Stochastic numerical modelling of groundwater flow and solute transport

Author:
Davis, Steven Richard

Publication Date:
2003

DOI:
https://doi.org/10.26190/unsworks/8042

STOCHASTIC NUMERICAL MODELLING OF GROUNDWATER FLOW AND SOLUTE TRANSPORT

S. DAVIS

A thesis submitted in fulfilment

of the requirements for the degree of

Doctor of Philosophy

University of New South Wales

February 2003

Abstract

A stochastic numerical perturbation method is applied to the groundwater flow and solute transport problems. The method involves modelling the hydraulic conductivity as a random field, discretising this random field into a vector of random variables, expanding this vector and the vectors representing, groundwater velocity, and solute concentration in Taylor series about their means and using stochastic finite element methods to relate the three parameters. Methods of calculating the second spatial moment of the mean solute plume and the mean second moment of individual solute plumes are presented. The method is implemented in a computer program and results for a range of integral scales are presented. These results are compared to analytical results found in the literature. It is found that the groundwater velocity results give very good agreement, except when mesh resolution becomes a problem, and that the solute concentration results give reasonable agreement for practical integral scales.

The work presented in this thesis builds on previous work that uses stochastic finite element methods in that it determines the statistics of the groundwater velocity field in the groundwater flow problem, determines the statistics of the concentration field in the solute transport problem from the statistics of the hydraulic conductivity field, and uses a second order method for the mean values, thus avoiding the problem of using the same result as obtained in the deterministic case.

Acknowledgements

I would like to thank my supervisor Garry Mostyn for all of his guidance and encouragement.

I would like to thank Russel Standish of the High Performance Computing Unit at UNSW for helping me to optimise my program for running on parallel supercomputers.

I would like to thank my family, especially my parents, my sister and my in-laws, for all of their help, support and child-minding.

Most importantly, I would especially like to thank my wife, not only for her help and support, but also for putting up with me. At last she will have a husband who is not "doing a PhD"!

Table of Contents

Table of Figures

1 Introduction

This thesis applies a stochastic numerical perturbation method to the groundwater flow and solute transport problems.

1.1 Statement of the Problem

It is well known that the dispersive behaviour of solute transport is much greater at field scales than that observed in the laboratory. Research has shown that this is due to variation in groundwater flow, caused by variation in aquifer properties (Dagan 1989, Gelhar, 1993). However, it is not possible to gather data and model an aquifer at small enough scale to capture all of this variation in a deterministic sense because of the huge amount of data involved, and because the act of gathering such data would greatly modify the properties being measured.

One approach that has been used to deal with this is to use a random field description of the aquifer (Vanmarke, 1983).

Analytical solutions to the governing stochastic equations have been developed using various approximations (for example Dagan,1982b, Gelhar and Axeness, 1983, Rajaram and Gelhar, 1993a & b, Kapoor and Gelhar 1994a & b). However, analytical solutions have the disadvantages that they are restricted to infinite (or semi-infinite) geometries, cannot deal with aquifers made up of distinct geological units, and the solute sources are limited in spatial geometries, often restricted to point sources, and temporal distribution, either instantaneous or a constant continuous value.

In the deterministic area these problems are often overcome using numerical methods. Stochastic finite element methods have been developed that use a random field description of the hydraulic conductivity. However, they have the limitations that they only determine the hydraulic head values (Sagar, 1978, Vanmarke and Hachich, 1983, Vanmarke, 1994, Hantush and Marino, 1995, Ghanem, 1998, Tartakovsky and Neuman, 1998, Guadini and Neuman, and Winter et al, 2002), assume that the groundwater velocity statistics are given (Graham and McLaughlin, 1989a & b), or use the first order result for mean values (which is equal to the deterministic result) and require cross covariances to be calculated (Kunstman, 2002).

There is a need for a stochastic finite element method that overcomes these limitations.

1.2 Objectives of Study

The objectives of the study are:

(i) To develop a stochastic finite element method that determines the statistics of the velocity field in the groundwater flow problem from the statistics of the hydraulic conductivity field of the aquifer.

(ii) To further develop this method to determine the statistics of the concentration field in the solute transport problem from the statistics of the hydraulic conductivity field of the aquifer.

(iii) To compare results from using this method with appropriate analytical solutions to demonstrate the effectiveness of the method.

These objectives have been met.

1.3 Thesis Outline

Chapter 2 presents the background to the problem explaining the physical processes involved in solute transport and why the most common method of modelling solute transport, the advection dispersion equation, is deficient and why aquifer heterogeneity is an important property. It also details different probabilistic methods of modelling the variation within the aquifer, including a random field model.

Chapter 3 provides a literature review of research that applies the random field model to groundwater flow and solute transport. It discusses this in the categories of Monte Carlo simulation, analytical methods, and other numerical methods.

Chapter 4 gives the methodology behind the perturbation method and its application to groundwater flow and solute transport. The algorithms for each step of the method are presented. Also algorithms for calculating other interesting properties of the solute plume from the results are presented. It also looks at issues related to upstream weighting and presents several methods of reducing the computation time required for the method.

Chapter 5 presents groundwater velocity results from use of the method and compares these to analytical results found in the literature.

Chapter 6 presents solute concentration results from use of the method and compares these to analytical results found in the literature.

Chapter 7 makes some suggestions about further work that could proceed from the thesis.

2 Background

This chapter looks at the contaminant transport problem and discusses the limitations of deterministic methods of modelling the problem. It does this by looking at the processes involved in the transport of contamination by groundwater and discusses how the current method models these processes. It explains why the heterogeneity of an aquifer is an important aquifer property when modelling solute transport and discusses ways of modelling this heterogeneity. This provides the background to the literature review outlined in Chapter 3 and the methodology presented in Chapter 4.

2.1 Solute Transport

This research deals with water-soluble contaminants. The significance of this is that when the contaminant enters an aquifer it dissolves in the groundwater and becomes a constituent or phase of the groundwater. As such it moves within the aquifer following the same travel path as the groundwater. This movement is a result of hydraulic head gradients and is referred to as advection, see Figure 2.1.

As time passes the contaminant also tends to spread out. The boundaries of the contamination plume become less sharp as the concentration gradients reduce through processes of diffusion and mixing, and less certain due to the uncertainty in the aquifer properties. This results in a plume that fills an increasingly larger volume but has an increasingly lower concentration, see Figure 2.1.

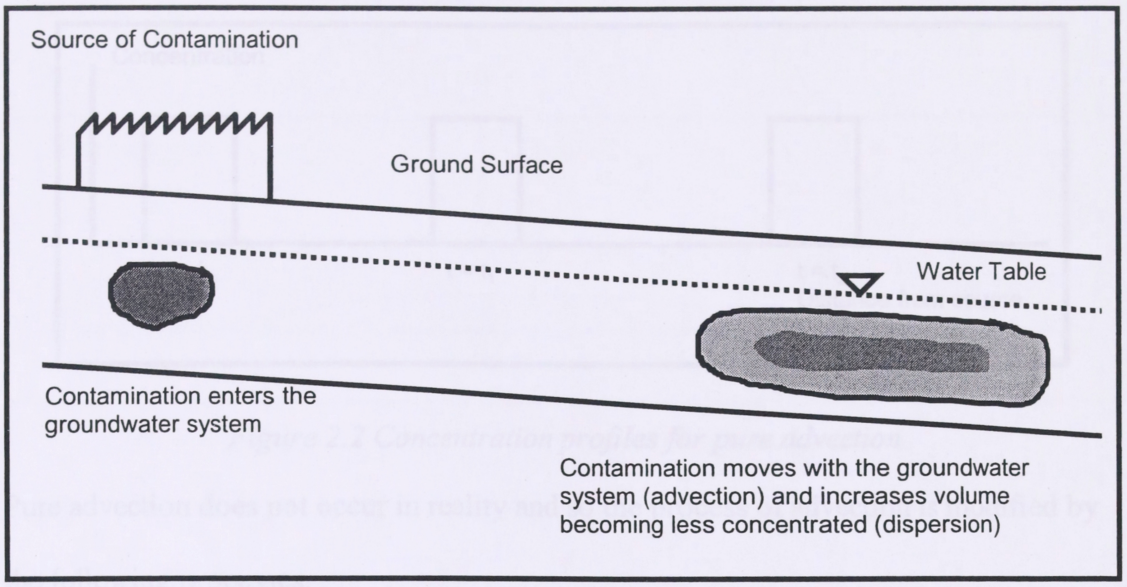


Figure 2.1 Movement of contamination in groundwater system

2.1.1 Advection

Advection is the movement of contaminants with the groundwater flow in an aquifer. When a contaminant enters an aquifer it is dissolved in, or displaces, the groundwater already present in the aquifer. Normally this groundwater is moving under the influence of hydraulic head gradients. Thus any contaminant dissolved in the groundwater is carried along with it. If pure advection existed it would involve a contaminant plume having sharp edges. Just outside the boundaries of the contaminant plume there would be no contamination and just inside the boundaries the contamination would be at the original concentration, see Figure 2.2.

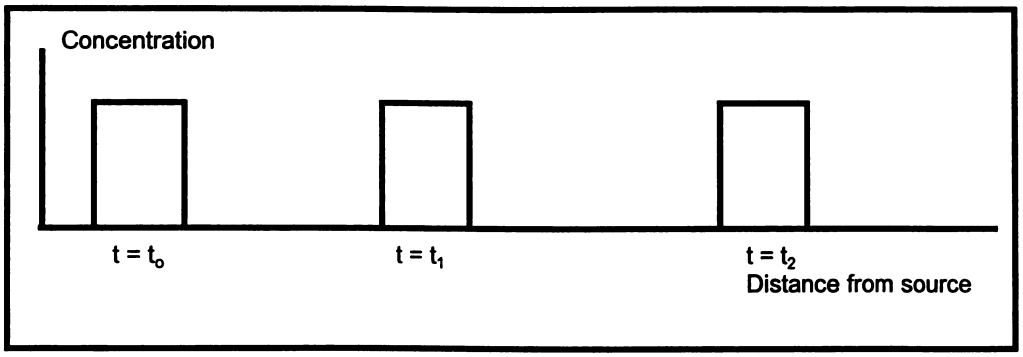


Figure 2.2 Concentration profiles for pure advection

Pure advection does not occur in reality and so the process of advection is modified by the following processes.

2.1.2 Diffusion

Diffusion is the spreading out of a solute in response to concentration gradients, see Figure 2.3. The process is the result of Brownian motion moving particles in random directions. Since there are more contaminant particles in high concentration areas than low concentration areas it is more likely for a particle to move from a high concentration area to a low one than vice versa. The solute flux occurring as a result of diffusion is a linear function of the concentration gradient. This type of behaviour is described as Fickian and is described by Fick's law:

$$\mathbf{F} = D\nabla C \qquad \text{Equation 2.1}$$

Where

$\mathbf{F}$ = Solute flux
D = Diffusion coefficient
∇ = Gradient operator
C = Concentration field

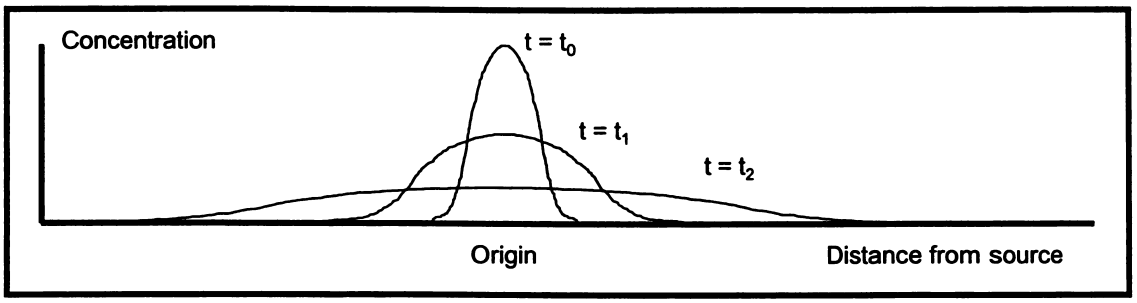


Figure 2.3 Effect of diffusion on a plug of contaminant

In a soil the behaviour is slightly different to a drop of solute in a beaker. Soil is a porous media and so the diffusion of the solute is impeded by the soil particles. Therefore the diffusion coefficient determined in the laboratory must be modified for use in an aquifer. This is done by a parameter called the tortuosity, τ. This allows for the tortuous paths that the solute must follow to diffuse throughout the aquifer. Tortuosity is an aquifer property, not a contaminant property, and typically varies from 0.05 to 0.5 for most soils.

