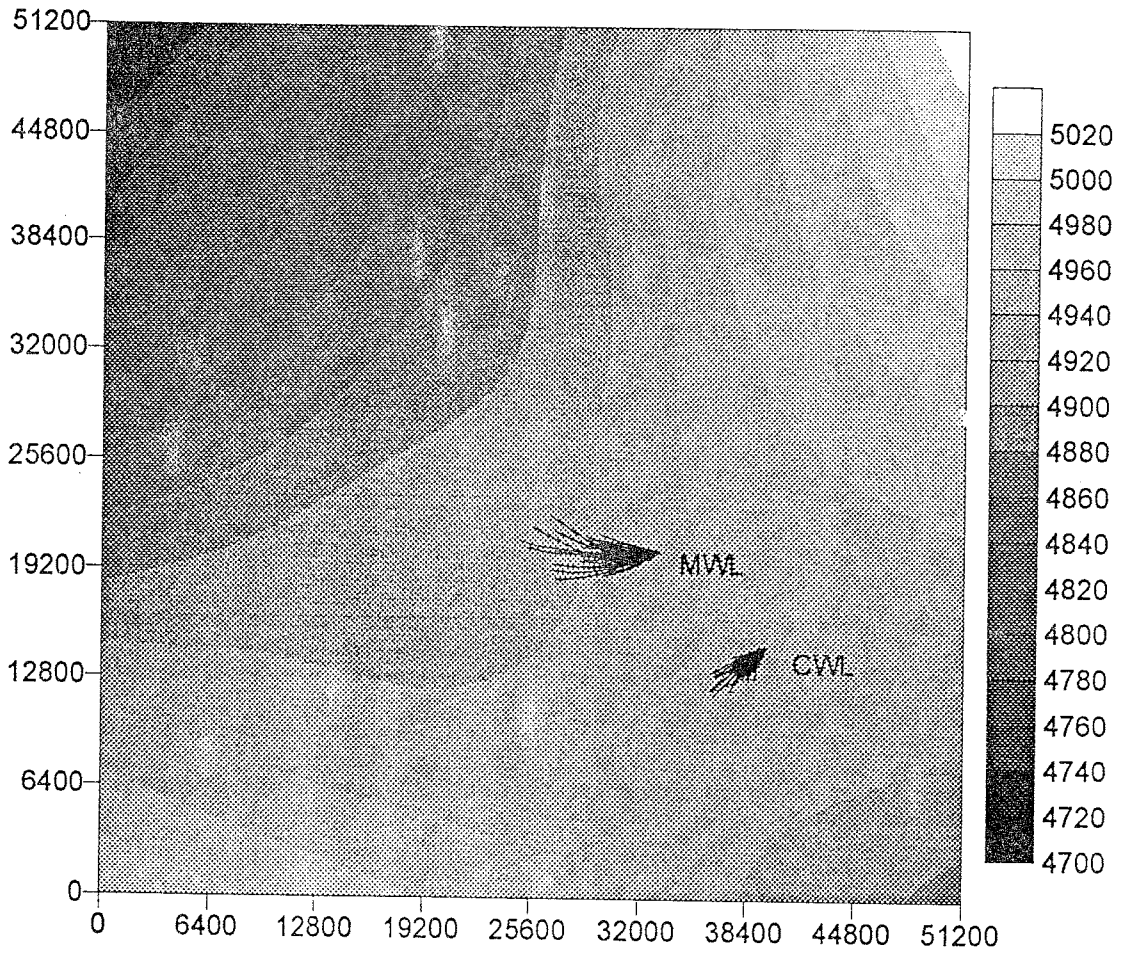


Figure 5.42: Average head field resulting from 100 simulations conditioned sequentially, without iteration, on f data and head data at times t=0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44. Particle tracks shown were released from the CWL and MWL sites.

A summary of the comparison measures: Maxf for maximum conditioned mean f field, Minf for minimum conditioned mean f field, and MVAR for maximum conditioned f or h variance is given in Table 5.4 and compared using bar graphs in Figures 5.43, and 5.44,. In the bar graph plots, data sets used for conditioning are represented by numbers as:

1. f data only,
2. f data and h data at times t=0 and 44 months,
3. f data and h data at times t=0, 6, 9, 36, 41, and 44 months, and
4. f data and h data at times t=0, 1, 2, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 36, 40, 41, 42, 43, and 44 months.

The letter C refers to collaterally conditioned simulations, S refers to sequentially conditioned simulations, and I indicates that the simulations have been iteratively conditioned to the AME criterion given in Table 5.4.

Based on the results obtained from conditioning on head data in the hypothetical reality exercise, section 5.6.1, I would expect collateral conditioning on head data at times t=0 and 4 months to significantly improve the estimation of both head and transmissivity fields accompanied by a reduction in estimated head variance and little change in estimated f variance over conditioning only on f data. Also I would expect iteration to slightly improve the estimation of the head field and, at the same time, slightly increase the estimated f variance. Except for the dramatic increase in estimated f variance with iteration, this is essentially what happened with collateral conditioning

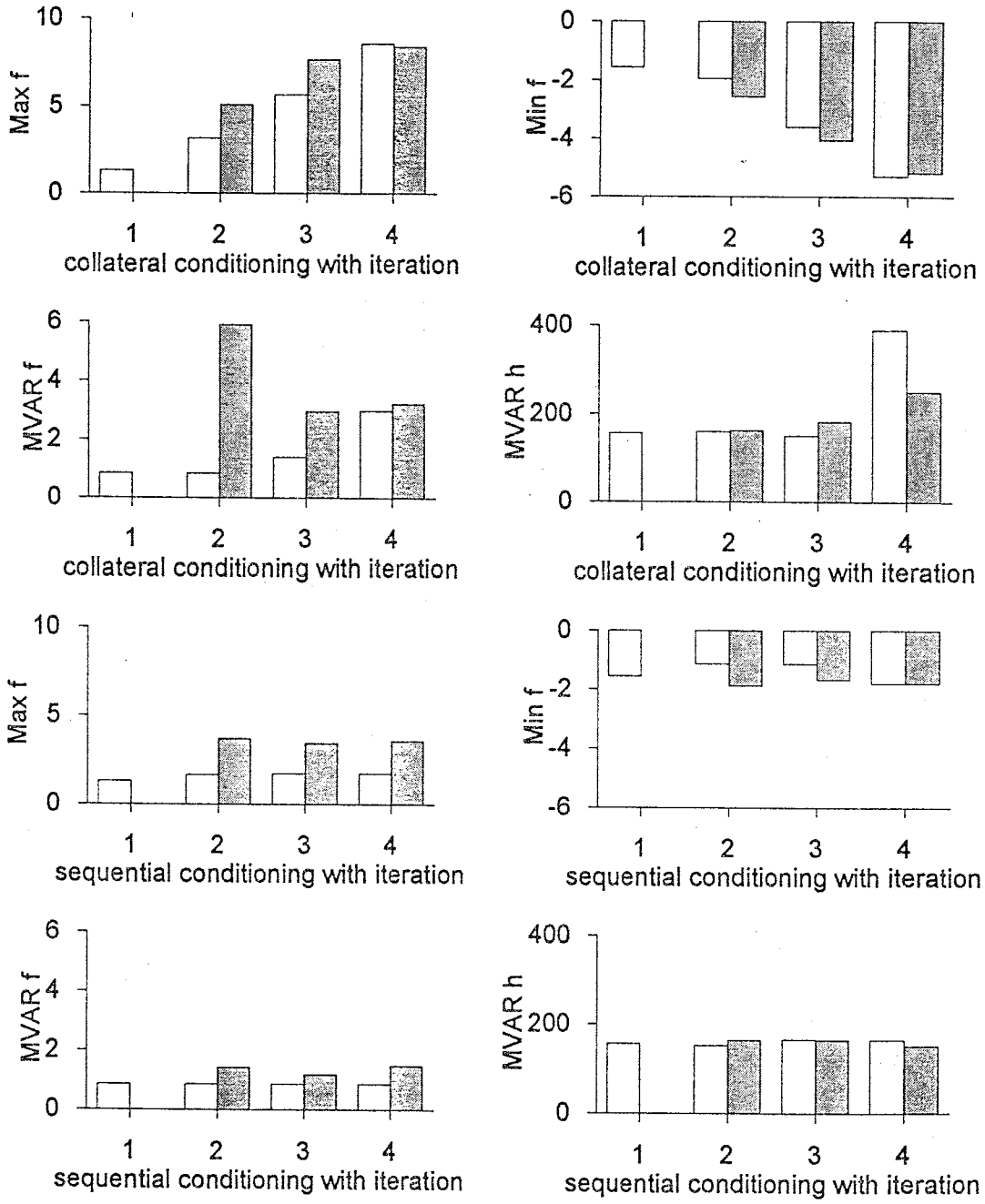


Figure 5.43: Comparison of the measures Maxf, Minf, and MVAR for simulations conditioned on the real KAFB data. The numbers 1, 2, 3, and 4 refer to the data set used for conditioning. White bars represent measures from simulations conditioned without iteration and filled bars represent measures from simulations conditioned with iteration.

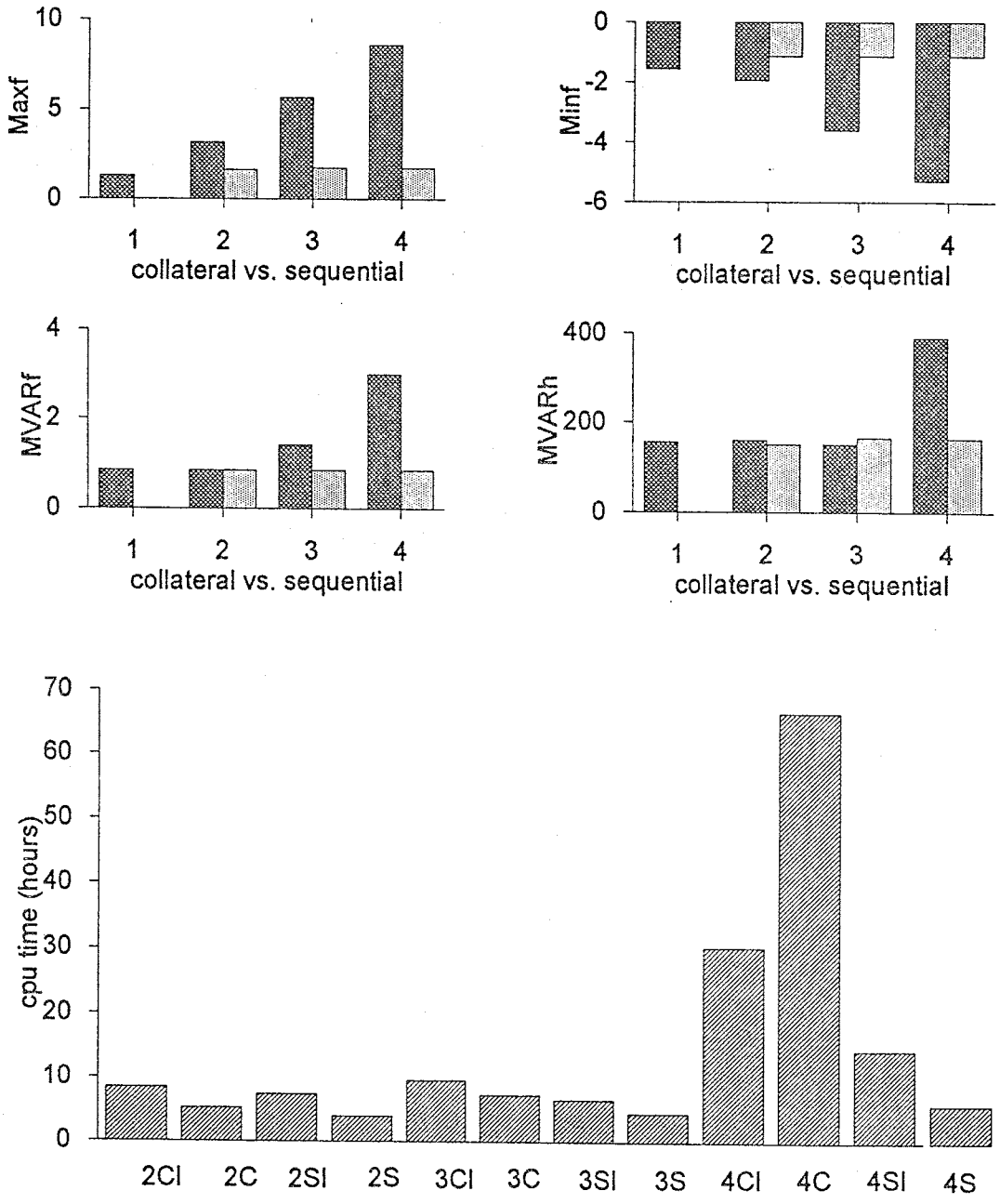


Figure 5.44: Comparison of the measures Maxf, Minf, and MVAR for simulations conditioned collaterally and sequentially on the real KAFB data. The numbers 1, 2, 3, and 4 refer to the data set used for conditioning. Darker filled bars represent measures from simulations conditioned collaterally without iteration and lighter filled bars represent measures from simulations conditioned sequentially without iteration. Lower graph shows cpu time in hours.

	CWL site			MWL site		
set	Maxf	Minf	MVARf	MVARh	M	AME
1	1.30	-1.55	.849	156	100	
2CI	5.09	-2.56	5.90	163	110	14.0
2C	3.18	-1.93	.850	161	100	∞
2SI	3.69	-1.88	1.42	164	110	14.0
2S	1.65	-1.13	.855	151	100	∞
3CI	7.58	-4.05	2.96	183	107	14.0
3C	5.66	-3.59	1.41	152	100	∞
3SI	3.43	-1.67	1.18	164	105	14.0
3S	1.72	-1.13	.853	165	100	∞
4CI	8.41	-5.18	3.22	251	67	26.0
4C	8.57	-5.30	2.99	390	100	∞
4SI	3.56	-1.80	1.48	150	110	14.0
4S	1.74	-1.14	.853	164	100	∞

Table 5.4: A summary of the comparison measures: Maxf for maximum conditioned mean f field, Minf for minimum conditioned mean f field, and MVAR for maximum conditioned f or h variance for simulations conditioned on the real KAFB data. M is the number of simulations, and AME is the iteration convergence criterion.

on the real h data at times t=0 and 44 months. Conditioning on head data provides significant added detail in spatial distribution of transmissivity over conditioning on just the f data alone, as seen in Figure 5.25 compared with Figure 5.23. Sequential conditioning on h data at times t=0 and 44 months should result in a slightly poorer, but fairly similar, estimation of the f and head fields as collateral conditioning on the same data. Iteration with either method would increase the extremes, the maximum and minimum, of the conditioned mean f field with a corresponding increase in the maximum estimated f variance and a decrease in the maximum estimated h variance.

One noteworthy consequence of the hypothetical data conditioning

exercise is that conditioning on more than a single head data interval using the collateral method can produce erroneous results and conditioning on more than a single head data interval using the sequential method increases computational effort without any substantial improvement in parameter estimation. Of course the transient induced by changing the boundaries through time with this KAFB model may account for the differences between the results of chapter 4 and the results seen here. Since the KAFB model is essentially in quasi-equilibrium with the boundaries, the quasi-steady state assumption of the sequential conditioning method is satisfied and there is no additional information to be gained by using the cross-time head covariances.

An interesting difference between the conditioning results of the real data versus the hypothetical data is that iterating to match the real head data does not decrease the maximum estimated h variance and produces a sharp increase in the maximum estimated f variance with both conditioning methods. This suggests to me a flaw in the conceptual model as it applies to the real groundwater flow system. A major assumption of this geostatistical inverse method is that all perturbations from the mean in head are due to heterogeneities in the transmissivity parameter. In the real natural system, perturbations in head could be due to localized recharge or discharge from sources unknown to the modeler. Iterating with this geostatistical inverse model to match the real head data, then, forces the transmissivity parameter to be modified more extremely in order to account for head variability which is not due to spatial heterogeneity of the transmissivity parameter.

A few observations can be made, at least qualitatively, from the re-

sults of these simulations. Neglecting the simulations which condition only on f data, no matter how many times of head data are used or whether the conditioning is done collaterally or by the sequential method, the head data strongly imply that the CWL and MWL sites are located in a low permeability zone and that a very high permeability zone occurs around x=20400 ft y=19600 ft. This pattern is consistently displayed in all simulations utilizing head data in the conditioning, the biggest difference between simulation methods being the relative magnitude of these zones. Another observation which can be made concerns the estimated boundary conditions. The procedure used to estimate boundary conditions resulted in a saddle-point in the vicinity of the CWL site. Results from the hypothetical data set showed all of the particles released from the CWL site traveling towards the bottom boundary. This occurred even though care was exercised in selecting a simulation as the hypothetical reality which closely resembled the pattern of heterogeneities displayed in the mean f fields conditioned on the real KAFB data. Head fields conditioned on the real KAFB data, however, seem indecisive as to the travel paths of particles released from the CWL site. Whether the particles generally are influenced to travel towards the left boundary or the bottom boundary depends on the head data used and the method utilized for conditioning. This implies some significant uncertainty as to the ultimate fate of contamination escaping from the CWL site. Finally, the observation can be made that a transient condition in the Albuquerque basin does appear to influence the destiny of contaminant transport at Kirtland Air Force Base. Even without a direct comparison of these results to the results of simulations performed by *McCord et al.* [1993], the uncertainty in predominantly southerly or westerly travel of the CWL con-

taminants due to the possible groundwater divide and the influence of time dependent head data on the ultimate outcome of this travel direction implies that the transient condition is an important factor.

Chapter 6

Discussion and Conclusions

This dissertation's primary purpose is to address the value of time dependent head data for conditioning probabilistic distributions of head, transmissivity, and velocity. A central issue of both forward and inverse models is identification of the correct conceptual model. This is a critical source of modeling uncertainty and remains beyond the scope of the geostatistical method presented here. The character of groundwater flow may vary from phreatic to confined, two-dimensional to three-dimensional, unsaturated to saturated within a region and, unless the model accounts for this, any amount of data has questionable value in conditioning model predictions. Conditioning on data may, conversely, lead to more instead of less erroneous predictions if the conceptual model fails to replicate important features of the natural system. For example: suppose leakage from a surface impoundment were causing mounding of the groundwater table and water levels recorded at a monitoring well in the vicinity were a part of the data set. The model (presented in this dissertation) lacking any other information indicating a source, would interpret a positive head anomaly as a low permeability zone. Thus a zone predicted by the model to have very low groundwater velocities could in fact have high velocities. Conditioning on head data with an incorrect conceptual model in this case leads further from the truth. No model, and certainly no inverse model,

can be treated as a "black box" solution to a natural flow system. Modeling is inherently an interactive process between the real system, measurements, the modeler, and the modeling results. All results and conclusions from the simulations in chapters 4 and 5 are thus presented with the caveat "given the conceptual model" and the recognition that methods of identifying the correct conceptual model should be the focus of future work.

Head covariances and cross-covariances between head and log transmissivity perturbations, and the method of numerical computation, are primary results of this research. An interesting side note is that all covariance computations were accomplished using a standard five point stencil block-centered finite difference numerical scheme. The cokriging data matrix and estimation vectors were calculated for a 4096 block grid over 440 time steps using less than one hour of an IBM RS/6000 CPU time. Key to this method is the calculation of covariances on a coarser grid than the mean or heterogeneous head and interpolation from the coarse grid back to the fine grid which is possible due to the smoothly varying covariances. The problem of scale-dependence, averaging of covariances over larger block sizes, may have been sidestepped by using the finer grid to estimate the divergence of the product of the input covariance and mean gradient load terms for the covariance equations. Changing the grid block size by a factor of two did not appear to significantly influence the covariance results (Figures 4.9e and 4.9f) nor did comparisons with covariances simulated on the fine grid show any errors which could be attributed to a scale-effect. The scaling question is certainly not resolved but these results indicate that, for the specific cases investigated here, scale-dependent covariance between the coarse and fine grids is not a major issue.

Covariance computations at the nodes of a grid more coarsely discretized than that used to calculate head and velocity is possible only when the mean gradient varies smoothly in space. Variation of head and gradient with time is related to the system time constant and may be captured by adjusting the time step. Errors induced by the coarse grid approximation are related to the change in gradient across the distance of 3 of the coarse grid blocks as a direct consequence of the central finite differencing used to approximate the covariance equation load terms. Extension of the method to account for a small number of concentrated sources or sinks may be possible by straightforward modification of the grid discretization in the vicinity of the source or sink. An interesting, and possibly productive, avenue to pursue in this direction would be to adapt the covariance grid during run-time to accommodate the head field gradient. Once the mean head field had been determined, the grid for covariance could be adjusted to concentrate covariance computations in regions were the gradient varies rapidly across small distances and space the covariance computations more widely where the gradient is changing slowly. In this manner wells, infiltration from surface impoundments, localized recharge, and other important small area sources and sinks could be accounted for by the model without compromising execution efficiency or memory storage requirements. Run-time efficiency may also be improved in the existing method by using a Crank-Nicolson time integration scheme to allow for larger time steps without inducing instability.

An adjoint method for computing time-dependent head covariances was proposed by *Townly and Wilson* [1984] and *Sun and Yeh* [1992]. No detailed comparison between this method and the method presented here is made

or is intended. However, the adjoint state equation method should at least be mentioned and a few comments made regarding computational efficiency. The adjoint state method is essentially a Green's function approach. For the grids and head data sets dealt with here, the size of the matrices needed to calculate the required covariances is quite large. Despite the fact that the adjoint state method requires only that sensitivities equal to the number of head measurements be computed, for problems involving large grids the coarse grid covariance approach presented here may be comparably efficient. In any case, the idea introduced here of using the smoothness of the covariances to reduce the size of these matrices may also be useful for decreasing the computational effort of the adjoint state method. An interesting study would be to compare run-times and results of time-dependent head covariance calculations using the methods of *Sun and Yeh* [1992], *Townly and Wilson* [1985], and the method presented here.

Intuitively the collateral cokriging matrix, equation (3.21), incorporating all cross-time head covariances and conditioning one f field simultaneously for all times would appear to extract more information from the time-dependent head data than the sequential conditioning covariance matrix, equation (3.19), where cross-time covariances are neglected and the conditioned f field changes with time. Operationally, however, the results are equivocal. Unquestionably, improvement in the estimation obtained by incorporating head data is significant with just a few head measurements but gains diminish quickly with even more additional head measurements for either method. In terms of head data and estimation accuracy, a little gains a lot but a lot more gains little. The same holds true for particle tracks in that the most noticeable ef-

fect is seen with conditioning on a single head data interval and conditioning on more head data intervals does not effect much change. For the collateral cokriging matrix, the tradeoff between estimation improvement obtained and computational effort expended reaches an optimum with only a very few additional times of head data. The size of the collateral cokriging matrix grows with the square of the number of head data and this matrix tends to become nearly singular. Small errors in the covariance matrix due to numerical approximation or coarse to fine grid interpolation can become magnified through the singular value matrix decomposition to large errors in the cokriged fields. The errors appear to become more severe for larger matrices, even with the addition of a stabilization term to the matrix diagonal.

Cokriging matrices used in the sequential conditioning method are never larger than the total number of data locations and suffer less from error magnification in inversion. Modifying the simulation by conditioning on head data at one or two additional times with the sequential method does improve the estimation of head and, slightly less, the f field. Conditioning on head data at every available time, however, just modifies the fields at each time to match the data at that time without improving the overall estimate since the cross-correlation between times is neglected. Using an f field sequentially conditioned on head data at all times to replicate data at the initial time is almost like starting over with a new unconditioned simulation.

Recognizing that the cross-time head covariances do make some difference in the effect of conditioning on head data through time, the question becomes at what time or few times does conditioning on head data provide

the most information. The answer is, of course, related to the nature of the transient and the time constant of the aquifer. By all performance measures examined here, head data at one time constant well into the transient provides more information than many times early in the transient. In the hypothetical test cases presented here, head data were available during a transient for more than one aquifer time constant. In the hypothetical test cases some of the better results were obtained by collaterally conditioning on only the initial time and head data at the last available time which was more than one aquifer time constant later. In both test cases, the initial head data was used and provided significant improvement in the estimation. Many questions remain to be answered here which could provide the subject of more simulations and possible some theoretical results. For example: if the driving force were periodic, how often should water levels be sampled? If only quasi-steady state measurements were available at the beginning and end of a transient, does the cross-time covariance play a significant role in conditioning the estimates? Is any information lost if the final state is known but the initial state is not? The simulations presented here, while leaving much unanswered, do point towards a few tentative conclusions.

- Conditioning on transient head data does improve the estimation of transmissivity, head, and advective transport.
- Transient head data has more value in improving the estimation if it is collected at a later rather than earlier time in a transient.
- Cross-time head covariances have some importance in extracting information from head data in time, depending on the nature of the transient

condition.

- There is an optimal amount of head data beyond which computational problems overwhelm any benefits obtained by conditioning on additional time intervals of head data.

- Collateral conditioning on transient head data shows the strongest influence in constraining advective transport to the narrowest pathway. While sequential conditioning may show comparable behavior in other performance measures, it allows particles more leeway in their direction of travel.

- Of the performance measures examined, particle pathlines appear to be the most sensitive to head data conditioning.

Information extracted from transient head data was found in this work to be dependent on just how well conditioning was able to match the actual head data values. An iterative technique introduced by *Gutjahr et al.* [1994] and extended here to transient flow was used to modify the conditioned f fields through the cokriging matrix so that the errors between the flow model results and the head data became smaller with successive iterations. Computationally this required solving the cokriging equations and the flow equation at each iteration. A similar iterative technique was developed by *Yeh et al.* [1996] which modified not only the cokriged fields at each iteration but also the covariance and cross-covariances between iterations. To accomplish this *Yeh et al.* [1996] relied upon an adjoint state sensitivity analysis similar to that of *Sun and Yeh* [1992]. The advantage of this is that the cokriged f approaches

the true conditional f field as higher order terms neglected by linearization are regained through iteration. The disadvantage, especially for transient problems with a long time duration, is the effort required to re-solve the adjoint state sensitivity at each iteration. The field size used by *Yeh et al.* [1996] had only 231 nodes yet a single steady state simulation required 30 minutes on a CPU identical to the ones used here. For the method presented here, steady state simulations on a grid of 4096 nodes required about 4 minutes per simulation. The method presented here is focused more towards conditioning a single random simulation on head and transmissivity data rather than obtaining a covariance matrix which replicates the conditional mean fields. Even though more accurate cokriged fields have been obtained by *Yeh et al.* [1996] with a single simulation, it may desirable to generate many simulations in order to observe variability in the velocity fields and advective transport. Details aside, the iterative method of *Yeh et al.* [1996] and the technique of *Gutjahr et al.* [1994], applied here in the time domain, agree that information extracted from head data is dependent on the accuracy of conditioning.

6.1 Some conclusions regarding KAFB

The problem of contaminant transport from war-era waste sites located on the Kirtland Air Force Base around Sandia National Laboratories is a difficult challenge for any modeling study. Solvents and hazardous chemicals, disposed of in unlined pits during the 1950's, have formed contaminant plumes in an aquifer overlain by up to a 500 ft thick vadose zone. The water supply for the City of Albuquerque is derived from wells scattered throughout the municipality and water levels in the area have been steadily declining since

the 1960's [*Kernodle et al.*, 1995]. The KAFB site lies across a heavily faulted region at the edge of a rift valley with north-south trending structural features and stratigraphic offset from the east to west. Significant localized ephemeral recharge may occur in the Tijeras drainage which crosses the site from northeast to southwest and an unknown quantity of leakage from surface impoundments may infiltrate through stages of perched water tables to the saturated zone. Uncertainty even exists as to whether the aquifer is predominately phreatic or confined. The basin-scale model of *Kernodle et al.* [1995] used 11 layers extending to a depth of 1730 ft. *Kernodle et al.* [1995] utilized the Sandia and Tijeras faults running through the KAFB site as an easterly boundary of their basin-scale model and estimated the flux crossing this boundary at 14.68 ft^3/sec. In view of this complexity the two-dimensional saturated flow model used here appears quite inadequate even though it was designed to deal with more complexity than existing stochastic models.

Previous stochastic modeling of contaminant transport at the KAFB site was conducted under the assumption that the system is in steady state and that mean gradient is constant [*McCord et al.*, 1993]. The main objective in the development of this method was to relax those restrictions in order to allow for unsteady flow and a spatially variable mean head gradient. The more sophisticated model could then be used to address the questions: is the transient a significant factor in contaminant transport and is a constant mean gradient an adequate conceptual model? Natural flow in the region trends in a southwesterly direction towards the Rio Grande, however, water supply development has reversed the gradient over part of the region in a northwesterly direction towards the City of Albuquerque wells. Thus the ultimate destination

of KAFB contaminants, the City of Albuquerque or the Rio Grande, is also uncertain. Regional transients and variable mean gradients were accommodated through the use of model boundary conditions which imposed the requirement that these boundary conditions be estimated. The method used to estimate boundary conditions, a combination of a quasi-steady state assumption and a weighted least squares optimization, is essentially a deterministic procedure yet it forms a major component of the conceptual model. Within this conceptual model, the only stochastic component is a randomly heterogeneous spatially correlated transmissivity. Observations and conclusions based on the modeling results thus stem from both the deterministic and stochastic components of the model.

The effect of conditioning on head data consistently results in the same pattern of heterogeneities underlying the KAFB site for all sets of head data and collateral or sequential methods. The marked difference between methods and head data sets occurs primarily in the magnitude of the heterogeneities. Thus conditioning on head data within this conceptual model implies that the CWL and MWL sites are located in a zone of low permeability. Furthermore, advective transport from the MWL site appears to be in the direction of a zone of high permeability. From a more deterministic approach, estimation of model boundary conditions by weighted least squares optimization on head data resulted in a stagnation zone and groundwater divide near the CWL site. While this tends to result in slower advective transport away from the CWL site it also introduces significant uncertainty as to whether transport is generally in a southerly or westerly direction. Collateral conditioning on head data using the correlation in time has the effect of forcing transport more towards the

westerly direction whereas conditioning under a quasi-steady state assumption using the sequential method allows more of the advecting particles to move in a southerly direction. The appearance of a groundwater divide as a result of deterministic boundary condition estimation and the effect of head data conditioning on the location of this divide lead to the observation that transient flow and a variable mean gradient are important aspects in the uncertainty of flow and transport at the KAFB site.

6.2 Research opportunities

Many unsolved problems and unanswered questions in the area of inverse modeling remain the topic of future research. Some of the questions raised earlier are summarized here.

- Developing a conceptual model is often an *ad hoc* interaction between the modeler, the data, and the real system. More rigorous methods of identifying an appropriate conceptual model or choosing the most applicable conceptual model are needed.

- Perhaps more importantly, uncertainty inherent in a conceptual model's simplifications and approximations needs to be quantified.

- Hierarchical, or nested models, which are often used deterministically to relate a modeled region to the larger domain, could be applied stochastically to infer boundary covariances and relate uncertainty in a larger flow regime to the area of interest.

- The influence of second-type boundaries on data-conditioned probabilities needs to be investigated. Random perturbations in boundary fluxes may propagate, affecting the covariances everywhere within the model domain.

- Perturbations in head due to parameter heterogeneity need to be distinguished from head perturbations due to localized sources or sinks. Uncertainty due to spatial and temporal random variability of recharge and localized sources and sinks needs to be accounted for in the stochastic model.

- The domain decomposition technique could be improved to accommodate more spatial variability in the mean head gradient.

- Uncertainty in model predictions due to uncertainty in the parameter covariance functions could be examined.

While modeling may be primarily an exercise in understanding a natural system, as long as models are used to make predictions those predictions require some measure of their inherent uncertainty. A true measure of uncertainty in model predictions is an accumulation of all sources of uncertainty and not merely those due to spatial heterogeneity of the model parameters. Research on data-conditioned probability should be posed as a data evaluation methodology within a framework designed towards accounting for all sources of uncertainty.

6.3 Final remarks

Stochastic concepts and methods for dealing with heterogeneity and uncertainty in subsurface models have been in existence for about 20 years as of the date on this manuscript. Textbooks have been written on the subject. But rarely does one see applications of these techniques to real world problems outside of a research context. How long does it usually take for a useful new method to migrate from the realm of research to regular use in a consulting practice? With workstations and high-end personal computers on nearly everyone's desk, excessive computational effort is no longer an excuse. While it is possible that stochastic modeling may have no practical use, the substantial research effort expended on stochastic model development is directed at addressing real world problems. Why then are applications of stochastic methods so much rarer than theoretical developments and common only in Nuclear Waste Repository investigations? What other field of science is as closely linked to legislation and regulation? Is any other science as dominated by civil court actions as subsurface hydrology? Perhaps stochastic models will not become standard operating procedure until regulators expect them, and only then if they survive litigation.

Appendix A

Algorithm Implementation in Fortran

This appendix contains a listing of Fortran code used to generate simulations, solve for head and covariances, and iteratively condition simulations by cokriging both collaterally and sequentially. But first a footnote about the block-centered finite difference implementation referred to in chapter 3.