2.1.3 Mechanical Dispersion

Mechanical dispersion is a pore scale process that results in the mixing of different concentrations of solution due to variations in velocity within a pore and length of flow path among different pores.

In a single pore there is a variation of pore water velocity. See Figure 2.4. In a throat between particles the pore water flows slower near the walls of the pore than in the centre. Thus the pore water in the centre reaches the next pore faster. If the concentration in the two pores is different then this will result in an averaging out of the

concentration through mixing rather than a simple replacement of the contents of one pore with another.

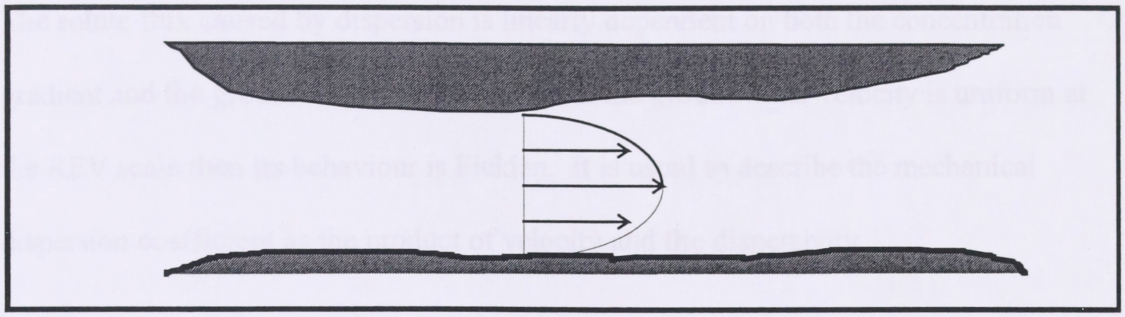


Figure 2.4 Pore water velocity variations

Pore water can also take different paths to travel from one position to another. See Figure 2.5. This enables the groundwater from different pores to mix together thus further diluting the solute and spreading out the plume.

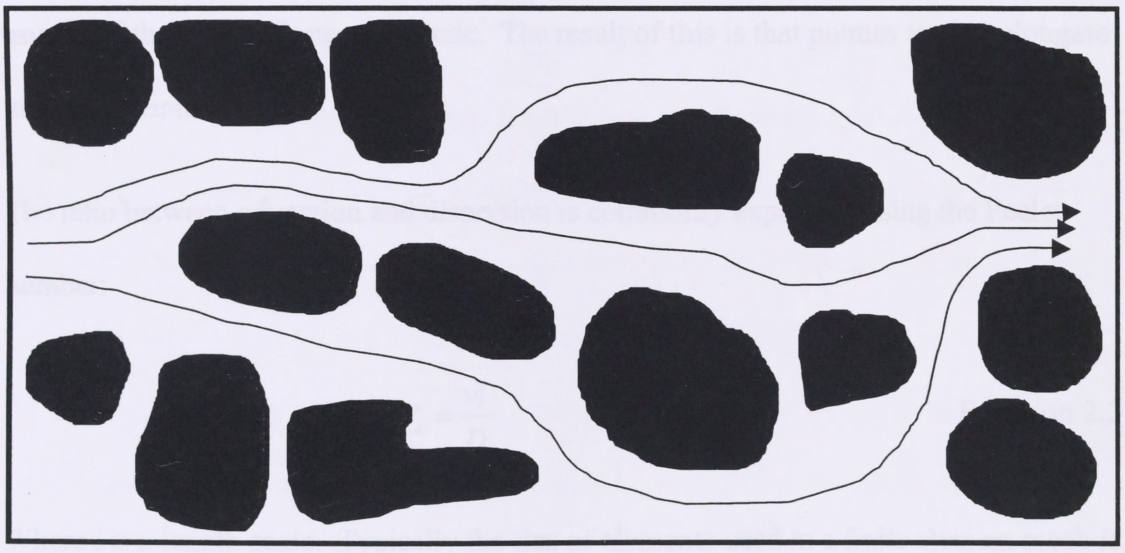


Figure 2.5 Diversity of flow path lengths

Modelling the details of the groundwater flow at the pore scale is impracticable. Therefore the concept of a representative elementary volume (REV) is introduced. This is a small volume that is just large enough so that when the groundwater velocity is

averaged out over the volume the result is consistent when small changes are made in the size of the volume.

The solute flux caused by dispersion is linearly dependent on both the concentration gradient and the groundwater velocity. Thus if the groundwater velocity is uniform at the REV scale then its behaviour is Fickian. It is usual to describe the mechanical dispersion coefficient as the product of velocity and the dispersivity.

$$D = \alpha v \qquad \text{Equation 2.2}$$

Where α is the dispersivity and v is the pore water velocity.

Experiments show that the longitudinal dispersivity, i.e. in the direction of the groundwater flow, is much larger than the transverse dispersivity. The difference is usually of the order of one magnitude. The result of this is that plumes tend to elongate in the direction of flow.

The ratio between advection and dispersion is commonly expressed using the Peclet number:

$$P_e = \frac{vl}{D} \qquad \text{Equation 2.3}$$

Where l is a length scale. Typically the size of elements used in a finite element mesh is used for l. Finite element meshes with high Peclet numbers tend to have oscillation problems. This will be discussed in more detail in Section 4.5.

2.1.4 Heterogeneity of the Aquifer

Aquifers are not uniform in the properties relating to contaminant transport. These properties vary in space and may even vary in time. The main properties relating to contaminant transport are hydraulic conductivity, porosity and the chemical properties of the aquifer that cause chemical reactions with the contaminant.

Hydraulic conductivity is one of the most variable properties in nature, varying over 15 orders of magnitude from gravel to clay. It also varies within an individual aquifer. This can be a discrete process caused by bedding, sand lenses, clay seams, etc or a gradual continuous process caused by localised variations within a coherent aquifer unit. The result of this non-uniformity is that groundwater tends to find preferential flow paths through which the bulk of the groundwater flow passes, see Figure 2.6. When a contaminant enters an aquifer some of it enters preferential flow paths of varying velocities. This fragmentation of the plume into sections travelling at different velocities is the dominant effect causing dispersion of the plume at engineering scales. This process is called macro dispersion. The effect of macro dispersion may be many orders of magnitude greater than diffusion and mechanical dispersion (Gelhar, 1993). Macro dispersion is non-Fickian and is greater for larger plumes than for smaller ones, i.e. it is scale dependent, because larger plumes contain greater variations than smaller ones.

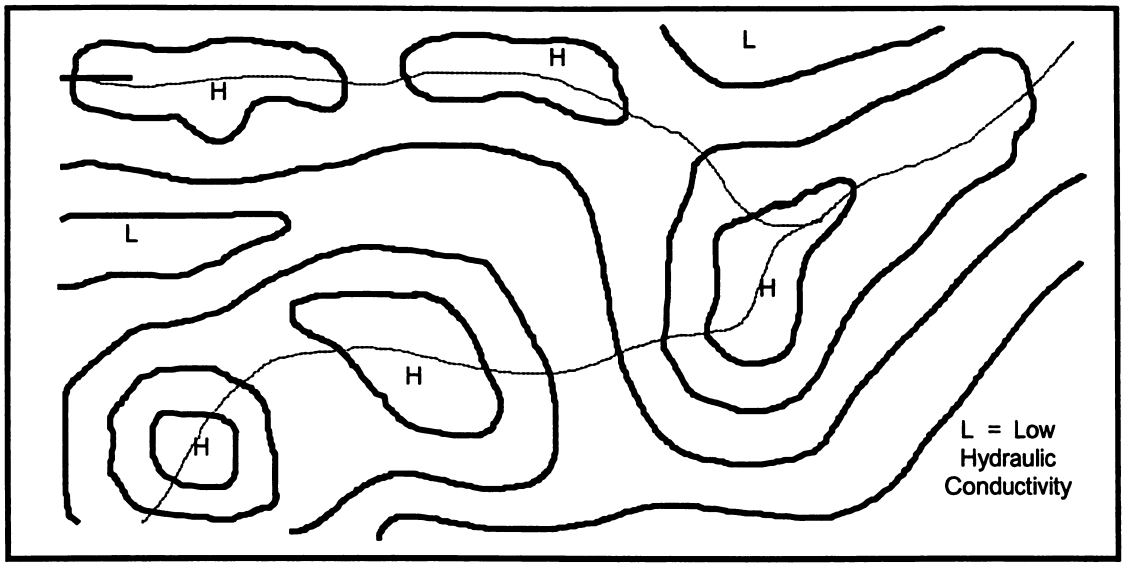


Figure 2.6 Preferential flow paths caused by heterogeneity in the aquifer

Porosity also affects contaminant transport. However generally the change in porosity is much smaller in magnitude than the change in hydraulic conductivity. Thus its effect is usually ignored in preference to investigation of the effect of variation of hydraulic conductivity.

Variation in the chemical properties that control any reactions between the contaminant and the aquifer can also have a large impact on the behaviour of non-conservative contaminants. This form of variation is not investigated in this thesis as only conservative contaminants are being modelled.

Other spatial heterogeneities that may occur in aquifers include variations in tortuosity, affecting diffusion, and dispersivity, affecting mechanical dispersion. However, the effect of the variability of these parameters is minor compared to the effect of the variability of hydraulic conductivity (Gelhar, 1983). Thus these parameters are neglected in this research to highlight the behaviour caused by variation in hydraulic conductivity.

2.1.5 Current Methods of Modelling the Problem

Normally this problem is modelled by the advection dispersion equation (Fetter, 1993):

$$D_L \frac{\partial^2 C}{\partial x^2} + D_T \frac{\partial^2 C}{\partial y^2} - v_x \frac{\partial C}{\partial x} = \frac{\partial C}{\partial t} \qquad \text{Equation 2.4}$$

where

C = Concentration

t = time

v_x = groundwater velocity in the x direction (the Cartesian coordinate axes are selected so that the direction of groundwater flow is in the x direction)

D_L = longitudinal dispersion coefficient

D_T = transverse dispersion coefficient

If Ficks Law is assumed for mechanical dispersion then the dispersion coefficients will be a linear function of velocity.

The velocity is determined from the hydraulic head field using Darcy's Law (cf Rushton and Redshaw, 1979):

$$v = -k\nabla H \qquad \text{Equation 2.5}$$

where

H = hydraulic head field

k = hydraulic conductivity tensor

∇ is the gradient operator.

Assuming steady state flow the hydraulic head field is determined from the hydraulic conductivity by (cf Rushton and Redshaw, 1979):

$$k_x \frac{\partial^2 H}{\partial x^2} + k_y \frac{\partial^2 H}{\partial y^2} = q \qquad \text{Equation 2.6}$$

Where q is an in inflow or outflow at a source or sink (for example a pumping well), and is zero at all other points in the aquifer, and k_x and k_y are the components of hydraulic conductivity, which may vary from point to point.

These equations can be solved using numerical methods such as finite element or finite differences. Equation 2.4 has been solved analytically for the two limiting cases in a uniform velocity over an infinite space of a finite mass being released at a point in space and time, and a constant value of source concentration at a point. Commonly these analytical solutions are used in engineering practice by finding which case most closely approximates the field conditions.

The first term on the right hand side of Equation 2.4 models advection, the second term models dispersion. This equation assumes Fickian dispersion, thus it is strictly only applicable to diffusion and mechanical dispersion. Laboratory testing implies using a small sample that can only model the effects of diffusion, mechanical dispersion, and very small-scale non-homogeneities, and not the large-scale heterogeneities that dominate the process in natural aquifers. Thus a tracer test equivalent in size to the actual contaminant plume is required to determine the contribution from heterogeneity in the conductivity field. Thus the value for the dispersion coefficient is either fitted to

the data or extrapolated from other sites, neither method caters for the scale dependence of the effect.

Thus this method is inappropriate for two reasons: it does not correctly model the dominant behaviour (i.e. macro dispersion) and the major parameter (i.e. the dispersion coefficient) is fitted to the data by back calculation rather than by determination prior to calculation.