A.1 Implementation of first-type boundary conditions

The second derivative finite difference operator is given in *Celia and Gray* [1992] as

$$\left.\frac{d^2u}{dx^2}\right|_{x_i} \simeq \frac{\frac{u_{i+1}-u_i}{x_{i+1}-x_i} - \frac{u_i-u_{i-1}}{x_i-x_{i-1}}}{x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}} \tag{A.1}$$

where $x_{i\pm\frac{1}{2}}$ represents the x-location of the midpoint between x_i and $x_{i\pm1}$. Following this example, I approximated the term $\frac{\partial}{\partial y}\left(T\frac{\partial \phi}{\partial y}\right)$ at the bottom boundary shown in Figure 3.2 for constant head, c, at the boundary as

$$\frac{T_u \frac{\phi_{i,j+1}-\phi_{i,j}}{\Delta y} - T_{i,j}\frac{\phi_{i,j}-c}{\frac{\Delta y}{2}}}{\frac{3\Delta y}{4}} \tag{A.2}$$

where $x_{i,j+\frac{1}{2}}$, the location midway between **i,j** and **i,j+1** (Figure 3.2), minus $x_{i,j-\frac{1}{2}}$, the location at the midpoint between **i,j** and c on the boundary, is $\frac{3}{4}\Delta y$. Developing this finite difference expression from the fundamental water

balance, however [1],

$$\Delta x \Delta y S \frac{\partial \phi}{\partial t} = B\left(\Delta y\left(q_{x_{i-\frac{1}{2},j}} - q_{x_{i+\frac{1}{2},j}}\right) + \Delta x\left(q_{x_{i,j-\frac{1}{2}}} - q_{x_{i,j+\frac{1}{2}}}\right)\right) \qquad \text{(A.3)}$$

where B is the aquifer thickness and q_x, q_y are the side of block fluxes, results in an expression

$$S\frac{\phi_{i,j}^k - \phi_{i,j}^{k-1}}{\Delta t} = \left(\frac{T_r\frac{\phi_{i+1,j}^k - \phi_{i,j}^k}{\Delta x} - T_l\frac{\phi_{i,j}^k - \phi_{i-1,j}^k}{\Delta x}}{\Delta x} + \frac{T_u\frac{\phi_{i,j+1}^k - \phi_{i,j}^k}{\Delta y} - T_d\frac{\phi_{i,j}^k - \phi_{i,j-1}^k}{\Delta y}}{\Delta y}\right) \qquad \text{(A.4)}$$

for blocks adjacent to the boundary, with similar expressions for other blocks along other boundaries. The error in equation A.2 results, in this case, from the denominator, Δy, being replaced by $\frac{3}{4}\Delta y$. The magnitude of this error was investigated by comparing results from the block-centered finite difference approximation with analytical solutions for steady state and transient flow under uniform recharge between constant head boundaries. The geometry for this comparison is that described in section 4.1.1.

In Figure A.1 the block-centered finite difference approximation is compared with the analytical solution,

$$\phi(y) = -\frac{RY^2}{2T} + \left(\frac{h_2 - h_1}{L} + \frac{RL}{2T}\right)y + H_1 \qquad \text{(A.5)}$$

for steady state flow between constant head boundaries $h(0) = h_1$ and $h(L) = h_2$ with uniform recharge, R. For parameters $L = 6400m$, $T = 8000m^2/year$, $R = .025m/year$, $h_1 = 0$, $h_2 = 10m$ and $\Delta y = 100m$ (64 blocks) the maximum error between the numerical approximation and the analytical solution is

[1] as was pointed out by John Wilson during my defense

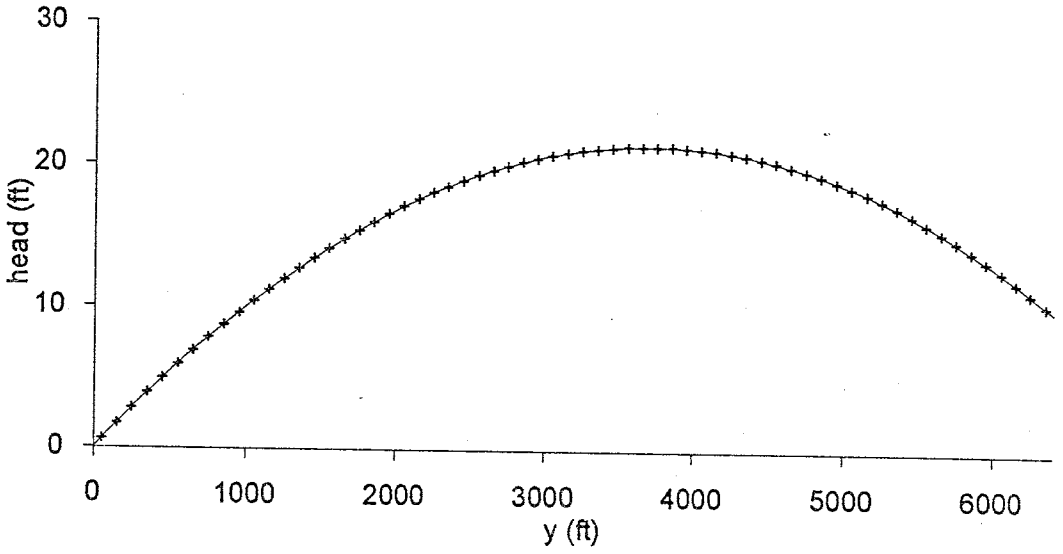


Figure A.1: A comparison of the block-centered finite difference approximation (+ symbols) to an analytical solution (solid line) for steady state.

-2.34×10^{-6}. Changing the block size to $\Delta y = 200m$ (32 blocks) resulted in a maximum error of -2.64×10^{-6} and to a block size of $\Delta y = 400m$ (16 blocks) resulted in a maximum error of -7.95×10^{-7}.

The finite difference approximation and an analytical solution for one dimensional transient flow between constant head boundaries due to a step change in uniform recharge,

$$\phi(y,t) = \frac{2L^2R}{T} \sum_{m=1}^{\infty} sin\left(\frac{m\pi y}{L}\right) \frac{(1-(-1)^m)}{(m\pi)^3} \left(1 - exp\left(\frac{-\lambda m^2 \pi^2 t}{L^2}\right)\right) \quad \text{(A.6)}$$

are compared at a block adjacent to the boundary in Figure A.2 and at a block in the center of the domain in Figure A.3 for a grid discretization of 64 blocks. Parameters used for the solution to transient flow are the same as used for the steady state solution with the additional parameter, storativity, taken as $S = 0.2$. The initial condition is taken to be steady state in equilibrium with

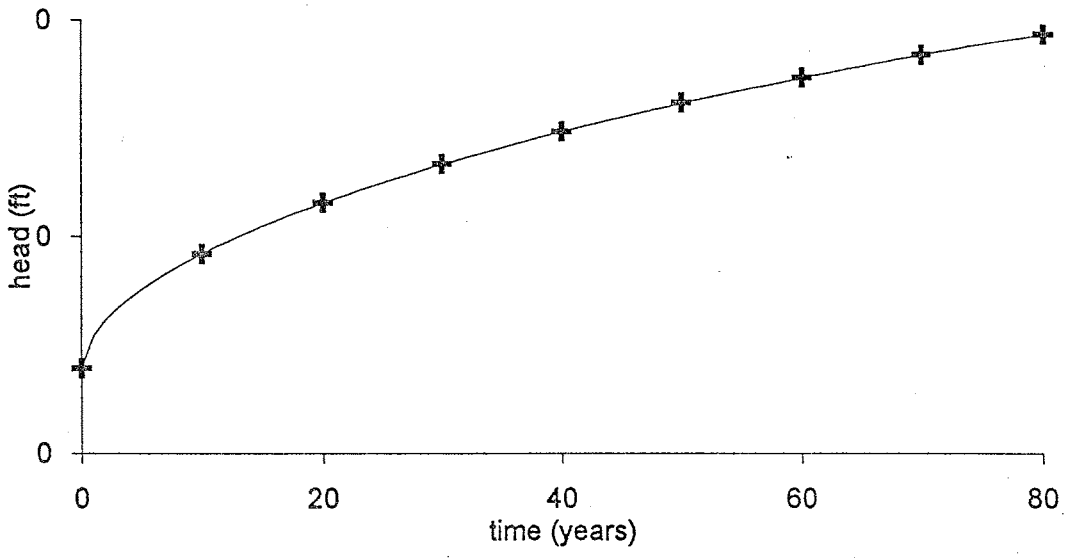


Figure A.2: A comparison of the block-centered finite difference approximation (+ symbols) to an analytical solution (solid line) for a transient state at a block adjacent to the constant head boundary.

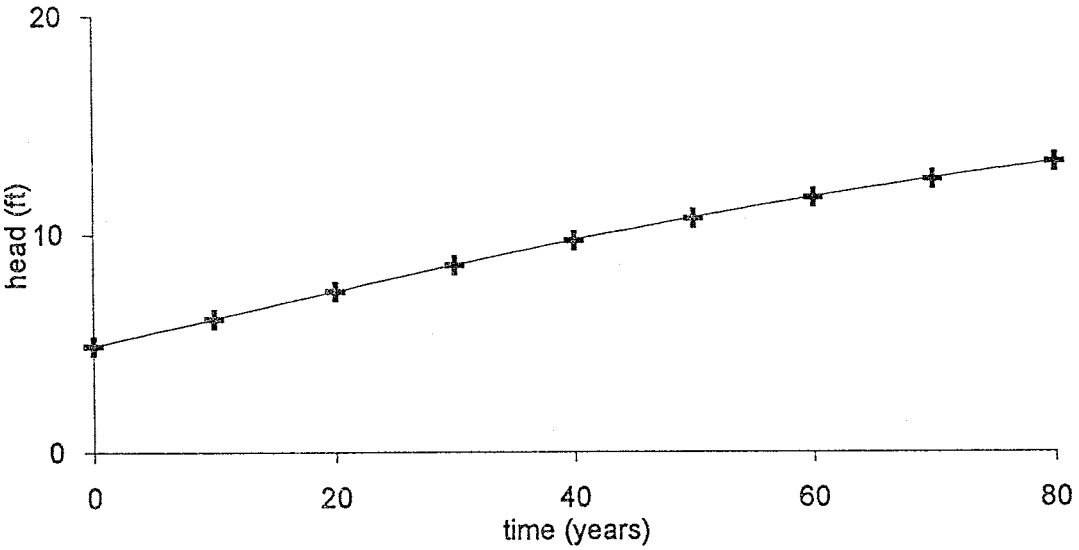


Figure A.3: A comparison of the block-centered finite difference approximation (+ symbols) to an analytical solution (solid line) for a transient state at a block in the center of the domain.

the boundary conditions and no recharge. The infinite series in equation A.6 was approximated by 50 terms. The largest error, analytical solution minus finite difference approximation, in Figure A.2 is -5×10^{-4} and the largest error in Figure A.3 is 0.03.

This then is the magnitude of the error which can be expected in the finite difference approximate solutions to head for the results presented in chapter 4. I don't believe that error of this magnitude will significantly effect any conclusions based on those results.

A.2 Interpolation of covariances from coarse to fine grid

Interpolation of smoothly varying covariances from the coarse to fine numerical grid is key to this geostatistical inverse algorithm. In this section I describe in detail the interpolation of a single cross-covariance between $f(d')$, log transmissivity perturbation at location d' with $h(d)$, head perturbation at location d. Shown schematically in Figure A.4 are the grid blocks surrounding the head perturbation datum, h_d, and grid blocks surrounding the log transmissivity perturbation, $f_{d'}$. Coarse grid blocks are numbered 1, 2, ... 9 around the h_d datum and 1', 2', ... 9' around the $f_{d'}$ datum for reference. Cross-covariances of log transmissivity perturbations at block centers 1', 2', ... 9' with head perturbations at block centers 1, 2, ... 9 are calculated, by finite difference numerical approximation, for a total of 81 cross-covariance block pairs as described in chapter 3. An interpolation of the cross-covariance of $f(d')$ with $h(d)$ is obtained then, by a sequence of one-dimensional quadratic interpolations using the routine, polint, from Numerical Recipes [*Press et al.*, 1992]. Referring to Figure A.4, a quadratic interpolation between cross-covariances

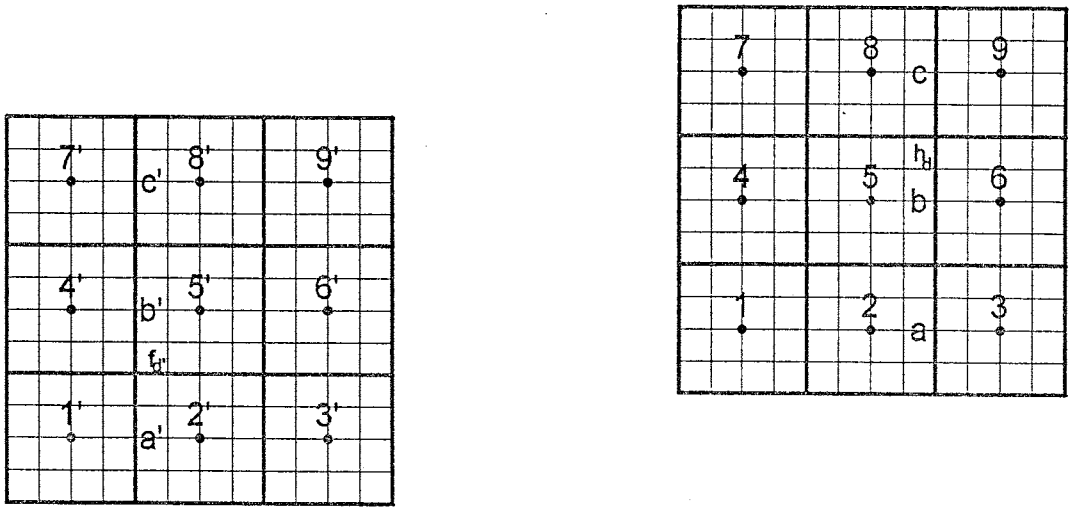


Figure A.4: Schematic of a portion of the numerical grid surround a head perturbation, h_d, and a log transmissivity perturbation, f_d.

$R_{1'1}$, $R_{1'2}$, and $R_{1'3}$ gives an estimate of the cross-covariance, $R_{1'a}$, of block center $1'$ with location a. Likewise a quadratic interpolation between the cross-covariances $R_{1'4}$, $R_{1'5}$, and $R_{1'6}$ gives an estimate of the cross-covariance, $R_{1'b}$, of block center $1'$ with location b. A third quadratic interpolation using $R_{1'7}$, $R_{1'8}$, and $R_{1'9}$ gives an estimate of the cross-covariance, $R_{1'c}$, of block center $1'$ with location c. Now a quadratic interpolation using $R_{1'a}$, $R_{1'b}$, and $R_{1'd}$ gives an estimate of the cross-covariance, $R_{1',d}$, of block 1' with the head perturbation datum location, d. This sequence is repeated 8 more times for blocks 2', 3', ... 9' resulting in estimates of the cross-covariance of all block centers around $f_{d'}$ with the location of h_d. Next the quadratic interpolation is done using $R_{1'd}$, $R_{2'd}$, and $R_{3'd}$ to give an estimate of the cross-covariance $R_{a'd}$. Likewise the interpolation using $R_{4'd}$, $R_{5'd}$, and $R_{6'd}$ gives an estimate of the cross-covariance $R_{b'd}$. An estimate of the cross-covariance $R_{c'd}$ is obtained by intepolation using

$R_{7'd}$, $R_{8'd}$, and $R_{9'd}$ and, finally, an interpolation using $R_{a'd}$, $R_{b'd}$, and $R_{c'd}$ gives the sought estimate $R_{d'd}$, cross-covariance of $f(d')$ with $h(d)$.

Estimates of all covariances, interpolated from the coarse to fine grid, are obtained in a similar manner as the cross-covariance described in this example except at the domain edges where bilinear interpolation is used instead of quadratic.

A.3 Fortran code

Fortran code for collateral and sequential conditioning is listed in this appendix. Subroutines and functions from Numerical Recipes [*Press et al.*, 1992] which were used include dsvdcmp and dsvbksb, singular value matrix decomposition, linbcg, a pre-conditioned conjugate gradient algorithm, bessk0 and bessk1, Bessel's functions of the third type order 0 and 1, ran2, a random number generator, fourn, a Fast Fourier Transform routine, and polint, a polynomial interpolation routine. These routines are listed in *Press et al.* [1992] and are not included here.

A.3.1 Main for calculating covariances used in conditioning

```
* PROGRAM LHAMANA simult
      parameter(maxit=2000)
      implicit double precision (a-h,o-z)
      include 'lhamana.inc'
      character bound(4),outc(4),pert(2),newbound(4),track
      integer i,j,k,kk,n,nxny,iter,nts,itmax,load,kappa,ntimes,ntrans
      integer ija(max*(5*max-2)),nxa,nya,mpr1,mpr2,jack,iot,int
      real*8 b(max*max),x(max*max),sa(max*(5*max-2)),tol,err
      real*8 stor,tran,hbound(max,4),dxa,dya,sdr,dt,ss,xo,yo
      real*8 tsm,cts,ttime,datmat(2,ndd,ndd),cfhvect(max*max,ndd)
      real*8 Hallk(max,max,0:maxnts),Htouse(max,max),ttime2
      real*8 newhbound(max,4),temp(max*max),hdtimes(0:maxntimes)
      real*8 startploc(2),tracktime,voids,thick
* get all the necessary parameters
```

```
      call inputpar(bound,modn,nsd,tol,tolc,nsims,stor
     c             ,tran,outc,sdr,nts,jack,tsm,dt,load,pert
     c             ,newbound,ntimes,hdtimes
     c             ,track,startploc,tracktime,voids,thick,ntrans)
      dxa=dx*real(nob)
      dya=dy*real(nob)
      nxa=nx/nob
      nya=ny/nob
      ss=0.
      nxny=nx*ny
      nd=nhdata
      itmax=maxit
* CALCULATE COVARIANCES required for cokriging conditioning
* output in file cokcov.out
* File cokcov.out for cokriging covariances to be input to
* simulation conditioning routine.
      open(unit=27,file='cokcov.in')
      if(outc(2).eq.'y') open(unit=77,file='hmean.out')
      call initialize
* Initialize sets qwells, Rfh, and Rhh arrays to zero.
* Set up the A coefficient matrix for steady state, S eq 0
* on fine grid.
      call setup(sa,ija,bound,ss,tran,0,dt)
      if(load.eq.0)then
          call setq(1,ss)
      else
          call setq(jack,sdr)
      endif
      call setbound(newhbound,ss)
      hbound=newhbound
      do j=1,ny
          do i=1,nx
              Htouse(i,j)=0.
          enddo
      enddo
      call loadf(b,bound,hbound,ss,tran,dt,Htouse)
* Solve for mean head for steady state.  Setq(1,ss) builds load
* vector for uniform recharge = ss plus boundary conditions.
* If the load vector is zero linbcg divides by 0.
      call linbcg(nxny,b,x,1,tol,itmax,iter,err,sa,ija)
* Copy x solution vector into Hallk(0) for steady state.
      do j=1,ny
          do i=1,nx
              n=i+(j-1)*nx
              Hallk(i,j,0)=x(n)
              Htouse(i,j)=x(n)
              if(outc(2).eq.'y') write(77,*) x(n)
          enddo
      enddo
* calculate the mean head for each time step kk=1,nts
* Implicit time step.
      cts=dt
      ttime=dt
      do kk=1,nts
* Set qwells recharge and pumping load to transient condition.
* either by changing the load or by changing the boundary
* conditions or both
          if(pert(1).eq.'y')then
              if(load.eq.0)then
                  call setq(jack,sdr)
```

```
            else
                call setq(1,ss)
            endif
        endif
        if(pert(2).eq.'y')then
            call setbound(newhbound,ttime)
            hbound=newhbound
        endif
        call setup(sa,ija,bound,stor,tran,0,cts)
        call loadf(b,bound,hbound,stor,tran,cts,Htouse)
        call linbcg(nxny,b,x,1,tol,itmax,iter,err,sa,ija)
* Copy b solution vector into Hallk for time step kk.
        do j=1,ny
            do i=1,nx
                n=i+(j-1)*nx
                Hallk(i,j,kk)=x(n)
                Htouse(i,j)=x(n)
                if(outc(2).eq.'y') write(77,*) x(n)
            enddo
        enddo
* Increment current time step by time step multiplier.
        ttime=ttime+cts
        cts=cts*tsm
    enddo
* Setup coefficient matrix for covariances on the coarse grid.
    call setup(sa,ija,bound,ss,tran,1,dt)
* Solve the coarse grid boundary value problem nxanya times for
* steady state cross-covariance matrix Rfh.
    do j=1,ny
        do i=1,nx
            Htouse(i,j)=Hallk(i,j,0)
        enddo
    enddo
    do j=1,nya
        do i=1,nxa
            n=i+(j-1)*nxa
            xo=real(i)*dxa-dxa/2.
            yo=real(j)*dya-dya/2.
* Load vector for cross-covariance uses derivatives from mean
* head in hmean, input function covar, and previous values in
* storage matrix Rfh times S/dt for time stepping.
            call loadrfh(b,xo,yo,modn,ss,tran,n,dt,Htouse)
            do k=1,nxa*nya
                x(k)=0.
            enddo
            call linbcg(nxa*nya,b,x,1,tol,itmax,iter,err,sa,ija)
* Copy solution into storage matrix.  Fixed location f, indexed
* by n or i,j, is stored by columns.  Variable location f, all
* other blocks in the domain, stored by row.
            do k=1,nxa*nya
                Rfh(k,n)=x(k)
            enddo
        enddo
    enddo
* compare Rfh test routine??????????????????
    if(outc(3).eq.'y')then
        open(unit=25,file='mpr.dat')
        read(25,*) mpr1
        read(25,*) mpr2
        close(25)
```

```
          open(unit=100,file='Rfht.out')
          write(*,*) 'Rfh ',0
          do ntest=1,nxa*nya
              write(100,99) Rfh(ntest,mpr1),Rfh(mpr1,ntest),
     c         Rfh(ntest,mpr2),Rfh(mpr2,ntest)
          enddo
       endif
    99 format(4f16.8)
* Solve for head covariance at steady state.
c       t1 = dtime_(tc,ts)
       do j=1,nya
          do i=1,nxa
              n=i+(j-1)*nxa
* Load vector for Rhh covariance at h(i,j) location uses
* derivatives on mean head from hmean, derivatives on Rhf
* from array Rfh taken by rows, and Rhh from last partial step.
              call loadrhh(b,n,ss,tran,1,dt,Htouse)
              do k=1,nxa*nya
                  x(k)=0.
              enddo
              call linbcg(nxa*nya,b,x,1,tol,itmax,iter,err,sa,ija)
* Copy solution into storage matrix.  Fixed location h, index n,
* stored by columns in first partial step.
              do k=1,nxa*nya
                  Rhh(k,n,1)=x(k)
              enddo
          enddo
       enddo
* compare Rhh test routine??????????????????
       if(outc(3).eq.'y')then
          open(unit=101,file='Rhht.out')
          write(*,*) 'Rhh',0,0
          do ntest=1,nxa*nya
              write(101,99) Rhh(ntest,mpr1,1),Rhh(mpr1,ntest,1),
     c         Rhh(ntest,mpr2,1),Rhh(mpr2,ntest,1)
          enddo
       endif
* Extract cokriging data matrix and estimation location cross-
* covariance vector for steady state and write to cokcov.out.
* This uses a polynomial interpolation scheme on Rfh(n,m) and
* Rhh(n,m,1).  The one extracts cfhvect, 0 for no cfhvect.
       call extract(datmat,cfhvect,1)
       do i=1,nfdata
          do j=1,nhdata
              write(27,*) datmat(1,i,j)
          enddo
       enddo
       do i=1,nhdata
          do j=1,nhdata
              write(27,*) datmat(2,i,j)
          enddo
       enddo
       do j=1,ny
          do i=1,nx
          n=i+(j-1)*nx
              do k=1,nhdata
                  write(27,*) cfhvect(n,k)
              enddo
          enddo
       enddo
```

```
      print*,'Rfh Rhh collect ',0,0,hdtimes(0),hdtimes(0)
      if(outc(3).eq.'y')then
          print*,'reference h location on coarse grid block ',mpr1,mpr2
          print*,'files Rhht.out and Rfht.out'
      endif
* BEGIN TIME STEP*********************************>>>>>>>>>>>>>>>>
* Steady state is time step 0.  Number of time steps nts is
* calculated in inputpar from total time, initial time step dt,
* and time step mulitplier tsm.  Current time step cts.
      cts=dt
      ttime=dt
      ttime2=0.
      iot=0
      int=1
      do kk=1,nts
          ttime2=ttime2+dt
c          write(*,*) 'calculating covariances for time ',ttime,
c     c                ' explicit -> implicit'
* Use kk-1 (0=steady state) hmean and Rfh to make a forward time
* evaluation of Rh(n,kk-1)h(m,kk).  This loads the h(n,kk-1)h(m,kk)
* covariance into Rhh array 2.  The transpose of this will be
* needed later for Rh(n,kk)h(m,kk) with an implicit time step.
          do j=1,ny
              do i=1,nx
                  Htouse(i,j)=Hallk(i,j,kk-1)
              enddo
          enddo
* Explicit Time Step
          do m=1,nxa*nya
              call loadrhh(b,m,stor,tran,1,cts,Htouse)
* Last time step*stor and + dt*tran*dot(Rhf del H)) in vector b.
              do j=1,nya
                  do i=1,nxa
                      n=i+(j-1)*nxa
                      Rhh(n,m,2)=
     c               (cts*tran*expliciteval(i,j,m,n,bound)+b(n))/stor
                  enddo
              enddo
          enddo
          do m=1,nxa*nya
              do n=1,nxa*nya
                  Rhh(n,m,1)=Rhh(n,m,2)
              enddo
          enddo
* Call extract with 0 since the estimation location vector is
* not needed.
          if(dabs((ttime-dt)-hdtimes(iot)).lt.tol.and.
     c    dabs(ttime2-hdtimes(int)).lt.tol)then
* compare Rhh test routine???????????????????
            if(outc(3).eq.'y')then
              write(*,*) 'Rhh',kk-1,kk
              do ntest=1,nxa*nya
                 write(101,99) Rhh(ntest,mpr1,1),Rhh(mpr1,ntest,1),
     c           Rhh(ntest,mpr2,1),Rhh(mpr2,ntest,1)
              enddo
            endif
            call extract(datmat,cfhvect,0)
            do i=1,nhdata
               do j=1,nhdata
                  write(27,*) datmat(2,i,j)
```

```
                enddo
              enddo
            print*,'Rhh collect steps ',kk-1,kk,hdtimes(iot),hdtimes(int)
              int=int+1
           endif
* Here's where I estimate h(n,kk-1)h(m,kk+1 to nts) covariance
* with an explicit time step.  Each of these goes into Rhh
* array 1 where the polynomial interpolation takes place.
* Copy Rhh array 2 by columns back into array 1 and run the
* polynomial extract to get off-diagonal cross-time components.

* Do the rest of the cross-time components.
          if(dabs((ttime-dt)-hdtimes(iot)).lt.tol)then
              tsk=cts*tsm
              do kappa=kk+1,nts
                  ttime2=ttime2+dt
c              print*,'entering inner time loop'
* Setup coefficient matrix for covariance on a coarse grid.
                  call setup(sa,ija,bound,stor,tran,1,tsk)
                  do j=1,ny
                      do i=1,nx
                          Htouse(i,j)=Hallk(i,j,kappa-1)
                      enddo
                  enddo
                  do m=1,nxa*nya
                      call loadrhh(b,m,stor,tran,1,tsk,Htouse)
* EXPLICIT TIME STEP
                      do j=1,nya
                          do i=1,nxa
                              n=i+(j-1)*nxa
                              temp(n)=
     c                        (tsk*tran*expliciteval(i,j,m,n,bound)
     c                        +b(n))/stor
                          enddo
                      enddo
                      do n=1,nxa*nya
                          Rhh(n,m,1)=temp(n)
                      enddo
                  enddo
                  if(dabs(ttime2-hdtimes(int)).lt.tol)then
                     if(outc(3).eq.'y')then
                        write(*,*) 'Rhh',kk-1,kappa
                        do ntest=1,nxa*nya
                           write(101,99) Rhh(ntest,mpr1,1),Rhh(mpr1,
     c                     ntest,1),Rhh(ntest,mpr2,1),Rhh(mpr2,ntest,1)
                        enddo
                     endif
                      call extract(datmat,cfhvect,0)
                      do i=1,nhdata
                          do j=1,nhdata
                              write(27,*) datmat(2,i,j)
                          enddo
                      enddo
                      print*,'Rhh collect steps ',kk-1,kappa,
     c                hdtimes(iot),hdtimes(int)
                      int=int+1
                  endif
                  tsk=tsk*tsm
              enddo
          endif
```

```
c leaving inner time loop
* Implicit Time Step.  Use Rh(n,kk-1)h(m,kk) array 2 transpose
* to solve for Rh(n,kk)h(m,kk) which goes into array 1.
* The implicit step uses Hallk(at time step kk) and the
* mean head at time step kk.
* Solve cross covariance Rfh(rows,cols) for time step kk.
* This is the coarse grid boundary value problem nxny times.
         do j=1,ny
             do i=1,nx
                 Htouse(i,j)=Hallk(i,j,kk)
             enddo
         enddo
         do j=1,nya
             do i=1,nxa
                 n=i+(j-1)*nxa
                 xo=real(i)*dxa-dxa/2.
                 yo=real(j)*dya-dya/2.
* Mean head and Rhf transpose of Rfh array into vector b.

                 call loadrfh(b,xo,yo,modn,stor,tran,n,cts,Htouse)
* A first initial guess may be the last solution.
                 do k=1,nxa*nya
                     x(k)=Rfh(k,n)
                 enddo
                 call linbcg(nxa*nya,b,x,1,tol,itmax,iter,err,sa,ija)
* Copy solution into storage matrix by columns
                 do k=1,nxa*nya
                     Rfh(k,n)=x(k)
                 enddo
             enddo
         enddo
* Solve for head covariance at time step kk.
         do j=1,nya
             do i=1,nxa
                 n=i+(j-1)*nxa
* nar = 2 loads Rhh array 2 by row n, the transpose of Rhh(2)
                 call loadrhh(b,n,stor,tran,2,cts,Htouse)
* A first guess as the last solution.
                 do k=1,nxa*nya
                     x(k)=Rhh(k,n,2)
                 enddo
                 call linbcg(nxa*nya,b,x,1,tol,itmax,iter,err,sa,ija)
* Copy solution into Rhh storage matrix first array by columns.
* This is h location, at this time step, covariance with all
* other locations at this time step.
                 do k=1,nxa*nya
                     Rhh(k,n,1)=x(k)
                 enddo
             enddo
         enddo
* Extract cokriging data matrix and estimation location cross-
* covariance vector and write to cokcov.out.
* Only save the times for which there's head data.
         if(dabs(ttime-hdtimes(iot+1)).lt.tol)then
* compare Rfh test routine??????????????????
            if(outc(3).eq.'y')then
               write(*,*) 'Rfh ',kk
               do ntest=1,nxa*nya
                  write(100,99) Rfh(ntest,mpr1),Rfh(mpr1,ntest),
     c            Rfh(ntest,mpr2),Rfh(mpr2,ntest)
```

```
               enddo
            endif
* compare Rhh test routine??????????????????
            if(outc(3).eq.'y')then
               write(*,*) 'Rhh ',kk,kk
               do ntest=1,nxa*nya
                  write(101,99) Rhh(ntest,mpr1,1),Rhh(mpr1,ntest,1),
     c            Rhh(ntest,mpr2,1),Rhh(mpr2,ntest,1)
               enddo
            endif
            call extract(datmat,cfhvect,1)
            do i=1,nfdata
               do j=1,nhdata
                  write(27,*) datmat(1,i,j)
               enddo
            enddo
            do i=1,nhdata
               do j=1,nhdata
                  write(27,*) datmat(2,i,j)
               enddo
            enddo
            do j=1,ny
               do i=1,nx
                  n=i+(j-1)*nx
                  do k=1,nhdata
                     write(27,*) cfhvect(n,k)
                  enddo
               enddo
            enddo
          print*,'Rfh Rhh collect ',kk,kk,hdtimes(iot+1),hdtimes(iot+1)
            iot=iot+1
            int=iot+1
         endif
* Increment current time step by time step multiplier.
         ttime2=ttime
         ttime=ttime+cts
         cts=cts*tsm
      enddo
* End of transient covariance calculations.
*****************************************************************

 13   continue
      if(outc(2).eq.'y') close(77)
      if(outc(3).eq.'y')then
          close(99)
          close(100)
      endif
      print*,'Covariance calculations completed for'
      print*,'Time steps ',nts
      print*,'Total time ',ttime-cts
      END
```

A.3.2 Main for collaterally conditioning simulations

```
* PROGRAM PEJUTA
* Uses covariances calculated from lhamana and saved on disk to
* condition a random field on f and h data.
      parameter(maxit=2000)
      implicit double precision (a-h,o-z)
      include 'lhamana.inc'
      logical hitb
      character bound(4),outc(4),pert(2),newbound(4),track
      integer i,j,k,nxny,na,nb,modn,ndata,iter,nsims,in,nts,ntrans
      integer ija(max*(5*max-2)),blav,nsd,maxl,maxt,ntimes,ihdt
      integer nsimsbust,itdiverged,iterc,jack,load,nf,kappa,mtry
      real*8 corx,cory,varf,datp(ndd+(maxntimes+1)*ndd),vhat(max*max)
      real*8 b(max*max),x(max*max),sa(max*(5*max-2)),tol,err
      real*8 stor,tran,tf(max,max),newhbound(max,4)
      real*8 hbound(max,4),amean,avar,ff(max,max),ssqh,diff
      real*8 xmn,xmx,ymn,ymx,cx,cy,lastssqh,uhat(max*max),tolc
      real*8 vs(max,max),sdr,dt,dx,dy,hcmem(max*max),ss,tsm
      real*8 Tlnm,ttime,husamp(ndd,0:maxntimes)
      real*8 cts,cfhvect(max*max,ndd,0:maxntimes),tracktime
      real*8 Htouse(max,max),cfh(ndd,ndd,0:maxntimes)
      real*8 chh(ndd,ndd,0:(maxntimes+1)**2),hdtimes(0:maxntimes)
      real*8 ufdata(ndd),Hallk(max,max,0:maxnts),startploc(4)
      real*8 partrack(max*10,3),voids,thick,pmem(4),ptim(2)
      real*8 condstats(max,max,4),xcoord,ycoord,aae,aaemax
* get all the necessary parameters
      call inputpar(bound,modn,nsd,tol,tolc,nsims,stor
     c             ,tran,outc,sdr,nts,jack,tsm,dt,load,pert
     c             ,newbound,ntimes,hdtimes
     c             ,track,startploc,tracktime,voids,thick,ntrans)
      ss=0.
      nsc=0
      nxny=nx*ny
      Tlnm=log(tran)
      itmax=maxit
      mtry=25
      na=nx
      nb=ny
      nf=nfdata
      ndata=nfdata+nhdata*(ntimes+1)
      xmn=0.
      ymn=0.
      xmx=real(nx)*dx-dx
      ymx=real(ny)*dy-dx
      vr=varf
      cx=corx
      cy=cory
      blav=0
      if(outc(4).eq.'y') open(unit=60,file='hu.out')
* Calculate mean head for all time steps
* Set up the mean head coefficient matrix for steady state.
      call setup(sa,ija,bound,ss,tran,0,dt)
      if(load.eq.0)then
          call setq(1,ss)
```

```
      else
          call setq(jack,sdr)
      endif
      call setbound(newhbound,ss)
      hbound=newhbound
      do j=1,ny
          do i=1,nx
              Htouse(i,j)=0.
          enddo
      enddo
      call loadf(b,bound,hbound,ss,tran,dt,Htouse)
* Solve for mean head for steady state.
* If the load vector b is zero, no-load, linbcg divides by zero.
      call linbcg(nxny,b,x,1,tol,itmax,iter,err,sa,ija)
* Copy x solution vector into hmean for steady state.
      do j=1,ny
          do i=1,nx
              n=i+(j-1)*nx
              Hallk(i,j,0)=x(n)
              Htouse(i,j)=x(n)
          enddo
      enddo
      cts=dt
      ttime=dt
      do kk=1,nts
          if(pert(1).eq.'y')then
              if(load.eq.0)then
                  call setq(jack,sdr)
              else
                  call setq(1,ss)
              endif
          endif
          if(pert(2).eq.'y')then
              call setbound(newhbound,ttime)
              hbound=newhbound
          endif
* Solve for mean head at time step kk.
          call setup(sa,ija,bound,stor,tran,0,cts)
          call loadf(b,bound,hbound,stor,tran,cts,Htouse)
          call linbcg(nxny,b,x,1,tol,itmax,iter,err,sa,ija)
* Copy b solution vector into hmean for transient step.
          do j=1,ny
              do i=1,nx
                  n=i+(j-1)*nx
                  Hallk(i,j,kk)=x(n)
                  Htouse(i,j)=x(n)
              enddo
          enddo
          ttime=ttime+cts
          cts=cts*tsm
      enddo
* Read in data covariance matrix and estimation location vector
* from file cokcov.in
      open(unit=27,file='cokcov.in')
c      print*,'cokcov'
* Steady state,
      do i=1,nfdata
          do j=1,nhdata
              read(27,*) cfh(i,j,0)
          enddo
```

```
      enddo
      do i=1,nhdata
          do j=1,nhdata
              read(27,*) chh(i,j,0)
c              print*,chh(i,j,0)
          enddo
      enddo
      do j=1,ny
          do i=1,nx
          n=i+(j-1)*nx
              do k=1,nhdata
                  read(27,*) cfhvect(n,k,0)
              enddo
          enddo
      enddo
* Time step cokcov
      kappa=1
      do kk=1,ntimes
          do kkk=kk,ntimes
              do i=1,nhdata
                  do j=1,nhdata
                      read(27,*) chh(i,j,kappa)
c                     print*,chh(i,j,kappa)
                  enddo
              enddo
              kappa=kappa+1
          enddo
          do i=1,nfdata
              do j=1,nhdata
                  read(27,*) cfh(i,j,kk)
              enddo
          enddo
          do i=1,nhdata
              do j=1,nhdata
                  read(27,*) chh(i,j,kappa)
c                 print*,chh(i,j,kappa)
              enddo
          enddo
          kappa=kappa+1
          do j=1,ny
              do i=1,nx
                  n=i+(j-1)*nx
                  do k=1,nhdata
                      read(27,*) cfhvect(n,k,kk)
                  enddo
              enddo
          enddo
       enddo
       close(27)
* SIMULATION RUN, condition a random f and h field on data of
* each a number of times.  For each conditioned simulation, calc
* and accumulate statistics.  Built in requirement that there is
* the same number of data at the same locations for each time
* step.
       nsimsbust=0
       if(track.eq.'y') open(unit=99,file='ptrack1.out')
       if(track.eq.'y') open(unit=100,file='ptrack2.out')
       do in=1,nsims
       lastssqh=1.E+7
       itdiverged=0
```

```
      iterc=0
* Generate the unconditional random f field with mean zero
* and specified variance using spectral methods and FFT.
      call ranfgen(modn,blav,na,nb,xmn,xmx,ymn,ymx,
     b     vr,cx,cy,nsd,ff,amean,avar)
* collect sample location distributions for unconditional f
      call samplef(ufdata,nf,ff)
      if(outc(3).eq.'y')then
          open(unit=65,file='fu.out')
          do j=1,ny
              ycoord=dble(j)*dy-dy/2.
              do i=1,nx
                  xcoord=dble(i)*dx-dx/2.
                  write(65,*) xcoord,ycoord,ff(i,j)
              enddo
          enddo
          close(65)
      endif
* Calculate unconditional head and sample for cokriging.
* Reset the pumping/recharge load conditions
      if(load.eq.0)then
          call setq(1,ss)
      else
          call setq(jack,sdr)
      endif
      call setbound(newhbound,ss)
      hbound=newhbound
      do j=1,ny
          do i=1,nx
              tf(i,j)=dexp(Tlnm+ff(i,j))
          enddo
      enddo
* Setup connectivity matrix using harmonic averages between
* random transmissivity blocks. ss=0 is steady state.
      call setupr(sa,ija,bound,tf,ss,dt)
      do i=1,nxny
          b(i)=0.
      enddo
      call loadfr(b,bound,hbound,tf,ss,dt)
      do j=1,ny
          do i=1,nx
              n=i+(j-1)*nx
              x(n)=Hallk(i,j,0)
          enddo
      enddo
      call linbcg(nxny,b,x,1,tol,itmax,iter,err,sa,ija)
      do n=1,nxny
          hcmem(n)=x(n)
      enddo
* Sample steady state unconditional head field.
      do i=1,nhdata
          k=int(hdata(i,1)/dx)+1
          l=int(hdata(i,2)/dy)+1
          n=k+(l-1)*nx
          if(hdata(i,3).lt.1.E+29)then
             husamp(i,0)=x(n)-Hallk(k,l,0)
          else
             husamp(i,0)=1.E+30
          endif
      enddo
```

```
      if(outc(4).eq.'y')then
          do j=1,ny
              ycoord=dble(j)*dy-dy/2.
              do i=1,nx
                  xcoord=dble(i)*dx-dx/2.
                  n=i+(j-1)*nx
                  write(60,6) xcoord,ycoord,x(n),Hallk(i,j,0),
     c             x(n)-Hallk(i,j,0)
              enddo
          enddo
      endif
    6 format(2f8.1,3f18.10)
* Solve for the rest of the time steps
* Steady state is time step 0.  Number of time steps nts is
* calculated in inputpar from total time, initial time step dt,
* and time step mulitplier tsm.  Current time step cts.
      cts=dt
      ttime=dt
      ihdt=1
      do kk=1,nts
          if(pert(1).eq.'y')then
              if(load.eq.0)then
                  call setq(jack,sdr)
              else
                  call setq(1,ss)
              endif
          endif
          if(pert(2).eq.'y')then
              call setbound(newhbound,ttime)
              hbound=newhbound
          endif
          call setupr(sa,ija,bound,tf,stor,cts)
          do n=1,nxny
              b(n)=hcmem(n)
              x(n)=b(n)
          enddo
          call loadfr(b,bound,hbound,tf,stor,cts)
          call linbcg(nxny,b,x,1,tol,itmax,iter,err,sa,ija)
          do n=1,nxny
              hcmem(n)=x(n)
          enddo
          if(dabs(ttime-hdtimes(ihdt)).lt.tol)then
              do i=1,nhdata
                  k=int(hdata(i,1)/dx)+1
                  l=int(hdata(i,2)/dy)+1
                  n=k+(l-1)*nx
                  if(htdat(i,ihdt).lt.1.E+29)then
                     husamp(i,ihdt)=x(n)-Hallk(k,l,kk)
                  else
                     husamp(i,ihdt)=1.E+30
                  endif
              enddo
              if(outc(4).eq.'y')then
                  do j=1,ny
                      ycoord=dble(j)*dy-dy/2.
                      do i=1,nx
                          xcoord=dble(i)*dx-dx/2.
                          n=i+(j-1)*nx
                          write(60,6) xcoord,ycoord,x(n),Hallk(i,j,kk)
      c                   ,x(n)-Hallk(i,j,kk)
```

```
                    enddo
                enddo
            endif
            ihdt=ihdt+1
        endif
        ttime=ttime+cts
        cts=cts*tsm
    enddo
* Use unconditional f and f.dat, sampled unconditional h, h.dat
* and ht.dat to obtain a conditioned f field by cokriging.
    do i=1,nfdata
        datp(i)=fdata(i,3)
    enddo
    do i=1,nhdata
        datp(i+nfdata)=hdata(i,3)
    enddo
    kappa=1
    do kk=1,ntimes
        do i=1,nhdata
            datp(nfdata+nhdata+kappa)=htdat(i,kk)
            kappa=kappa+1
        enddo
    enddo
    if(outc(3).eq.'y')then
        open(unit=61,file='vdatp.out')
        do i=1,ndata
            write(61,*) datp(i)
        enddo
        close(61)
    endif
    call cokrig(ndata,modn,datp,vhat,cfh,chh,cfhvect,ntimes)
    if(outc(3).eq.'y')then
        open(unit=63,file='vhat.out')
        do j=1,ny
            ycoord=dble(j)*dy-dy/2.
            do i=1,nx
                n=i+(j-1)*nx
                xcoord=dble(i)*dx-dx/2.
                write(63,*) xcoord,ycoord,vhat(n)
            enddo
        enddo
        close(63)
    endif
    do i=1,nfdata
        datp(i)=ufdata(i)
    enddo
    kappa=1
    do kk=0,ntimes
        do i=1,nhdata
            datp(nfdata+kappa)=husamp(i,kk)
            kappa=kappa+1
        enddo
    enddo
 17 continue
    if(outc(3).eq.'y')then
        open(unit=62,file='udatp.out')
        do i=1,ndata
            write(62,*) datp(i)
        enddo
        close(62)
```

```
      endif
      call cokrig(ndata,modn,datp,uhat,cfh,chh,cfhvect,ntimes)
      if(outc(3).eq.'y')then
          open(unit=64,file='uhat.out')
          do j=1,ny
              ycoord=dble(j)*dy-dy/2.
              do i=1,nx
                  n=i+(j-1)*nx
                  xcoord=dble(i)*dx-dx/2.
                  write(64,*) xcoord,ycoord,uhat(n)
              enddo
          enddo
          close(64)
      endif
* Subtract to get the cokrig conditioned random f field.
      do j=1,ny
          do i=1,nx
              n=i+(j-1)*nx
              vs(i,j)=vhat(n)+(ff(i,j)-uhat(n))
          enddo
      enddo
* Solve for conditioned head field given cokriged conditioned
* f field.
* Reset the pumping/recharge load conditions
      if(load.eq.0)then
          call setq(1,ss)
      else
          call setq(jack,sdr)
      endif
      call setbound(newhbound,ss)
      hbound=newhbound
      do j=1,ny
          do i=1,nx
              tf(i,j)=dexp(Tlnm+vs(i,j))
          enddo
      enddo
* Setup connectivity matrix using harmonic averages between
* random transmissivity blocks. ss=0 is steady state.
      call setupr(sa,ija,bound,tf,ss,dt)
      do i=1,nxny
          b(i)=0.
      enddo
      call loadfr(b,bound,hbound,tf,ss,dt)
      do j=1,ny
          do i=1,nx
              n=i+(j-1)*nx
              x(n)=Hallk(i,j,0)
          enddo
      enddo
      call linbcg(nxny,b,x,1,tol,itmax,iter,err,sa,ija)
      do n=1,nxny
          hcmem(n)=x(n)
      enddo
* Add up SSQ errors for steady state
c      open(unit=69,file='testh.out')
      ssqh=0.
      aae=0.
      aaemax=0.
      do i=1,nhdata
          k=int(hdata(i,1)/dx)+1
```

```
          l=int(hdata(i,2)/dy)+1
          n=k+(l-1)*nx
          if(hdata(i,3).lt.1.E+29)then
             diff=(x(n)-Hallk(k,l,0))-hdata(i,3)
          else
             diff=0.
          endif
          datp(i+nfdata)=diff+datp(i+nfdata)
          ssqh=ssqh+(diff)**2
          temp=dabs(diff)
          aae=aae+temp
          if(temp.gt.aaemax)then
              aaemax=temp
              maxl=i
              maxt=0
          endif
       enddo
* Solve for the rest of the time steps
* Steady state is time step 0.  Number of time steps nts is
* calculated in inputpar from total time, initial time step dt,
* and time step mulitplier tsm.  Current time step cts.
       cts=dt
       ttime=dt
       kappa=1
       ihdt=1
       do kk=1,nts
           if(pert(1).eq.'y')then
               if(load.eq.0)then
                   call setq(jack,sdr)
               else
                   call setq(1,ss)
               endif
           endif
           if(pert(2).eq.'y')then
               call setbound(newhbound,ttime)
               hbound=newhbound
           endif
* Calculate the time step perturbed head using kriged f
           call setupr(sa,ija,bound,tf,stor,cts)
           do n=1,nxny
               b(n)=hcmem(n)
               x(n)=b(n)
           enddo
           call loadfr(b,bound,hbound,tf,stor,cts)
           call linbcg(nxny,b,x,1,tol,itmax,iter,err,sa,ija)
           do n=1,nxny
               hcmem(n)=x(n)
           enddo
           if(dabs(ttime-hdtimes(ihdt)).lt.tol)then
               do i=1,nhdata
                   k=int(hdata(i,1)/dx)+1
                   l=int(hdata(i,2)/dy)+1
                   n=k+(l-1)*nx
                   if(htdat(i,ihdt).lt.1.E+29)then
                      diff=(x(n)-Hallk(k,l,kk))-htdat(i,ihdt)
c                    print*,x(n)-Hallk(k,l,kk),htdat(i,ihdt)
                   else
                      diff=0.
                   endif
                   datp(nfdata+nhdata+kappa)=
```

```
c              diff+datp(nfdata+nhdata+kappa)
                ssqh=ssqh+(diff)**2
                temp=dabs(diff)
                aae=aae+temp
                if(temp.gt.aaemax)then
                    aaemax=temp
                    maxl=i
                    maxt=kk
                endif
                kappa=kappa+1
              enddo
              ihdt=ihdt+1
            endif
            ttime=ttime+cts
            cts=cts*tsm
          enddo
          iterc=iterc+1
          write(*,1) in,iterc
         c           ,' ASQE ',ssqh/dble(nhdata*(ntimes+1))
         c           ,' AAE ',aae/dble(nhdata*(ntimes+1))
         c           ,' AME ',aaemax
          write(*,10) 'maximum error at ',hdata(maxl,1),hdata(maxl,2),
         c            ' time step',kk-1
       10 format(a18,2f6.0,a11,i5)
c       write(69,*) 'iteration ',iterc
        1 format(2i4,a7,e12.6,a7,e12.6,a7,e12.6)
          if(aaemax.gt.tolc.and.iterc.lt.mtry) then
            if(ssqh.ge.lastssqh)then
              print*,'ITERATION DIVERGENCE at simulation',in
              nsimsbust=nsimsbust+1
              itdiverged=1
              go to 18
            endif
            lastssqh=ssqh
            go to 17
          endif
* collect conditioned f field
          if(outc(3).eq.'y')then
            open(unit=71,file='fc.out')
            do j=1,ny
              ycoord=dble(j)*dy-dy/2.
              do i=1,nx
                xcoord=dble(i)*dx-dx/2.
                write(71,*) xcoord,ycoord,vs(i,j)
              enddo
            enddo
            close(71)
          endif
       18 continue
          if(aaemax.le.tolc) then
******* solve head one more time for particle tracking
* Reset the pumping/recharge load conditions
            if(load.eq.0)then
              call setq(1,ss)
            else
              call setq(jack,sdr)
            endif
            call setbound(newhbound,ss)
            hbound=newhbound
            do j=1,ny
```

```
            do i=1,nx
                tf(i,j)=dexp(Tlnm+vs(i,j))
            enddo
        enddo
        call setupr(sa,ija,bound,tf,ss,dt)
        do i=1,nxny
            b(i)=0.
        enddo
        call loadfr(b,bound,hbound,tf,ss,dt)
        do j=1,ny
            do i=1,nx
                n=i+(j-1)*nx
                x(n)=Hallk(i,j,0)
            enddo
        enddo
        call linbcg(nxny,b,x,1,tol,itmax,iter,err,sa,ija)
        do j=1,ny
            do i=1,nx
                n=i+(j-1)*nx
                hcmem(n)=x(n)
                Htouse(i,j)=Hallk(i,j,0)
            enddo
        enddo
        aaemax=0.
        do i=1,nhdata
            k=int(hdata(i,1)/dx)+1
            l=int(hdata(i,2)/dy)+1
            n=k+(l-1)*nx
            if(hdata(i,3).lt.1.E+29)then
               diff=(x(n)-Hallk(k,l,0))-hdata(i,3)
            else
               diff=0.
            endif
            temp=dabs(diff)
            if(temp.gt.aaemax) aaemax=temp
        enddo
* call to particle track
***** Particle tracking at steady state
        if(track.eq.'y')then
            hitb=.false.
            pmem=startploc
            ptim(1)=0.
            ptim(2)=0.
            partrack(1,1)=startploc(1)
            partrack(1,2)=startploc(2)
            partrack(1,3)=0.
            if(ntrans.eq.0)then
                call annette(x,tf,voids,thick,partrack,npoints
     c           ,tracktime,hitb)
                do i=1,npoints
                    write(99,13) partrack(i,1),partrack(i,2)
     c               ,partrack(i,3)
                enddo
            else
                write(99,13) partrack(1,1),partrack(1,2)
     c           ,partrack(1,3)
                write(100,13) pmem(3),pmem(4),0.
            endif
        endif
* Solve for the rest of the time steps
```

```
      cts=dt
      ttime=dt
      ihdt=1
      do kk=1,ntrans
          if(pert(1).eq.'y')then
              if(load.eq.0)then
                  call setq(jack,sdr)
              else
                  call setq(1,ss)
              endif
          endif
          if(pert(2).eq.'y')then
              call setbound(newhbound,ttime)
              hbound=newhbound
          endif
          call setupr(sa,ija,bound,tf,stor,cts)
          do n=1,nxny
              b(n)=hcmem(n)
              x(n)=b(n)
          enddo
          call loadfr(b,bound,hbound,tf,stor,cts)
          call linbcg(nxny,b,x,1,tol,itmax,iter,err,sa,ija)
          do n=1,nxny
              hcmem(n)=x(n)
          enddo
          if(dabs(ttime-hdtimes(ihdt)).lt.tol)then
              do i=1,nhdata
                  k=int(hdata(i,1)/dx)+1
                  l=int(hdata(i,2)/dy)+1
                  n=k+(l-1)*nx
                  if(htdat(i,ihdt).lt.1E+29)then
                     diff=(x(n)-Hallk(k,l,kk))-htdat(i,ihdt)
                  else
                     diff=0.
                  endif
                  temp=dabs(diff)
                  if(temp.gt.aaemax) aaemax=temp

              enddo
              ihdt=ihdt+1
          endif
          ttime=ttime+cts
          cts=cts*tsm
          if(track.eq.'y'.and.(.not.hitb))then
             partrack(1,1)=pmem(1)
             partrack(1,2)=pmem(2)
             partrack(1,3)=ptim(1)
              call annette(x,tf,voids,thick,partrack,npoints
     c         ,tracktime,hitb)
     c        print*,npoints
              do i=2,npoints
                  write(99,13) partrack(i,1),partrack(i,2)
     c             ,partrack(i,3)
              enddo
              pmem(1)=partrack(npoints,1)
              pmem(2)=partrack(npoints,2)
              ptim(1)=partrack(npoints,3)
              partrack(1,1)=pmem(3)
              partrack(1,2)=pmem(4)
              partrack(1,3)=ptim(2)
```

```
                call annette(x,tf,voids,thick,partrack,npoints
     c           ,tracktime,hitb)
c               print*,npoints
                do i=2,npoints
                    write(100,13) partrack(i,1),partrack(i,2)
     c               ,partrack(i,3)
                enddo
                pmem(3)=partrack(npoints,1)
                pmem(4)=partrack(npoints,2)
                ptim(2)=partrack(npoints,3)
              endif
          enddo
* End of particle tracking
          print*,'AAE for particle track head ',aaemax
* End of collect
***** collect ensemble conditional mean variance fields at
***** steady state head
          nsc=nsc+1
          call var_mean(vs,x,nsc,condstats)
      endif
* End of simulations.
      enddo
 13   format(2f10.2,f15.5)
* print out ensemble mean and var
      open(unit=50,file='fens.out')
      open(unit=51,file='hens.out')
      do j=1,ny
          ycoord=dble(j)*dy-dy/2.
          do i=1,nx
          xcoord=dble(i)*dx-dx/2.
          write(50,15) xcoord,ycoord,condstats(i,j,1),condstats(i,j,2)
          write(51,15) xcoord,ycoord,condstats(i,j,3),condstats(i,j,4)
          enddo
      enddo
 15   format(2f10.2,2f20.10)
      close(50)
      close(51)
      if(track.eq.'y') close(99)
      if(outc(4).eq.'y') close(60)
      print*,nts,' time steps',nsims,' simulations',nsc,' used'
      end
```

Routine for input of parameters and data

```
c file for various input output routines
      subroutine inputpar(bound,modn,nsd,tol,tolc,nsims,stor
     c              ,tran,outc,sdr,nts,jack,tsm,dt,load,pert
     c              ,newbound,ntimes,hdtimes
     c              ,track,startploc,tracktime,voids,thick,ntrans)
      implicit double precision (a-h,o-z)
      include 'lhamana.inc'
      character bound(4),outc(4),pert(2),newbound(4),track
      integer i,k,modn,nsd,nts,jack,load,ntimes,ntrans
      real*8 corx,cory,varf,sdr,stor,tran
      real*8 ttime,tsm,t,temp,hdtimes(0:maxntimes),voids,thick
      real*8 tol,tolc,startploc(4),tracktime,dt
      open(unit=29,file='lhamana.par')
      read(29,*) nx
      read(29,*) ny
      read(29,*) ttime
      read(29,*) dt
      read(29,*) tsm
      ntrans=0
      t=0.
      temp=dt
      do while(t.lt.ttime)
          t=t+temp
          temp=temp*tsm
          ntrans=ntrans+1
      enddo
c      print*,'number of time steps ',nts
      read(29,*) dx
      read(29,*) dy
      read(29,*) tran
      read(29,*) stor
      read(29,*) sdr
      read(29,*) bound
c      read(29,*) hbound
      read(29,*) corx
      read(29,*) cory
      read(29,*) varf
      read(29,*) fer
      read(29,*) her
      read(29,*) modn
      read(29,*) nsd
      read(29,*) tol
      read(29,*) tolc
      read(29,*) nsims
      read(29,*) outc
      read(29,*) jack
      read(29,*) load
      read(29,*) pert
      read(29,*) newbound
c      read(29,*) newhbound
      read(29,*) track
      read(29,*) startploc
      read(29,*) tracktime
```

```
      read(29,*) voids
      read(29,*) thick
      close(29)
      open(unit=30,file='f.dat')
      read(30,*) nfdata
      do i=1,nfdata
          read(30,*) (fdata(i,j),j=1,3)
      enddo
      close(30)
      open(unit=31,file='h.dat')
      read(31,*) nhdata
      do i=1,nhdata
          read(31,*) (hdata(i,j),j=1,3)
      enddo
      close(31)
      open(unit=32,file='ht.dat')
      read(32,*) ntimes
      hdtimes(0)=0.
      do k=1,ntimes
          read(32,*) hdtimes(k)
          do i=1,nhdata
              read(32,*) htdat(i,k)
          enddo
      enddo
      nts=0
      t=0.
      temp=dt
      do while(t.lt.hdtimes(ntimes))
          t=t+temp
          temp=temp*tsm
          nts=nts+1
      enddo
      close(32)
      return
      end
c
      subroutine initialize
      implicit double precision (a-h,o-z)
      include 'lhamana.inc'
      do j=1,max
          do i=1,max
              Rfh(i,j)=0.
          enddo
      enddo
      do j=1,max
          do i=1,max
              qwells(i,j)=0.
          enddo
      enddo
      do j=1,max
          do i=1,max
              do k=1,2
                  Rhh(i,j,k)=0
              enddo
          enddo
      enddo
      return
      end
*
      subroutine setq(jack,to)
```

```
      implicit double precision (a-h,o-z)
      include 'lhamana.inc'
      integer i,j,jack,nqdata,k,l
      real*8 to,x,y,q
      if(jack.eq.1)then
          do j=1,ny
              do i=1,nx
                  qwells(i,j)=to
              enddo
          enddo
      elseif(jack.eq.2)then
          do j=1,ny
              do i=1,nx
                  qwells(i,j)=0.
              enddo
          enddo
          open(unit=61,file='q.dat')
          read(61,*) nqdata
          do i=1,nqdata
              read(61,*) x,y,q
              k=int(x/dx)+1
              l=int(y/dy)+1
              qwells(k,l)=q/(dx*dy)
          enddo
          close(61)
      endif
      return
      end
*
      subroutine samplef(datp,ndata,tf)
* samples the random field in tf as given by the locations
* in the fdata and hdata common arrays
      implicit double precision (a-h,o-z)
      integer i,j,k
      include 'lhamana.inc'
      real*8 tf(max,max),datp(ndata)
      do k=1,ndata
          i=int(fdata(k,1)/dx)+1
          j=int(fdata(k,2)/dy)+1
          if(fdata(k,3).lt.1.E+29)then
             datp(k)=tf(i,j)
          else
             datp(k)=1.E+30
          endif
      enddo
      return
      end
```