However if the heterogeneity could be modelled exactly by knowing the hydraulic conductivity at every point then the equation would accurately represent real world behaviour. Effectively this means transferring the macro dispersion component of the behaviour from the dispersion term in the advection dispersion equation to the advection term, thus dealing with it in Equation 2.5 and Equation 2.6. To do this we need a method of characterising the aquifer that contains more detail.

2.2 Characterisation of an Aquifer

There are many ways of characterising an aquifer. Different methods are appropriate for different purposes. This section will describe different methods leading up the exposition of a random field as a way of characterising a heterogenous hydraulic conductivity field.

2.2.1 Deterministic

The simplest way to characterise an aquifer is deterministically. This means that the hydraulic conductivity at every point is assumed to be a known definite value. This is the most commonly used method. Generally a particular value of hydraulic

conductivity is assumed for each aquifer unit and that value is uniform throughout each unit.

If an elaborate system of testing is carried out, the hydraulic conductivity may be defined on a fine enough grid to accurately and reasonably characterise the aquifer. However this level of testing would leave the aquifer perforated with bore holes such that the hydraulic conductivity could be drastically altered.

Instead it is much more common to use test data to derive an effective hydraulic conductivity that gives a behaviour equivalent to the average behaviour of the aquifer. This characterisation obviously contains no information about the variability of the aquifer.

2.2.2 Single Random Variable

The next simplest method of characterising an aquifer is with a random variable representing each unit of the aquifer. If a simple model is taken of an aquifer that consists only of a single unit then the aquifer is assumed to have a single value for hydraulic conductivity but this value is unknown and so a probabilistic analysis can be carried out. For example the probability of groundwater flow into an excavation exceeding a certain level could be determined. However, features that depend on the non-uniform structure of the aquifer cannot be modelled. This is because the aquifer is assumed to have a uniform structure even though the actual uniform value is assumed to be variable.

This method was used in Chowdhury and Zhang (1988).

2.2.3 White Noise

Another way that the aquifer can be characterised is by a system of white noise. This means that the aquifer is assumed to have a different, random value for the hydraulic conductivity at each point, see Figure 2.7. Each point would be represented by a different random variable. Thus an infinite number of random variables would be needed. Practically this could be modelled by dividing the aquifer into a large number of very small elements and assigning an independent random variable to each element. This gives the model of the aquifer some form of structure by which the features that depend on the non-uniform structure of the aquifer might be able to be modelled.

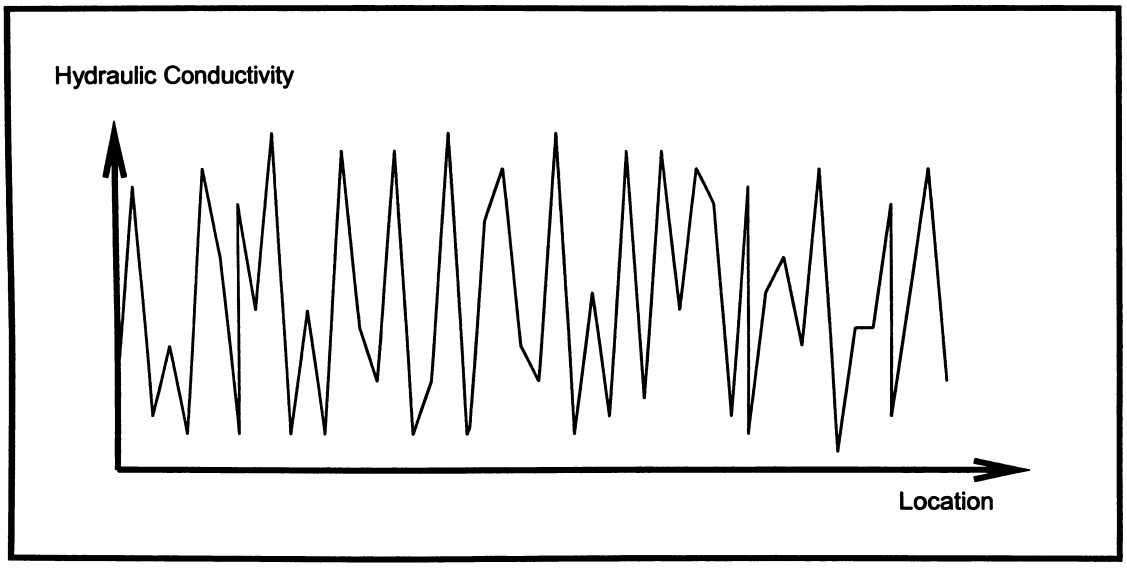


Figure 2.7 Example of one dimensional white noise variation of hydraulic conductivity

The single random variable model can also be thought of as this model but with all the random variables being perfectly correlated with each other instead of being independent.

This method was used in Freeze (1975), mainly because more sophisticated techniques were unavailable.

2.2.4 Random Field

In the real world the hydraulic conductivities of points that are closer together are more likely to have similar values than points that are far apart, see Figure 2.8. Thus for very small parts of the aquifer the single random variable appears appropriate, while for large distances the white noise approach appears better. For intermediate distances there will be variation between points but when one point has a higher than average value we would expect nearby points to have higher than average values, or both to have lower than average values but still both values are unknown and not expected to be the same.

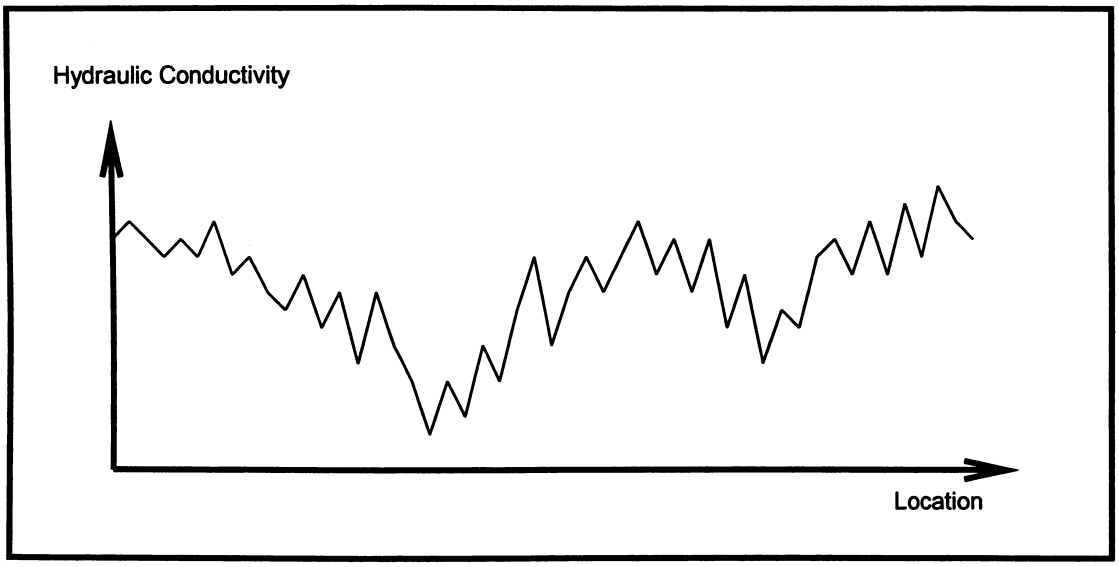


Figure 2.8 Example of aquifer with positive auto correlation function

A form of model with these properties is the random field (Vanmarcke, 1983). A random field is characterised by an auto correlation function. It is assumed that each point has a value that can be represented by a random variable. However each random variable is not independent from every other random variable, nor are they perfectly correlated. Instead they are correlated according to the separation vector between them.

The auto correlation between two points is the expected value of the product of the deviations of the values from the means at the two points divided by the product of the standard deviations of the random variables at the two points. Commonly it is assumed that over a large enough length/area/volume the mean is constant (i.e. no trend) and the variability is constant (the field is stationary). For a stationary random field the autocorrelation function is defined by (VanMarcke, 1983):

$$\rho(X_1, X_2) = \frac{\mathrm{E}[(K(X_1) - \mathrm{E}(K))(K(X_2) - \mathrm{E}(K))]}{\sigma^2} \qquad \text{Equation 2.7}$$

Where

$\rho(X_1, X_2)$ = Auto correlation between points X_1 and X_2
$K(X)$ = Hydraulic conductivity at point X
$\mathrm{E}[K]$ = Mean value of hydraulic conductivity for the random field
σ^2 = Variance of the point values of hydraulic conductivity

The auto correlation can vary from minus one to one. If the random variables are independent then the auto correlation is zero. The reverse is not necessarily true. An auto correlation of one means that the two points always have the same value. For the case of normally distributed random variables, an auto correlation of zero implies that the random variables are independent. A negative auto correlation implies that when one value is high the other is low. This may occur for example where there are alternating beds with only two uniform values and each bed has the same thickness. This case would give an auto correlation of minus one at the distance equivalent to the thickness of the beds. See Figure 2.9.

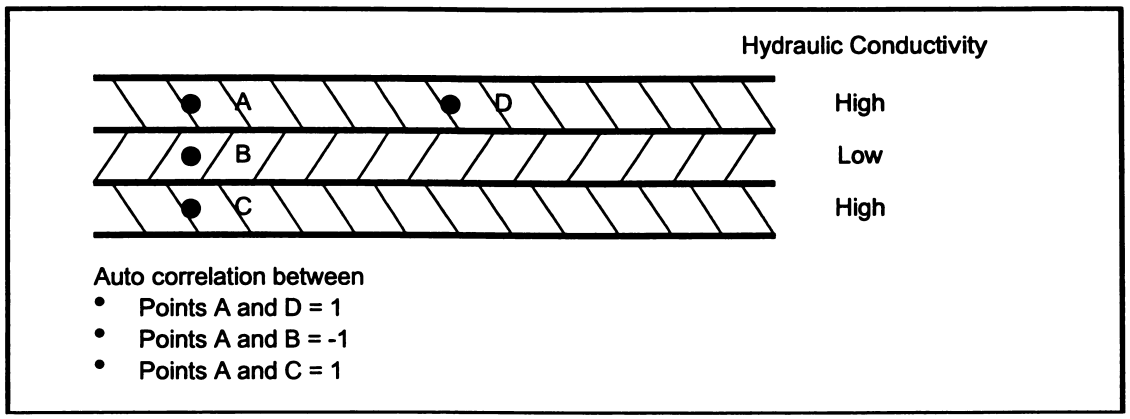


Figure 2.9 Auto correlation for uniform alternating constant thickness beds

The auto correlation is usually modelled as a function of the separation vector only. This is referred to as a homogeneous random field since the auto correlation is the same regardless of the location of the two points provided they have the same separation. If the auto correlation function is the same for a given separation distance regardless of direction then the random field is also isotropic. If it varies differently in one direction than in the other it is anisotropic. Bedding is an example of an anisotropic random field.

Statistics for a random field model can be generated for any other type of model of an aquifer. For example the above case of uniform alternating constant thickness beds has an anisotropic auto correlation function where the correlation (i) is a wave function oscillating between minus one and one with the wavelength being twice the bed thickness in one direction and (ii) is equal to one within the plane of the beds. Similarly auto correlation functions can be generated for fractal realisations. Using these more particular models may give a more accurate model, but the random field model is more general and so can model more situations just by modifying the auto correlation function. It should be remembered that all of these models are only approximations of reality, but for many purposes a random field has been found to be a very useful approximation (Gelhar, 1993).