A.3.3 Matrix routines

```
* file matrix.for contains coefficient matrix setup subroutines
* for lhamana
* set up load vector for mean head
      subroutine loadf(fload,bound,hbound,stor,tran,dt,hmean)
      implicit double precision (a-h,o-z)
      character bound(4)
      integer i,j,n
      include 'lhamana.inc'
      real*8 fload(*),hbound(max,4),stor,hmean(max,max)
      do j=1,ny
          do i=1,nx
              n=i+(j-1)*nx
* load from wells, recharge and last time step head
              fload(n)=dt*qwells(i,j)+hmean(i,j)*stor
* add in load from boundary conditions corner and edge
* lower left
              if((i.eq.1).and.(j.eq.1))then
                  if((bound(2).eq.'d').and.(bound(4).eq.'d'))then
                      fload(n)=fload(n)+8./3.*tran*dt*
     c                  (hbound(j,2)/dx**2+hbound(i,4)/dy**2)
                  elseif((bound(2).eq.'n').and.(bound(4).eq.'d'))then
                      fload(n)=fload(n)+dt*
     c                  (-hbound(j,2)/dx+8./3.*tran*hbound(i,4)/dy**2)
                  elseif((bound(2).eq.'d').and.(bound(4).eq.'n'))then
                      fload(n)=fload(n)+dt*
     c                  (8./3.*tran*hbound(j,2)/dx**2-hbound(i,4)/dy)
                  elseif((bound(2).eq.'n').and.(bound(4).eq.'n'))then
                      fload(n)=fload(n)+dt*
     c                  (-hbound(j,2)/dx-hbound(i,4)/dy)
                  else
                      print*,'error in boundary conditions LL'
                  endif
              endif
* left
              if((i.eq.1).and.(j.gt.1).and.(j.lt.ny))then
                  if(bound(2).eq.'d')then
                      fload(n)=fload(n)+8./3.*tran*dt*
     c                  hbound(j,2)/dx**2
                  elseif(bound(2).eq.'n')then
                      fload(n)=fload(n)-hbound(j,2)/dx*dt

                  else
                      print*,'error in boundary conditions L'
                  endif
              endif
* upper left
              if((i.eq.1).and.(j.eq.ny))then
                  if((bound(2).eq.'d').and.(bound(1).eq.'d'))then
                      fload(n)=fload(n)+8./3.*tran*dt*
     c                  (hbound(j,2)/dx**2+hbound(i,1)/dy**2)
                  elseif((bound(2).eq.'n').and.(bound(1).eq.'d'))then
                      fload(n)=fload(n)+dt*
     c                  (-hbound(j,2)/dx+8./3.*tran*hbound(i,1)/dy**2)
```

```
                elseif((bound(2).eq.'d').and.(bound(1).eq.'n'))then
                    fload(n)=fload(n)+dt*
     c               (8./3.*tran*hbound(j,2)/dx**2-hbound(i,1)/dy)
                elseif((bound(2).eq.'n').and.(bound(1).eq.'n'))then
                    fload(n)=fload(n)+dt*
     c               (-hbound(j,2)/dx-hbound(i,1)/dy)
                else
                    print*,'error in boundary conditions UL'
                endif
            endif
* top
            if((i.gt.1).and.(i.lt.nx).and.(j.eq.ny))then
                if(bound(1).eq.'d')then
                    fload(n)=fload(n)+8./3.*tran*dt*
     c               hbound(i,1)/dy**2
                elseif(bound(1).eq.'n')then
                    fload(n)=fload(n)-hbound(i,1)/dy*dt

                else
                    print*,'error in boundary conditions T'
                endif
            endif
* upper right
            if((i.eq.nx).and.(j.eq.ny))then
                if((bound(3).eq.'d').and.(bound(1).eq.'d'))then
                    fload(n)=fload(n)+8./3.*tran*dt*
     c               (hbound(j,3)/dx**2+hbound(i,1)/dy**2)
                elseif((bound(3).eq.'n').and.(bound(1).eq.'d'))then
                    fload(n)=fload(n)+dt*
     c               (-hbound(j,3)/dx+8./3.*tran*hbound(i,1)/dy**2)
                elseif((bound(3).eq.'d').and.(bound(1).eq.'n'))then
                    fload(n)=fload(n)+dt*
     c               (8./3.*tran*hbound(j,3)/dx**2-hbound(i,1)/dy)
                elseif((bound(3).eq.'n').and.(bound(1).eq.'n'))then
                    fload(n)=fload(n)+dt*
     c               (-hbound(j,3)/dx-hbound(i,1)/dy)
                else
                    print*,'error in boundary conditions UR'
                endif
            endif
* right
            if((i.eq.nx).and.(j.lt.ny).and.(j.gt.1))then
                if(bound(3).eq.'d')then
                    fload(n)=fload(n)+8./3.*tran*dt*
     c               hbound(j,3)/dx**2
                elseif(bound(3).eq.'n')then
                    fload(n)=fload(n)-hbound(j,3)/dx*dt

                else
                    print*,'error in boundary conditions R'
                endif
            endif
* lower right
            if((i.eq.nx).and.(j.eq.1))then
                if((bound(3).eq.'d').and.(bound(4).eq.'d'))then
                    fload(n)=fload(n)+8./3.*tran*dt*
     c               (hbound(j,3)/dx**2+hbound(i,4)/dy**2)
                elseif((bound(3).eq.'n').and.(bound(4).eq.'d'))then
                    fload(n)=fload(n)+dt*
     c               (-hbound(j,3)/dx+8./3.*tran*hbound(i,4)/dy**2)
```

```
                elseif((bound(3).eq.'d').and.(bound(4).eq.'n'))then
                    fload(n)=fload(n)+dt*
     c               (8./3.*tran*hbound(j,3)/dx**2-hbound(i,4)/dy)
                elseif((bound(3).eq.'n').and.(bound(4).eq.'n'))then
                    fload(n)=fload(n)+dt*
     c               (-hbound(j,3)/dx-hbound(i,4)/dy)
                else
                    print*,'error in boundary conditions LR'
                endif
            endif
* bottom
            if((i.gt.1).and.(i.lt.nx).and.(j.eq.1))then
                if(bound(4).eq.'d')then
                    fload(n)=fload(n)+8./3.*tran*dt*
     c               hbound(i,4)/dx**2
                elseif(bound(4).eq.'n')then
                    fload(n)=fload(n)-hbound(i,4)/dx*dt
                else
                    print*,'error in boundary conditions B'
                endif
            endif
        enddo
      enddo
c      open(unit=69,file='testb.out')
c      do i=1,nx*ny
c          write(69,*) i,fload(i)
c      enddo
c      close(69)
      return
      end
* subroutine setup ************************************************
      subroutine setup(sa,ija,bound,stor,tran,iscov,dt)
      implicit double precision (a-h,o-z)
      include 'lhamana.inc'
      character bound(4)
      integer i,j,n,ija(*),iscov,nxa,nya
      real*8 stor,tran,dxa,dya,dt
      real*8 sa(*),bmat(max*max,5)
      if(iscov.eq.1)then
          dxa=dx*real(nob)
          dya=dy*real(nob)
          nxa=nx/nob
          nya=ny/nob
      else
          dxa=dx
          dya=dy
          nxa=nx
          nya=ny
      endif
      do j=1,5
          do i=1,max*max
              bmat(i,j)=0.
          enddo
      enddo
* build coefficient matrix
      do j=1,nya
          do i=1,nxa
              n=i+(j-1)*nxa
* corner conditions, lower left
              if((i.eq.1).and.(j.eq.1))then
```

```
                if((bound(2).eq.'d').and.(bound(4).eq.'d'))then
                    bmat(n,3)=stor+dt*tran*(4./dxa**2+4./dya**2)
                    bmat(n,4)=-4.*dt*tran/(3.*dxa**2)
                    bmat(n,5)=-4.*dt*tran/(3.*dya**2)
                elseif((bound(2).eq.'d').and.(bound(4).eq.'n'))then
                    bmat(n,3)=stor+dt*tran*(4./dxa**2+1./dya**2)
                    bmat(n,4)=-4.*dt*tran/(3.*dxa**2)
                    bmat(n,5)=-dt*tran/(dya**2)
                elseif((bound(2).eq.'n').and.(bound(4).eq.'d'))then
                    bmat(n,3)=stor+dt*tran*(1./dxa**2+4./dya**2)
                    bmat(n,4)=-dt*tran/(dxa**2)
                    bmat(n,5)=-4.*dt*tran/(3.*dya**2)
                elseif((bound(2).eq.'n').and.(bound(4).eq.'n'))then
                    bmat(n,3)=stor+dt*tran*(1./dxa**2+1./dya**2)
                    bmat(n,4)=-dt*tran/(dxa**2)
                    bmat(n,5)=-dt*tran/(dya**2)
                else
                    print*,'error lower left'
                endif
            endif
* bottom edge
            if((i.gt.1).and.(i.lt.nxa).and.(j.eq.1))then
                if(bound(4).eq.'d')then
                    bmat(n,3)=stor+dt*tran*(2./dxa**2+4./dya**2)

                    bmat(n,4)=-dt*tran/(dxa**2)
                    bmat(n,2)=-dt*tran/(dxa**2)
                    bmat(n,5)=-4.*dt*tran/(3.*dya**2)

                elseif(bound(4).eq.'n')then
                    bmat(n,3)=stor+dt*tran*(2./dxa**2+1./dya**2)
                    bmat(n,4)=-dt*tran/(dxa**2)
                    bmat(n,2)=-dt*tran/(dxa**2)
                    bmat(n,5)=-dt*tran/(dya**2)
                else
                    print*,'error bottom edge'
                endif
            endif
* corner conditions, lower right
            if((i.eq.nxa).and.(j.eq.1))then
                if((bound(3).eq.'d').and.(bound(4).eq.'d'))then
                    bmat(n,3)=stor+dt*tran*(4./dxa**2+4./dya**2)
                    bmat(n,2)=-4.*dt*tran/(3.*dxa**2)
                    bmat(n,5)=-4.*dt*tran/(3.*dya**2)
                elseif((bound(3).eq.'d').and.(bound(4).eq.'n'))then
                    bmat(n,3)=stor+dt*tran*(4./dxa**2+1./dya**2)
                    bmat(n,2)=-4.*dt*tran/(3.*dxa**2)
                    bmat(n,5)=-dt*tran/(dya**2)
                elseif((bound(3).eq.'n').and.(bound(4).eq.'d'))then
                    bmat(n,3)=stor+dt*tran*(1./dxa**2+4./dya**2)
                    bmat(n,2)=-dt*tran/(dxa**2)
                    bmat(n,5)=-4.*dt*tran/(3.*dya**2)
                elseif((bound(3).eq.'n').and.(bound(4).eq.'n'))then
                    bmat(n,3)=stor+dt*tran*(1./dxa**2+1./dya**2)
                    bmat(n,2)=-dt*tran/(dxa**2)
                    bmat(n,5)=-dt*tran/(dya**2)
                else
                    print*,'error lower right'
                endif
            endif
```

```
* left edge
              if((i.eq.1).and.(j.gt.1).and.(j.lt.nya))then
                  if(bound(2).eq.'d')then
                      bmat(n,3)=stor+dt*tran*(4./dxa**2+2./dya**2)
                      bmat(n,4)=-4.*dt*tran/(3.*dxa**2)
                      bmat(n,5)=-dt*tran/(dya**2)
                      bmat(n,1)=-dt*tran/(dya**2)

                  elseif((bound(2).eq.'n'))then
                      bmat(n,3)=stor+dt*tran*(1./dxa**2+2./dya**2)
                      bmat(n,4)=-dt*tran/(dxa**2)
                      bmat(n,5)=-dt*tran/(dya**2)
                      bmat(n,1)=-dt*tran/(dya**2)
                  else
                      print*,'error left edge'
                  endif
              endif
* right edge
              if((i.eq.nxa).and.(j.gt.1).and.(j.lt.nya))then
                  if(bound(3).eq.'d')then
                      bmat(n,3)=stor+dt*tran*(4./dxa**2+2./dya**2)

                      bmat(n,2)=-4.*dt*tran/(3.*dxa**2)
                      bmat(n,5)=-dt*tran/(dya**2)
                      bmat(n,1)=-dt*tran/(dya**2)

                  elseif((bound(3).eq.'n'))then
                      bmat(n,3)=stor+dt*tran*(1./dxa**2+2./dya**2)
                      bmat(n,2)=-dt*tran/(dxa**2)
                      bmat(n,5)=-dt*tran/(dya**2)
                      bmat(n,1)=-dt*tran/(dya**2)
                  else
                      print*,'error right edge'
                  endif
              endif
* corner conditions, upper left
              if((i.eq.1).and.(j.eq.nya))then
                  if((bound(2).eq.'d').and.(bound(1).eq.'d'))then
                      bmat(n,3)=stor+dt*tran*(4./dxa**2+4./dya**2)
                      bmat(n,4)=-4.*dt*tran/(3.*dxa**2)
                      bmat(n,1)=-4.*dt*tran/(3.*dya**2)
                  elseif((bound(2).eq.'d').and.(bound(1).eq.'n'))then
                      bmat(n,3)=stor+dt*tran*(4./dxa**2+1./dya**2)
                      bmat(n,4)=-4.*dt*tran/(3.*dxa**2)
                      bmat(n,1)=-dt*tran/(dya**2)
                  elseif((bound(2).eq.'n').and.(bound(1).eq.'d'))then
                      bmat(n,3)=stor+dt*tran*(1./dxa**2+4./dya**2)
                      bmat(n,4)=-dt*tran/(dxa**2)
                      bmat(n,1)=-4.*dt*tran/(3.*dya**2)
                  elseif((bound(2).eq.'n').and.(bound(1).eq.'n'))then
                      bmat(n,3)=stor+dt*tran*(1./dxa**2+1./dya**2)
                      bmat(n,4)=-dt*tran/(dxa**2)
                      bmat(n,1)=-dt*tran/(dya**2)
                  else
                      print*,'error upper left'
                  endif
              endif
* top edge
              if((i.gt.1).and.(i.lt.nxa).and.(j.eq.nya))then
                  if(bound(1).eq.'d')then
```

```
                    bmat(n,3)=stor+dt*tran*(2./dxa**2+4./dya**2)
                    bmat(n,4)=-dt*tran/(dxa**2)
                    bmat(n,2)=-dt*tran/(dxa**2)
                    bmat(n,1)=-4.*dt*tran/(3.*dya**2)

                elseif((bound(1).eq.'n'))then
                    bmat(n,3)=stor+dt*tran*(2./dxa**2+1./dya**2)
                    bmat(n,4)=-dt*tran/(dxa**2)
                    bmat(n,2)=-dt*tran/(dxa**2)
                    bmat(n,1)=-dt*tran/(dya**2)
                else
                    print*,'error top edge'
                endif
            endif
* corner conditions, upper right
            if((i.eq.nxa).and.(j.eq.nya))then
                if((bound(1).eq.'d').and.(bound(3).eq.'d'))then
                    bmat(n,3)=stor+dt*tran*(4./dxa**2+4./dya**2)
                    bmat(n,2)=-4.*dt*tran/(3.*dxa**2)
                    bmat(n,1)=-4.*dt*tran/(3.*dya**2)
                elseif((bound(1).eq.'d').and.(bound(3).eq.'n'))then
                    bmat(n,3)=stor+dt*tran*(4./dya**2+1./dxa**2)
                    bmat(n,2)=-dt*tran/(dxa**2)
                    bmat(n,1)=-4.*dt*tran/(3.*dya**2)
                elseif((bound(1).eq.'n').and.(bound(3).eq.'d'))then
                    bmat(n,3)=stor+dt*tran*(1./dya**2+4./dxa**2)
                    bmat(n,2)=-4.*dt*tran/(3.*dxa**2)
                    bmat(n,1)=-dt*tran/(dya**2)
                elseif((bound(1).eq.'n').and.(bound(3).eq.'n'))then
                    bmat(n,3)=stor+dt*tran*(1./dxa**2+1./dya**2)
                    bmat(n,2)=-dt*tran/(dxa**2)
                    bmat(n,1)=-dt*tran/(dya**2)
                else
                    print*,'error upper right'
                endif
            endif
* all the interior
            if((i.gt.1).and.(i.lt.nxa).and.(j.gt.1).and.(j.lt.nya))then
                    bmat(n,3)=stor+dt*tran*(2./dxa**2+2./dya**2)
                    bmat(n,4)=-dt*tran/(dxa**2)
                    bmat(n,2)=-dt*tran/(dxa**2)
                    bmat(n,5)=-dt*tran/(dya**2)
                    bmat(n,1)=-dt*tran/(dya**2)
              endif
c              write(*,6) (bmat(n,l),l=1,5)
          enddo
      enddo
c      open(unit=69,file='tests.out')
c      do i=1,nxa*nya
c          write(*,6) (bmat(i,j),j=1,5)
c      enddo
c    6 format(5f12.5)
c      close(69)
* store matrix in row indexed sparse storage mode for NR
* subroutine linbcg.  Reference Numerical Recipies 1992 pg 71.
      n=nxa*nya
      do 11 j=1,n
        sa(j)=bmat(j,3)
11    continue
      ija(1)=n+2
```

```
      k=n+1
c here the i index is for rows of the coefficient matrix a, j is col
      do 13 i=1,n
          j=i
          if(bmat(i,1).ne.0.)then
              k=k+1
              sa(k)=bmat(i,1)
              ija(k)=j-nxa
          endif
          if(bmat(i,2).ne.0.)then
              k=k+1
              sa(k)=bmat(i,2)
              ija(k)=j-1
          endif
          if(bmat(i,4).ne.0)then
              k=k+1
              sa(k)=bmat(i,4)
              ija(k)=j+1
          endif
          if(bmat(i,5).ne.0)then
              k=k+1
              sa(k)=bmat(i,5)
              ija(k)=j+nxa
          endif
12       continue
         ija(i+1)=k+1
13    continue
c      open(unit=70,file='testsa.out')
c      do i=1,nxa*(5*nxa-2)
c          write(70,*) i,ija(i),sa(i)
c      enddo
c      close(70)
      return
      end
*****************************************************************
* Subroutine loadrfh builds load vector for cross-covariance
* using mean head at time step and input covariance model.
* Derivatives on mean head are approximated by
* finite difference at the node center of the coarse grid of
* covariance calculation.
      subroutine loadrfh(b,xo,yo,modn,stor,tran,m,dt,hmean)
      implicit double precision (a-h,o-z)
      integer i,j,n,modn,nxa,nya,ia,ja,m
      include 'lhamana.inc'
      real*8 b(*),stor,tran,dt,hmean(max,max)
      real*8 dhldx,dhrdx,dhudy,dhddy,Rffl,Rffr,Rffu,Rffd
      real*8 xa,xo,ya,yo,dxa,dya,xl,xr,yd,yu,term1,term2
      dxa=dx*real(nob)
      dya=dy*real(nob)
      nxa=nx/nob
      nya=ny/nob
      do i=1,nxa*nya
          b(i)=0.
      enddo
      do j=1,nya
          do i=1,nxa
              n=i+(j-1)*nxa
* center of coarse grid block fd approximation to mean head
* derivative.  Average of 4 blocks around the center node.
              ia=i*nob-nob/2
```

```
              ja=j*nob-nob/2
* f covariance evaluated at neighboring or edge blocks
              xa=real(i-1)*dxa+dxa/2.
              ya=real(j-1)*dya+dya/2.
              xl=xa-dx
              xr=xa+dx
              yd=ya-dy
              yu=ya+dy
              dhldx=.5*((hmean(ia,ja+1)-hmean(ia-1,ja+1))/dx+
     c              (hmean(ia,ja)-hmean(ia-1,ja))/dx)
              dhrdx=.5*((hmean(ia+2,ja+1)-hmean(ia+1,ja+1))/dx+
     c              (hmean(ia+2,ja)-hmean(ia+1,ja))/dx)
              Rffl=covar(xl,xo,ya,yo,modn)
              Rffr=covar(xr,xo,ya,yo,modn)
              dhddy=.5*((hmean(ia+1,ja)-hmean(ia+1,ja-1))/dy+
     c              (hmean(ia,ja)-hmean(ia,ja-1))/dy)
              dhudy=.5*((hmean(ia+1,ja+2)-hmean(ia+1,ja+1))/dy+
     c              (hmean(ia,ja+2)-hmean(ia,ja+1))/dy)
              Rffd=covar(xa,xo,yd,yo,modn)
              Rffu=covar(xa,xo,yu,yo,modn)
              term1=(Rffr*dhrdx-Rffl*dhldx)/(2*dx)
              term2=(Rffu*dhudy-Rffd*dhddy)/(2*dy)
              b(n)=dt*tran*(term1+term2)+Rfh(n,m)*stor
c              write(*,*) m,n,b(n)
          enddo
       enddo
c     5 format(2i4,3e18.7)
       return
       end
*****************************************************************
       real*8 function covar(x,xo,y,yo,modn)
       implicit double precision (a-h,o-z)
       include 'lhamana.inc'
       real*8 x,xo,y,yo,treble,pi,arg
       pi=3.1415926535898
       if(modn.eq.2)then
           treble=sqrt(((xo-x)/corx)**2+((yo-y)/cory)**2)
           covar=varf*dexp(-treble)
       elseif(modn.eq.1)then
           treble=((xo-x)/corx)**2+((yo-y)/cory)**2
           covar=varf*dexp(-treble)
       elseif(modn.eq.4)then
           if(corx.ne.cory)then
               print*,'Mizell-A covariance must be isotropic'
               stop
           endif
           treble=sqrt((xo-x)**2+(yo-y)**2)
           arg=1.33*pi*treble/(4.*corx)
           if(treble.eq.0.)then
               covar=varf
           else
               covar=varf*(arg*besskl(arg)-0.5*(arg)**2*besskO(arg))
           endif
       else
           print*,'covariance model number?'
       endif
*       print*,Rff,dRdx,dRdy
       return
       end
*****************************************************************
```

```
* subroutine loadrhh builds load vector for head-covariance
* using mean head at time step and Rfh cross covariance
* b is the return vector, m is the row of h in Rfh matrix,
      subroutine loadrhh(b,m,stor,tran,nar,dt,hmean)
* nar, if nar .eq. load Rhh array 1 else load Rhh array 2
      implicit double precision (a-h,o-z)
      include 'lhamana.inc'
      integer i,j,n,m,nxa,nya,ia,ja,ir,il,ju,jd,nar
      real*8 b(*),stor,tran,dxa,dya,hmean(max,max)
      real*8 dhldx,dhrdx,dhddy,dhudy,Rhfl,Rhfr,Rhfd,Rhfu
      real*8 dhcdx,dhcdy,Rhfc
      dxa=dx*real(nob)
      dya=dy*real(nob)
      nxa=nx/nob
      nya=ny/nob
      do i=1,nxa*nya
          b(i)=0.
      enddo
      do j=1,nya
          do i=1,nxa
              n=i+(j-1)*nxa
* derivatives on hmean and Rfh for Rhh covariance load
              ia=i*nob-nob/2
              ja=j*nob-nob/2
              ir=(i+1)*nob-nob/2
              il=(i-1)*nob-nob/2
              ju=(j+1)*nob-nob/2
              jd=(j-1)*nob-nob/2
* left edge, no blocks to the left
              if(i.eq.1)then
                  dhcdx=.5*((hmean(ia+1,ja+1)-hmean(ia,ja+1))/dx+
     c              (hmean(ia+1,ja)-hmean(ia,ja))/dx)
                  dhrdx=.5*((hmean(ir+1,ja+1)-hmean(ir,ja+1))/dx+
     c              (hmean(ir+1,ja)-hmean(ir,ja))/dx)
                  Rhfc=Rfh(m,n)
                  Rhfr=Rfh(m,n+1)
              endif
* right edge, no blocks to the right
              if(i.eq.nxa)then
                  dhldx=.5*((hmean(il+1,ja+1)-hmean(il,ja+1))/dx+
     c              (hmean(il+1,ja)-hmean(il,ja))/dx)
                  dhcdx=.5*((hmean(ia+1,ja+1)-hmean(ia,ja+1))/dx+
     c              (hmean(ia+1,ja)-hmean(ia,ja))/dx)
                  Rhfl=Rfh(m,n-1)
                  Rhfc=Rfh(m,n)
              endif
* interior
              if(i.gt.1.and.i.lt.nxa)then
                  dhldx=.5*((hmean(il+1,ja+1)-hmean(il,ja+1))/dx+
     c              (hmean(il+1,ja)-hmean(il,ja))/dx)
                  dhrdx=.5*((hmean(ir+1,ja+1)-hmean(ir,ja+1))/dx+
     c              (hmean(ir+1,ja)-hmean(ir,ja))/dx)
                  dhcdx=.5*((hmean(ia+1,ja+1)-hmean(ia,ja+1))/dx+
     c              (hmean(ia+1,ja)-hmean(ia,ja))/dx)
                  Rhfl=Rfh(m,n-1)
                  Rhfr=Rfh(m,n+1)
                  Rhfc=Rfh(m,n)
              endif
* bottom edge, no blocks below
              if(j.eq.1)then
```

```
                dhcdy=.5*((hmean(ia+1,ja+1)-hmean(ia+1,ja))/dy+
     c            (hmean(ia,ja+1)-hmean(ia,ja))/dy)
                dhudy=.5*((hmean(ia+1,ju+1)-hmean(ia+1,ju))/dy+
     c            (hmean(ia,ju+1)-hmean(ia,ju))/dy)
                Rhfc=Rfh(m,n)
                Rhfu=Rfh(m,n+nxa)
              endif
* top edge, no blocks above
              if(j.eq.nya)then
                dhddy=.5*((hmean(ia+1,jd+1)-hmean(ia+1,jd))/dy+
     c            (hmean(ia,jd+1)-hmean(ia,jd))/dy)
                dhcdy=.5*((hmean(ia+1,ja+1)-hmean(ia+1,ja))/dy+
     c            (hmean(ia,ja+1)-hmean(ia,ja))/dy)
                Rhfd=Rfh(m,n-nxa)
                Rhfc=Rfh(m,n)
              endif
* interior
              if(j.gt.1.and.j.lt.nya)then
                dhddy=.5*((hmean(ia+1,jd+1)-hmean(ia+1,jd))/dy+
     c            (hmean(ia,jd+1)-hmean(ia,jd))/dy)
                dhudy=.5*((hmean(ia+1,ju+1)-hmean(ia+1,ju))/dy+
     c            (hmean(ia,ju+1)-hmean(ia,ju))/dy)
                dhcdy=.5*((hmean(ia+1,ja+1)-hmean(ia+1,ja))/dy+
     c            (hmean(ia,ja+1)-hmean(ia,ja))/dy)
                Rhfd=Rfh(m,n-nxa)
                Rhfu=Rfh(m,n+nxa)
                Rhfc=Rfh(m,n)
              endif
* forward or backward or centered difference options
              if(i.eq.1)then
                term1=(Rhfr*dhrdx-Rhfc*dhcdx)/dxa
c                write(*,13) i,j,term1
              elseif(i.eq.nxa)then
                term1=(Rhfc*dhcdx-Rhfl*dhldx)/dxa
c                write(*,13) i,j,term1
              else
                term1=(Rhfr*dhrdx-Rhfl*dhldx)/(2*dxa)
                term1f=(Rhfr*dhrdx-Rhfc*dhcdx)/dxa
                term1b=(Rhfc*dhcdx-Rhfl*dhldx)/dxa
c                write(*,14) i,j,term1,term1f,term1b
              endif
              if(j.eq.1)then
                term2=(Rhfu*dhudy-Rhfc*dhcdy)/dya
c                write(*,13) i,j,term2
              elseif(j.eq.nya)then
                term2=(Rhfc*dhcdy-Rhfd*dhddy)/dya
c                write(*,13) i,j,term2
              else
                term2=(Rhfu*dhudy-Rhfd*dhddy)/(2*dya)
                term2u=(Rhfu*dhudy-Rhfc*dhcdy)/dya
                term2d=(Rhfc*dhcdy-Rhfd*dhddy)/dya
c                write(*,14) i,j,term2,term2u,term2d
              endif
* well location for differencing choices
c             if(i.eq.kw.and.j.eq.lw) then
c                term1=(term1f+term1b)/2.
c                term2=(term2u+term2d)/2.
c             endif
c             if(i.eq.kw.and.j.eq.lw+1) term2=term2u
c             if(i.eq.kw-1.and.j.eq.lw) term1=term1b
```

```
c               if(i.eq.kw+1.and.j.eq.lw) term1=term1f
c               if(i.eq.kw.and.j.eq.lw-1) term2=term2d
                if(nar.eq.1)then
                    b(n)=dt*tran*(term1+term2)+Rhh(n,m,1)*stor
                elseif(nar.eq.2)then
                    b(n)=dt*tran*(term1+term2)+Rhh(m,n,2)*stor
                else
                    print*,'error in loadrhh'
                endif
            enddo
        enddo
c   13 format(2i5,f15.6)
c   14 format(2i5,3f15.6)
        return
        end
*****************************************************************
        real*8 function expliciteval(i,j,m,n,bound)
        implicit double precision (a-h,o-z)
* returns the del squared (Rh(n),(m)) where n is row of column m
        character bound(4)
        include 'lhamana.inc'
        integer i,j,m,n,nxa,nya
        real*8 derx,delx,deuy,dedy,dxa,dya,term1,term2
        dxa=dx*real(nob)
        dya=dy*real(nob)
        nxa=nx/nob
        nya=ny/nob
        if(i.eq.1)then
            if(bound(2).eq.'d') delx=2.*Rhh(n,m,1)/dxa
            if(bound(2).eq.'n') delx=0.
            derx=(Rhh(n+1,m,1)-Rhh(n,m,1))/dxa
        elseif(i.eq.nxa)then
            if(bound(3).eq.'d') derx=-2.*Rhh(n,m,1)/dxa
            if(bound(3).eq.'n') derx=0.
            delx=(Rhh(n,m,1)-Rhh(n-1,m,1))/dxa
        else
            derx=(Rhh(n+1,m,1)-Rhh(n,m,1))/dxa
            delx=(Rhh(n,m,1)-Rhh(n-1,m,1))/dxa
        endif
        if(j.eq.1)then
            if(bound(4).eq.'d') dedy=2.*Rhh(n,m,1)/dya
            if(bound(4).eq.'n') dedy=0.
            deuy=(Rhh(n+nxa,m,1)-Rhh(n,m,1))/dya
        elseif(j.eq.nya)then
            if(bound(1).eq.'d') deuy=-2.*Rhh(n,m,1)/dya
            if(bound(1).eq.'n') deuy=0.
            dedy=(Rhh(n,m,1)-Rhh(n-nxa,m,1))/dya
        else
            dedy=(Rhh(n,m,1)-Rhh(n-nxa,m,1))/dya
            deuy=(Rhh(n+nxa,m,1)-Rhh(n,m,1))/dya
        endif
        if(i.eq.1)then
            if(bound(2).eq.'d') term1=4.*(derx-delx)/(3.*dxa)
            if(bound(2).eq.'n') term1=(derx-delx)/dxa
        elseif(i.eq.nxa)then
            if(bound(3).eq.'d') term1=4.*(derx-delx)/(3.*dxa)
            if(bound(3).eq.'n') term1=(derx-delx)/dxa
        else
            term1=(derx-delx)/dxa
        endif
```

```
      if(j.eq.1)then
          if(bound(4).eq.'d') term2=4.*(deuy-dedy)/(3.*dya)
          if(bound(4).eq.'n') term2=(deuy-dedy)/dya
      elseif(j.eq.nya)then
          if(bound(1).eq.'d') term2=4.*(deuy-dedy)/(3.*dya)
          if(bound(1).eq.'n') term2=(deuy-dedy)/dya
      else
          term2=(deuy-dedy)/dya
      endif
      expliciteval=term1+term2
c      write(*,7) i,j,term1,term2,term1+term2
c    7 format(2i5,3f14.6)
      return
      end
*****************************************************************
c subroutine setupr for solving actual head given random T
c given some random ln(T) field
      subroutine setupr(sa,ija,bound,tf,stor,dt)
      implicit double precision (a-h,o-z)
      include 'lhamana.inc'
      character bound(4)
      integer i,j,k,n,ija(*)
      real*8 tl,tr,tu,td,amat(max*max,5),tf(max,max),sa(*),stor
c build A coefficient matrix
      do j=1,5
          do i=1,max*max
              amat(i,j)=0.
          enddo
      enddo
      do j=1,ny
          do i=1,nx
              n=i+(j-1)*nx
c constant dx and dy harmonic mean
              if(i.eq.1)then
                  tl=0.
              else
                  tl=2*tf(i-1,j)*tf(i,j)/(tf(i-1,j)+tf(i,j))
              endif
              if(i.eq.nx)then
                  tr=0.
              else
                  tr=2*tf(i+1,j)*tf(i,j)/(tf(i+1,j)+tf(i,j))
              endif
              if(j.eq.ny)then
                  tu=0.
              else
                  tu=2*tf(i,j+1)*tf(i,j)/(tf(i,j+1)+tf(i,j))
              endif
              if(j.eq.1)then
                  td=0.
              else
                  td=2*tf(i,j-1)*tf(i,j)/(tf(i,j-1)+tf(i,j))
              endif
c neumann conditions bound=n should be taken care of by the
c zeros at the edges of the tf field
              amat(n,3)=stor+dt*((tl+tr)/dx**2+(tu+td)/dy**2)
              amat(n,4)=-dt*tr/dx**2
              amat(n,5)=-dt*tu/dy**2
              if(n.gt.1) amat(n,2)=-dt*tl/dx**2
              if(n.gt.nx) amat(n,1)=-dt*td/dy**2
```

```
c replace coefficients for dirchlet conditions corner and edge

c lower left
               if((i.eq.1).and.(j.eq.1)) then
                   if((bound(2).eq.'d').and.(bound(4).eq.'d'))then
                       amat(n,3)=stor+dt*(4./3.*(2.*tf(i,j)
     c                  +tr)/dx**2+4./3.*(tu+2.*tf(i,j))/dy**2)
                       amat(n,4)=-dt*tr*4./(3.*dx**2)
                       amat(n,5)=-dt*tu*4./(3.*dy**2)
                   elseif((bound(2).eq.'d').and.(bound(4).eq.'n'))then
                       amat(n,3)=stor+dt*(4./3.*(2.*tf(i,j)
     c                  +tr)/dx**2+tu/dy**2)
                       amat(n,4)=-dt*tr*4./(3.*dx**2)
                   elseif((bound(2).eq.'n').and.(bound(4).eq.'d'))then
                       amat(n,3)=stor+dt*(
     c                  tr/dx**2+4./3.*(tu+2.*tf(i,j))/dy**2)
                       amat(n,5)=-dt*tu*4./(3.*dy**2)
                   endif
               endif
c left edge
               if((i.eq.1).and.(j.gt.1).and.(j.lt.ny))then
                   if(bound(2).eq.'d')then
                       amat(n,3)=stor+dt*(4./3.*(2.*tf(i,j)
     c                  +tr)/dx**2+(tu+td)/dy**2)
                       amat(n,4)=-dt*tr*4./(3.*dx**2)
                   endif
               endif
c upper left
               if((i.eq.1).and.(j.eq.ny))then
                   if((bound(1).eq.'d').and.(bound(2).eq.'d'))then
                       amat(n,3)=stor+dt*(4./3.*(2.*tf(i,j)
     c                  +tr)/dx**2+4./3.*(td+2.*tf(i,j))/dy**2)
                       amat(n,4)=-dt*tr*4./(3.*dx**2)
                       amat(n,1)=-dt*td*4./(3.*dy**2)
                   elseif((bound(1).eq.'d').and.(bound(2).eq.'n'))then
                       amat(n,3)=stor+dt*(
     c                  tr/dx**2+4./3.*(td+2.*tf(i,j))/dy**2)
                       amat(n,1)=-dt*td*4./(3.*dy**2)
                   elseif((bound(1).eq.'n').and.(bound(2).eq.'d'))then
                       amat(n,3)=stor+dt*(4./3.*(2.*tf(i,j)
     c                  +tr)/dx**2+td/dy**2)
                       amat(n,4)=-dt*tr*4./(3.*dx**2)
                   endif
               endif
c top
               if((i.gt.1).and.(i.lt.nx).and.(j.eq.ny))then
                   if(bound(1).eq.'d')then
                       amat(n,3)=stor+dt*((tl
     c                  +tr)/dx**2+4./3.*(td+2*tf(i,j))/dy**2)
                       amat(n,1)=-dt*td*4./(3.*dy**2)
                   endif
               endif
c upper right
               if((i.eq.nx).and.(j.eq.ny))then
                   if((bound(1).eq.'d').and.(bound(3).eq.'d'))then
                       amat(n,3)=stor+dt*(4./3.*(2.*tf(i,j)
     c                  +tl)/dx**2+4./3.*(td+2.*tf(i,j))/dy**2)
                       amat(n,2)=-dt*tl*4./(3.*dx**2)
                       amat(n,1)=-dt*td*4./(3.*dy**2)
                   elseif((bound(1).eq.'d').and.(bound(3).eq.'n'))then
```

```
                    amat(n,3)=stor+dt*(
     c                tl/dx**2+4./3.*(td+2.*tf(i,j))/dy**2)
                    amat(n,1)=-dt*td*4./(3.*dy**2)
                elseif((bound(1).eq.'n').and.(bound(3).eq.'d'))then
                    amat(n,3)=stor+dt*(4./3.*(2.*tf(i,j)
     c               +tl)/dx**2+td/dy**2)
                    amat(n,2)=-dt*tl*4./(3.*dx**2)
                endif
            endif
c right
            if((i.eq.nx).and.(j.gt.1).and.(j.lt.ny))then
                if(bound(3).eq.'d')then
                    amat(n,3)=stor+dt*(4./3.*(2.*tf(i,j)
     c               +tl)/dx**2+(td+tu)/dy**2)
                    amat(n,2)=-dt*tl*4./(3.*dx**2)
                endif
            endif
c lower right
            if((i.eq.nx).and.(j.eq.1))then
                if((bound(3).eq.'d').and.(bound(4).eq.'d'))then
                    amat(n,3)=stor+dt*(4./3.*(2.*tf(i,j)
     c               +tl)/dx**2+4./3.*(tu+2.*tf(i,j))/dy**2)
                    amat(n,2)=-dt*tl*4./(3.*dx**2)
                    amat(n,5)=-dt*tu*4./(3.*dy**2)
                elseif((bound(3).eq.'d').and.(bound(4).eq.'n'))then
                    amat(n,3)=stor+dt*(4./3.*(2.*tf(i,j)
     c               +tl)/dx**2+tu/dy**2)
                    amat(n,2)=-dt*tl*4./(3.*dx**2)
                elseif((bound(3).eq.'n').and.(bound(4).eq.'d'))then
                    amat(n,3)=stor+dt*(
     c               tl/dx**2+4./3.*(2.*tf(i,j)+tu)/dy**2)
                    amat(n,5)=-dt*tu*4./(3.*dy**2)
                endif
            endif
c bottom
            if((i.gt.1).and.(i.lt.nx).and.(j.eq.1))then
                if(bound(4).eq.'d')then
                    amat(n,3)=stor+dt*((tr
     c               +tl)/dx**2+4./3.*(tu+2.*tf(i,j))/dy**2)
                    amat(n,5)=-dt*tu*4./(3.*dy**2)
                endif
            endif
        enddo
      enddo
* store matrix in row indexed sparse storage mode for NR
* subroutine linbcg.  Reference Numerical Recipies 1992 pg 71.
      n=nx*ny
      do 11 j=1,n
        sa(j)=amat(j,3)
11    continue
      ija(1)=n+2
      k=n+1
c here the i index is for rows of the coefficient matrix a, j is col
      do 13 i=1,n
          j=i
          if(amat(i,1).ne.0.)then
              k=k+1
              sa(k)=amat(i,1)
              ija(k)=j-nx
          endif
```

```
          if(amat(i,2).ne.0.)then
              k=k+1
              sa(k)=amat(i,2)
              ija(k)=j-1
          endif
          if(amat(i,4).ne.0)then
              k=k+1
              sa(k)=amat(i,4)
              ija(k)=j+1
          endif
          if(amat(i,5).ne.0)then
              k=k+1
              sa(k)=amat(i,5)
              ija(k)=j+nx
          endif
12      continue
        ija(i+1)=k+1
13    continue
      return
      end
*****************************************************************
      subroutine loadfr(fload,bound,hbound,tf,stor,dt)
      implicit double precision (a-h,o-z)
      include 'lhamana.inc'
      character bound(4)
      integer i,j,n
      real*8 fload(*),hbound(max,4),tf(max,max),stor
      do i=1,nx
          do j=1,ny
              n=i+(j-1)*nx
c load from wells, recharge and last time step head
              fload(n)=dt*qwells(i,j)+fload(n)*stor
c add in load from boundary conditions corner and edge
c lower left
              if((i.eq.1).and.(j.eq.1))then
                  if((bound(2).eq.'d').and.(bound(4).eq.'d'))then
                      fload(n)=fload(n)+8./3.*dt*tf(i,j)*
     c                 (hbound(j,2)/dx**2+hbound(i,4)/dy**2)
                  elseif((bound(2).eq.'n').and.(bound(4).eq.'d'))then
                      fload(n)=fload(n)+dt*
     c                 (-hbound(j,2)/dx+8./3.*tf(i,j)*hbound(i,4)/dy**2)
                  elseif((bound(2).eq.'d').and.(bound(4).eq.'n'))then
                      fload(n)=fload(n)+dt*
     c                 (8./3.*tf(i,j)*hbound(j,2)/dx**2-hbound(i,4)/dy)
                  elseif((bound(2).eq.'n').and.(bound(4).eq.'n'))then
                      fload(n)=fload(n)+dt*
     c                 (-hbound(j,2)/dx-hbound(i,4)/dy)
                  else
                     print*,'error in boundary conditions LL'
                  endif
              endif
c left
              if((i.eq.1).and.(j.gt.1).and.(j.lt.ny))then
                  if(bound(2).eq.'d')then
                      fload(n)=fload(n)+8./3.*dt*tf(i,j)*
     c                 hbound(j,2)/dx**2
                  elseif(bound(2).eq.'n')then
                      fload(n)=fload(n)-dt*hbound(j,2)/dx

                  else
```

```
                    print*,'error in boundary conditions L'
                endif
            endif
c upper left
            if((i.eq.1).and.(j.eq.ny))then
                if((bound(2).eq.'d').and.(bound(1).eq.'d'))then
                    fload(n)=fload(n)+8./3.*dt*tf(i,j)*
     c               (hbound(j,2)/dx**2+hbound(i,1)/dy**2)
                elseif((bound(2).eq.'n').and.(bound(1).eq.'d'))then
                    fload(n)=fload(n)+dt*
     c               (-hbound(j,2)/dx+8./3.*tf(i,j)*hbound(i,1)/dy**2)
                elseif((bound(2).eq.'d').and.(bound(1).eq.'n'))then
                    fload(n)=fload(n)+dt*
     c               (8./3.*tf(i,j)*hbound(j,2)/dx**2-hbound(i,1)/dy)
                elseif((bound(2).eq.'n').and.(bound(1).eq.'n'))then
                    fload(n)=fload(n)+dt*
     c               (-hbound(j,2)/dx-hbound(i,1)/dy)
                else
                    print*,'error in boundary conditions UL'
                endif
            endif
c top
            if((i.gt.1).and.(i.lt.nx).and.(j.eq.ny))then
                if(bound(1).eq.'d')then
                    fload(n)=fload(n)+8./3.*dt*tf(i,j)*
     c               hbound(i,1)/dy**2
                elseif(bound(1).eq.'n')then
                    fload(n)=fload(n)-dt*hbound(i,1)/dy

                else
                    print*,'error in boundary conditions T'
                endif
            endif
c upper right
            if((i.eq.nx).and.(j.eq.ny))then
                if((bound(3).eq.'d').and.(bound(1).eq.'d'))then
                    fload(n)=fload(n)+8./3.*dt*tf(i,j)*
     c               (hbound(j,3)/dx**2+hbound(i,1)/dy**2)
                elseif((bound(3).eq.'n').and.(bound(1).eq.'d'))then
                    fload(n)=fload(n)+dt*
     c               (-hbound(j,3)/dx+8./3.*tf(i,j)*hbound(i,1)/dy**2)
                elseif((bound(3).eq.'d').and.(bound(1).eq.'n'))then
                    fload(n)=fload(n)+dt*
     c               (8./3.*tf(i,j)*hbound(j,3)/dx**2-hbound(i,1)/dy)
                elseif((bound(3).eq.'n').and.(bound(1).eq.'n'))then
                    fload(n)=fload(n)+dt*
     c               (-hbound(j,3)/dx-hbound(i,1)/dy)
                else
                    print*,'error in boundary conditions UR'
                endif
            endif
c right
            if((i.eq.nx).and.(j.lt.ny).and.(j.gt.1))then
                if(bound(3).eq.'d')then
                    fload(n)=fload(n)+8./3.*dt*tf(i,j)*
     c               hbound(j,3)/dx**2
                elseif(bound(3).eq.'n')then
                    fload(n)=fload(n)-dt*hbound(j,3)/dx

                else
```

```
                    print*,'error in boundary conditions R'
                endif
            endif
c lower right
            if((i.eq.nx).and.(j.eq.1))then
                if((bound(3).eq.'d').and.(bound(4).eq.'d'))then
                    fload(n)=fload(n)+8./3.*dt*tf(i,j)*
     c               (hbound(j,3)/dx**2+hbound(i,4)/dy**2)
                elseif((bound(3).eq.'n').and.(bound(4).eq.'d'))then
                    fload(n)=fload(n)+dt*
     c               (-hbound(j,3)/dx+8./3.*tf(i,j)*hbound(i,4)/dy**2)
                elseif((bound(3).eq.'d').and.(bound(4).eq.'n'))then
                    fload(n)=fload(n)+dt*
     c               (8./3.*tf(i,j)*hbound(j,3)/dx**2-hbound(i,4)/dy)
                elseif((bound(3).eq.'n').and.(bound(4).eq.'n'))then
                    fload(n)=fload(n)+dt*
     c               (-hbound(j,3)/dx-hbound(i,4)/dy)
                else
                    print*,'error in boundary conditions LR'
                endif
            endif
c bottom
            if((i.gt.1).and.(i.lt.nx).and.(j.eq.1))then
                if(bound(4).eq.'d')then
                    fload(n)=fload(n)+8./3.*dt*tf(i,j)*
     c               hbound(i,4)/dx**2
                elseif(bound(4).eq.'n')then
                    fload(n)=fload(n)-dt*hbound(i,4)/dx
                else
                    print*,'error in boundary conditions B'
                endif
            endif
        enddo
    enddo
    return
    end
```