In a deterministic model of an aquifer all properties are defined exactly at all points. In a stochastic model the properties are defined probabilistically. Ideally this would mean that the joint probability distributions were known. However, normally it is assumed that the first two moments of the probability distributions are known (Gelhar, 1983). Thus the mean at any point and the covariance between any two points are known. If X_1 is the value of the random field at point $\mathbf{x}_1$ and X_2 is the value at $\mathbf{x}_2$ then the covariance is given by (VanMarcke, 1983):

$$\begin{aligned}\operatorname{cov}(X_1, X_2) &= \mathrm{E}\left[\left(X_1 - \mathrm{E}[X_1]\right)\left(X_2 - \mathrm{E}[X_2]\right)\right] \\ &= \mathrm{E}[X_1 X_2] - \mathrm{E}[X_1]\mathrm{E}[X_2]\end{aligned} \qquad \text{Equation 2.8}$$

where E[X] is the mean of X. Commonly the covariance is given in terms of the autocorrelation function (VanMarcke, 1983):

$$\rho(\mathbf{x}_1, \mathbf{x}_2) = \frac{\operatorname{cov}(X_1, X_2)}{\sigma_{X_1}\sigma_{X_2}} \qquad \text{Equation 2.9}$$

where ρ is the autocorrelation and σ_X is the standard deviation of X. If a random field is stationary then ρ is only dependent on the separation vector $\mathbf{x} = \mathbf{x}_1 - \mathbf{x}_2$. If it only depends on the magnitude and not the direction of $\mathbf{x}$ then it is isotropic, otherwise it is anisotropic.

Commonly an autocorrelation function is characterised by its integral scale. The integral scale in a particular direction is defined as the integral of the autocorrelation function over all positive values of the separation vector in that direction.

$$I = \int_0^{\infty} \rho(x)\, dx \qquad \text{Equation 2.10}$$

Where I is the integral scale.

I is commonly used as a parameter in the autocorrelation function. Small values of the integral scale indicate that the related random variable fluctuates over short distances, large values indicate that the random variable is constant over short distances and fluctuates over large ones. In the extremes a value of zero for the integral scale gives the aforementioned white noise model and an infinite value gives the aforementioned single random variable model.

Another parameter that is commonly used to characterise an auto correlation function is the scale of fluctuation. The scale of fluctuation is the integral of the autocorrelation function over all values, both positive and negative, of the separation vector in that dimension. Since the autocorrelation function is symmetrical the scale of fluctuation is exactly twice the integral scale. It is usually denoted by θ. Most authors in the groundwater area tend to use integral scale rather than scale of fluctuation and so that practice will be adopted here.

3 Potential Solution Schemes

The purpose of this Chapter is to enumerate solution schemes to the problem developed in the last chapter. As such it will also function as a literature review discussing methods pursued by other authors. It will begin by discussing the Monte Carlo simulation method through which many of the concepts of stochastic solutions can be presented in an easy to understand manner. It will then look at analytical solutions that have been extensively pursued. Next numerical solutions will be considered. This will involve showing how stochastic solutions are built from deterministic solutions and the different types of stochastic solutions that are available. Finally a perusal of discretisation methods will be made.

3.1 Monte Carlo Simulation

A stochastic problem involves determining the probability distributions of outputs for a particular system given the probability distributions of the inputs, see Figure 3.1.

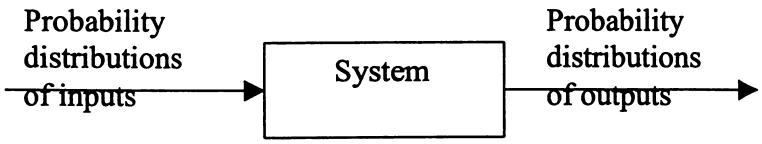


Figure 3.1 Schema for a stochastic problem

Monte Carlo simulation solves the stochastic problem by turning it into a large number of deterministic problems, see Figure 3.3.

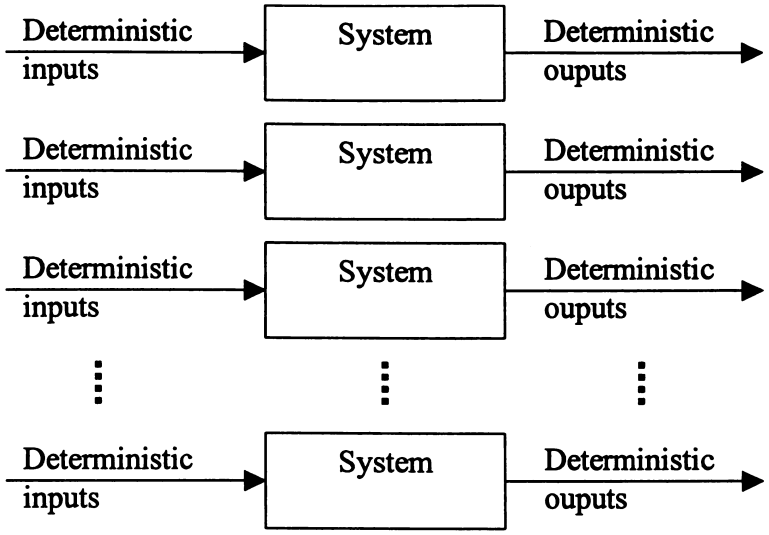


Figure 3.2 Schema for Monte Carlo simulation

The deterministic inputs are selected so that their combined statistics match the probability distributions of the stochastic inputs. Each of the deterministic problems is called a realisation. The statistics of the stochastic outputs are assumed to be equal to the statistics of the set of deterministic outputs.

To give an example of this take the case of a problem involving a single random input variable and a single random dependent output variable. See Figure 3.3.

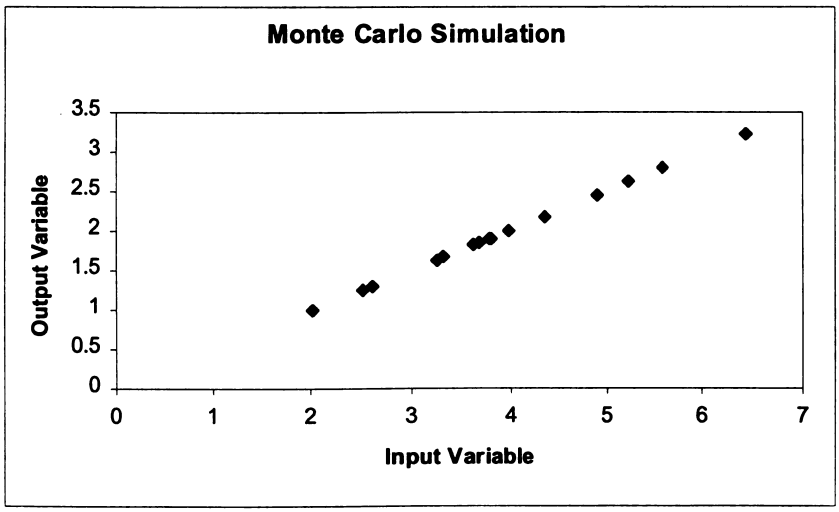


Figure 3.3 Example of a Monte Carlo simulation

Firstly a set of realisations is generated. This example has sixteen realisations of the input variable between 2 and 7. Notice that they do not come in even increments but have random values designed to fit the statistics of the input variable. Each of these realisations is processed to give a value of the output variable. In the example the results range from 1 to 3. The statistics of the output variable can be estimated from the set of results. In this particular case this could have been done directly from the statistics of the input variable since the operator involved is linear. However in most engineering situations this is not the case and the Monte Carlo Method is the simplest way to obtain the required statistics.

The input required for the contaminant transport problem is the hydraulic conductivity field. Usually the deterministic solution of each realisation involves a finite element or finite difference approach. Therefore the realisation is defined by the values chosen for each element or finite difference point in the domain of the problem. This realisation needs to be carefully generated to fit the statistics of the random field.

In practice the application of the Monte Carlo method to the contaminant transport problem involves:

1. Discretising the random field into a set of random variables
2. Determining the statistics of the random variables from the random field
3. Generating a set of numbers that satisfy the statistics of the random variables
4. Determining the flow field from these values for hydraulic conductivity
5. Determining the groundwater velocities within each element

6. Determining the location of the plume at the desired time

7. Repeating until a statistically significant set of data has been gathered

8. Collating the statistics of all of this data.

Various statistics that can be collected of the transported plume are:

- The mean concentration at each point in the aquifer
- The variance of the concentration at each point in the aquifer
- The mean of the position of the centre of the plume
- The variance of the position of the centre of the plume
- The mean and variance of the second moment of each realisation about the centre of the plume
- The mean and variance of the second moment of the sum of realisations of the plume

Each of these statistics will give us different information about the plume and will be used in different contexts. For example if an investigation of a well for extracting groundwater is being carried out then the concentration statistics are desired. If a remediation program is being proposed then the second moment statistics will be required to delineate the extent of the problem.

3.1.1 Groundwater Flow

The earliest work looking at solutions for the stochastic groundwater flow problem, Freeze (1975), applies Monte Carlo simulation to a one dimensional finite element problem and determines the statistics of the values of hydraulic head along the flow. This work assumes that the hydraulic conductivity values in each element is independent of the values in all other elements, similar to the white noise model discussed in Section 2.2.3.

Later Smith and Freeze (1979a) use a nearest neighbour stochastic process model to include the effects of correlation between elements that are close together. Smith and Freeze (1979b) extend this to two dimensions. In the nearest neighbour model the values for the hydraulic conductivity only depend on the nearest elements. Obviously all elements are connected by neighbours, however, this limits the shape of the auto correlation function.

Later, Bellin et al (1992) perform two-dimensional simulations using a fast Fourier transform method to generate the realisations.

Fenton and Griffiths (1993) use two dimensional simulations to determine effective block conductivities and demonstrated that local averaging (see Section 3.4.2) of log conductivities provides good results.

The Turning Bands Method, originally presented in Mantoglou and Wilson (1982), was developed so that random fields could be generated efficiently for any form of autocorrelation function. It is probably the most commonly used method of generating random fields in the groundwater area. It is a lot faster than using matrix methods, but

it is not as accurate, however, the minor loss in accuracy is not seen as significant. A program for applying a matrix decomposition method is presented in El-Kadi and Williams (2000). Dietrich and Newsam (1993) develop an exact method that is similar in speed to the Turning Bands Method. Bellin et al (1994) develop another method based on conditioning successive random variables on previous random variables.

Shrestha and Loganathan (1994) give a method to calculate how many realisations are needed to ensure that the sample mean for hydraulic head calculations is within a given limit of the population mean for a given confidence interval.

One of the problems with Monte Carlo simulation is that it may require a large number of realisations to obtain the statistically significant number of outputs required to obtain meaningful results. This may require a large amount of computational time. Therefore these early works tended to use finite element meshes with very small numbers of elements. Attempts to use large meshes have sometimes compromised on the number of realisations. For example, Ababou et al (1989) perform a single realisation study on a three-dimensional aquifer.

Meyer et al (1989) present methods of performing fast solutions of the finite difference method optimised for use on supercomputers. Ashby and Falgout (1996) investigate optimising this further for parallel computations by using preconditioning. They use an iterative solution technique for solving the large matrices required and show that preconditioning techniques can help. Dykaar and Kitanidis (1993) use this method to perform three dimensional simulations to determine effective transmissivity and find that simple methods, such as depth averaging, leads to significant errors.

3.1.2 Concentration

Application of Monte Carlo methods to the contaminant transport problem is much more difficult. This is because the aquifer must be divided into much smaller elements to generate the required behaviour. The large amount of computing power required to generate the values for hydraulic conductivity and repeatedly solve the problem for all of the realisations make this problem very time consuming.

Russo (1984) uses Monte Carlo simulation with 50 realisations to model salination of a soil profile focussing on vertical flow through the unsaturated zone. However, he models the aquifer as a cluster of one dimensional columns of soil through which the groundwater percolates and there is no flow between columns.

The problem has been undertaken in two dimensions by Bellin et al (1992). The earliest large scale example in three dimensions is by Tompson and Gelhar (1990) who generate one realisation of the hydraulic conductivity field and simulate four realisations of the contaminant transport problem by placing the initial plume in different parts of the aquifer. Burr et al (1994) perform 5 three dimensional simulations of a tracer test at Bordern in Ontario, Canada. Moreno and Tsang (1994) use single realisation studies with varying standard deviations and integral scales to show that for large standard deviations the flow tends to mainly pass through distinct channels. Jussel et al (1994a&b) use ten realisations to model a gravel deposit in Switzerland. They were limited to ten by supercomputer time constraints. Naff et al (1998a & b) perform three dimensional simulations to determine the effects of integral scale and initial plume size. They use approximately 20 simulations per parameter set and state that 80 would have given much better results, but that they were limited in computer time.

Hassan et al (1998) provide a comparison table of Monte Carlo simulation studies in the literature. Most of them are two dimensional and of those that are three dimensional the number of realisations is quite small, often only one.