A.3.4 Polynomial interpolation routine

```
      subroutine extract(datmat,cfhvect,idocfh)
* Extract the interpolated data covariance matrix and the
* interpolated estimation location cross-covariance vector
* from the big storage arrays.
      implicit double precision (a-h,o-z)
      include 'lhamana.inc'
      integer i,j,k,m,n,nxa,nya,ii,jj,nn,l,kk,ll,idocfh
      real*8 dxa,dya,datmat(2,ndd,ndd),cfhvect(max*max,ndd)
      dxa=dx*real(nob)
      dya=dy*real(nob)
      nxa=nx/nob
      nya=ny/nob
      do i=1,nfdata
          k=int(fdata(i,1)/dx)+1
          l=int(fdata(i,2)/dy)+1
          kk=int(fdata(i,1)/dxa)+1
          ll=int(fdata(i,2)/dxa)+1
          m=(int(fdata(i,1)/dxa)+1)+(int(fdata(i,2)/dya))*nxa

          do j=1,nhdata
* Interpolate between coarse grid blocks to approximate fine
* grid covariances.
              n=(int(hdata(j,1)/dxa)+1)+(int(hdata(j,2)/dya))*nxa
              call polyfh(k,l,j,kk,ll,n,m,ans)
c             datmat(id)=Rfh(n,m)
              datmat(1,i,j)=ans
          enddo
      enddo
      do i=1,nhdata
          do j=1,nhdata
              m=(int(hdata(i,1)/dxa)+1)+(int(hdata(i,2)/dya))*nxa

              n=(int(hdata(j,1)/dxa)+1)+(int(hdata(j,2)/dya))*nxa

              call polyhh(n,j,m,i,ans)
c              datmat(id)=Rhh(n,m,1)
              datmat(2,i,j)=ans
          enddo
      enddo
* Extract interpolated cross-covariance vector for each estimation
* location for each h datum if idocfh=1
      if(idocfh.eq.1)then
          do j=1,ny
              do i=1,nx
                  n=i+(j-1)*nx
                  ii=(i+(nob-1))/nob
                  jj=(j+(nob-1))/nob
                  nn=ii+(jj-1)*nxa
                  do k=1,nhdata
                     m=(int(hdata(k,1)/dxa)+1)+(int(hdata(k,2)/dya))*nxa
                      call polyfh(i,j,k,ii,jj,m,nn,ans)
c                 cfhvect(id)=Rfh(m,nn)
                      cfhvect(n,k)=ans
                  enddo
```

```
            enddo
          enddo
        endif
        end
*****************************************************************
        subroutine polyfh(i,j,kh,ii,jj,m,n,ans)
* i,j are fine grid indices of f block, kh is hdata number
* ii,jj are coarse grid block containing f block indices
* m is coarse grid block number containing h datum
* n is coarse grid block number containing f fine block
* x,y coordinates of f fine block are determined from i,j
* x,y coordinates of h fine block are from kh and hdata(kh,1:2)
        implicit double precision (a-h,o-z)
        include 'lhamana.inc'
        integer i,j,m,ii,jj,n,nxa,nya,kh,l,nh,nfx,nfy
        real*8 ans,xf,yf,dxa,dya,fh(9,9),fx(3),fy(3),hx(3),hy(3),
      c       xh,yh,tmpx(3),tmpy(3),er,festh(9)
        dxa=dx*real(nob)
        dya=dy*real(nob)
        nxa=nx/nob
        nya=ny/nob
* extract neighboring values from the Rfh matrix
* case where coarse grid block containing f fine block is interior
* use a quadratic, n is cgrid block with f, m is cgrid blk w/ h
        if(ii.gt.1.and.ii.lt.nxa.and.jj.gt.1.and.jj.lt.nya)then
            call fetchc(-1,1,-1,1,fh,m,n,fx,fy,hx,hy,nfx,nfy)
* on the corners, use bilinear
* lower left
        elseif(ii.eq.1.and.jj.eq.1)then
            call fetchc(0,1,0,1,fh,m,n,fx,fy,hx,hy,nfx,nfy)
*lower right
        elseif(ii.eq.nxa.and.jj.eq.1)then
            call fetchc(0,1,-1,0,fh,m,n,fx,fy,hx,hy,nfx,nfy)
* upper left
        elseif(ii.eq.1.and.jj.eq.nya)then
            call fetchc(-1,0,0,1,fh,m,n,fx,fy,hx,hy,nfx,nfy)
* upper right
        elseif(ii.eq.nxa.and.jj.eq.nya)then
            call fetchc(-1,0,-1,0,fh,m,n,fx,fy,hx,hy,nfx,nfy)
* on the edges use a combination bilinear-quadratic
* bottom edge
        elseif(ii.gt.1.and.ii.lt.nxa.and.jj.eq.1)then
            call fetchc(0,1,-1,1,fh,m,n,fx,fy,hx,hy,nfx,nfy)
* right edge
        elseif(ii.eq.nxa.and.jj.gt.1.and.jj.lt.nya)then
            call fetchc(-1,1,-1,0,fh,m,n,fx,fy,hx,hy,nfx,nfy)
* left edge
        elseif(ii.eq.1.and.jj.gt.1.and.jj.lt.nya)then
            call fetchc(-1,1,0,1,fh,m,n,fx,fy,hx,hy,nfx,nfy)
* top edge
        elseif(ii.gt.1.and.ii.lt.nxa.and.jj.eq.nya)then
            call fetchc(-1,0,-1,1,fh,m,n,fx,fy,hx,hy,nfx,nfy)
        endif
* with up to 81 values of covariance and locations of blocks
* interpolate to f location with h datum
* f block is located at i,j.  Find the x,y coordinates
        xf=real(i-1)*dx+dx/2.
        yf=real(j-1)*dy+dy/2.
        xh=hdata(kh,1)
        yh=hdata(kh,2)
```

```
c      print*,kh
c      print*,hdata(kh,1),hdata(kh,2),hdata(kh,3)
* interpolate f 1...nfx*nfy for each h
      do nh=1,9
          do k=1,nfy
              do l=1,nfx
                  tmpx(l)=fh(l+nfx*(k-1),nh)
              enddo
              call polint(fx,tmpx,nfx,xf,tmpy(k),er)
          enddo
          call polint(fy,tmpy,nfy,yf,festh(nh),er)
      enddo
* now interpolate between the h locations
      do k=1,3
          do l=1,3
              tmpx(l)=festh(l+3*(k-1))
          enddo
          call polint(hx,tmpx,3,xh,tmpy(k),er)
      enddo
      call polint(hy,tmpy,3,yh,ans,er)
      return
      end
*****************************************************************

      subroutine fetchc(inkb,inke,inlb,inle,fh,m,n,fx,fy,hx,hy,nfx,nfy)
      implicit double precision (a-h,o-z)
      include 'lhamana.inc'
      integer m,n,nxa,nya,k,l,nf,mm,nn,nh,inkb,inke,inlb,inle
      integer nfx,nfy,iif,jjf,iih,jjh,nb,mb
      real*8 fh(9,9),dxa,dya,fx(3),hx(3),fy(3),hy(3)
      nxa=nx/nob
      nya=ny/nob
      dxa=dx*real(nob)
      dya=dy*real(nob)
      nh=1
      do mm=-1,1,1
          do nn=-1,1,1
              nf=1
              mb=m+nxa*mm+nn
              do k=inkb,inke,1
                  do l=inlb,inle,1
                      nb=n+nxa*k+1
                      fh(nf,nh)=Rfh(mb,nb)
* n+nxa*k+1 is the block number on cgrid for the f loc
* m+nxa*mm+nn is block number on cgrid for h datum
                      nf=nf+1
                  enddo
              enddo
              nh=nh+1
          enddo
      enddo
* x and y coordinates of blocks around the h datum
      nh=1
      do mm=-1,1,1
          mb=m+nxa*mm
          jjh=(mb-1)/nxa+1
          hy(nh)=real(jjh-1)*dya+dya/2.
          nh=nh+1
      enddo
      nh=1
```

```
      do nn=-1,1,1
          mb=m+nn
          iih=mod(mb,nxa)
          if(iih.eq.0)iih=nxa
          hx(nh)=real(iih-1)*dxa+dxa/2.
          nh=nh+1
      enddo
* x and y coordinates of blocks around the f datum
      nf=1
      do k=inkb,inke,1
          nb=n+nxa*k
          jjf=(nb-1)/nxa+1
          fy(nf)=real(jjf-1)*dya+dya/2.
          nf=nf+1
      enddo
      nf=1
      do l=inlb,inle,1
          nb=n+l
          iif=mod(nb,nxa)
          if(iif.eq.0)iif=nxa
          fx(nf)=real(iif-1)*dxa+dxa/2.
          nf=nf+1
      enddo
c      print*,'hx',hx
c      print*,'hy',hy
c      print*,'fx',fx
c      print*,'fy',fy
c      print*,m,n
      nfy=abs(inkb)+abs(inke)+1
      nfx=abs(inlb)+abs(inle)+1
      return
      end
*****************************************************************
      subroutine polyhh(m,k2,n,k1,ans)
* m is coarse block of h2, n is coarse block of h1
* k2 and k1 are data numbers of hdata
      implicit double precision (a-h,o-z)
      include 'lhamana.inc'
      integer nxa,nya,nh1,mm,nn,nh2,mb,k,l,nb
      real*8 ans,dxa,dya,hh(9,9),h1x(3),h1y(3),h2x(3),h2y(3),
     c       x1h,y1h,x2h,y2h,tmpx(3),tmpy(3),er,hesth(9)
      dxa=dx*real(nob)
      dya=dy*real(nob)
      nxa=nx/nob
      nya=ny/nob
* extract neighboring values from the Rhh matrix
      nh2=1
      do mm=-1,1,1
          do nn=-1,1,1
              nh1=1
              mb=m+nxa*mm+nn
              do k=-1,1,1
                  do l=-1,1,1
                      nb=n+nxa*k+l
                      hh(nh1,nh2)=Rhh(mb,nb,1)
* n+nxa*k+l is the block number on cgrid for the h1 datum
* m+nxa*mm+nn is block number on cgrid for h2 datum
                      nh1=nh1+1
                  enddo
              enddo
```

```
            nh2=nh2+1
          enddo
        enddo
* x and y coordinates of blocks around the h2 datum
        nh2=1
        do mm=-1,1,1
          mb=m+nxa*mm
          jjh=(mb-1)/nxa+1
          h2y(nh2)=real(jjh-1)*dya+dya/2.
          nh2=nh2+1
        enddo
        nh2=1
        do nn=-1,1,1
          mb=m+nn
          iih=mod(mb,nxa)
          if(iih.eq.0)iih=nxa
          h2x(nh2)=real(iih-1)*dxa+dxa/2.
          nh2=nh2+1
        enddo
* x and y coordinates of blocks around the h1 datum
        nh1=1
        do k=-1,1,1
          nb=n+nxa*k
          jjf=(nb-1)/nxa+1
          h1y(nh1)=real(jjf-1)*dya+dya/2.
          nh1=nh1+1
        enddo
        nh1=1
        do l=-1,1,1
          nb=n+1
          iif=mod(nb,nxa)
          if(iif.eq.0)iif=nxa
          h1x(nh1)=real(iif-1)*dxa+dxa/2.
          nh1=nh1+1
        enddo
* interpolate with polint
        x1h=hdata(k1,1)
        y1h=hdata(k1,2)
        x2h=hdata(k2,1)
        y2h=hdata(k2,2)
* interpolate h1 1...nfx*nfy for each h2
        do nh2=1,9
          do k=1,3
            do l=1,3
              tmpx(l)=hh(l+3*(k-1),nh2)
            enddo
            call polint(h1x,tmpx,3,x1h,tmpy(k),er)
          enddo
          call polint(h1y,tmpy,3,y1h,hesth(nh2),er)
        enddo
* now interpolate between the h2 locations
        do k=1,3
          do l=1,3
            tmpx(l)=hesth(l+3*(k-1))
          enddo
          call polint(h2x,tmpx,3,x2h,tmpy(k),er)
        enddo
        call polint(h2y,tmpy,3,y2h,ans,er)
        return
        end
```

A.3.5 Cokriging routine

```
* Cokriging subroutine
* INPUT  ndata -- number of f and h data.
*        modn  -- model number for ff covariance.
*        datp  -- vector containing f and h data.
*        datmat-- matrix of data fh and hh covariances.
*        cfhvect- vector of estimation location fh covariance.
* OUTPUT fk -- field cokriged on f and h data in datp.
*
* Cokrig calls a singular value decomposition to decompose
* the data matrix then calculates cokriging weights for each
* f estimation location.
*
      subroutine cokrig(ndata,modn,datp,fk,cfh,chh,cfhvect,nts)
      implicit double precision (a-h,o-z)
* cokrigs f given data on f and h and the cross/covariances
* f data come first in the data value vector ndatp
      include 'lhamana.inc'
      integer i,j,k,n,np,kk,kappa
      real*8 a(ndd+ndd*(maxntimes+1),ndd+ndd*(maxntimes+1))
      real*8 u(ndd+ndd*(maxntimes+1),ndd+ndd*(maxntimes+1))
      real*8 w(ndd+ndd*(maxntimes+1))
      real*8 v(ndd+ndd*(maxntimes+1),ndd+ndd*(maxntimes+1))
      real*8 b(ndd+ndd*(maxntimes+1)),x(ndd+ndd*(maxntimes+1))
      real*8 fk(max*max),datp(ndata)
      real*8 x1,y1,x2,y2,wtsum,wmax,wmin
      real*8 cfhvect(max*max,ndd,0:maxntimes)
      real*8 cfh(ndd,ndd,0:maxntimes),chh(ndd,ndd,0:(maxntimes+1)**2)

      np=ndd+ndd*(maxntimes+1)
c      do i=1,ndata
c          write(*,*) i,datp(i)
c      enddo
c      write(*,*) ' '
* assemble the data covariance matrix
      do i=1,nfdata
          do j=1,nfdata
              x1=fdata(i,1)
              y1=fdata(i,2)
              x2=fdata(j,1)
              y2=fdata(j,2)
              a(i,j)=covar(x2,x1,y2,y1,modn)
          enddo
      enddo
c      write(*,*) 'cfh matrix'
c      write(*,*) ' '
      do kk=0,nts
c          write(*,*) 'time step ',kk
c          write(*,*) ' '
          do i=1,nfdata
              do j=1,nhdata
                  a(i,j+nfdata*(kk+1))=cfh(i,j,kk)
                  a(j+nfdata*(kk+1),i)=cfh(i,j,kk)
c                  temp(j)=cfh(i,j,kk)
```

```
                enddo
c            write(*,11) (temp(j),j=1,4)
            enddo
c            write(*,*) ' '
        enddo
c     11 format(4f12.6)
        kappa=0
c        write(*,*) 'chh matrix'
c        write(*,*) ' '
        do kk=0,nts
            do k=kk,nts
c                write(*,*) kk,k
c                write(*,*) ' '
                do i=1,nhdata
                    do j=1,nhdata
                        a(i+nfdata*(kk+1),j+nfdata*(k+1))=chh(i,j,kappa)
                        a(j+nfdata*(k+1),i+nfdata*(kk+1))=chh(i,j,kappa)
c                         temp(j)=chh(i,j,kappa)
                    enddo
c                    write(*,11) (temp(j),j=1,4)
                enddo
c                write(*,*) ' '
                kappa=kappa+1
            enddo
        enddo
        do i=1,ndata
            do j=1,ndata
                u(i,j)=a(i,j)
            enddo
        enddo
        do i=1,nfdata
            u(i,i)=u(i,i)+fer
            a(i,i)=a(i,i)+fer
        enddo
        do i=nfdata+1,ndata
            u(i,i)=u(i,i)+her
            a(i,i)=a(i,i)+her
        enddo
c        open(unit=69,file='matest.out')
c        do j=1,ndata
c            do i=1,ndata
c                u(i,j)=a(i,j)
c                write(69,*) i,j, a(i,j)
c            enddo
c        enddo
c        close(69)
c        write(*,*) 'u-matrix'
c        do i=1,ndata
c            write(*,7) (u(i,j),j=1,ndata)
c        enddo
c     7 format(7f10.3)
* Ship data covariance matrix to singular value decomposition.
        call dsvdcmp(u,ndata,ndata,np,np,w,v)
* Then deal with the residuals. (ref. NR ed.2 p. 57)
        wmax=0.
        do j=1,ndata
            if(w(j).gt.wmax)wmax=w(j)
        enddo
        wmin=wmax*1.0E-12
        do j=1,ndata
```

```
          if(w(j).lt.wmin)w(j)=0.
        enddo
* Cokrig estimate a f value for each block location.
c       bigadiff=0.
        do j=1,ny
          do i=1,nx
            n=i+(j-1)*nx
            x1=real(i-1)*dx+dx/2.
            y1=real(j-1)*dy+dy/2.
* Build estimation location fh cross-covariance load vector.
            do k=1,nfdata
              x2=fdata(k,1)
              y2=fdata(k,2)
              b(k)=covar(x2,x1,y2,y1,modn)
            enddo
            do kk=0,nts
              do k=1,nhdata
                b(k+nfdata*(kk+1))=cfhvect(n,k,kk)
c                if(n.eq.496) write(*,*) cfhvect(n,k,kk)
              enddo
            enddo
* Backsubtitution, b is load, x is solution, no input's
* are modified.
            call dsvbksb(u,w,v,ndata,ndata,np,np,b,x)
c* test multiply check
c             do it=1,ndata
c               tsum=0.
c               do jt=1,ndata
c                 tsum=tsum+a(it,jt)*x(jt)
c                 write(*,*) a(it,jt),b(jt)
c               enddo
c               diff=abs(tsum-b(it))
c               if(bigadiff.lt.diff) bigadiff=diff
c             enddo
* Cokriged estimate for block n.
            fk(n)=0.
            wtsum=0.
            do k=1,ndata
              if(datp(k).lt.1.E+29) fk(n)=fk(n)+datp(k)*x(k)
c               write(*,*) n,k,datp(k)
              wtsum=wtsum+x(k)
            enddo
          enddo
        enddo
c       print*,'maximum svd error ',bigadiff
        return
        end
*****************************************************************
```

A.3.6 Routine for iteratively calculating mean and variance

```
      subroutine var_mean(farray,harray,nsc,cds)
      implicit double precision (a-h,o-z)
      include 'lhamana.inc'
      integer nsc,i,j,n
      real*8 farray(max,max),harray(max*max)
      real*8 cds(max,max,4)
* nsc realizations of the array(x,y)
* iterative accumulate mean and variance on cds array
* cds(x,y,loc)loc = 4
*     1 average farray
*     2 variance array
*     3 average harray
*     4 variance harray
      if(nsc.eq.1)then
          do j=1,ny
              do i=1,nx
                  n=i+(j-1)*nx
                  cds(i,j,1)=farray(i,j)
                  cds(i,j,2)=farray(i,j)
                  cds(i,j,3)=harray(n)
                  cds(i,j,4)=harray(n)
              enddo
          enddo
      else
          do j=1,ny
              do i=1,nx
                  n=i+(j-1)*nx
                  temp=cds(i,j,1)
                  cds(i,j,1)=cds(i,j,1)+(farray(i,j)-cds(i,j,1))
     c            /dble(nsc)
                  cds(i,j,2)=(1.-1./dble(nsc-1))*cds(i,j,2)+
     c            dble(nsc)*(cds(i,j,1)-temp)**2
                  htmp=harray(n)
                  temp=cds(i,j,3)
                  cds(i,j,3)=cds(i,j,3)+(htmp-cds(i,j,3))
     c            /dble(nsc)
                  cds(i,j,4)=(1.-1./dble(nsc-1))*cds(i,j,4)+
     c            dble(nsc)*(cds(i,j,3)-temp)**2
              enddo
          enddo
      endif
      return
      end
```

A.3.7 Routine to set boundary conditions

```
      subroutine setbound(newhbound,ttime)
      implicit double precision (a-h,o-z)
      include 'lhamana.inc'
      real*8 newhbound(max,4),ttime,a1,a2,a3,a4
* hardwired function parameters for sides
      a1=4819.9126-.077161673*ttime**1.6385662
      a2=4927.5568+7.593934E-5*ttime**3
      a3=4957.0389+.066884312*ttime**1.5742475
      a4=4918.4995-.09921578*ttime**1.3725252
c      open(unit=69,file='corner.par')
c      read(69,*) r1,s1
c      read(69,*) r2,s2
c      read(69,*) r3,s3
c      read(69,*) r4,s4
c      close(69)
c      if(ttime.eq.0.)then
c         a1=r1
c         a2=r2
c         a3=r3
c         a4=r4
c      else
c         a1=s1
c         a2=s2
c         a3=s3
c         a4=s4
c      endif
      do i=1,nx
c         newhbound(i,1)=a1
c         newhbound(i,4)=a4
          newhbound(i,1)=(a3-a1)/(real(nx)*dx)*(real(i)*dx-dx/2.)+a1
          newhbound(i,4)=(a4-a2)/(real(nx)*dx)*(real(i)*dx-dx/2.)+a2
      enddo
      do j=1,ny
c         newhbound(j,2)=a2
c         newhbound(j,3)=a3
          newhbound(j,2)=(a1-a2)/(real(ny)*dy)*(real(j)*dy-dy/2.)+a2
          newhbound(j,3)=(a3-a4)/(real(ny)*dy)*(real(j)*dy-dy/2.)+a4
      enddo
      return
      end
```

A.3.8 Include file and sample parameter file

```
* nob should be 4, 8 may work but has not been tested
      parameter(max=64,ndd=20,nob=4,maxnts=500,maxntimes=20)
      common/para/nx,ny
      common/data/nfdata,nhdata,fdata(ndd,3),hdata(ndd,3)
     c            ,htdat(ndd,maxntimes)
      common/opie/dx,dy,varf,corx,cory,fer,her
      common/vectors/qwells(max,max)
      common/matrix/Rfh((max/nob)**2,(max/nob)**2),
     c              Rhh((max/nob)**2,(max/nob)**2,2)
```

```
64      nx nodes on the x-axis
64     ny nodes on the y-axis
88.         ttime total time of simulation time units
1.         dt time step increment time units
1.          tsm time step multiplier
800        dx delta x spacing length unit
800        dy delta y spacing length units
597000         tran transmissivity length squared per time
.05         storativity unitless, 0 is steady state
.0       spatially distr. uniform recharge rate length/time
  d
d  d
  d        d is constant head n is prescribed fluxd
6000     correlation in the x direction
6000     correlation in the y direction
.6       input variance
.1       f data measurement error
6.0      h data measurement error
4         model number 2 is exp, 1 is bell, 4 is Mizell-A
823456789  starting random seed
1.E-7  conjugate gradient convergence criterium
3.0E+15  iteration convergence tolerance
100   number of simulations runs
n n n n random f, h mean, Rfh & Rhh, uncond h
1 jack=1 set uniform constant, 2 read per block from q.dat
0  load=0 for steady state load, load.ne.0 uniform or qwells.dat
n y  perturb system by changing load? y/n boundaries? y/n
  d
d  d
  d
y     y to particle track
39600  14800  33200   20400  starting locations
1.  track time, should be time step for transient, total time for ss
.2   porosity
150.   vertical aquifer thickness
```

A.3.9 Main program for sequential conditioning

```
* PROGRAM PEJUTA
* Uses covariances calculated from lhamana and saved on disk to
* condition a random field on f and h data.
      parameter(maxit=2000)
      implicit double precision (a-h,o-z)
      include 'lhamana.inc'
      logical hitb
      character bound(4),outc(4),pert(2),newbound(4),track
      integer i,j,k,nxny,na,nb,modn,ndata,iter,nsims,in,nts,ntrans
      integer ija(max*(5*max-2)),blav,nsd,ntimes,ihdt
      integer nsimsbust,itdiverged,iterc,jack,load,nf,kappa,mtry
      integer matindex(maxntimes),ntss
      real*8 corx,cory,varf,datp(ndd*2),vhat(max*max),himem(max*max)
      real*8 b(max*max),x(max*max),sa(max*(5*max-2)),tol,err
      real*8 stor,tran,tf(max,max),newhbound(max,4),t,temp
      real*8 hbound(max,4),amean,avar,ff(max,max),ssqh,diff
      real*8 xmn,xmx,ymn,ymx,cx,cy,lastssqh,uhat(max*max),tolc
      real*8 vs(max,max),sdr,dt,dx,dy,hcmem(max*max),ss,tsm
      real*8 Tlnm,ttime,husamp(ndd,0:maxntimes),datmat(2,ndd,ndd)
      real*8 cts,cfhvect(max*max,ndd,0:maxntimes),tracktime
      real*8 Htouse(max,max),cfh(ndd,ndd,0:maxntimes)
      real*8 chh(ndd,ndd,0:(maxntimes+1)**2),hdtimes(0:maxntimes)
      real*8 ufdata(ndd),Hallk(max,max,0:maxnts),startploc(4)
      real*8 partrack(max*10,3),voids,thick,cfhv(max*max,ndd)
      real*8 condstats(max,max,4),xcoord,ycoord,aae,aaemax
      real*8 pmem(4),ptim(2)
* get all the necessary parameters
      call inputpar(bound,modn,nsd,tol,tolc,nsims,stor
     c             ,tran,outc,sdr,nts,jack,tsm,dt,load,pert
     c             ,newbound,ntimes,hdtimes
     c             ,track,startploc,tracktime,voids,thick,ntrans)
      ss=0.
      nsc=0
      nxny=nx*ny
      Tlnm=log(tran)
      itmax=maxit
      mtry=25
      na=nx
      nb=ny
      nf=nfdata
      ndata=nfdata+nhdata
      xmn=0.
      ymn=0.
      xmx=real(nx)*dx-dx
      ymx=real(ny)*dy-dx
      vr=varf
      cx=corx
      cy=cory
      blav=0
* Calculate mean head for all time steps
* Set up the mean head coefficient matrix for steady state.
      call setup(sa,ija,bound,ss,tran,0,dt)
      if(load.eq.0)then
```

```
            call setq(1,ss)
        else
            call setq(jack,sdr)
        endif
        call setbound(newhbound,ss)
        hbound=newhbound
        do j=1,ny
            do i=1,nx
                n=i+(j-1)*nx
                Htouse(i,j)=0.
                b(n)=0.
            enddo
        enddo
        call loadf(b,bound,hbound,ss,tran,dt,Htouse)
* Solve for mean head for steady state.
* If the load vector b is zero, no-load, linbcg divides by zero.
        call linbcg(nxny,b,x,1,tol,itmax,iter,err,sa,ija)
* Copy x solution vector into hmean for steady state.
        do j=1,ny
            do i=1,nx
                n=i+(j-1)*nx
                Hallk(i,j,0)=x(n)
                Htouse(i,j)=x(n)
            enddo
        enddo
        cts=dt
        ttime=dt
        do kk=1,nts
            if(pert(1).eq.'y')then
                if(load.eq.0)then
                    call setq(jack,sdr)
                else
                    call setq(1,ss)
                endif
            endif
            if(pert(2).eq.'y')then
                call setbound(newhbound,ttime)
                hbound=newhbound
            endif
* Solve for mean head at time step kk.
            call setup(sa,ija,bound,stor,tran,0,cts)
            call loadf(b,bound,hbound,stor,tran,cts,Htouse)
            call linbcg(nxny,b,x,1,tol,itmax,iter,err,sa,ija)
* Copy b solution vector into hmean for transient step.
            do j=1,ny
                do i=1,nx
                    n=i+(j-1)*nx
                    Hallk(i,j,kk)=x(n)
                    Htouse(i,j)=x(n)
                enddo
            enddo
            ttime=ttime+cts
            cts=cts*tsm
        enddo
* Read in data covariance matrix and estimation location vector
* from file cokcov.in
        open(unit=27,file='cokcov.in')
* Steady state,
        do i=1,nfdata
            do j=1,nhdata
```

```
                read(27,*) cfh(i,j,0)
            enddo
        enddo
        do i=1,nhdata
            do j=1,nhdata
                read(27,*) chh(i,j,0)
            enddo
        enddo
        do j=1,ny
            do i=1,nx
            n=i+(j-1)*nx
                do k=1,nhdata
                    read(27,*) cfhvect(n,k,0)
                enddo
            enddo
        enddo
* Time step cokcov
        kappa=1
        do kk=1,ntimes
            do kkk=kk,ntimes
                do i=1,nhdata
                    do j=1,nhdata
                        read(27,*) chh(i,j,kappa)
                    enddo
                enddo
                kappa=kappa+1
            enddo
            do i=1,nfdata
                do j=1,nhdata
                    read(27,*) cfh(i,j,kk)
                enddo
            enddo
            do i=1,nhdata
                do j=1,nhdata
                    read(27,*) chh(i,j,kappa)
                enddo
            enddo
            matindex(kk)=kappa
            kappa=kappa+1
            do j=1,ny
                do i=1,nx
                    n=i+(j-1)*nx
                    do k=1,nhdata
                        read(27,*) cfhvect(n,k,kk)
                    enddo
                enddo
            enddo
        enddo
        close(27)
* SIMULATION RUN, condition a random f and h field on data of
* each a number of times.  For each conditioned simulation, calc
* and accumulate statistics.  Built in requirement that there is
* the same number of data at the same locations for each time
* step.
        nsimsbust=0
        if(track.eq.'y')then
            open(unit=99,file='ptrack1.out')
            open(unit=100,file='ptrack2.out')
        endif
        do in=1,nsims
```

```
      lastssqh=1.E+7
      itdiverged=0
      iterc=0
* Generate the unconditional random f field with mean zero
* and specified variance using spectral methods and FFT.
      call ranfgen(modn,blav,na,nb,xmn,xmx,ymn,ymx,
     b     vr,cx,cy,nsd,ff,amean,avar)
* collect sample location distributions for unconditional f
      call samplef(ufdata,nf,ff)
* Calculate unconditional head and sample for cokriging.
* Reset the pumping/recharge load conditions
      if(load.eq.0)then
          call setq(1,ss)
      else
          call setq(jack,sdr)
      endif
      call setbound(newhbound,ss)
      hbound=newhbound
      do j=1,ny
          do i=1,nx
              tf(i,j)=dexp(Tlnm+ff(i,j))
          enddo
      enddo
* Setup connectivity matrix using harmonic averages between
* random transmissivity blocks. ss=0 is steady state.
      call setupr(sa,ija,bound,tf,ss,dt)
      do i=1,nxny
          b(i)=0.
      enddo
      call loadfr(b,bound,hbound,tf,ss,dt)
      do j=1,ny
          do i=1,nx
              n=i+(j-1)*nx
              x(n)=Hallk(i,j,0)
          enddo
      enddo
      call linbcg(nxny,b,x,1,tol,itmax,iter,err,sa,ija)
      do n=1,nxny
          hcmem(n)=x(n)
      enddo
* Sample steady state unconditional head field.
      do i=1,nhdata
          k=int(hdata(i,1)/dx)+1
          l=int(hdata(i,2)/dy)+1
          n=k+(l-1)*nx
          if(hdata(i,3).lt.1.E+29)then
             husamp(i,0)=x(n)-Hallk(k,l,0)
          else
             husamp(i,0)=1.E+30
          endif
      enddo
* Now use the covariance matrices for cokriging at steady state.
* Read in the steady state data covariance matrices and estimation
* location cross-covariance vector from file cokcov.out.
      do i=1,nfdata
          do j=1,nhdata
              datmat(1,i,j)=cfh(i,j,0)
          enddo
      enddo
      do i=1,nhdata
```

```
            do j=1,nhdata
                datmat(2,i,j)=chh(i,j,0)
            enddo
        enddo
        do j=1,ny
            do i=1,nx
            n=i+(j-1)*nx
                do k=1,nhdata
                    cfhv(n,k)=cfhvect(n,k,0)
                enddo
            enddo
        enddo
* Cokrig with the original data from files h.dat and f.dat.
        do i=1,nfdata
            datp(i)=fdata(i,3)
        enddo
        do i=1,nhdata
            datp(i+nfdata)=hdata(i,3)
        enddo
        call cokrig(ndata,modn,datp,vhat,datmat,cfhv)
* Cokrig with the unconditional random f samples
        do i=1,nfdata
            datp(i)=ufdata(i)
        enddo
c        print*,'Steady state cokriging iteration'
        do i=1,nhdata
* and unconditional head hu data.
            datp(i+nfdata)=husamp(i,0)
        enddo
        iterc=0
        lastssqh=1.E+7
* The *make it better BETA* iterc loop returns here to 17
* for steady state cokriging iteration.
   17 continue
        call cokrig(ndata,modn,datp,uhat,datmat,cfhv)
* Subtract to get the cokrig co-conditioned random f field.
        do j=1,ny
            do i=1,nx
                n=i+(j-1)*nx
                vs(i,j)=vhat(n)+(ff(i,j)-uhat(n))
            enddo
        enddo
* Solve for new hetero head given random co-conditioned f in
* array vs(i,j).
        do j=1,ny
            do i=1,nx
                n=i+(j-1)*nx
                tf(i,j)=exp(Tlnm+vs(i,j))
                b(n)=0.
                x(n)=Hallk(i,j,0)
            enddo
        enddo
        call setupr(sa,ija,bound,tf,ss,dt)
        call loadfr(b,bound,hbound,tf,ss,dt)
        call linbcg(nxny,b,x,1,tol,itmax,iter,err,sa,ija)
* Check to see how closely the solved/conditioned h perturbations
* match with the hdata.
        ssqh=0.
        aae=0.
        aaemax=0.
```

```
      do i=1,nhdata
          k=int(hdata(i,1)/dx)+1
          l=int(hdata(i,2)/dy)+1
          n=k+(l-1)*nx
          if(hdata(i,3).lt.1.E+29)then
              difft=hdata(i,3)-(x(n)-Hallk(k,l,0))
          else
              difft=0.
          endif
          temp=dabs(difft)
          aae=aae+temp
          if(temp.gt.aaemax) aaemax=temp
          ssqh=ssqh+(difft)**2
      enddo
      iterc=iterc+1
      write(*,1) in,0,iterc,' ASQE ',ssqh/dble(nhdata),
     c                      ' AAE  ',aae/dble(nhdata),
     c                      ' AME  ',aaemax
    1 format(3i4,a5,e12.6,a5,e12.6,a5,e12.6)
* The is the BETA to make the iterated h fields match the data
* better.
      if(aaemax.gt.tolc.and.iterc.lt.maxit) then
          if(ssqh.ge.lastssqh)then
              print*,'ITERATION DIVERGENCE at time step',0,
     c             ' simulation',in
* If steady state diverges, bail out of the simulation.
              nsimsbust=nsimsbust+1
              itdiverged=1
              go to 13
          endif
          lastssqh=ssqh
          do i=1,nhdata
              k=int(hdata(i,1)/dx)+1
              l=int(hdata(i,2)/dy)+1
              n=k+(l-1)*nx
              if(hdata(i,3).lt.1.E+29)then
                  diff=(x(n)-Hallk(k,l,0))-hdata(i,3)
              else
                  diff=0.
              endif
              datp(i+nfdata)=diff+datp(i+nfdata)
          enddo
          go to 17
      endif
      do n=1,nxny
          hcmem(n)=x(n)
          himem(n)=x(n)
      enddo
 93   format(2f10.2,f15.5)
* BEGIN TIME STEP*********************************>>>>>>>>>>>>>>>>
      if(track.eq.'y')then
          hitb=.false.
          pmem=startploc
          ptim(1)=0.
          ptim(2)=0.
          partrack(1,1)=startploc(1)
          partrack(1,2)=startploc(2)
          partrack(1,3)=0.
          write(99,93) partrack(1,1),partrack(1,2)
     c   ,partrack(1,3)
```

```
            write(100,93) pmem(3),pmem(4),0.
        endif
        cts=dt
        ttime=dt
        ihdt=1
        do kk=1,ntrans
            if(pert(1).eq.'y')then
                if(load.eq.0)then
                    call setq(jack,sdr)
                else
                    call setq(1,ss)
                endif
            endif
            if(pert(2).eq.'y')then
                call setbound(newhbound,ttime)
                hbound=newhbound
            endif
* Calculate the time step perturbed head using f conditioned on
* last time data and conditioned h perturbations plus mean from
* last time step.
            do j=1,ny
                do i=1,nx
                    tf(i,j)=exp(Tlnm+vs(i,j))
                enddo
            enddo
            call setupr(sa,ija,bound,tf,stor,cts)
            do n=1,nxny
                b(n)=hcmem(n)
                x(n)=b(n)
            enddo
            call loadfr(b,bound,hbound,tf,stor,cts)
            call linbcg(nxny,b,x,1,tol,itmax,iter,err,sa,ija)
* Go on to next time step if no cokriging is to be done here.
            if(dabs(ttime-hdtimes(ihdt)).lt.tol)then
                t=hdtimes(ihdt-1)
                temp=cts
                ntss=0
                do while(t.lt.hdtimes(ihdt))
                    t=t+temp
                    temp=temp*tsm
                    ntss=ntss+1
                enddo
                print*,'conditioning at time ',hdtimes(ihdt)
* Collect the hu head field samples for this time step.
                do i=1,nhdata
                    k=int(hdata(i,1)/dx)+1
                    l=int(hdata(i,2)/dy)+1
                    n=k+(l-1)*nx
                    if(htdat(i,ihdt).lt.1.E+29)then
                        husamp(i,ihdt)=x(n)-Hallk(k,l,kk)
                    else
                        husamp(i,ihdt)=1.E+30
                    endif
                enddo
* Read in data covariance matrix and estimation location cross-
* covariance vector from disk file cokcov.out.
                do i=1,nfdata
                    do j=1,nhdata
                        datmat(1,i,j)=cfh(i,j,ihdt)
                    enddo
```

```
                enddo
                kappa=matindex(ihdt)
                do i=1,nhdata
                    do j=1,nhdata
                        datmat(2,i,j)=chh(i,j,kappa)
                    enddo
                enddo
                do j=1,ny
                    do i=1,nx
                        n=i+(j-1)*nx
                        do k=1,nhdata
                            cfhv(n,k)=cfhvect(n,k,ihdt)
                        enddo
                    enddo
                enddo
* Cokrig with the original f data
                do i=1,nfdata
                    datp(i)=fdata(i,3)
                enddo
* and the head data for this time
                do i=1,nhdata
                    datp(i+nfdata)=htdat(i,ihdt)
                enddo
                call cokrig(ndata,modn,datp,vhat,datmat,cfhv)
* Reload the unconditional f
                do i=1,nfdata
                    datp(i)=ufdata(i)
                enddo
                do i=1,nhdata
                    datp(i+nfdata)=husamp(i,ihdt)
                enddo
                iterc=0
                lastssqh=1.E+7
* The *make it better BETA* iterc cokriging iteration loop
* returns here to 18 for time step.
   18           continue
                call cokrig(ndata,modn,datp,uhat,datmat,cfhv)
* Subtract to get the cokrig co-conditioned random f field.
                do j=1,ny
                    do i=1,nx
                        n=i+(j-1)*nx
                        vs(i,j)=vhat(n)+(ff(i,j)-uhat(n))
                    enddo
                enddo
* Solve for new hetero head given co-conditioned f field.
                do j=1,ny
                    do i=1,nx
                        n=i+(j-1)*nx
                        tf(i,j)=exp(Tlnm+vs(i,j))
                        Htouse(i,j)=himem(n)
                    enddo
                enddo
                ctss=cts
                do kkk=kk-ntss+1,kk
                    if(pert(1).eq.'y')then
                        if(load.eq.0)then
                            call setq(jack,sdr)
                        else
                            call setq(1,ss)
                        endif
```

```
                endif
                if(pert(2).eq.'y')then
                    call setbound(newhbound,ttime)
                    hbound=newhbound
                endif
                call setupr(sa,ija,bound,tf,stor,ctss)
                do j=1,ny
                    do i=1,nx
                        n=i+(j-1)*nx
                        b(n)=Htouse(i,j)
                        x(n)=b(n)
                    enddo
                enddo
                call loadfr(b,bound,hbound,tf,stor,ctss)
                call linbcg(nxny,b,x,1,tol,itmax,iter,err,sa,ija)

                do j=1,ny
                    do i=1,nx
                        n=i+(j-1)*nx
                        Htouse(i,j)=x(n)
                    enddo
                enddo
                ctss=ctss*tsm
            enddo
* Check to see how close the solved/conditioned h perturbations
* match with the htdat time step data.
            ssqh=0.
            aae=0.
            aaemax=0.
            do i=1,nhdata
                k=int(hdata(i,1)/dx)+1
                l=int(hdata(i,2)/dy)+1
                n=k+(l-1)*nx
                if(htdat(i,ihdt).lt.1.E+29)then
                    difft=htdat(i,ihdt)-(x(n)-Hallk(k,l,kk))
                else
                    difft=0.
                endif
                temp=dabs(difft)
                aae=aae+temp
                if(temp.gt.aaemax) aaemax=temp
                ssqh=ssqh+(difft)**2
            enddo
     write(*,1) in,kk,iterc,' ASQE ',ssqh/dble(nhdata),
    c                       ' AAE  ',aae/dble(nhdata),
    c                       ' AME  ',aaemax

            iterc=iterc+1
   3        format(3i7,a19,e20.10)
* This is the BETA to make the cokriging iteration match the
* time step h data better.
            if(aaemax.gt.tolc.and.iterc.lt.maxit) then

* What to do if the iteration diverges, gets worse.
                if(ssqh.ge.lastssqh)then
                    print*,'ITERATION DIVERGENCE at time step',kk,
    c                   ' simulation',in
                    nsimsbust=nsimsbust+1
                    itdiverged=1
                    go to 13
```

```
                endif
                lastssqh=ssqh
                do i=1,nhdata
                    k=int(hdata(i,1)/dx)+1
                    l=int(hdata(i,2)/dy)+1
                    n=k+(l-1)*nx
                    if(htdat(i,ihdt).lt.1.E+29)then
                        diff=(x(n)-Hallk(k,l,kk))-htdat(i,ihdt)
                    else
                        diff=0.
                    endif
                    datp(i+nfdata)=diff+datp(i+nfdata)
                enddo
                go to 18
            else
                do j=1,ny
                    do i=1,nx
                        n=i+(j-1)*nx
                        Htouse(i,j)=himem(n)
                    enddo
                enddo
                ctss=cts
                do kkk=kk-ntss+1,kk
                    if(pert(1).eq.'y')then
                        if(load.eq.0)then
                            call setq(jack,sdr)
                        else
                            call setq(1,ss)
                        endif
                    endif
                    if(pert(2).eq.'y')then
                        call setbound(newhbound,ttime)
                        hbound=newhbound
                    endif
                    call setupr(sa,ija,bound,tf,stor,ctss)
                    do j=1,ny
                        do i=1,nx
                            n=i+(j-1)*nx
                            b(n)=Htouse(i,j)
                            x(n)=b(n)
                        enddo
                    enddo
                    call loadfr(b,bound,hbound,tf,stor,ctss)
                    call linbcg(nxny,b,x,1,tol,itmax,iter,err,sa,ija)
                    do j=1,ny
                        do i=1,nx
                            n=i+(j-1)*nx
                            Htouse(i,j)=x(n)
                        enddo
                    enddo
                    if(track.eq.'y'.and.(.not.hitb))then
                        partrack(1,1)=pmem(1)
                        partrack(1,2)=pmem(2)

                        partrack(1,3)=ptim(1)
                        call annette(x,tf,voids,thick,partrack,npoints
      c                 ,tracktime,hitb)
c                        print*,kkk,npoints,hitb
                        do i=2,npoints
                            write(99,93) partrack(i,1),partrack(i,2)
```

```
c                       ,partrack(i,3)
                    enddo
                    pmem(1)=partrack(npoints,1)
                    pmem(2)=partrack(npoints,2)
                    ptim(1)=partrack(npoints,3)
                    partrack(1,1)=pmem(3)

                    partrack(1,2)=pmem(4)
                    partrack(1,3)=ptim(2)
                    call annette(x,tf,voids,thick,partrack,npoints
c                           ,tracktime,hitb)
                    do i=2,npoints
                        write(100,93) partrack(i,1),partrack(i,2)
c                       ,partrack(i,3)
                    enddo
                    pmem(3)=partrack(npoints,1)
                    pmem(4)=partrack(npoints,2)
                    ptim(2)=partrack(npoints,3)

                endif
                ctss=ctss*tsm
            enddo
            do n=1,nxny
                himem(n)=x(n)
            enddo
        endif
        ihdt=ihdt+1
    elseif(ttime.gt.hdtimes(ntimes))then
        if(track.eq.'y'.and.(.not.hitb))then
            partrack(1,1)=pmem(1)
            partrack(1,2)=pmem(2)
            partrack(1,3)=ptim(1)
            call annette(x,tf,voids,thick,partrack,npoints
c                   ,tracktime,hitb)
            do i=2,npoints
                write(99,93) partrack(i,1),partrack(i,2)
c                       ,partrack(i,3)
            enddo
            pmem(1)=partrack(npoints,1)
            pmem(2)=partrack(npoints,2)
            ptim(1)=partrack(npoints,3)
            partrack(1,1)=pmem(3)
            partrack(1,2)=pmem(4)
            partrack(1,3)=ptim(2)
            call annette(x,tf,voids,thick,partrack,npoints
c                       ,tracktime,hitb)
            do i=2,npoints
                write(100,93) partrack(i,1),partrack(i,2)
c               ,partrack(i,3)
            enddo
            pmem(3)=partrack(npoints,1)
            pmem(4)=partrack(npoints,2)
            ptim(2)=partrack(npoints,3)
        endif
    endif
    do n=1,nxny
        hcmem(n)=x(n)
    enddo
    ttime=ttime+cts
```

```
          cts=cts*tsm
        enddo
* End of the time step loop.
***** collect ensemble conditional mean variance fields
        nsc=nsc+1
        call var_mean(vs,x,nsc,condstats)
   13   continue
* End of simulations.
        enddo
* print out ensemble mean and var
        open(unit=50,file='fens.out')
        open(unit=51,file='hens.out')
        do j=1,ny
            ycoord=dble(j)*dy-dy/2.
            do i=1,nx
            xcoord=dble(i)*dx-dx/2.
            write(50,15) xcoord,ycoord,condstats(i,j,1),condstats(i,j,2)
            write(51,15) xcoord,ycoord,condstats(i,j,3),condstats(i,j,4)
            enddo
        enddo
 15     format(2f10.2,2f20.10)
        close(50)
        close(51)
        if(track.eq.'y')then
            close(99)
            close(100)
        endif
        if(outc(4).eq.'y') close(60)
        print*,ntrans,' time steps',nsims,' simulations',nsc,' used'
        end
```

A.3.10 Random field generator

```
      SUBROUTINE ranfgen(model,blav,nx,ny,xmin,xmax,ymin,ymax,
     b     var,lambdax,lambday,seed,a,amean,avar)
*
*
*     The following set of routines are designed to generate random
*     fields of a user input variance, size, and scale.  All of the
*     input and output variables are explained below.  Note: In
*     theory, the generated field should be mean zero with variance
*     given by the user.  But in reality, the means are not quite
*     zero and the variances of an ensemble are uniformly distributed
*     around the desired value.  Also the covariance is computed by
*     means of an FFT trick so that it is not very expensive to
*     compute.
*
*     NOTE:  maxn is a parameter that must be correctly set in each
*            of the following subroutines: ranfgen, generation,
*            meanvar, sxcov, conv1d, conv2d.  Its value corresponds
*            to the maximum field size that can be generated (i.e.
*            the the dimension of the field a in the calling program.
*
*     INPUT:  model--the type of spectral density to use
*                    1) Bell      2) Exponential  3) Whittle
*                    4) Mizell-A  5) Mizell-B
*             blav--0-do not block average,1-block average
*             nx--the size of the field in the x direction
*                 (this size must be a power of two)
*             ny--the size of the field in the y direction
*                 (this size must be a power of two)
*             xmin--the minimum x value (used to compute dx)
*             xmax--the maximum x value (used to compute dx)
*             ymin--the minimum y value (used to compute dy)
*             ymax--the maximum y value (used to compute dy)
*             var--the theoretical variance of the field
*             lambdax--the e-fold scale length in the x direction
*             lambday--the e-fold scale length in the y direction
*                      (this value is not used in isotropic models)
*             seed--the number that is used to seed the random # gen.
*     OUTPUT: a--the random field
*             ccv--the covariance of the above random field
*             amean--the computed mean of the field
*             avar--the computed variance of the field
*
* NOTICE!!!!!
* The array a must be the same dimension as the array in the
* calling routine.  The parameter maxn must be a power of 2
* This parameter occurs in three places in the following code.
* To change this parameter, change all 3 occurrences to the
* same power of 2.
*
      PARAMETER(maxn=64)
      INTEGER model,nx,ny,mx,my,seed,blav
      DOUBLE PRECISION xmin,xmax,ymin,ymax,dx,dy,dfx,dfy,dfxy,
     b     var,lambdax,lambday,alphasq,lambdaxy,amean,avar,
```

```
      c     pi,pi2,pisq,picube,sq5,a(maxn,maxn)
       DOUBLE COMPLEX im
*
*      *****Define all the constants necessary for the program.
       print*,'simulation seed ',seed
       CALL constants(nx,ny,mx,my,xmin,xmax,ymin,ymax,dx,dy,
      b     dfx,dfy,dfxy,pi,pi2,pisq,picube,sq5,im,seed)
*
*      *****Define constants relevant to the spectral model chosen.
       CALL specmodel(model,pi,lambdax,lambday,lambdaxy,alphasq)
*
*      *****Generate the random field.
       CALL generation(model,blav,nx,ny,mx,my,dx,dy,dfx,dfy,dfxy,
      b     var,lambdax,lambday,lambdaxy,alphasq,a,pi,pi2,pisq,
      c     picube,sq5,im,seed)
*
*      *****Compute the mean and variance.
       CALL meanvar(a,nx,ny,amean,avar)
*
*
       RETURN
       END
*
*************************************************************************
***
*************************************************************************
***
*
       SUBROUTINE constants(nx,ny,mx,my,xmin,xmax,ymin,ymax,
      b     dx,dy,dfx,dfy,dfxy,pi,pi2,pisq,picube,sq5,im,seed)
*
       INTEGER nx,ny,mx,my,seed
       DOUBLE PRECISION xmin,xmax,ymin,ymax,dx,dy,dfx,dfy,dfxy,
      b     pi,pi2,pisq,picube,sq5
       DOUBLE COMPLEX im
       REAL stuff,ran2
*
       stuff=ran2(-seed)
*
       pi=3.14159265358979
       pi2=2.0*pi
       pisq=pi*pi
       picube=pisq*pi
       im=(0.0,1.0)
       sq5=DSQRT(0.5D0)
       mx=nx/2
       my=ny/2
       dx=(xmax-xmin)/(nx-1)
       dy=(ymax-ymin)/(ny-1)
       dfx=1.0/(nx*dx)
       dfy=1.0/(ny*dy)
       dfxy=dfx*dfy
*
       RETURN
       END
*
*************************************************************************
***
*************************************************************************
***
```

```
*
      SUBROUTINE specmodel(model,pi,lambdax,lambday,
     b     lambdaxy,alphasq)
*
*     *****If model<3 then it can be non-isotropic, but in the other
*     *****  cases the model is isotropic-->lambdax=lambday.
*
*     *****The models that are currently avalable are:
*     *****       1) Bell-shaped            4) Mizell-A
*     *****       2) Exponential            5) Mizell-B
*     *****       3) Whittle
*
      INTEGER model
      DOUBLE PRECISION lambdax,lambday,lambdaxy,
     b     pi,alpha,alphasq
*
      IF (model.GE.3) THEN
         IF (model.EQ.3) alpha=1.05*pi/(2.0*lambdax)
         IF (model.EQ.4) alpha=1.33*pi/(4.0*lambdax)
         IF (model.EQ.5) alpha=4.2*pi/(16.0*lambdax)
         alphasq=alpha*alpha
      ELSE
         lambdaxy=lambdax*lambday
      END IF
*
      RETURN
      END
*
*******************************************************************************
***
*******************************************************************************
***      *
      SUBROUTINE generation(model,blav,nx,ny,mx,my,dx,dy,
     b     dfx,dfy,dfxy,var,lambdax,lambday,lambdaxy,alphasq,
     c     a,pi,pi2,pisq,picube,sq5,im,seed)
*
      PARAMETER(maxn=64)
      INTEGER model,nx,ny,mx,my,seed,ka,kb,nn(2),
     b     i,j,i1,j1,i2,j2,blav
      DOUBLE PRECISION dx,dy,dfx,dfy,dfxy,var,lambdax,lambday,
     b     pi,pi2,pisq,picube,sq5,a(maxn,maxn),
     c     alphasq,lambdaxy,ux,uy,nran2,sqtdz,yreal,
     d     yimag,xx(2*maxn*maxn),spectral,dsincf
      DOUBLE COMPLEX im,xa(maxn,maxn),zx,z1
*
*     *****Pack a complex array with random numbers so it can be
*     ***** transformed into a random field.
      DO ka=1,mx
         ux=(ka-0.5)*dfx
         DO kb=1,my
            uy=(kb-0.5)*dfy
            sqtdz=spectral(ux,uy,model,var,lambdax,lambday,
     b           lambdaxy,alphasq,pi,pi2,pisq,picube)
            sqtdz=DSQRT(sqtdz*dfxy)
            yreal=nran2(seed)*sqtdz*sq5
            yimag=nran2(seed)*sqtdz*sq5
            xa(ka,kb)=DCMPLX(yreal,yimag)
            IF (blav.EQ.1) THEN
               xa(ka,kb)=xa(ka,kb)*DSINCF(pi*ux*dx)*
     b              DSINCF(pi*uy*dy)
```

```
            END IF
         END DO
      END DO
*
      DO ka=1,mx
         ux=(ka-0.5)*dfx
         DO kb=1,my
            uy=-(kb-0.5)*dfy
            sqtdz=spectral(ux,uy,model,var,lambdax,lambday,
     b           lambdaxy,alphasq,pi,pi2,pisq,picube)
            sqtdz=DSQRT(sqtdz*dfxy)
            yreal=nran2(seed)*sqtdz*sq5
            yimag=nran2(seed)*sqtdz*sq5
            xa(ka,ny+1-kb)=DCMPLX(yreal,yimag)
            IF (blav.EQ.1) THEN
               xa(ka,kb)=xa(ka,kb)*DSINCF(pi*ux*dx)*
     b              DSINCF(pi*uy*dy)
            END IF
         END DO
      END DO
*
      DO ka=mx+1,nx
         DO kb=1,ny
            xa(ka,kb)=DCMPLX(0.0,0.0)
         END DO
      END DO
*
*     *****Perform the inverse FFT.
      nn(1)=nx
      nn(2)=ny
      num=1
      DO j=1,ny
         DO i=1,nx
            xx(num)=DREAL(xa(i,j))
            xx(num+1)=DIMAG(xa(i,j))
            num=num+2
         END DO
      END DO
      CALL fourn(xx,nn,2,1)
      num=1
      DO j=1,ny
         DO i=1,nx
            xa(i,j)=DCMPLX(xx(num),xx(num+1))
            num=num+2
         END DO
      END DO
*
*     *****Extract the field from the complex array.
      DO i=1,mx
         i1=i-1
         i2=i1+mx+1
         DO j=1,my
            j1=j-1
            j2=j1+my+1
            zx=im*pi*(i1/DBLE(nx)+j1/DBLE(ny))
            z1=2.0*xa(i,j)*CDEXP(zx)
            a(i2,j2)=DREAL(z1)
         END DO
         DO j=my+1,ny
            j1=j-1-ny
```

```
            j2=j1+my+1
            zx=im*pi*(i1/DBLE(nx)+j1/DBLE(ny))
            z1=2.0*xa(i,j)*CDEXP(zx)
            a(i2,j2)=DREAL(z1)
         END DO
      END DO
      DO i=mx+1,nx
         i1=i-1-nx
         i2=i1+mx+1
         DO j=1,my
            j1=j-1
            j2=j1+my+1
            zx=im*pi*(i1/DBLE(nx)+j1/DBLE(ny))
            z1=2.0*xa(i,j)*CDEXP(zx)
            a(i2,j2)=DREAL(z1)
         END DO
         DO j=my+1,ny
            j1=j-1-ny
            j2=j1+my+1
            zx=im*pi*(i1/DBLE(nx)+j1/DBLE(ny))
            z1=2.0*xa(i,j)*CDEXP(zx)
            a(i2,j2)=DREAL(z1)
         END DO
      END DO
*
      RETURN
      END
*
*************************************************************************
**
*************************************************************************
**
*
      SUBROUTINE meanvar(a,nx,ny,amean,avar)
*
      PARAMETER(maxn=64)
      INTEGER nx,ny
      DOUBLE PRECISION a(maxn,maxn),amean,avar,atemp,count1,count
*
*     *****Compute the mean and variance of the field.
      amean=0.0
      avar=0.0
      count=0.0
      count1=1.0
*
      DO i=1,nx
         DO j=1,ny
            count=count+1.0
            atemp=amean
            amean=amean+(a(i,j)-amean)/count
            atemp=amean-atemp
            avar=(1.0-1.0/count1)*avar+count*atemp*atemp
            count1=count
         END DO
      END DO
c      PRINT 11,amean,avar
 11   FORMAT (1X,2(E21.14))
*
      RETURN
      END
```

```
*
*************************************************************************
***
*************************************************************************
***
*
      DOUBLEPRECISION FUNCTION spectral(ux,uy,model,var,lambdax,
     b           lambday,lambdaxy,alphasq,pi,pi2,pisq,picube)
*
      INTEGER model
      DOUBLE PRECISION ux,uy,usq,tempx,tempy,ulambdasq,temp,g,
     b     var,lambdax,lambday,pi,pi2,pisq,picube,alphasq,lambdaxy
*
      usq=ux*ux+uy*uy
      tempx=ux*lambdax
      tempy=uy*lambday
      ulambdasq=tempx*tempx+tempy*tempy
      IF (model.GE.3) temp=4.0*pisq*usq+alphasq
      IF (model.EQ.1) g=pi*lambdaxy*DEXP(-pisq*ulambdasq)
      IF (model.EQ.2) g=pi2*lambdaxy/(1.0+4.0*pisq*ulambdasq)**1.5
      IF (model.EQ.3) g=2.0*pi2*alphasq/(temp*temp)
      IF (model.EQ.4) g=32.0*alphasq*picube*usq/(temp*temp*temp)
      IF (model.EQ.5) g=192.0*alphasq*picube*pisq*usq*usq/
     b     (temp*temp*temp*temp)
      spectral=g*var
*
      RETURN
      END
*
*************************************************************************
**
*************************************************************************
**
*
      DOUBLE PRECISION FUNCTION dsincf(x)
*
      DOUBLE PRECISION x
*
      IF (DABS(x).LT.1d-8) THEN
         dsincf=1.0
      ELSE
         dsincf=DSIN(x)/x
      ENDIF
*
      RETURN
      END
*
*************************************************************************
**
*************************************************************************
**
*
      REAL FUNCTION ran2(idum)
*
*     Numerical Recipes in Fortran, 2nd Edition, p. 272.
*
      INTEGER idum,IM1,IM2,IMM1,IA1,IA2,IQ1,IQ2,IR1,IR2,NTAB,NDIV
      REAL AM,EPS,RNMX
      PARAMETER (IM1=2147483563,IM2=2147483399,AM=1.0/IM1,IMM1=IM1-1,
     b     IA1=40014,IA2=40692,IQ1=53668,IQ2=52774,IR1=12211,
```

```
      c     IR2=3791,NTAB=32,NDIV=1+IMM1/NTAB,EPS=1.2e-7,RNMX=1.0-EPS)
       INTEGER idum2,j,k,iv(NTAB),iy
       SAVE iv,iy,idum2
       DATA idum2/123456789/, iv/NTAB*0/, iy/0/
*
       IF (idum.LE.0) THEN
          idum=MAX(-idum,1)
          idum2=idum
          DO j=NTAB+8,1,-1
             k=idum/IQ1
             idum=IA1*(idum-k*IQ1)-k*IR1
             IF (idum.LT.0) idum=idum+IM1
             IF (j.LE.NTAB) iv(j)=idum
          END DO
          iy=iv(1)
       END IF
*
       k=idum/IQ1
       idum=IA1*(idum-k*IQ1)-k*IR1
       IF (idum.LT.0) idum=idum+IM1
       k=idum2/IQ2
       idum2=IA2*(idum2-k*IQ2)-k*IR2
       IF (idum2.LT.0) idum2=idum2+IM2
       j=1+iy/NDIV
       iy=iv(j)-idum2
       iv(j)=idum
       IF (iy.LT.1) iy=iy+IMM1
       ran2=MIN(AM*iy,RNMX)
*
       RETURN
       END
*
*************************************************************************
***
*************************************************************************
***
*
       DOUBLEPRECISION FUNCTION nran2(idum)
*
*      Based on discussion on Normal Deviates in Numerical Recipes
*      in Fortran, 2nd Edition on p. 279.
*
       INTEGER idum
       REAL x1,x2,ran2
       DOUBLE PRECISION pi
*
       pi=3.14159269365
       x1=ran2(idum)
       x2=ran2(idum)
       nran2=DSQRT(-2.0d0*DLOG(DBLE(x1)))*DCOS(2.0d0*pi*x2)
*
       RETURN
       END
*
*************************************************************************
***
*************************************************************************
***
       SUBROUTINE fourn(data,nn,ndim,isign)
*
```

```
*     Numerical Recipes in Fortran, 2nd Edition, p.518.
*
*     Replaces data by its 2-dimensional discrete Fourier
*     transform, if isign is 1.  n is the size of the square
*     array which MUST be a power of 2.  Data is a real array
*     of length twice n*n, in which the data are stored as in
*     a multidimensional complex FORTRAN array.  If isign is
*     input as -1, data is replaced by its inverse transform
*     times n*n.
*
      INTEGER ndim,nn(ndim),isign,ntot,nprev,nrem,idim,n,k1,k2
      INTEGER i1,i2,i2rev,i3,i3rev,ibit,ifp1,ifp2,ip1,ip2,ip3
      DOUBLEPRECISION wr,wi,wpr,wpi,wtemp,tempr,tempi
      DOUBLEPRECISION theta
      DOUBLEPRECISION data(*)
*
*     *****Compute total number of complex values.
      ntot=1
      DO idim=1,ndim
         ntot=ntot*nn(idim)
      END DO
      nprev=1
*
*     *****Main loop over the dimensions.
      DO idim=1,ndim
         n=nn(idim)
         nrem=ntot/(n*nprev)
         ip1=2*nprev
         ip2=ip1*n
         ip3=ip2*nrem
         i2rev=1
*        *****Bit reversal section of routine.
         DO i2=1,ip2,ip1
            IF (i2.LT.i2rev) THEN
               DO i1=i2,i2+ip1-2,2
                  DO i3=i1,ip3,ip2
                     i3rev=i2rev+i3-i2
                     tempr=data(i3)
                     tempi=data(i3+1)
                     data(i3)=data(i3rev)
                     data(i3+1)=data(i3rev+1)
                     data(i3rev)=tempr
                     data(i3rev+1)=tempi
                  END DO
               END DO
            END IF
            ibit=ip2/2
            DO WHILE((ibit.GE.ip1).AND.(i2rev.GT.ibit))
               i2rev=i2rev-ibit
               ibit=ibit/2
            END DO
            i2rev=i2rev+ibit
         END DO
         ifp1=ip1
*        *****Danielson-Lanczos section of routine.
         DO WHILE(ifp1.lt.ip2)
            ifp2=2*ifp1
*           *****Initialize for the trig. recurrence.
            theta=isign*6.28318530717959d0/(ifp2/ip1)
            wpr=-2.d0*DSIN(0.5d0*theta)**2
```

```
            wpi=DSIN(theta)
            wr=1.d0
            wi=0.d0
            DO i3=1,ifp1,ip1
               DO i1=i3,i3+ip1-2,2
                  DO i2=i1,ip3,ifp2
                     k1=i2
*                    *****Danielson-Lanczos formula.
                     k2=k1+ifp1
                     tempr=wr*data(k2)-wi*data(k2+1)
                     tempi=wr*data(k2+1)+wi*data(k2)
                     data(k2)=data(k1)-tempr
                     data(k2+1)=data(k1+1)-tempi
                     data(k1)=data(k1)+tempr
                     data(k1+1)=data(k1+1)+tempi
                  END DO
               END DO
*              *****Trignometric recurrence.
               wtemp=wr
               wr=wr*wpr-wi*wpi+wr
               wi=wi*wpr+wtemp*wpi+wi
            END DO
            ifp1=ifp2
         END DO
         nprev=n*nprev
      END DO
*
      RETURN
      END
```