Scheibe and Cole (1994) use Monte Carlo simulation at using both coarse and fine element meshes. They ascribe the difference to dispersive effects and then remodel using the coarse mesh with the dispersive effects included and find that they get good agreement with the original fine mesh results.

The Monte Carlo method itself is an approximation based on the assumption that the behaviour in the field, no matter what the precise details of the realisation in the field, will reach some form of average value for all the realisations. This is referred to as ergodicity. Other methods are used to overcome the Monte Carlo method's problem with computational load, but they are quite often an approximation of the model used in the Monte Carlo method and thus it must be recognised that they are another step away from the field situation when their results are interpreted.

Rubin (1990) uses analytical results to generate the covariance field for the groundwater velocity covariance matrix and then uses particle tracking to generate the concentration field. Particle tracking involves placing a "particle" at an initial location of the velocity field for a particular realisation and simulating over time where it will move. The concentration for any small volume is then determined at a particular time as being proportional to the number of particles resident in the volume. This gives the mean of the concentration, but not higher order moments. Goode (1990) presents an improved method of interpolating the groundwater velocities between the nodes for use in particle tracking. Rubin (1991) extends the particle tracking method to determine the spatial

moments of the plume and their variance. Tompkins et al (1994) use a finite difference method to generate the velocity field realisations from the hydraulic conductivity realisations and then use particle tracking to determine the concentrations. Saladin and Fiorotto (1998) use particle tracking in two dimensions to investigate the effects of using large log hydraulic conductivity variances. Feyen et al (2001) use particle tracking to delineate the capture zone for a well.

Crane and Blunt (1999) present a streamline based method. For each realisation of the hydraulic conductivity field they determine the locations of the streamlines. They then carry out one dimensional analyses along each streamline. The main limitation of the method is that it does not deal with local dispersion.

In other fields, such as structural mechanics, methods have been developed to reduce the computations involved in Monte Carlo simulation. These include importance sampling (Melchers, 1990, Yaacob, 1991, and Fu and Moses, 1992) and preconditioning the realisations of the random field to exactly match the statistics of the random field, thus requiring fewer realisations (Yamazaki and Shinozuka, 1990).

3.2 Analytical Solutions

3.2.1 Groundwater Flow

In analytic methods groundwater flow is solved by substituting the probability density functions, auto correlation functions, and/or spectral density functions into the flow equation and integrating with respect to the random input variables to obtain the probability density functions or statistical moments of the output variables. The advantages of this approach are exactness, and transparency of solution to third party

inspection. However they are often limited in application to very prescribed circumstances (eg simple geometries). Early work determined relationships between head variance and log conductivity variance (Bakr et al, 1978; Gutjahar and Gelhar, 1981; Mizell et al,1982) or determined effective conductivities (Gutjahar et al, 1978, Gelhar and Axness, 1983, Paleologos et al, 1996, Indelman et al, 1996). The effective conductivity is the value of hydraulic conductivity that a uniform aquifer with the same geometric properties would require to obtain the same average groundwater velocity as the non-uniform aquifer. Tartakovsky et al (2000) determines the effective transmissivity from the statistics of the hydraulic conductivity field.

Dagan (1979) determined limits on the effective conductivity and hydraulic head variance using a model of the aquifer involving uniform lumps of constant hydraulic permeability. This was then extended to transient flow where it was shown that the arithmetic average of the flow field satisfies Darcy's Law and the continuity equation if effective values for hydraulic conductivity storativity are used and the average head varies slowly in space and time.

Investigations into the applicability of some of the assumptions made by earlier work were made by Gutjahr (1984) and Dagan (1985). Van Lent and Kitanidis, (1996) investigated the applicability of the perturbation approximations used in many of the analytical solutions.

Li and McLaughlin (1991), Li and McLaughlin (1995), Indelman And Rubin (1995) remove the stationary (no trend) assumption. Zhang et al (2000) develops a general approach to nonstationary random fields.

Another thread of research work looks at conditional probabilities (Dagan 1982a).

A spectral approach was used by Dykaar and Kitanidis (1992a&b) to find effective conductivity. Neuman and Orr (1993) show that effective conductivities may not exist in many cases, for example pumping from a well. Beckie et al (1994) shows that effective conductivities are appropriate when there is a spectral gap, and investigates appropriate resolving scales in other cases. Beckie et al (1996) develops a model that resolves subgrid heterogeneity to the grid scale. Beckie (1996) extends this to data gathering measurements.

Rubin and Dagan (1992) determine analytical solutions for the cross covariances between each of log-conductivity, groundwater velocity, and hydraulic head. Hsu and Neuman (1997) look at the effects on the mean, variance and autocorrelation structure of the groundwater of extending the analysis to second order in the variance of the hydraulic conductivity.

Karakas and Kavvas (2000) develop a conservation equation for the mean groundwater velocity that includes random spatial variability in the hydraulic conductivity, storativity and porosity.

Chan and Govindarau (2001) use an interval computing method that enables them to convert the probability distributions into fuzzy numbers to determine the mean concentrations.

Another thread of research looks at the inverse problem (Neumen and Yakowitz, 1979). This means determining the hydraulic conductivities or transmissivities from measurements of the hydraulic head at various points. This tends to give results with lower point variances near the measurements and higher ones further away.

3.2.2 Concentration

Generally the majority of published research in the area of contaminant transport in random fields is analytical in nature. That is, a solution to the advection dispersion equation is sought that is stochastic in nature and based on the statistics of the random field, especially on the auto correlation function. Common practice has been to seek a closed form expression for the dispersion coefficient and then substitute this into the advection dispersion equation. This uses the property that the dispersion coefficient is half of the rate of change of the second spatial moment of the concentration profile of the plume (Dagan, 1989).

Dagan (1982b) gives analytical results for the second moments of the ensemble average of the plumes in a two dimensional aquifer. Dagan (1984) shows that conditioning these results on measured values of the hydraulic head field has little effect on the variance (uncertainty) of the concentration field. Sudicky (1986) and Barry et al (1988) show that these results give good approximations to a large scale tracer test carried out at Borden in Ontario, Canada. Dagan (1990) derives expressions for the variance of the second moment of the ensemble. Cvetkovic and Dagan (1994) include the effects of sorption of the solute with soil particles. Russo (1995) derives expressions for the second moments of the ensemble average for three dimensional anisotropic cases with flow parallel and perpendicular to the beds.

Neuman (1993) shows that conditioning results on measurements transfers information from the dispersive flux term to the advective flux term.

Gelhar and Axness (1983) determine the three dimensional macrodispersivity tensor for the anisotropic case, at arbitrary angles to the bedding, and include the effects of local

dispersion. Neuman et al (1987) shows that these results are dependent on Peclet number and give differing results for high Peclet numbers. Rehfeldt and Gelhar (1992) extend this work to unsteady flow and show that this yields better results than the steady flow when applied to tracer tests, particularly for the transverse macrodispersivity. Neuweiler et al (2001) and Attinger et al (2001) determine effective macrodispersivities for radial flow and find that without diffusion the macrodispersivity is constant but that with diffusion it depends on the radius, horizontal diffusion increases macrodispersion for larger radii while vertical diffusion decreases it for larger radii.

Shapiro and Cvetkovic (1988) and Dagan and Nguyen (1989) investigate the properties of the breakthrough curve. In other words instead of looking at where the solute is at a particular time they focus on one point and determine when the solute arrives there and what the concentration is at different times. Rubin and Dagan (1992) investigate the effect on the breakthrough curve of conditioning on measurements. Selroos (1995) examines the temporal moments, that is, how spread out the breakthrough curve is.

Dagan et al (1992) and Cvetkovic et al (1992) take a solute flux approach. That is, rather than looking at how much concentration exists at a point in space or time they investigate the rate that the solute flows through a control plane.

Jury and Scotter (1994) use a streamtube approach to determine the probability density functions of travel time and travel distance. Their approach neglects local dispersion. Toride and Leij (1996a & b) add the effects of local dispersion. However, it appears that this is within a streamtube not across them.

Other work has investigated evolving integral scales (Dagan, 1994b). This means that as a larger portion of the aquifer is sampled the shape of the auto correlation function

changes so that the integral scale increases. The cause of this may be that formation units are made up of smaller units and the smaller units have less variability between them than the larger ones. As the plume travels and grows larger and encompasses larger formation units the macrodispersion will jump to a higher level. Dagan (1994b) shows that under these circumstances the area under the auto correlation function may become infinite and if this occurs a constant dispersion coefficient cannot be found. Fiori (2001a) extends this by adding the effects of local dispersion. Serrano (1994) and Serrano (1997) derive a non-Fickian dispersion equation by assuming two scales.

Dagan (1991), Rajaram and Gelhar (1993a & b), Zhang et al (1996), Zhang and Zhang (1997) and Dentz et al (2000a & b) recognise that there is a difference between the average of the second moment of each realisation of the plume and the second moment of the sum of the plumes. The second moment of the sum of the plumes is necessarily larger since it also includes the variance in the position of the centre of the plume. As the travel time becomes large these two types of second moment should converge to a single value. Dagan (1991) gives charts for the dispersion coefficient based on the average of the second moment of each realisation of the plume as time approaches infinity. Dagan (1994a) adds time varying behaviour to this. Zhang (1997) looks at the variance of the second moment of individual realisations. Rajaram and Gelhar (1995) investigate how the second moment of individual realisations is affected by evolving scales. Most of these works use an exponential covariance function. Zhang and Di Federico (1998) extend this work to use a Gaussian autocorrelation function and Di Federico and Zhang (1999) extend it to a fractional Gaussian noise autocorrelation function.

Kapoor and Gelhar (1994a & b) investigate the fluctuations in the concentration field. The size of these fluctuations at a point is proportional to the standard deviation of the concentration at that point. They show that the processes that cause macrodispersion create these fluctuations, for example as a preferential flow path carries solute into a solute free area it creates a fluctuation or difference between the high concentration in the flow path and the much lower (or zero) concentration outside it. They also show that the only process that can destroy these fluctuations is local dispersion, for example the solute will diffuse and/or mechanically disperse from the area of high concentration to low concentration. They give analytical expressions for this creation and destruction. Adricevic (1998) includes the effects of sampling volume to this. Kitanidis (1994) investigates this idea from a different direction by defining a dilution index, which describes how much the solute has diluted rather than simply how much it has spread. Zhang and Neuman (1996) investigate this using a different method and conclude that when the ratio of local dispersion to macrodispersion is small the local dispersion can be ignored. Fiori (2001b) uses some of these concepts to determine estimates of the maximum concentration, which would be higher than the maximum mean concentration.

Some attempts have been made at looking at higher than second moments by using a second order approach. Naff (1992 & 1994) uses a lot of complicated operator development and provides a set of integrals requiring numerical solution. Dagan (1994c) uses a fourth order analysis of the transverse macrodispersivity to show that the effects of the higher order moments are quite small.

Destouni and Graham (1995) and Zhang and Lu (2002) investigate the effect on transport regimes of coupling the unsaturated and saturated zones.

Dagan et al (1996) shows the effect on solute transport of periodic (eg seasonal) changes in direction of groundwater flow.

Lessof et al (2000) investigate the effect of a constant groundwater velocity boundary and find that it suppresses macrodispersion in a zone near the boundary.

Most of the previous work uses a perturbation technique and the results assume that the stochastic system is in some sense linear and therefore higher order terms are ignored. Serrano (1996) attemps to overcome this limitation by using the method of decomposition and finds good agreement with the Bordern aquifer tests.

Vanderborght (2001) takes a different approach in deriving the analytical expressions determining the probability distribution of whether a particle in a given location at a given time was in the initial volume at the start, instead of integrating over the initial volume.

Rubin (2000) determines the effective macrodispersivity that can be used to resolve subgrid heterogeneity in deterministic numerical applications.

In summary perturbation methods have been used to obtain analytical solutions to the groundwater flow and solute transport problems. As in the deterministic cases these are generally restricted to a point source in an infinite media or a continuous source in a semi-infinite media. One of the major benefits closed form analytical solutions is that they help with understanding the problem. They are also useful if the approximations involved in applying them (such as assuming infinite boundaries) are inconsequential. However, for solving problems where boundaries are important, particularly boundaries between the units that make up the aquifer, numerical solutions are required.

3.3 Numerical Solutions

3.3.1 Deterministic Solution

For the contaminant transport problem there are three stages in solution.

Firstly the hydraulic head field must be determined from the hydraulic conductivity field. This is done using the flow equation. The hydraulic head field may be steady or unsteady. If it is steady then the problem is a simple boundary value problem. If it is unsteady then it is a more complicated initial value problem. Unsteady effects can arise from pumping, infiltration or the lack thereof, and fluctuation in boundaries (surface water levels).

Secondly the velocity field is determined from the hydraulic head field. This is done using Darcy's law (Equation 2.5). For steady flow a single set of velocities will be obtained. For unsteady flow a continually evolving set of velocities will be obtained.

Finally the concentration field is determined using the advection dispersion equation. This is an initial value problem whether the flow field is steady or unsteady since it depends on the initial plume condition.

These three stages are generally performed sequentially and are uncoupled. However if the contaminant is very dense and/or concentrated then it can affect the flow of the groundwater and so a more complicated set of flow equations must be coupled to the advection dispersion equation. In this research work it will be assumed that these density effects are negligible.

3.3.2 Stochastic Solutions

For stochastic problems there are three forms of numerical solution. The primary numerical method in stochastic problems is the Monte Carlo method discussed in Section 3.1. This method gives complete information about the probability distributions of the output values for a given set of input values (provided a sufficiently large number of realisations has been used). However the Monte Carlo Method can be prohibitively expensive in computation time. Therefore other methods have been developed to provide specific information about the output variables. These methods are less computationally intensive but also give less information.

The two main groups of other numerical methods are perturbation methods and reliability methods. These methods have not been applied extensively in the groundwater area, but have been in the structural mechanics area. Therefore a quick overview of some of the developments from this area will be given.

3.3.2.1 Perturbation Methods

Perturbation methods involve calculating the solution at a particular point, usually the mean point, and determining how this solution changes as the various input variables change. The latter is equivalent to determining the derivatives of the output variables with respect to the input variables. When both the input variables and output variables are described by Taylor series it is possible to relate the statistics of the output to the input. Of course this relation is only as accurate as the Taylor series, which depends both on the closeness of the point to the point about which the Taylor series is being made and on the accuracy of the Taylor series (i.e. the number of terms in the Taylor series). Sections 4.2 and 4.3 give more details about implementing this method.

Perturbation methods have been extensively researched in the structural mechanics area where they have been taken from simple static problems to more complex dynamic problems (for example see Hisada and Nakagiri, 1981; Liu et al, 1986a & b; Vanmarke et al,1986; Nakagiri, 1987; Ditlevsen, 1988). Nakagiri (1987) discusses both the static and eigenvalue problems using this approach. Liu et al (1987) apply the method to elastic plastic dynamic problems. They also use a diagonalisation of the covariance matrix to reduce the amount of computation required. Liu et al (1988) apply the method to linear dynamic problems and further refine the solution by decoupling and using Lanczos basis and give a method for removing secular terms from the results.

A similar alternative method investigated by Katafygiotis and Beck (1995) is to use a Fourier series expansion instead of a Taylor series.

3.3.2.2 Reliability Methods

The perturbation method seeks to determine the behaviour of the system about the mean point. This suffers from loss of accuracy in the domain away from the mean point. Engineering is often interested in extremes rather than in averages. For example, with proverbial chain that is only as strong as its weakest link, it is the extreme value that matters not the average. If the Taylor series were exact in the perturbation method then the probability of failure could easily be determined from it. However, the response is rarely normally distributed and this most strongly influences the behaviour at the tails of the statistical distributions. Therefore the reliability method has been developed specifically for determining the probability of failure. See Der Kiureghan and Ke (1988).

For the reliability method a performance function is defined G(X) such that when G(X) > 0 the structure will not fail and when G(X) < 0 the structure will fail. The domain, X, is the vector of random variables upon which the response of the structure depends. The surface G(X) = 0 is called the failure surface. The probability of failure is the volume under the probability density function integrated over the area where G(X) < 0.

To determine the reliability a point known as either the most probable point or the design point is found. This point is the point on the failure surface that is closest to the mean point. Various methods for determining this point are compared in Liu and Der Kiureghian (1991b). The distance from the mean point to the design point is called the reliability and given the symbol β, see Figure 3.4.

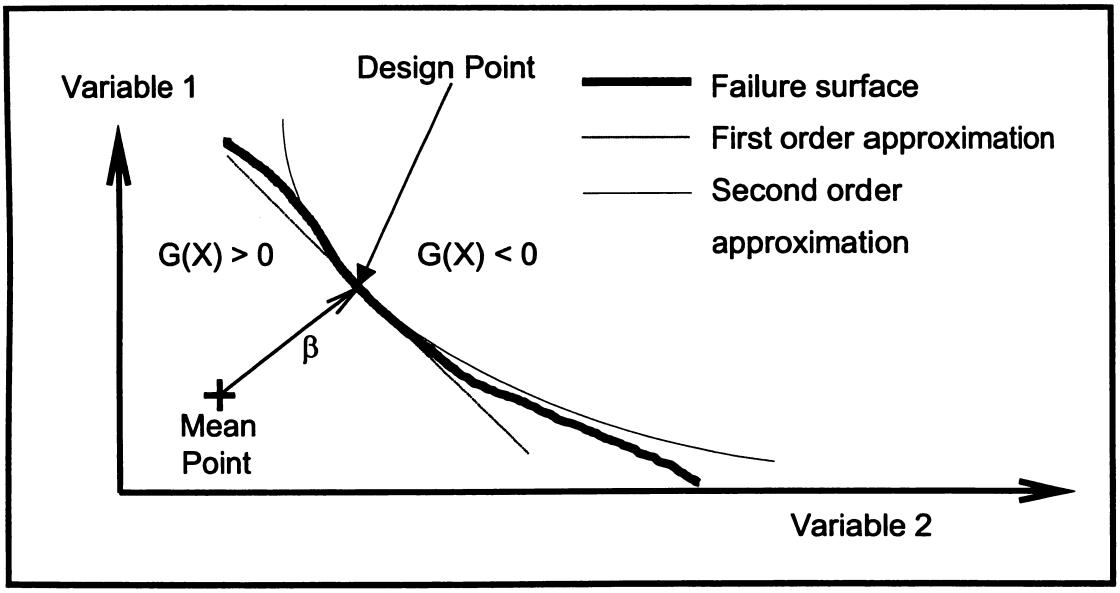


Figure 3.4 Two variable reliability example

Hassofer and Lind (1974) suggest that the random variables be transformed into standard normal distributions before determining beta to give it a consistent definition and hence a consistent value for a given problem.

A first order approximation to the probability of failure can be calculated directly from the reliability with the use of standard normal probability tables. This approximation assumes that the failure surface can be approximated by a plane tangent to the failure surface at the design point.

A second order approximation to the probability of failure can be obtained if the second derivatives of the failure surface at the design point are known. This approximation uses a second order Taylor series to approximate the failure surface. Liu and Der Kiureghan (1991a) use a (structural) finite element package to develop a second order solution. Der Kiureghan and De Stefano (1991) develop an algorithm that finds the second derivatives in order of significance so that only curvatures of a given significance are used. This reduces the calculation time.

The next couple of subsections will discuss how these methods have been applied in the groundwater flow and solute transport areas.

3.3.3 Velocity

Sagar (1978) applied a form of perturbation method to the groundwater flow problem. It outlines a methodology for obtaining the means and variances of the hydraulic head in the groundwater flow problem from the means and covariances of the hydraulic conductivity, storativity and forcing function using the Galerkin finite element method. In this method a matrix equation is derived, then a perturbation is added to a matrix element and the matrix is inverted to enable the derivatives of hydraulic head with respect to the matrix elements to be obtained. These derivatives are used in a Taylor series from which the mean and variance can be determined up to second order. A one-

dimensional example with three elements is provided. The work is not extended to groundwater velocity.

Dettinger and Wilson (1981) develop a perturbation method for obtaining the mean and covariance of the hydraulic head field. However, their method involves matrix inversion and no application is provided.

Hantush and Marino (1995) also develop a form of perturbation method for computing hydraulic head values in unsteady flow with random recharge, storativity and transmissivity. However, the method involves inverting matrices rather than just solving them and so may be computationally intensive. A one dimensional application is provided using up to 40 nodes.

Ghanem (1998) uses Khahunen-Loeve expansions to discretise the random field in the spectral dimension. The hydraulic head at each node is then considered to consist of the superposition of the effects of each of these random variables. This is substituted into the flow equation and the ensemble average is taken. Solution of this equation gives the covariance matrix for the hydraulic heads.

Tartakovsky and Neuman (1998), Guadini and Neuman (1999a & b) and Winter et al (2002) expand the hydraulic head directly into a Taylor series using Green's functions. These are then numerically solved. Tartakovsky and Neuman (1998) deal with transient flow, Guadini and Neuman (1999a & b) deal with steady state flow, and Winter et al (2002) model the effect of random boundaries between aquifer units as well as random spatial variation within them.

The forms of Sagar's (1978), Hantush and Marino's (1995) and Ghanem's (1998) perturbation methods are quite different to that discussed in Section 3.3.2.1. The latter

is derived more directly in terms of the derivatives of the stiffness matrix, thus not requiring any matrix inversions. Vanmarcke and Hachich (1983) and Vanmarcke (1994) use a method similar to that discussed in Section 3.3.2.1 to determine the covariance matrix of the hydraulic head from the covariance matrix of the hydraulic conductivity. They then use Bayesian analysis to determine the reduction in uncertainty that can be obtained by gathering more data. This can be used for making decisions about future testing, for example. Both papers use the local averaging method described later in Section 3.4.2. However, they only determine the hydraulic head and not the groundwater velocity as they are interested in geotechnical aspects of the pore water pressure rather than in solute transport.

Little has been done to apply this method to determination of groundwater velocity, or to perform numerical experiments to obtain results of groundwater velocities from using the method.

3.3.4 Concentration

Literature describing the application of non Monte Carlo numerical methods to solute transport is less common than that applying them to hydraulic head calculation.

Graham and McLaughlin (1989a) develop a set of three coupled matrix equations, the first is similar to the advection dispersion equation and relates the mean concentration to the cross covariance between the velocity and concentration perturbations, the other two relate the concentration covariances to the mean concentrations and aforementioned cross covariances. The paper shows how the coupled equations can be solved using a modified Galerkin finite element method. Graham and McLaughlin (1989b) show how

the results of this method can be conditioned on measured values using Kalman filtering to give improved accuracy.

Hamed et al (1995) demonstrate the use of reliability methods, however, spatial variability of aquifer properties was not considered.

Very recently, Kunstmann et al (2002) have used a first order second moment method to determine the nodal concentration variances from the covariance matrices of the hydraulic heads, hydraulic conductivity and aquifer recharge. In doing so they need to calculate the cross covariance matrices between the hydraulic heads and hydraulic conductivity and between the hydraulic heads and the aquifer recharge. It is to be noted that they are using a first order method and so only focus on covariances. To first order the mean is the same as the deterministic result. It is necessary to go to second order for the mean to be affected by the second moment of the input variables because the input variables are assumed to be normal (Gaussian). Thus their mean values are assumed to be the deterministic result.

3.4 Random Field Discretisation

In order to use a numerical solution scheme it is necessary to discretise the random field into random variables.

The result of the discretisation process is a covariance matrix. This is a square matrix with dimensions equal to the number of random variables. Each element of the matrix represents the covariance between the random variables represented by the row and column of the matrix. Thus the matrix is symmetric and the diagonal elements represent the variances at each point. By finding suitable eigenvectors it is possible to

transform the covariance matrix into a diagonal matrix (Liu et al, 1987). The vector of correlated random variables can be transformed into a vector of uncorrelated random variables using the same transformation matrix. This has three uses. Firstly using the inverse matrix a vector of independent random variables can be transformed into a vector of correlated random variables. This is useful since it is easier to generate realisations of independent random variables than dependant ones. Secondly it can be used to simplify the Taylor series for the output variables. In the transformed matrix all covariances are zero except for diagonal entries. This eliminates the necessity to calculate the cross second derivatives of the hydraulic head and concentration fields with respect to the random variables for the perturbation and reliability methods. This can be a major reduction in calculations.