A.3.11 Particle tracking routine

```
* annette.f
* Track a single particle through a flow field
* INPUT
* array head x(1:n), array transmissivity tf(i,j)
* real*8 porosity=voids, thickness=b
* array particle track = partrack(max*10,2)
* where only the first partrack(1,1) and partrack(1,3) which
* specifiy the particle starting point are required input
* real*8 tracktime total time of particle tracking
* OUTPUT
* array particle track = partrack(max*10,2) where the first
* partrack(1,1),partrack(1,2),partrack(1,3) are unchanged
* integer npoints number of significant locations in partrack
* for a maximum of 5 per fine grid block across the domain
* logical hitb, returns true if particle has reached the edge
      subroutine annette(x,tf,voids,thick,partrack,npoints
     c                  ,tracktime,hitb)
      implicit double precision (a-h,o-z)
      parameter(eps=1.E-7)
      include 'lhamana.inc'
      logical hitb
      integer npoints,k,l,n
      real*8 x(max*max),tf(max,max),partrack(max*10,3)
      real*8 voids,thick,tracktime,tinit,a,b,c,d
      real*8 tleft,curx,cury,xbc,ybc,xr,yr,ql,qr,qu,qd,qx,qy
      real*8 tl,tr,tu,td,delt,tmp,epsx,epsy,ttime
      curx=partrack(1,1)
      cury=partrack(1,2)
      tinit=partrack(1,3)
      npoints=1
      tleft=tracktime
      ttime=0.
      epsx=0.
      epsy=0.
   10 continue
* find the block location of the particle
         k=int((curx+epsx)/dx)+1
         l=int((cury+epsy)/dy)+1
c         print*,'k ',k,'   l',l
         n=k+(l-1)*nx
* bail out if particle is in an edge block
         if(k.eq.1.or.k.eq.nx.or.l.eq.1.or.l.eq.ny)then
            print*,'hit boundary ',k,l
             hitb=.true.
             return
         endif
* interpolate side of block velocities
         xbc=dble(k)*dx-dx/2.
         ybc=dble(l)*dy-dy/2.
         xr1=curx-xbc+dx/2.
         yr1=cury-ybc+dy/2.
         tl=2*tf(k-1,l)*tf(k,l)/(tf(k-1,l)+tf(k,l))
         tr=2*tf(k+1,l)*tf(k,l)/(tf(k+1,l)+tf(k,l))
```

```
            tu=2*tf(k,l+1)*tf(k,l)/(tf(k,l+1)+tf(k,l))
            td=2*tf(k,l-1)*tf(k,l)/(tf(k,l-1)+tf(k,l))
            ql=-(tl/thick)*(x(n)-x(n-1))/dx
            qr=-(tr/thick)*(x(n+1)-x(n))/dx
            qd=-(td/thick)*(x(n)-x(n-nx))/dy
            qu=-(tu/thick)*(x(n+nx)-x(n))/dy
c            print*,'x dir ',ql,qr
c            print*,'y dir ',qd,qu
            a=ql
            b=(qr-ql)/dx
            c=qd
            d=(qu-qd)/dy
            qx=a+b*xr1
            qy=c+d*yr1
c            print*,'qx',qx,abs((a+b*dx)/qx),abs(a/qx)
c            print*,'qy',qy,abs((c+d*dy)/qy),abs(c/qy)
* find the time step to a boundary
            if(qx.gt.0..and.((a+b*dx)/qx).ne.0.)then
                delt=(voids/b)*dlog(abs((a+b*dx)/qx))
            elseif(qx.lt.0..and.(a/qx).ne.0.)then
                delt=(voids/b)*dlog(abs(a/qx))
            endif
            if(qy.gt.0..and.((c+d*dy)/qy).ne.0.)then
                tmp=(voids/d)*dlog(abs((c+d*dy)/qy))
            elseif(qy.lt.0..and.(c/qy).ne.0.)then
                tmp=(voids/d)*dlog(abs(c/qy))
            endif
c            print*,'set delt to the smaller of'
c            print*,'delt',delt,'  tmp',tmp
            if(delt.lt.0..and.tmp.gt.0.)then
               delt=tmp
            elseif(delt.gt.0..and.tmp.lt.0.)then
               delt=delt
            elseif(delt.gt.0..and.tmp.gt.0.)then
               if(tmp.lt.delt) delt=tmp
            else
               write(*,*)'error in particle track'
               return
            endif
            if(delt.lt.eps.and.tmp.gt.0.) delt=tmp
c            print*,'delt',delt
* if the delt to a boundary is greater than the tracktime
            temp=delt
c            print*,'if delt is greater than tleft'
            if(delt.gt.tleft) delt=tleft
c            print*,'delt ',delt,'  temp',temp
* take a step inside the block and return
           xr=((a+b*xr1)*dexp(b*delt/voids)-a)/b
           yr=((c+d*yr1)*dexp(d*delt/voids)-c)/d
           if(abs(xr-dx).lt.eps)then
               epsx=dx/100.
           elseif(abs(xr).lt.eps)then
               epsx=-dx/100.
           else
               epsx=0.
           endif
           if(abs(yr-dy).lt.eps)then
               epsy=dy/100.
           elseif(abs(yr).lt.eps)then
               epsy=-dy/100.
```

```
            else
                epsy=0.
            endif
c            print*,'xr ',xr,'yr   ',yr
            npoints=npoints+1
            curx=xr+xbc-dx/2.
            cury=yr+ybc-dy/2.
            partrack(npoints,1)=curx
            partrack(npoints,2)=cury
            partrack(npoints,3)=partrack(npoints-1,3)+delt
            ttime=ttime+delt
c            print*,'delt ',delt,'   tleft',tleft
c            print*,partrack(npoints,1),partrack(npoints,2)
c     c             ,partrack(npoints,3)
c            pause
c            print*,npoints,ttime
c            print*,temp,tleft
            if(temp.gt.tleft)then
c            print*,temp,tleft
                return
            else
                tleft=tracktime-ttime
                go to 10
            endif
        end
```

Appendix B

Head and transmissivity data from KAFB

	State Plane		Transmissivity	Pumping rate	
number	*East (ft)*	*North (ft)*	*gal/day/ft*	*(gal/min)*	*well name*
1	423121.2	1482561	127033	1300	LOMAS-1
2	417100.1	1491069	35005	800	LOVE-6
3	407663.8	1491947	420062	2750	CHARLES-4
4	404045.6	1492282	420062	3000	CHARLES-2
5	406259.8	1494660	444096	3700	CHARLES-1
6	380161.7	1475293	45006	1150	SAN JOSE-2
7	413622.5	1480171	45000	1350	RIDGECREST-1
8	406943.6	1484195	280000	2700	RIDGECREST-4
9	395364.1	1483677	188983	2000	BURTON-3
10	387909	1479499	145991	3150	YALE-2
11	409426.1	1480830	249000	2800	RIDGECREST-3
12	394362.2	1481820	157038	1331	BURTON-2
13	386311.8	1474358	72025	2400	MILES-1
14	414173	1492473	140020	2200	LOVE-7
15	382204	1473809	45977	2500	SAN JOSE-3
16	385223.4	1483270	60008	2000	YALE-3
17	411990.4	1484897	110016	1500	LOVE-5
18	413767.6	1486923	210031	1600	LOVE-3
19	410476.3	1486943	177041	1700	LOVE-4
20	393272.3	1479600	130000	3020	BURTON-1
21	403025.8	1495874	400000	2986	CHARLES-3
22	421392.2	1485670	130000	500	LOMAS-2
23	418213.3	1481745	130000	2300	LOMAS-5
24	419374	1480429	130000	2500	LOMAS-6
25	381319.4	1475298	130000	188	SAN JOSE-1
26	381884	1473808	130000	2000	SAN JOSE-4
27	407859	1473515	130000	331	KAFB-1
28	400524.8	1470763	130000	629	KAFB-2
29	406572.7	1479062	130000	122	KAFB-3
30	407500.2	1468116	678365	1102	KAFB-4
31	404433.5	1470732	300000	1354	KAFB-7
32	410840.8	1466904	344448	500	KAFB-8
33	418585.6	1470267	23586	924	KAFB-11
34	394246.2	1476294	130000	456	KAFB-12
35	396550.5	1473886	300000	1781	KAFB-14

number	East (ft)	North (ft)	Head (ft)	well name
1	414690.0	1444690	4939.33	BW-3
2	414300.0	1445110	4936.49	MW-1A
3	414170.0	1444980	4937.09	MW-2A
4	414140.0	1444770	4937.24	MW-3A
5	414200.0	1445110	4935.25	MW-4
6	411760.0	1451700	4921.51	MWL-BW1
7	411660.0	1452660	4921.58	MWL-MW1
8	411450.0	1452690	4921.66	MWL-MW2
9	411410.0	1452480	4920.06	MWL-MW3
10	407300.0	1455310	4889.75	NW-TA3
11	407370.0	1444910	4899.57	SW-TA3
12	421092.0	1457410	4954.95	KAFB-9
13	413850.0	1455090	4921.44	KAFB-10
14	399890.0	1470770	4880.31	LF/DM-01
15	409390.0	1465450	4894.46	LF/DM-02
16	391920.0	1461770	4904.18	MVMW-J
17	399450.0	1464320	4897.86	MVMW-K
18	417070.0	1471440	4910.21	TIJERAS EAST
19	397810.0	1467750	4881.67	TIJERAS WEST
20	406860.0	1470180	4870.87	LAGOON NE
21	406300.0	1470190	4870.89	LAGOON NW
22	406930.0	1468870	4872.45	LAGOON SE
23	406220.0	1468990	4873.13	LAGOON SW
24	404000.0	1444000	4896.10	IRP-1005
25	404000.0	1436500	4899.00	IRP-1001
26	400500.0	1436500	4900.37	IRP-1002
27	399800.0	1438800	4900.35	IRP-1003
28	399800.0	1442500	4896.66	IRP-1004
29	399700.0	1469000	4874.85	IRP-0110
30	400500.0	1470600	4879.34	IRP-0111
31	398900.0	1468900	4874.29	IRP-0113
32	399400.0	1470600	4871.18	IRP-0114
33	400700.0	1470400	4870.36	IRP-0115
34	398700.0	1469500	4874.22	IRP-0117
35	415700.0	1469400	4924.57	IRP-0307
36	416200.0	1469400	4925.80	IRP-0308
37	417000.0	1469400	4923.78	IRP-0309
38	415200.0	1468400	4924.53	IRP-0311
39	394400.0	1469600	4880.36	IRP-0417
40	377371.7	1453478	4905.03	CHAVA
41	385030.6	1454762	4896.60	SBLF-4
42	423121.2	1482561	4870.00	LOMAS-1
43	417100.1	1491069	4853.60	LOVE-6
44	416667.3	1487300	4852.90	LOVE-1
45	407663.8	1491947	4864.50	CHARLES-4
46	406259.8	1494660	4851.90	CHARLES-1
47	404045.6	1492282	4863.50	CHARLES-2
48	378368.4	1493912	4925.92	BIA

number	East (ft)	North (ft)	Head (ft)	well name
49	384169.0	1492837	4903.94	SBM-1
50	380161.7	1475293	4885.60	SAN JOSE-2
51	377586.9	1465406	4916.47	RIO BRAVO-3
52	379747.9	1465094	4912.34	RIO BRAVO-4
53	419374.0	1480429	4863.50	LOMAS-6
54	418213.3	1481745	4824.50	LOMAS-5
55	413622.5	1480171	4859.80	RIDGECREST-1
56	406943.6	1484195	4863.40	RIDGECREST-4
57	395364.1	1483677	4874.00	BURTON-3
58	387909.0	1479499	4878.20	YALE-2
59	381319.4	1475298	4906.30	SAN JOSE-1
60	384890.0	1474104	4893.15	YALE-MW5
61	388941.0	1473234	4891.02	YALE-MW4
62	409426.1	1480831	4858.20	RIDGECREST-3
63	394362.2	1481820	4866.10	BURTON-2
64	393272.2	1479600	4849.00	BURTON-1
65	389122.0	1474640	4885.39	YALE-MW3
66	386311.8	1474358	4868.40	MILES-1
67	396999.2	1427322	4904.75	ISL ECW-3
68	415213.0	1473367	4890.15	EUBANK-1
69	382094.3	1490662	4911.55	CTY OBS-1
70	383218.0	1480209	4900.50	JMC-1
71	407413.2	1436885	4925.00	HELIPORT WELL
72	380742.3	1448577	4900.07	SBLF-3
73	385318.0	1454102	4896.46	SBLF-4
74	382204.0	1473809	4893.32	SAN JOSE-3
75	384676.4	1449203	4897.89	SBLF-2
76	383818.5	1451674	4898.70	SBLF-1
77	413767.6	1486923	4868.10	LOVE-3
78	410476.3	1486943	4851.60	LOVE-4
79	414834.1	1444714	4937.19	CWL-BW-2
80	414354.2	1445105	4936.79	CWL-MW-1
81	414119.4	1444978	4933.96	CWL-MW-2
82	414131.1	1444825	4934.39	CWL-MW-3
83	414195.2	1445112	4935.61	CWL-MW-4
84	432117.1	1454186	5766.80	GRAYSTONE
85	433939.1	1454779	5844.86	COYOTE SPRING
86	443186.6	1456295	NA	TSA-1
87	427752.8	1442618	5659.34	LAKE CHRISTIAN WEST
88	431771.5	1451216	5699.01	SCHOOL HOUSE
89	426599.3	1448040	5685.57	EOD HILL
90	416650.7	1463845	5044.95	GC EAST/KAFB-0609
91	416589.5	1463575	5041.31	GC SOUTH/KAFB-0602
92	416185.6	1463528	5049.53	GC WEST/KAFB-0610
93	416312.4	1463917	5044.89	GC NORTH/KAFB-0608
94	415955.3	1430598	NA	HUBBEL SPRING
95	410840.8	1466903	NA	KAFB-8

month(ft)	CWL-BW-2	CWL-MW-1	CWL-MW-2	CWL-MW-3	CWL-BW-1
Jan-86	4938.93	4940.15	4939.12	4939.72	
Feb-86					
Mar-86					
Apr-86	4939.21	4939.48	4938.80	4939.25	
May-86					
Jun-86					
Jul-86					
Aug-86	4940.21	4939.35	4939.86	4940.24	
Sep-86					
Oct-86					
Nov-86					
Dec-86					
Jan-87					
Feb-87	4940.51	4943.39	4938.71	4939.19	
Mar-87					
Apr-87					
May-87					
Jun-87					
Jul-87					
Aug-87	4939.81	4938.28	4937.66	4937.99	4966.93
Sep-87					
Oct-87					
Nov-87					
Dec-87					
Jan-88					
Feb-88	4939.75	4938.18	4937.74	4938.13	4967.82
Mar-88					
Apr-88					
May-88	4939.24	4937.57	4937.15	4937.43	4967.70
Jun-88	4939.44	4937.57	4937.27	4937.65	4968.40
Jul-88	4939.44	4937.61	4937.26	4937.63	4969.38
Aug-88	4939.47		4937.25	4937.65	4957.44
Sep-88	4938.87		4937.23	4937.63	4957.67
Oct-88	4938.60		4936.77	4936.90	4957.18
Nov-88					
Dec-88	4939.52		4937.55	4938.00	4958.28
Jan-89	4938.99		4937.21	4937.64	4957.73
Feb-89	4938.74		4936.66	4937.10	4957.54
Mar-89	4939.12		4937.48	4937.60	4957.68
Apr-89	4939.33		4937.48	4937.85	4958.28
May-89	4938.80		4936.62	4937.02	4957.63
Jun-89	4938.92		4937.05	4937.36	4958.00
Jul-89	4938.70		4936.34	4936.72	4957.58
Aug-89	4938.73		4936.25	4936.67	4957.65
Sep-89	4938.72		4936.26	4936.66	4957.63

month(ft)	CWL-BW-2	CWL-MW-1	CWL-MW-2	CWL-MW-3	CWL-BW-1
Oct-89	4938.40		4935.84	4936.26	4957.46
Nov-89	4938.49		4935.93	4936.35	4957.43
Dec-89	4938.60		4935.98	4936.41	4957.58
Jan-90	4938.61		4935.90	4936.34	4957.66
Feb-90	4938.82		4936.11	4936.54	4957.95
Mar-90	4938.42		4935.68	4936.10	4957.55
Apr-90	4938.35		4935.55	4935.98	4957.53
May-90	4938.29		4935.51	4935.95	4957.53
Jun-90	4938.45		4935.71	4936.09	4957.75
Jul-90	4938.54		4935.65	4936.04	4957.84
Aug-90	4938.46		4935.57	4935.97	4957.84
Sep-90	4938.31		4935.48	4935.88	4957.83
Oct-90	4938.12		4935.16	4935.54	4957.55
Nov-90	4938.07		4935.12	4935.51	4957.56
Dec-90	4938.52		4935.46	4935.86	4958.02
Jan-91	4938.47		4935.36	4935.77	4957.94
Feb-91	4938.26		4935.11	4935.52	4957.75
Mar-91	4938.47		4935.35	4935.74	4957.73
Apr-91	4938.14		4934.95	4935.38	4957.92
May-91	4938.03		4934.94		4958.07
Jun-91	4937.62		4934.58	4934.98	4957.75
Jul-91	4937.78		4934.62	4935.03	4957.89
Aug-91	4937.53		4934.38	4934.79	4957.69
Sep-91	4937.56		4934.40	4934.80	4957.76
Oct-91	4937.88		4934.68	4935.11	4958.02
Nov-91	4938.13		4934.94	4935.37	4958.24
Dec-91	4937.52		4934.61	4934.97	4957.99
Jan-92	4938.13	4937.34	4934.95	4939.48	4958.36
Feb-92	4938.89	4938.15	4935.37	4934.37	4958.36
Mar-92	4937.19	4936.79	4933.96	4934.39	4957.91
Apr-92	4938.15	4937.60	4934.70	4935.87	4958.26
May-92	4937.38	4938.20	4935.73	4936.15	4958.72
Jun-92	4937.13	4937.66	4933.88	4934.29	4958.29
Jul-92	4937.00	4936.47	4933.72	4934.13	4958.22
Aug-92	4937.10	4936.49	4933.76	4934.16	4958.31
Sep-92	4937.02	4936.58	4933.69	4934.08	4958.17
Oct-92	4937.09	4936.61	4933.72	4934.15	4958.47
Nov-92	4936.80	4936.27	4933.47	4933.87	4957.16
Dec-92	4936.71	4936.24	4933.35	4933.76	4958.00
Jan-93	4936.41	4935.96	4932.97	4933.40	4957.84
Feb-93	4938.07	4936.78	4933.97	4933.35	4957.80
Mar-93	4935.70	4935.25	4932.39	4932.82	4957.31
Apr-93		4935.20	4932.21	4933.24	4957.24
May-93	4935.50	4935.07	4932.14	4932.56	4957.20
Jun-93	4936.03	4936.05	4932.65	4933.04	4957.85

month(ft)	CWL-BW-2	CWL-MW-1	CWL-MW-2	CWL-MW-3	CWL-BW-1
Jul-93					
Aug-93	4935.85	4935.38	4932.77	4932.47	4957.63
Sep-93	4936.20	4935.80	4933.37	4933.20	4958.08
Oct-93	4936.79	4936.33	4933.37	4933.79	4958.76
Nov-93	4938.55	4933.92	4932.92	4936.25	4959.17
Dec-93	4938.45	4934.55	4932.58	4935.38	4960.04
Jan-94	4936.39		4932.85	4933.28	4958.48
Feb-94	4935.74		4932.23	4932.65	4957.88
Mar-94	4935.39		4931.93	4932.36	4957.58
Apr-94	4936.13		4932.24	4933.68	4958.70
May-94	4935.34		4932.00	4932.43	4957.88
Jun-94	4935.38		4932.04	4932.37	4957.82
Jul-94	4935.37		4932.03	4932.39	4957.90
Aug-94	4935.41		4932.06	4932.20	4958.00
Sep-94	4935.49		4931.99	4932.33	4958.04

month(ft)	CWL-BW-3	CWL-MW-1A	CWL-MW-2A	CWL-MW-3A	MWL-MW-1
Oct-88	4941.66	4939.13	4939.73	4939.93	
Nov-88	4942.70	4941.44	4940.56	4940.80	
Dec-88	4941.75	4940.02	4940.40	4940.80	
Jan-89	4942.02	4939.51	4940.24	4940.35	4924.59
Feb-89	4941.70	4939.04	4939.65	4939.88	
Mar-89	4942.11	4939.59	4940.63		4927.46
Apr-89	4942.34	4939.79	4940.51	4940.67	4925.13
May-89	4941.35	4939.09	4939.68	4939.89	4924.56
Jun-89	4941.92	4939.76	4940.17	4940.24	
Jul-89	4941.44	4938.80	4939.40	4939.59	4924.12
Aug-89	4941.39	4938.82	4939.36	4939.58	4924.13
Sep-89	4940.81	4938.81	4939.36	4939.56	4924.11
Oct-89	4941.05	4938.35	4938.85	4939.12	4923.62
Nov-89	4941.15	4938.43	4938.97	4939.21	4923.78
Dec-89	4941.19	4938.50	4939.08	4939.31	4923.89
Jan-90	4941.15	4938.45	4939.02	4939.23	4923.78
Feb-90	4941.39	4938.59	4939.31	4939.46	4924.02
Mar-90	4940.96	4938.24	4939.00	4939.05	4923.73
Apr-90	4940.82	4938.13	4938.67	4938.89	4923.44
May-90	4940.76	4938.14	4938.69	4938.92	4923.44
Jun-90	4940.89	4938.19	4938.75	4938.98	4923.49
Jul-90	4940.89	4938.19	4938.72	4938.93	4923.45
Aug-90	4940.86	4938.12	4938.66	4938.87	4923.36
Sep-90	4940.77	4938.08	4938.62	4938.84	4923.33
Oct-90	4940.52	4937.84	4938.29	4938.50	4922.97
Nov-90	4940.42	4937.79	4938.32	4938.56	4923.03

month(ft)	CWL-BW-3	CWL-MW-1A	CWL-MW-2A	CWL-MW-3A	MWL-MW-1
Dec-90	4940.19	4938.10	4938.64	4938.82	4923.34
Jan-91	4940.67	4937.96		4938.76	4922.92
Feb-91	4940.43	4937.72	4938.28	4938.55	4923.36
Mar-91	4940.81	4938.02	4938.58	4938.75	4922.74
Apr-91		4937.56	4938.09	4938.29	4922.72
May-91		4937.16	4938.10	4938.30	4923.30
Jun-91	4940.01	4937.19	4937.80	4937.85	4923.30
Jul-91	4940.06	4937.31	4937.87	4938.04	4922.38
Aug-91	4939.85	4937.03	4937.63	4937.81	4922.22
Sep-91	4939.88	4937.11	4937.70	4937.91	4922.27
Oct-91	4940.15	4937.38	4937.97	4938.14	4922.23
Nov-91	4939.91	4937.63	4938.20		4922.45
Dec-91	4939.74	4937.15	4938.15		4922.04
Jan-92		4937.34	4938.38	4938.59	4921.87
Feb-92		4938.15	4937.30	4937.53	4921.93
Mar-92	4939.51	4936.79	4937.32	4937.53	4921.92
Apr-92	4939.66	4937.60	4938.16	4938.33	4922.96
May-92	4941.34	4938.20	4939.04	4939.47	
Jun-92		4937.66	4937.25	4937.45	
Jul-92	4939.28	4936.47	4937.14	4937.19	4921.35
Aug-92	4939.33	4936.49	4937.09	4937.24	4921.58
Sep-92	4939.30	4936.58	4937.14	4937.33	4921.68
Oct-92	4939.39	4936.61	4937.17	4937.35	4921.69
Nov-92		4936.27	4936.87	4937.01	4921.35
Dec-92	4939.16	4936.24	4936.78	4937.03	4921.30
Jan-93	4938.72	4935.96	4936.49	4936.71	4917.07
Feb-93	4940.44	4936.78	4935.94	4936.89	
Mar-93	4938.14	4935.25	4935.88	4935.97	4917.07
Apr-93	4937.99	4935.20	4935.83	4935.89	4920.67
May-93	4937.90	4935.07	4935.62	4935.81	
Jun-93	4938.41	4936.05	4935.69	4936.28	
Jul-93					
Aug-93	4938.80	4935.38	4935.96	4936.04	
Sep-93	4938.61	4935.80	4936.40	4936.52	
Oct-93	4940.80	4936.33	4938.00	4937.05	
Nov-93	4936.77	4933.92	4935.24	4934.92	
Dec-93	4937.42	4934.55	4936.51	4935.52	
Jan-94	4938.73	4935.84	4936.58	4936.52	4920.85
Feb-94	4938.10	4935.28	4936.00	4935.99	4920.55
Mar-94	4937.83	4935.04	4935.68	4935.68	4920.09
Apr-94	4938.70	4934.29	4935.63	4934.61	
May-94	4937.58	4934.99	4935.73	4935.69	4920.10
Jun-94	4937.72	4935.05	4935.84	4935.73	4920.21
Jul-94	4938.17	4935.19	4935.88	4935.80	4920.04
Aug-94	4938.19	4935.04	4935.74	4935.68	4920.10

month(ft)	CWL-BW-3	CWL-MW-1A	CWL-MW-2A	CWL-MW-3A	MWL-MW-1
Sep-94	4937.91	4935.01	4935.68	4935.73	4920.08

month(ft)	KAFB-10	KAFB-9	LF/DM-1	LF/DM-2	MV-MWJ
Jan-89	4926.39				
Feb-89	4926.78				
Mar-89	4926.62				
Apr-89	4926.63				
May-89	4925.81				
Jun-89	4926.11	4955.68	4899.27	4901.43	4914.68
Jul-89	4925.36	4955.07	4895.95	4900.28	4910.41
Aug-89	4925.23	4955.04	4900.45	4899.98	4910.29
Sep-89	4925.18	4955.09	4895.39	4899.91	4910.27
Oct-89	4924.98		4894.65	4899.54	4909.61
Nov-89	4924.98		4894.89	4900.26	4909.67
Dec-89	4924.91	4954.77	4895.34	4900.56	4909.62
Jan-90	4924.80	4954.87	4895.43	4900.66	4909.60
Feb-90	4924.80	4955.03	4895.52	4900.99	4909.61
Mar-90	4924.49	4954.69	4895.07	4900.76	4909.31
Apr-90	4924.55	4954.69	4894.72	4900.24	4909.04
May-90	4924.53	4954.62	4895.03	4900.10	4908.99
Jun-90		4954.90	4894.99	4900.01	4908.87
Jul-90	4924.62	4954.88	4894.53	4899.41	4908.49
Aug-90	4924.47	4954.98	4894.01	4899.00	4908.36
Sep-90	4924.37	4954.74	4893.55	4898.59	4908.24
Oct-90	4924.27	4954.82	4893.32	4898.09	4907.77
Nov-90	4923.97	4954.66	4893.39	4897.95	4907.57
Dec-90	4924.20	4954.95	4894.04	4898.34	4907.87
Jan-91	4923.44	4954.71	4894.09	4898.16	4907.52
Feb-91	4923.27	4954.77	4894.14	4898.18	4907.35
Mar-91	4923.57	4955.22	4894.35	4898.41	4906.99
Apr-91	4923.29	4954.78		4897.81	4907.12
May-91	4922.50	4954.27	4893.39	4897.30	4906.82
Jun-91	4922.97	4954.46	4893.06	4896.80	4906.51
Jul-91	4922.87	4955.04	4892.50	4898.09	4906.31
Aug-91	4922.76	4954.40	4892.28	4896.35	4906.32
Sep-91	4922.83	4954.39	4891.71	4895.84	4905.83
Oct-91	4923.11	4954.56	4892.87	4896.61	4907.33
Nov-91	4922.75	4954.55	4893.11	4897.47	4907.33
Dec-91	4922.48	4954.14	4892.52	4897.16	
Jan-92	4922.44	4954.10	4893.43	4897.43	4906.96
Feb-92	4924.17	4954.48	4893.83	4896.26	4904.94
Mar-92	4922.17	4954.62	4891.53	4896.09	4905.10
Apr-92	4923.31	4954.44	4893.67	4897.98	4907.21