There are several ways to discretise the random field into random variables. If the integral scale is large compared with the fluctuation of the hydraulic head or concentration field then it will be necessary to have a finer discretisation of these fields than of the random field. This means that one random variable may be used to model several finite elements. This is often used in structural mechanics problems where stress patterns can be very fine. In contaminant transport problems the structure of the random field is normally more detailed than the structure of the hydraulic head and concentration fields. Therefore at least one random variable is required for each finite element.

This allows the options available for discretisation to be midpoint discretisation, local averaging and weighted integrals.

3.4.1 Midpoint Discretisation

The simplest form of random field discretisation is the midpoint method presented in Der Kiureghian and Ke (1988). It simply requires choosing the value of the random field at the centre of the area or volume corresponding to the random variable as the random variable. Thus the covariances between the random variables will simply be the value of the auto correlation function at the separation of the midpoints of the areas or volumes corresponding to the random variables.

This is the method adopted for this research. Reasons for this are presented in Section 3.4.4.

3.4.2 Local Averages

Local averaging requires some form of averaging of the random field over the area or volume corresponding to the random variable as the random variable. Vanmarcke (1983) shows that local averaging gives covariances that are similar to those provided by the midpoint method acting on a random field with a longer integral scale and lower point variance, depending on the element. As the size of the areas or volumes being averaged decrease this difference will reduce until it reaches zero when the areas or volumes are identical with their midpoints.

$$\mathrm{Cov}\left(k_i, k_j\right) = \sigma^2 \iint_{D_i, D_2} \rho\left(d_i - d_j\right) dd_i dd_j \qquad \text{Equation 3.1}$$

D_i, D_j = the area or volume of elements k_i, k_j

Vanmarcke (1983) suggests using a variance reduction function γ to simplify the computations.

In one dimension this is:

$$\mathrm{Var}(k) = \gamma(T)\sigma^2 \qquad \text{Equation 3.2}$$

where

T is the length of the element

σ^2 is the point variance of the random field

This allows the covariance function to be calculated from (one dimension):

$$\mathrm{Cov}(k_i, k_j) = \frac{\sigma^2 \sum_{n=0}^{3} (-1)^n \gamma(T_n) T_n^2}{2T_i T_j \sqrt{\gamma(T_i)\gamma(T_j)}} \qquad \text{Equation 3.3}$$

Similar expressions can be determined for the covariance function for multiple dimension elements.

Analytic expressions of the one dimensional γ function can easily be determined for common auto-correlation functions based on I, the integral scale, a parameter used in the auto-correlation function.

For two dimensional elements:

$$\gamma(T_1, T_2) = \gamma(T_1)\gamma(T_2 \mid T_1)$$
$\gamma(T_2 \mid T_1) = \gamma(T_2)$ but with a different integral scale I_2 Equation 3.4
I_2 is a function of T_1 for most common autocorrelation functions.

This creates problems because I_2 is often different to I. One case where I_2 equals I is when the auto correlation function is separable. An example of this is $\rho(\tau) = \exp(\tau^2)$, which can be separated into a product of functions of the x, y and z components $\rho(\tau) = \exp(x^2) \times \exp(y^2) \times \exp(z^2)$. Most research is done using a linear exponential auto-correlation function $\rho(\tau) = \exp(\tau)$. This function is not separable.

For non-separable auto-correlation functions Vanmarcke (1983) suggests using a piecewise linear approximation for I_2.

Zhu et al (1992) show that local averages give faster convergence than midpoint discretisation. However this is for a structural problem rather than a contaminant transport problem. As contaminant transport requires modelling structures of variability rather than probabilities of variability this may not apply.

3.4.3 Weighted Integrals

Weighted integrals involve breaking each finite element up into a separate random variable for each term in the first derivative of each shape function used in the finite element method (Takada, 1990, Takada and Masanobu, 1990, Deodatis, 1991, Deodatis and Shinozuka, 1991a, b & c).

$$X_{i,j,k} = \iiint_e x^i y^j z^k f(x,y,z)\,dx\,dy\,dz \qquad \text{Equation 3.5}$$
$$f(x,y,z) = \text{random field}$$

For linear shape functions, which have constant first derivatives, i = j = k = 0, this reduces to a local average. For higher order functions more random variables are required.

Weighted integrals have been developed in the structural area and not applied to other areas as yet. Deodatis (1991) develops two forms of the structural stiffness matrix, one using the principal of stationary potential energy, a stiffness approach, and the other using the principle of virtual work, a flexibility approach. The approach that relates to the contaminant transport is the stationary potential energy approach. This method involves separating the shape functions into their separate powers of the space vector before integrating.

When higher dimension elements are used more random variables are required. It is noticed that the derivative of each sub matrix is the next sub matrix in the list (Deodatis, 1991). This is useful for using the perturbation method with the weighted integral method.

Deodatis (1991) points out that this method has inaccuracies because of the use of deterministic instead of stochastic shape functions. He then develops a set of stochastic shape functions. Stochastic shape functions are irrelevant to contaminant transport problems. In mechanics problems involving linear bars shape functions are exact and refer to the deformed shape of the bar. Thus a stochastic shape function can be derived. In field problems, such as the grounder flow and solute transport the shape functions are only approximations so it does not matter the exact shape they have as long as they are compatible at the edges of the element.

The research outlined in this thesis uses linear shape functions. Therefore weighted integrals are effectively equivalent to local averaging and so will not be further considered.

3.4.4 Method Adopted

The results presented in Chapters 5 and 6 use the midpoint method rather than the local averaging method. This is because groundwater seeks preferential flow paths. The average distance between these flow paths is modelled by the integral scale. The higher relative conductivity of these flow paths is modelled by the standard deviation. Vanmarcke (1983) points out that local averaging has the effect of increasing the integral scale and decreasing the standard deviation. The size of this effect depends upon the size of the elements being used. In many applications in structural mechanics the behaviour depends on the average properties of the materials and so this is not a problem. However, it is not the averages that matter when discussing macrodispersion of a solute, but rather the magnitude and frequency of the random fluctuations. Thus it appears to indicate that if local averaging were to be used in this work it would be akin to changing the aquifer properties depending upon the element size chosen. Therefore the midpoint method has been adopted for determining the results presented in Chapters 5 and 6.

However, it is to be noted that the stochastic finite element method presented is general and other methods can be used for determining the covariances.

4 Problem Definition, Algorithms and Implementation Issues

4.1 Problem Definition and Aims

Now that the nature of contaminant transport and a variety of aquifer models have been put forward it is possible to define the problem being solved.

Most of the research covered in Chapter 3 has been analytical in nature and only has solutions for a limited range of conditions, for example an infinite extent of boundary conditions, uniform groundwater flow, instantaneous point source or constant concentration source. Much of the remainder has involved the use of Monte Carlo simulation, which has the attendant problem of requiring very large amounts of computer time to achieve a reasonable degree of accuracy. There is need for a numerical method that can solve this problem for arbitrary boundary conditions, arbitrary groundwater flow regimes and arbitrary source and sink characteristics of solute.

The aim of this research is to develop a numerical method that models the contaminant transport problem using a random field to model the hydraulic conductivity. Thus input data for hydraulic conductivity will involve the mean value, the standard deviation of the point value, and the auto correlation function for each aquifer unit. A perturbation method based on the finite element method will be used. The advection dispersion equation will be used to formulate the method, but the heterogeneity of the field will be

accounted for in the advection term by determining the variation in velocity throughout the aquifer rather than artificially fitting the dispersion coefficient.

The results of applying this method will be compared with various analytical results found in the literature.

4.2 Algorithms for Velocity Calculations

The stochastic finite element method presented here is a numerical method based on the standard Galerkin finite element method. In this way following Equation 2.6 is expressed as:

$$\mathbf{KH} = \mathbf{F} \qquad \text{Equation 4.1}$$

where **K** is the global conductance matrix, **H** is the vector of nodal heads, and **F** is the global specified flow matrix (Istok, 1989).

The particular formulation used here assumes that the hydraulic conductivity is uniform within each finite element. The most important implication of this is that heterogeneity smaller than the element size cannot be modelled. Thus the results may be sensitive to element size because for a given value of the integral scale a smaller element will model more of the heterogeneity than a larger element, particularly when the element size is a substantial fraction of the integral scale. However, given that the midpoint method is being used to discretise the random field this is the most appropriate type of formulation to use as the midpoint method is only capable of modelling the hydraulic conductivity of each element by a single value.

Other formulations are possible, for example, a formulation where the hydraulic conductivity is defined at the nodes of each finite element and calculated at other points using the interpolation functions. To use this formulation it would be necessary to discretise the random field using the weighted integral method presented in Section 3.4.3. In that section it is noted that using linear interpolation functions (as used in this research) the weighted integral method reduces to the local averages method and that using higher order polynomial interpolation functions requires more random variables per finite element, greatly increasing computation time. For these reasons this alternative formulation was not used.

4.2.1 Normal-k Model

The statistics of the hydraulic head field are determined by expanding each nodal head value into a Taylor Series in a similar manner to that used for structural response in Nakagiri (1987):

$$H_n = \overline{H}_n + \sum_{i=1}^{E} \frac{\partial H_n}{\partial k_i} \Delta k_i + \frac{1}{2} \sum_{i,j=1}^{E} \frac{\partial^2 H_n}{\partial k_i \partial k_j} \Delta k_i \Delta k_j + \frac{1}{6} \sum_{i,j,l=1}^{E} \frac{\partial^3 H_n}{\partial k_i \partial k_j \partial k_l} \Delta k_i \Delta k_j \Delta k_l + \frac{1}{24} \sum_{i,j,l,m=1}^{E} \frac{\partial^4 H_n}{\partial k_i \partial k_j \partial k_l \partial k_m} \Delta k_i \Delta k_j \Delta k_l \Delta k_m + \ldots \quad \text{Equation 4.2}$$

where H_n is the head value at node n, E is the total number of elements being considered, k_i is the hydraulic conductivity for element i, and Δk_i is the deviation of k_i from its mean value. The overbar on H_n implies that this is the value determined using the mean hydraulic conductivity field, that is, the deterministic result.

Once again adapting the method used by Nakagiri (1987) and assuming that k is normally distributed (lognormal distributions will be considered later), and deleting

terms involving statistics of k of fourth order or more the mean and covariance are given by:

$$\mathrm{E}[H_n] = \overline{H}_n + \frac{1}{2}\sum_{i,j=1}^{E} \frac{\partial^2 H_n}{\partial k_i \partial k_j} \mathrm{cov}(k_i, k_j) \qquad \text{Equation 4.3}$$

$$\mathrm{cov}(H_m, H_n) = \sum_{i,j=1}^{E} \left(\frac{\partial H_m}{\partial k_i}\right)\left(\frac{\partial H_n}{\partial k_j}\right) \mathrm{cov}(k_i, k_j) \qquad \text{Equation 4.4}$$

where $\mathrm{E}[H_n]$ and $\mathrm{cov}(H_m, H_n)$ are the mean and covariance of the given values over all possible realisations of the hydraulic conductivity field.

The derivative terms in Equation 4.3 and Equation 4.4 can be determined by taking the derivatives of Equation 4.1 with respect to k_i and k_j and rearranging. Note that $\mathbf{F}$ is independent of k. This leads to:

$$\overline{\mathbf{K}} \frac{\partial \mathbf{H}}{\partial k_i} = -\frac{\partial \mathbf{K}}{\partial k_i} \overline{\mathbf{H}} \qquad \text{Equation 4.5}$$

$$\overline{\mathbf{K}} \frac{\partial^2 \mathbf{H}}{\partial k_i \partial k_j} = -\left(\frac{\partial \mathbf{K}}{\partial k_i}\right)\left(\frac{\partial \mathbf{H}}{\partial k_j}\right) - \left(\frac{\partial \mathbf{K}}{\partial k_j}\right)\left(\frac{\partial \mathbf{H}}{\partial k_i}\right) - \frac{\partial^2 \mathbf{K}}{\partial k_i \partial k_j} \overline{\mathbf{H}} \qquad \text{Equation 4.6}$$

Now noting that $\mathbf{K}$ is a linear function of k, the derivatives of $\mathbf{K}$ can be determined as:

$$\frac{\partial \mathbf{K}}{\partial k_i} = \frac{1}{k_i} \mathbf{K}^i \qquad \text{Equation 4.7}$$

$$\frac{\partial^2 \mathbf{K}}{\partial k_i \partial k_j} = 0 \qquad \text{Equation 4.8}$$

where K^i is the expanded elemental conductance matrix for element i.