month(ft)	KAFB-10	KAFB-9	LF/DM-1	LF/DM-2	MV-MWJ
May-92	4923.59	4954.95	4891.06	4895.44	4904.82
Jun-92	4922.04			4895.11	4903.82
Jul-92	4921.69		4889.49	4894.49	4904.24
Aug-92	4921.44		4880.31	4894.46	4904.18
Sep-92	4921.61		4879.92	4893.94	4903.89
Oct-92	4921.65			4893.84	4903.78
Nov-92				4893.94	4903.82
Dec-92	4921.30			4893.71	4903.12
Jan-93	4921.08			4894.11	4903.25
Feb-93	4921.28		4895.98	4894.28	4904.13
Mar-93	4920.49			4893.59	4902.35
Apr-93	4920.44			4893.72	4902.46
May-93	4920.41			4893.01	4901.95
Jun-93	4918.92			4894.46	4903.16
Jul-93					
Aug-93	4918.88		4878.06	4878.06	4902.18
Sep-93				4893.14	4902.12
Oct-93			4881.24	4878.45	4905.00
Nov-93	4920.43		4878.13	4903.95	4911.15
Dec-93	4920.78		4878.68	4893.64	4904.53
Jan-94	4921.31		4878.07	4893.38	4901.54
Feb-94	4920.78		4892.48	4893.16	
Mar-94	4920.73		4877.22	4892.84	4900.92
Apr-94	4921.47			4896.93	4901.44
May-94	4920.53		4877.18	4892.21	4900.59
Jun-94	4920.54		4876.76	4892.01	4900.39
Jul-94	4920.58		4876.54	4891.99	4900.21
Aug-94	4920.62		4876.56	4891.78	4900.00
Sep-94	4920.55		4876.28	4891.41	4900.02

month(ft)	MV-MWK	NW-TA3	SW-TA3	MWL-MW-2	MWL-MW-3
Jun-89	4911.99				
Jul-89	4904.35				
Aug-89	4904.08				
Sep-89	4904.03				
Oct-89	4903.40				
Nov-89	4903.52	4895.34	4903.52	4923.96	4920.14
Dec-89	4903.57	4895.26	4896.64	4924.01	
Jan-90	4903.66	4895.06	4898.84	4923.94	
Feb-90	4903.69	4895.17	4900.67	4924.19	
Mar-90	4903.38	4894.79		4923.75	4920.82
Apr-90	4903.13	4894.47		4922.97	4920.72
May-90	4903.02	4894.43	4897.78	4923.27	4920.86

month(ft)	MV-MWK	NW-TA3	SW-TA3	MWL-MW-2	MWL-MW-3
Jun-90	4902.84	4894.35	4893.94	4923.38	4921.93
Jul-90	4902.35	4894.21	4898.09	4923.29	4921.93
Aug-90	4902.13	4893.97	4900.52	4923.29	4921.92
Sep-90	4901.91	4893.95	4901.93	4923.23	4921.82
Oct-90		4893.30	4902.69	4922.95	4921.46
Nov-90	4901.46	4893.24	4902.98	4922.90	4921.31
Dec-90		4893.52	4903.10	4923.25	4921.67
Jan-91	4901.76	4893.32	4903.02	4923.08	4921.53
Feb-91	4901.54	4893.02	4902.92	4922.84	4921.31
Mar-91	4901.45	4893.13	4902.88	4923.24	4921.53
Apr-91	4901.16	4891.64	4898.75	4922.73	4921.24
May-91				4922.67	4921.10
Jun-91	4900.78			4922.31	4920.79
Jul-91	4900.47	4891.91	4901.26	4922.29	4920.75
Aug-91	4899.96	4891.60	4901.37	4922.21	4920.55
Sep-91	4900.05	4891.50	4901.61	4922.26	4920.71
Oct-91	4899.52	4891.38	4901.40	4922.26	4923.13
Nov-91	4901.42	4891.57	4901.60	4922.48	4923.36
Dec-91	4899.38	4891.17	4901.38	4921.98	4920.39
Jan-92	4900.93	4893.03	4903.18	4921.93	4920.54
Feb-92	4901.23	4890.57	4900.99	4921.93	4920.37
Mar-92	4899.21	4890.43	4900.95	4921.96	4920.46
Apr-92	4900.91	4891.63	4898.68	4923.37	4921.26
May-92	4898.84	4892.82	4902.50		
Jun-92	4897.42	4890.13	4900.65		
Jul-92	4898.53	4889.76	4900.06	4921.49	4919.99
Aug-92	4897.86	4889.75	4899.57	4921.66	4920.06
Sep-92	4897.44			4921.76	4920.12
Oct-92	4897.34	4889.42	4899.86	4921.76	4920.11
Nov-92	4898.16	4889.04	4899.80	4921.57	4919.10
Dec-92	4896.95	4888.91	4899.78	4921.47	4920.05
Jan-93	4897.21	4888.48	4899.33	4920.99	4919.41
Feb-93	4899.59	4887.78	4899.21		
Mar-93	4896.92	4887.83	4898.66	4920.99	4919.41
Apr-93	4896.46	4888.25		4920.66	4919.57
May-93	4895.85	4887.33	4895.67		
Jun-93	4896.64	4887.71	4896.81		
Jul-93					
Aug-93	4895.88	4888.13	4896.19		
Sep-93	4895.74	4887.56			
Oct-93	4898.22	4888.04	4890.66		
Nov-93	4904.06	4888.44	4899.21		
Dec-93	4898.32	4887.30	4895.82		
Jan-94	4895.44	4887.45	4892.94	4920.92	4969.98
Feb-94		4886.79	4893.79	4920.56	4918.86

month(ft)	MV-MWK	NW-TA3	SW-TA3	MWL-MW-2	MWL-MW-3
Mar-94	4894.78	4859.37	4894.72	4920.21	
Apr-94		4886.82	4894.26		
May-94	4894.42	4886.22	4894.24	4920.32	4918.73
Jun-94	4894.06	4886.12	4893.82	4920.40	4918.71
Jul-94	4893.85	4885.93	4894.47	4920.43	4918.74
Aug-94	4893.68	4885.91	4895.12	4920.47	4918.86
Sep-94	4893.72	4885.75	4895.44	4920.47	4918.70

month(ft)	MWL-BW-1	CWL-MW-4	Lagoon NE	Lagoon SE	Lagoon SW
Nov-89	4923.62				
Dec-89	4923.72				
Jan-90	4923.64				
Feb-90	4923.15				
Mar-90	4923.48				
Apr-90	4923.25				
May-90	4923.30	4937.08			
Jun-90	4923.35	4937.19			
Jul-90	4923.31	4937.12			
Aug-90	4923.22	4937.08			
Sep-90	4923.19	4937.01			
Oct-90	4922.84	4936.73			
Nov-90	4922.90	4936.65			
Dec-90	4923.24	4936.87	4875.55	4876.99	4877.74
Jan-91	4923.16	4936.75	4875.96	4876.75	4877.61
Feb-91	4922.84	4936.53	4875.88	4877.37	4878.13
Mar-91	4923.27	4936.82	4876.42	4876.43	4877.27
Apr-91	4922.65	4936.37	4875.40	4875.53	4876.44
May-91	4922.42	4936.37	4874.45	4874.74	4875.62
Jun-91	4922.21	4936.07	4873.55	4874.66	4875.25
Jul-91	4922.24	4936.11	4873.18	4874.85	4875.26
Aug-91	4922.13	4935.88	4873.05	4873.78	4874.62
Sep-91	4922.19	4935.95	4872.53	4875.24	4875.66
Oct-91	4922.14	4936.18	4873.40	4875.58	4875.98
Nov-91	4922.35	4936.40	4873.74	4875.14	4875.45
Dec-91	4921.95	4935.99	4873.52	4875.59	4876.25
Jan-92	4921.83	4936.54	4874.51	4875.67	4876.47
Feb-92	4921.84	4936.58	4875.34	4876.66	4877.41
Mar-92	4921.86	4935.61	4874.18	4875.20	4875.94
Apr-92	4922.82	4936.52	4875.54	4876.48	4877.31
May-92		4937.21	4872.82	4874.01	4874.81
Jun-92		4935.41	4871.04	4872.65	4873.36
Jul-92	4921.30	4935.31	4871.08	4872.56	4873.33
Aug-92	4921.51	4935.31	4870.87	4872.45	4873.13

month(ft)	MWL-BW-1	CWL-MW-4	Lagoon NE	Lagoon SE	Lagoon SW
Sep-92	4921.65	4935.25	4870.36	4871.85	4872.62
Oct-92	4921.57	4935.34	4870.50	4872.04	4872.71
Nov-92	4921.29	4935.15	4870.91	4872.10	4872.88
Dec-92	4921.27	4935.13	4871.46	4873.01	4873.35
Jan-93	4920.96	4934.76	4872.15	4873.91	4874.10
Feb-93		4933.61	4872.53	4874.24	4874.43
Mar-93	4920.96	4933.99	4872.07	4874.00	4874.09
Apr-93		4933.89	4872.57	4874.07	4874.15
May-93		4933.88	4871.13	4873.33	4873.35
Jun-93		4934.47	4869.83	4871.24	4869.75
Jul-93					
Aug-93		4934.16	4870.81	4872.46	4872.46
Sep-93		4934.56	4870.44	4872.83	4872.82
Oct-93		4935.09	4881.24	4873.03	4873.73
Nov-93		4933.76	4876.47	4878.75	4877.37
Dec-93		4934.34	4880.98	4872.60	4873.30
Jan-94	4972.82	4937.68	4871.00	4873.17	4873.14
Feb-94	4920.56	4934.12	4873.01	4872.99	4873.98
Mar-94	4920.06	4927.98	4865.21	4872.79	4872.77
Apr-94		4933.24	4874.07	4873.38	4873.64
May-94	4920.07	4933.68	4869.91	4872.18	4876.57
Jun-94	4919.95	4931.72	4869.86	4871.95	4872.10
Jul-94	4920.02	4933.78	4869.55	4871.84	4871.81
Aug-94	4920.09	4933.69	4875.92	4871.64	4871.57
Sep-94		4933.71	4868.22	4869.89	4870.62

month(ft)	Lagoon NW	Tijeras E	Tijeras W	Greystone	Coyote Spring
Nov-90		4911.71	4885.26	5764.90	5845.06
Dec-90				5765.86	5845.00
Jan-91	4876.24	4911.63	4884.99	5766.28	5845.11
Feb-91	4875.18	4911.44	4884.97	5766.21	5845.05
Mar-91	4876.67	4911.90	4885.13	5766.97	5845.00
Apr-91	4875.56	4915.53	4884.79	5767.11	5844.98
May-91	4874.44	4911.40	4884.38	5767.11	5845.00
Jun-91	4873.58	4910.90	4884.16	5766.95	5845.00
Jul-91	4872.18	4910.81	4883.79	5766.07	5845.10
Aug-91	4872.97	4910.73	4883.73	5766.24	5844.92
Sep-91	4872.06	4910.69	4883.24	5767.28	5844.85
Oct-91	4873.31	4910.84	4883.99	5766.32	5844.94
Nov-91	4873.62	4911.08	4884.15	5766.35	5844.92
Dec-91	4873.17	4910.74	4883.17	5767.27	5844.95
Jan-92	4874.48	4911.37	4884.51	5766.77	5845.03
Feb-92	4875.10	4910.59	4881.27	5766.80	5844.93

month(ft)	Lagoon NW	Tijeras E	Tijeras W	Greystone	Coyote Spring
Mar-92	4874.34	4910.65	4882.78	5766.80	5844.86
Apr-92	4875.76	4915.56	4884.87	5766.58	5844.99
May-92	4872.96	4910.54	4882.45	5766.48	5844.99
Jun-92	4872.18	4910.40	4881.45	5766.62	5844.76
Jul-92	4872.04	4910.09	4881.60	5765.81	5844.77
Aug-92	4870.89	4910.21	4881.67	5766.38	
Sep-92	4870.44	4910.12	4881.06	5765.67	5844.77
Oct-92	4870.78	4910.06	4880.93	5765.93	5844.88
Nov-92	4871.42	4910.21		5766.57	5844.85
Dec-92	4871.68	4909.24	4880.52	5765.53	5844.89
Jan-93	4872.31	4909.58	4880.57	5766.35	5849.32
Feb-93	4872.98	4909.75	4880.14	5767.19	5849.30
Mar-93	4872.12	4909.19	4879.90	5766.59	5849.28
Apr-93	4872.08	4909.24	4879.92	5766.03	5849.20
May-93	4871.20	4908.41	4879.36	5766.03	5848.69
Jun-93	4869.38	4909.28	4879.76	5765.96	5844.39
Jul-93					
Aug-93	4870.12	4909.44	4879.45	5765.35	5844.43
Sep-93	4870.37	4909.35	4879.19		
Oct-93	4870.77	4909.92	4880.44	5765.75	5849.06
Nov-93	4876.31	4914.34	4885.19	5766.81	5844.96
Dec-93	4870.41	4909.86	4881.19	5766.37	
Jan-94	4871.17	4909.76	4878.91	5767.14	5845.07
Feb-94	4872.67	4909.48	4877.42	5767.13	
Mar-94	4870.64	4909.53	4878.15	5767.12	5844.92
Apr-94	4871.55		4881.60	5766.65	5843.93
May-94	4869.92	4909.98	4877.89	5767.07	5844.92
Jun-94	4869.81	4908.83	4877.60	5766.95	
Jul-94	4869.42	4908.98	4877.46	5766.02	5845.14
Aug-94	4869.16	4908.95	4877.22	5765.28	5844.92
Sep-94	4868.21	4908.92	4876.96	5766.66	5845.07

month(ft)	TSA-1	School House	EOD Hill	Lake C.	GC East
Nov-90	5925.32			5659.35	
Dec-90	5924.55	5698.65	5687.38	5659.42	
Jan-91	5924.48	5697.86	5687.35	5659.38	5041.75
Feb-91	5924.44	5697.85	5687.30	5659.28	5041.92
Mar-91	5924.48	5698.64	5687.44	5659.32	5042.52
Apr-91		5698.68	5687.10	5659.14	5042.41
May-91		5698.71	5687.20	5659.10	5042.55
Jun-91		5699.73	5687.15	5659.06	5042.52
Jul-91		5697.97	5687.21	5656.75	5042.75
Aug-91		5698.90	5687.09	5656.85	5043.01

month(ft)	TSA-1	School House	EOD Hill	Lake C.	GC East
Sep-91		5699.64	5687.11	5659.52	5043.25
Oct-91		5699.02	5687.27	5656.95	5043.26
Nov-91		5699.05	5687.34	5656.95	5043.41
Dec-91		5698.99	5687.00		5043.90
Jan-92		5698.57	5685.79	5659.28	5043.50
Feb-92		5698.99	5685.54	5656.91	5044.57
Mar-92		5699.01	5685.57	5659.34	5044.95
Apr-92		5698.15	5685.63	5659.19	5042.55
May-92		5698.12	5685.83	5658.49	5045.72
Jun-92		5699.72	5684.06	5657.15	5042.78
Jul-92		5698.88	5685.60	5656.84	5045.87
Aug-92		5699.57	5685.66	5657.25	5046.23
Sep-92		5698.66	5684.00	5657.18	5046.42
Oct-92		5699.18	5685.69	5657.34	5046.87
Nov-92			5687.26	5657.34	5047.26
Dec-92		5699.00	5685.67	5657.65	5047.57
Jan-93		5698.92	5685.55	5657.55	5047.22
Feb-93		5699.58	5687.25	5657.02	5046.55
Mar-93		5698.47	5685.70	5657.08	5047.78
Apr-93		5698.62	5685.68		5047.74
May-93		5698.70	5685.08	5656.99	5047.57
Jun-93				5657.22	5048.40
Jul-93					
Aug-93		5697.53	5685.38	5658.03	5049.19
Sep-93		5698.85		5657.33	5049.33
Oct-93		5699.36	5686.30	5657.97	5050.20
Nov-93		5699.65	5689.71	5660.97	5045.56
Dec-93		5698.94	5686.30	5657.75	5049.29
Jan-94		5698.96	5685.94		5050.45
Feb-94		5698.89	5687.83	5660.72	5050.40
Mar-94		5698.94	5685.61	5657.81	5050.50
Apr-94			5686.28	5657.94	5049.73
May-94		5699.04	5685.86	5657.88	5050.95
Jun-94		5699.08	5685.78	5657.86	5050.90
Jul-94		5699.09	5685.79	5657.86	5051.09
Aug-94		5699.03	5685.81	5657.95	5051.24
Sep-94		5699.08	5685.82	5657.98	5051.53

month(ft)	GC South	GC West	GC North	Hubbel Spring	KAFB-8
Jan-91	5039.62	5047.55	5043.65		
Feb-91	5039.62	5047.67	5043.06		
Mar-91	5040.20	5048.16	5043.27		
Apr-91	5039.60	5048.05	5042.85		

month(ft)	GC South	GC West	GC North	Hubbel Spring	KAFB-8
May-91	5039.78	5049.15	5043.38		
Jun-91	5039.84	5048.14	5042.48		
Jul-91	5039.96	5048.24	5042.72		
Aug-91	5040.04	5048.35	5042.18		
Sep-91	5040.36	5048.56	5043.49		
Oct-91	5040.34	5048.59	5043.42	5441.17	
Nov-91	5040.51	5048.75	5043.59		
Dec-91	5041.25	5049.12	5044.23		
Jan-92	5040.62	5050.06	5044.25	5443.43	
Feb-92	5041.14	5049.29	5044.46		
Mar-92	5041.31	5049.53	5044.89		
Apr-92	5039.35	5048.33	5042.91		
May-92	5042.26	5049.92	5045.20	5443.42	
Jun-92	5041.04	5049.68	5041.16		
Jul-92	5042.21	5049.96	5045.40	5443.75	
Aug-92	5042.39	5050.15	5045.80		
Sep-92	5042.44	5050.22	5045.94		
Oct-92	5042.72	5050.59	5046.43		
Nov-92	5043.09	5050.92	5046.85		
Dec-92	5042.92	5050.70	5046.76		4866.98
Jan-93	5042.76	5050.66	5046.83		
Feb-93	5043.22	5051.69	5047.31		
Mar-93	5042.63	5050.49	5047.37	5442.75	
Apr-93	5043.00	5050.88	5047.41	5443.43	
May-93	5042.88	5050.69	5047.24		
Jun-93	5044.36	5052.87	5048.34		
Jul-93					
Aug-93	5044.37	5052.02			
Sep-93	5044.30	5052.08	5048.94	4879.19	5765.84
Oct-93	5045.06	5052.87	5049.80		5766.37
Nov-93	5042.92	5051.35	5046.91		5766.81
Dec-93	5045.20	5052.51	5049.76		5765.75
Jan-94	5045.04	5052.92	5050.12		
Feb-94	5044.89	5052.83	5050.13		
Mar-94	5044.94	5052.80	5050.15		
Apr-94	5046.80	5053.68	5050.14		
May-94	5045.24	5053.10	5050.53		
Jun-94	5045.22	5053.03	5050.56		
Jul-94	5045.79	5053.14	5049.70		
Aug-94	5045.97	5053.24	5050.82		
Sep-94	5046.14	5053.44	5051.08	5443.05	

References

[Bakr et al., 1978] Bakr, A., Gelhar, L. W., Gutjahr, A., and MacMillan, J. R. (1978). Stochastic analysis of spatial variability in subsurface flows: 1. comparison of one- and three-dimensional flows. *Water Resour. Res.*, 14(2):263–271.

[Carrera and Neuman, 1986] Carrera, J. and Neuman, S. P. (1986). Estimation of aquifer parameters under transient and steady state conditions, 1. maximum likelihood method incorporating prior information. *Water Resour. Res.*, 22(2):199–210.

[Celia and Gray, 1992] Celia, M. A. and Gray, W. G. (1992). *Numerical Methods for Differential Equations: Fundamental Concepts for Scientific and Engineering Applications.* Prentice Hall, Englewood Cliffs, New Jersey 07632. 436 pp.

[Clifton and Neuman, 1982] Clifton, P. M. and Neuman, S. P. (1982). Effects of kriging and inverse modeling on conditional simulation of the avra valley aquifer in southern arizona. *Water Resour. Res.*, 18(4):1215–1254.

[Cooley, 1985] Cooley, R. L. (1985). Regression modeling of ground-water flow. Technical report, U.S. Geological Survey, Open-File Report, Lakewood, Colorado.

[Dagan, 1982a] Dagan, G. (1982a). Analysis of flow through heterogeneous random aquifers, 2, unsteady flow in confined formations. *Water Resour. Res.*, 18(5):1571–1585.

[Dagan, 1982b] Dagan, G. (1982b). Stochastic modeling of groundwater flow by unconditional and conditional probabilities, 1, conditional simulation and the direct problem. *Water Resour. Res.*, 18(4):813–833.

[Dagan, 1985a] Dagan, G. (1985a). A note on higher-order corrections of the head covariances in steady aquifer flow. *Water Resour. Res.*, 21(4):573–578.

[Dagan, 1985b] Dagan, G. (1985b). Stochastic modeling of groundwater flow by unconditional and conditional probabilities: The inverse problem. *Water Resour. Res.*, 21(1):65–72.

[Dagan, 1986] Dagan, G. (1986). Statistical theory of groundwater flow and transport: Pore to laboratory, laboratory to formation, and formation to regional scale. *Water Resour. Res.*, 22(9):120S–134S.

[Dagan, 1989] Dagan, G. (1989). *Flow and Transport in Porous Formations.* Springer-Verlag, New York, N.Y. 465 pp.

[Dagan and Rubin, 1988] Dagan, G. and Rubin, Y. (1988). Stochastic identification of recharge, transmissivity, and storativity in aquifer transient flow: A quasi-steady approach. *Water Resour. Res.*, 24(10):1698–1710.

[Davis, 1987] Davis, M. (1987). Production of conditional simulations via the lu triangular decomposition of the covariance matrix. *Math. Geol.*, 19(2):91–98.

[Delhomme, 1979] Delhomme, J. P. (1979). Spatial variability and uncertainty in groundwater flow parameters: A geostatistical approach. *Water Resour. Res.*, 15(2):269–280.

[Dettinger and Wilson, 1981] Dettinger, M. D. and Wilson, J. L. (1981). First order analysis of uncertainty in numerical models of groundwater flow part 1. mathematical development. *Water Resour. Res.*, 17(1):149–161.

[Deutsch and Journel, 1992] Deutsch, C. V. and Journel, A. G. (1992). *GSLIB Geostatistical Software Library and User's Guide.* Oxford University Press. 340 pp.

[Dietrich and Newsam, 1989] Dietrich, C. R. and Newsam, G. N. (1989). A stability analysis of the geostatistical approach to aquifer transmissivity identification. *Stochastic Hydrol. Hydraul.*, 3:293–316.

[Doherty et al., 1994] Doherty, J., Brebber, L., and Whyte, P. (1994). *PEST: Parameter estimation for any model.* Watermark Computing, 336 Cliveden Ave., Corinda 4075, Australia. 132 pp.

[Freeze, 1975] Freeze, R. A. (1975). A stochastic-conceptual analysis of one-dimensional groundwater flow in nonuniform homogeneous media. *Water Resour. Res.*, 11(5):725–741.

[Freeze et al., 1991] Freeze, R. A., Massmann, J., Smith, L., Sperling, T., and James, B. (1991). Hydrogeological decision analysis: 1. a framework. *Groundwater*, 28(5):738–766.

[Gelhar, 1986] Gelhar, L. W. (1986). Stochastic subsurface hydrology from theory to applications. *Water Resour. Res.*, 22(9):135S–145S.

[Gelhar, 1993] Gelhar, L. W. (1993). *Stochastic Subsurface Hydrology.* Prentice-Hall, Englewood Cliffs, N.J. 390 pp.

[Gelhar et al., 1977] Gelhar, L. W., Bakr, A., Gutjahr, A., and MacMillan, J. R. (1977). Comments on 'a stochastic-conceptual analysis of one-dimensional groundwater flow in nonuniform homogeneous media' by r. a. freeze. *Water Resour. Res.*, 13(2):477–479.

[Gelhar et al., 1974] Gelhar, L. W., Ko, P. Y., Kwai, H. H., and Wilson, J. L. (1974). Stochastic modeling of groundwater systems. Technical Report 189, Ralph M. Parsons Laboratory for Water Resources and Hydrodynamics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02193. 313 pp.

[Ginn and Cushman, 1990] Ginn, T. R. and Cushman, J. H. (1990). Inverse methods for subsurface flow: A critical review of stochastic techniques. *Stochastic Hydrol. Hydraul.*, 4(1):1–26.

[Ginn et al., 1990] Ginn, T. R., Cushman, J. H., and Houck, M. H. (1990). A continuous-time inverse operator for groundwater and contaminant transport modeling: Deterministic case. *Water Resour. Res.*, 26(2):241–252.

[Gotway, 1994] Gotway, C. A. (1994). The use of conditional simulation in nuclear waste-site performance assessment. *Technometrics*, 36(2):129–161.

[Gutjahr, 1981] Gutjahr, A. (1981). Kriging in stochastic hydrology: Assessing the worth of data. In *AGU Chapman Conference on Spatial Variability in Hydrologic Modeling*, Fort Collins, Colorado.

[Gutjahr, 1989] Gutjahr, A. (1989). Fast fourier transforms for random field generation. Project report, New Mexico Tech. 106 pp.

[Gutjahr and Bras, 1990] Gutjahr, A. and Bras, R. (1990). Spatial variability in subsurface flow and transport: A review. *Reliability Engineering and System Safety*, 42:293–316.

[Gutjahr et al., 1994] Gutjahr, A., Bullard, B., Hatch, S., and Hughson, L. (1994). Joint conditional simulations and the spectral approach for flow modeling. *Stochastic Hydrol. Hydraul.*, 8(1):79–108.

[Gutjahr et al., 1978] Gutjahr, A., Gelhar, L. W., Bakr, A., and MacMillan, J. R. (1978). Stochastic analysis of spatial variability in subsurface flows: 2. evaluation and applications. *Water Resour. Res.*, 14(5):953–960.

[Gutjahr and Wilson, 1989] Gutjahr, A. and Wilson, J. L. (1989). Co-kriging for stochastic models. *Transport in Porous Media*, 4(6):585–598.

[Hantush and Marino, 1995] Hantush, M. M. and Marino, M. A. (1995). Continuous time stochastic analysis of groundwater flow in heterogeneous aquifers. *Water Resour. Res.*, 31(3):565–575.

[Hoeksema and Kitanidis, 1984] Hoeksema, R. J. and Kitanidis, P. K. (1984). An application of the geostatistical approach to the inverse problem in two-dimensional groundwater modeling. *Water Resour. Res.*, 20(7):1003–1020.

[Hoeksema and Kitanidis, 1985] Hoeksema, R. J. and Kitanidis, P. K. (1985). Analysis of the spatial structure of properties of selected aquifers. *Water Resour. Res.*, 21(4):563–572.

[Kernodle et al., 1995] Kernodle, J. M., McAda, D. P., and Thorn, C. R. (1995). Simulation of ground-water flow in the albuquerque basin, central new mexico, 1901-1994, with projections to 2020. Technical report, U.S. Geological Survey, Water-Resources Investigations Report 94-4251, Albuquerque, New Mexico.

[Kitanidis and Vomvoris, 1983] Kitanidis, P. K. and Vomvoris, E. G. (1983). A geostatistical approach to the inverse problem in groundwater modeling (steady state) and one-dimensional simulations. *Water Resour. Res.*, 19(3):677–690.

[LaVenue and Pickens, 1992] LaVenue, A. M. and Pickens, J. F. (1992). Application of a coupled adjoint-sensitivity and kriging approach to calibrate a ground-water flow model. *Water Resour. Res.*, 28(6):1543–1570.

[Lent and Kitanidis, 1996] Lent, T. V. and Kitanidis, P. K. (1996). Effects of first-order approximations on head and specific discharge covariances in high-contrast log conductivity. *Water Resour. Res.*, 32(5):1197–1207.

[Mantoglou and Wilson, 1982] Mantoglou, A. and Wilson, J. L. (1982). The turning bands method for simulation of random fields using line generation by a spectral method. *Water Resour. Res.*, 18(5):1379–1394.

[Marquardt, 1963] Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society of Industrial and Applied Mathematics*, 11(2):431–441.

[Marsily et al., 1984] Marsily, G., Lavedan, G., Boucher, M., and Fasanino, G. (1984). Interpretation of interference tests in a well field using geostatistical techniques to fit the permeability distribution in a reservoir model. In et al., G. V., editor, *Geostatistics for Natural Resources Characterization, Part 2*, number 122 in C. Reidel, Dordrecht, The Netherlands.

[Matheron, 1973] Matheron, G. (1973). The intrinsic random functions and their applications. *Adv. Appl. Prob.*, 5:439–468.

[McCord et al., 1993] McCord, J., McCord, J., Burck, P., Gibson, J. D., Neel, D., Lauffer, F., Wolford, R., Treadway, A., Kelson, K., Thomas, E., Fourz, W., Gaither, B., and Hyndman, D. (1993). Site-wide hydrogeologic characterization project. Annual report, Sandia National Laboratories.

[McLaughlin, 1976] McLaughlin, D. B. (1976). Application of kalman filtering to groundwater basin modeling and prediction. Presented at the IIASA Workshop on Recent Developments in Real Time Forecasting/Control of Water Resource Systems.

[Mizell et al., 1982] Mizell, S. A., Gutjahr, A., and Gelhar, L. W. (1982). Stochastic analysis of spatial variability in two-dimensional steady groundwater flow assuming stationary and non-stationary heads. *Water Resour. Res.*, 18(4):1053–1067.

[Myers, 1982] Myers, D. E. (1982). Matrix formulation of co-kriging. *Math. Geol.*, 14(3):249–257.

[Press et al., 1992] Press, W. H., Teukoisky, S. A., Vettering, W. T., and Flannery, B. R. (1992). *Numerical recipes in FORTRAN: The art of scientific computing, second editon.* Cambridge University Press. 963 pp.

[Rubin and Dagan, 1988] Rubin, Y. and Dagan, G. (1988). Stochastic analysis of boundaries effects on head spatial variability in heterogeneous aquifers: 1. constant head boundary. *Water Resour. Res.*, 24(10):1689–1697.

[Rubin and Dagan, 1989] Rubin, Y. and Dagan, G. (1989). Stochastic analysis of boundaries effects on head spatial variability in heterogeneous media: 1. impervious boundary. *Water Resour. Res.*, 25(4):707–712.

[Rushton and Redshaw, 1979] Rushton, K. R. and Redshaw, S. C. (1979). *Seepage and Groundwater Flow.* Wiley, New York. 339 pp.

[Schafer-Perini and Wilson, 1991] Schafer-Perini, A. and Wilson, J. L. (1991). Efficient and accurate front tracking for two-dimensional groundwater flow models. *Water Resour. Res.*, 27(7):1471–1485.

[Smith and Freeze, 1979] Smith, L. and Freeze, R. A. (1979). Stochastic analysis of steady state groundwater flow in bounded domain, 2: Two-dimensional simulations. *Water Resour. Res.*, 15:1543–1559.

[Sun and Yeh, 1992] Sun, N.-Z. and Yeh, W. W.-G. (1992). A stochastic inverse solution for transient groundwater flow: Parameter identification and reliability analysis. *Water Resour. Res.*, 28(12):3269–3280.

[Sykes et al., 1985] Sykes, J. F., Wilson, J. L., and Andrews, P. W. (1985). Sensitivity analysis for steady state groundwater flow using adjoint operators. *Water Resour. Res.*, 21(3):359–371.

[Tompson et al., 1989] Tompson, A. F. B., Ababou, R., and Gelhar, L. W. (1989). Implementation of the three-dimensional turning bands random field generator. *Water Resour. Res.*, 25(10):2227–2243.

[Townley and Wilson, 1985] Townley, L. R. and Wilson, J. L. (1985). Computationally efficient algorithms for parameter estimation and uncertainty propagation in numerical models of groundwater flow. *Water Resour. Res.*, 21(12):1851–1860.

[Townly and Wilson, 1983] Townly, L. R. and Wilson, J. L. (1983). Conditional second moment analysis of groundwater flow: The cumulative effects of transmissivity and head measurements. In *Papers of the International Conference on Groundwater and Man.* Australian Gevernment Publishing Service, Canberra.

[Wang and Anderson, 1982] Wang, H. F. and Anderson, M. P. (1982). *Introduction to Groundwater Modeling: Finite Difference and Finite Element Methods.* W. H. Freeman and Company, New York. 237 pp.

[Wilson and Dettinger, 1978] Wilson, J. L. and Dettinger, M. D. (1978). Steady-state vs. transient parameter estimation in groundwater systems, verfication of mathematical models in hydraulic engineering. In *Proceedings of the 26th Annual Hydraulics Division Special Conference.* Am. Soc. Civ. Eng., New York.

[Yeh et al., 1996] Yeh, T.-C. J., Jin, M., and Hanna, S. (1996). An iterative stochastic invers method: Conditional effective transmissivity and hydraulic head fields. *Water Resour. Res.*, 32(1):85–92.

This dissertation is accepted on behalf of the faculty of the Institute by the following committee:

John L Wilson 3/31/97
Adviser

allan Stuhy 4/21/97

Fred M. Phillips 21 April 97

James T McLord 3/18/97

Date

I release this document to New Mexico Institute of Mining and Technology.

Debra L Hughes Apr 22 1997
Students Signature Date