Substituting Equation 4.7 and Equation 4.8 into Equation 4.5 and Equation 4.6 yields:

$$\overline{\mathbf{K}}\frac{\partial \mathbf{H}}{\partial k_i} = -\frac{1}{k_i}\mathbf{K}^i\overline{\mathbf{H}} \qquad \text{Equation 4.9}$$

$$\overline{\mathbf{K}}\frac{\partial^2 \mathbf{H}}{\partial k_i \partial k_j} = -\frac{1}{k_i}\mathbf{K}^i\frac{\partial \mathbf{H}}{\partial k_j} - \frac{1}{k_j}\mathbf{K}^j\frac{\partial \mathbf{H}}{\partial k_i} \qquad \text{Equation 4.10}$$

To perform the calculations firstly Equation 4.1 is solved using the mean hydraulic conductivity values to obtain the H-bar values. These are substituted into Equation 4.9 to obtain the first order derivatives. These are substituted into Equation 4.10 to obtain the second order derivatives. Both the first and second order derivatives can then be substituted into Equation 4.3 and Equation 4.4 to obtain the mean and covariance of the hydraulic head field.

Most of the computational effort goes into solving Equation 4.5 and Equation 4.6. An important part of this algorithm is that the **K**-bar matrix on the left hand side of Equation 4.5 and Equation 4.6 is the same for all equations. This matrix is factorised at the beginning while solving Equation 4.1 and then used in all forward and backward substitutions. Also, the derivatives of the **K** matrix used on the right hand side are sparse matrices containing only terms relating to the nodes belonging to the elements for which they are derivatives and so not all elements need to be involved in the multiplication. These two facts greatly speed up the computation of the algorithm.

The above development is similar to Sagar (1978) except that in the present work the Taylor series is developed in terms of the material properties rather than matrix coefficients. It is also similar to Vanmarcke (1994) except that in the present work the

mean is determined as well as the covariance. The present work also differs from both of the above in that it determines the velocity statistics as well as the head statistics.

The statistics of the velocity field can be determined in a similar way to Equation 4.3 and Equation 4.4:

$$\mathrm{E}[v_n] = \bar{v}_n + \frac{1}{2}\sum_{i,j=1}^{E} \frac{\partial^2 v_n}{\partial k_i \partial k_j} \mathrm{cov}(k_i, k_j) \qquad \text{Equation 4.11}$$

$$\mathrm{cov}(v_m, v_n) = \sum_{i,j=1}^{E} \left(\frac{\partial v_m}{\partial k_i}\right)\left(\frac{\partial v_n}{\partial k_j}\right) \mathrm{cov}(k_i, k_j) \qquad \text{Equation 4.12}$$

To determine the derivatives in Equation 4.11 and Equation 4.12 the finite element deterministic case form of Darcy's Law is required. This is given by (Istok, 1989):

$$v_e = -k_e \sum_{m=1}^{M} \nabla N_m . H_m \qquad \text{Equation 4.13}$$

where v_e is the velocity at a point in element e, ∇ is the gradient operator, N_m is the value of interpolation function for node m at the point where the velocity is being determined, H_m is the nodal hydraulic head value at node m, and M is the number of nodes in element e. Differentiating Equation 4.13 with respect to k_i and k_j leads to:

$$\frac{\partial v_e}{\partial k_i} = -\delta_{e,i} \sum_{m=1}^{M} \nabla N_m . H_m - k_e \sum_{m=1}^{M} \nabla N_m . \frac{\partial H_m}{\partial k_i} \qquad \text{Equation 4.14}$$

$$\frac{\partial^2 v_e}{\partial k_i \partial k_j} = -\delta_{e,i} \sum_{m=1}^{M} \nabla N_m . \frac{\partial H_m}{\partial k_j} - \delta_{e,j} \sum_{m=1}^{M} \nabla N_m . \frac{\partial H_m}{\partial k_i} - k_e \sum_{m=1}^{M} \nabla N_m . \frac{\partial^2 H_m}{\partial k_i \partial k_j} \qquad \text{Equation 4.15}$$

where $\delta_{e,i}$ is the Dirac delta function.

4.2.2 Lognormal-k Model

It is generally accepted that hydraulic conductivity is lognormally distributed (Freeze, 1975), that is $Y = \ln(k)$ where Y is normally distributed. The method outlined above can also be used with lognormally distributed hydraulic conductivity. In this case Equation 4.3 and Equation 4.4 are replaced by:

$$\mathrm{E}[H_n] = \overline{H}_n + \frac{1}{2}\sum_{i,j=1}^{E} \frac{\partial^2 H_n}{\partial Y_i \partial Y_j} \mathrm{cov}(Y_i, Y_j) \qquad \text{Equation 4.16}$$

$$\mathrm{cov}(H_m, H_n) = \sum_{i,j=1}^{E} \left(\frac{\partial H_m}{\partial Y_i}\right)\left(\frac{\partial H_n}{\partial Y_j}\right) \mathrm{cov}(Y_i, Y_j) \qquad \text{Equation 4.17}$$

This time we will need to know the derivatives of **K** with respect to Y. Now:

$$k_e = \mathrm{e}^{Y_e} \qquad \text{Equation 4.18}$$

therefore:

$$\frac{\partial k_e}{\partial Y_i} = \delta_{e,i}\, \mathrm{e}^{Y_e} \qquad \text{Equation 4.19}$$

$$\frac{\partial^2 k_e}{\partial Y_i \partial Y_j} = \delta_{e,i,j}\, \mathrm{e}^{Y_e} \qquad \text{Equation 4.20}$$

Once again, noting that **K** is a linear function of Y, the derivatives of **K** with respect to Y can be determined as:

$$\frac{\partial \mathbf{K}}{\partial Y_i} = \mathbf{K}^i \qquad \text{Equation 4.21}$$

$$\frac{\partial^2 \mathbf{K}}{\partial Y_i \partial Y_j} = \delta_{i,j}\mathbf{K}^i \qquad \text{Equation 4.22}$$

Substituting these into equations similar to Equation 4.5 and Equation 4.6 but using Y instead of k gives:

$$\overline{\mathbf{K}}\frac{\partial \mathbf{H}}{\partial Y_i} = -\mathbf{K}^i\overline{\mathbf{H}} \qquad \text{Equation 4.23}$$

$$\overline{\mathbf{K}}\frac{\partial^2 \mathbf{H}}{\partial Y_i \partial Y_j} = -\mathbf{K}^i\frac{\partial \mathbf{H}}{\partial Y_j} - \mathbf{K}^j\frac{\partial \mathbf{H}}{\partial Y_i} - \delta_{i,j}\mathbf{K}^i\overline{\mathbf{H}} \qquad \text{Equation 4.24}$$

These can be used with Equation 4.16 and Equation 4.17 to obtain the statistics of H in the same manner as in the normal-k model above. Similarly the derivatives of velocity with respect to Y are given by:

$$\frac{\partial v_e}{\partial Y_i} = -\delta_{e,i}k_e\sum_{m=1}^{M}\nabla N_m.H_m - k_e\sum_{m=1}^{M}\nabla N_m.\frac{\partial H_m}{\partial Y_i} \qquad \text{Equation 4.25}$$

$$\begin{aligned}\frac{\partial^2 v_e}{\partial Y_i \partial Y_j} = &-\delta_{e,i,j}k_e\sum_{m=1}^{M}\nabla N_m.H_m - \delta_{e,i}k_e\sum_{m=1}^{M}\nabla N_m.\frac{\partial H_m}{\partial Y_j} \\ &-\delta_{e,j}k_e\sum_{m=1}^{M}\nabla N_m.\frac{\partial H_m}{\partial Y_i} - k_e\sum_{m=1}^{M}\nabla N_m.\frac{\partial^2 H_m}{\partial Y_i \partial Y_j}\end{aligned} \qquad \text{Equation 4.26}$$

These can be substituted into:

$$\mathrm{E}\left[v_n\right] = \bar{v}_n + \frac{1}{2}\sum_{i,j=1}^{E}\frac{\partial^2 v_n}{\partial Y_i \partial Y_j}\mathrm{cov}\left(Y_i, Y_j\right) \qquad \text{Equation 4.27}$$

$$\operatorname{cov}(v_m, v_n) = \sum_{i,j=1}^{E} \left(\frac{\partial v_m}{\partial Y_i}\right)\left(\frac{\partial v_n}{\partial Y_j}\right) \operatorname{cov}(Y_i, Y_j) \qquad \text{Equation 4.28}$$

to give the statistics of the velocity distribution.

It should be noted that the value of k_e to be used in Equation 4.9, Equation 4.10, Equation 4.13, Equation 4.14 and Equation 4.15 is the mean hydraulic conductivity, whereas in Equation 4.23, Equation 4.24, Equation 4.25 and Equation 4.26 it corresponds to the mean of Y, which is the geometric mean of k.

4.3 Algorithms for Concentration Calculations

4.3.1 Means and Covariances

The perturbation method presented here is a numerical method based on the standard Galerkin finite element method. In this way Equation 2.4 is expressed as:

$$\mathbf{A}\frac{\partial \mathbf{C}}{\partial t} + \mathbf{D}\mathbf{C} = \mathbf{F} \qquad \text{Equation 4.29}$$

where **A** is the global sorption matrix, **C** is the vector of nodal concentrations, t is time, **D** is the global advection-dispersion matrix, and **F** is the global specified flow matrix (Istok, 1989).

To solve this a finite difference formulation is used:

$$[\mathbf{A} + \omega\Delta t\mathbf{D}]\mathbf{C}_{t+\Delta t} = [\mathbf{A} - (1-\omega)\Delta t\mathbf{D}]\mathbf{C}_t + \Delta t[(1-\omega)\mathbf{F}_t + \omega\mathbf{F}_{t+\Delta t}] \qquad \text{Equation 4.30}$$

The statistics of the concentration field are determined by expanding each nodal head value into a Taylor Series:

$$C_n = \bar{C}_n + \sum_{i=1}^{E} \frac{\partial C_n}{\partial Y_i} \Delta Y_i + \frac{1}{2} \sum_{i,j=1}^{E} \frac{\partial^2 C_n}{\partial Y_i \partial Y_j} \Delta Y_i \Delta Y_j + \frac{1}{6} \sum_{i,j,l=1}^{E} \frac{\partial^3 C_n}{\partial Y_i \partial Y_j \partial Y_l} \Delta Y_i \Delta Y_j \Delta Y_l + \frac{1}{24} \sum_{i,j,l,m=1}^{E} \frac{\partial^4 C_n}{\partial Y_i \partial Y_j \partial Y_l \partial Y_m} \Delta Y_i \Delta Y_j \Delta Y_l \Delta Y_m + \ldots \quad \text{Equation 4.31}$$

where C_n is the head value at node n, E is the total number of elements being considered, Y_i is the logarithm of the hydraulic conductivity for element i, and ΔY_i is the deviation of Y_i from its mean value. The overbar on C_n implies that this is the value determined using the mean of the log hydraulic conductivity field, that is, the deterministic result.

Y was chosen as the basis for the Taylor series because it is generally accepted to be normally distributed (Freeze, 1975). Also Section 5.1.1 shows that the normal-k model does not give sensible results. Given this and deleting terms involving statistics of k of fourth order or more the mean and covariance are given by:

$$\mathrm{E}[C_n] = \bar{C}_n + \frac{1}{2} \sum_{i,j=1}^{E} \frac{\partial^2 C_n}{\partial Y_i \partial Y_j} \mathrm{cov}(Y_i, Y_j) \quad \text{Equation 4.32}$$

$$\mathrm{cov}(C_m, C_n) = \sum_{i,j=1}^{E} \left(\frac{\partial C_m}{\partial Y_i} \right) \left(\frac{\partial C_n}{\partial Y_j} \right) \mathrm{cov}(Y_i, Y_j) \quad \text{Equation 4.33}$$

where $\mathrm{E}[C_n]$ and $\mathrm{cov}(C_m,C_n)$ are the mean and covariance of the given values over all possible realisations of the Y field.

The derivatives in Equation 4.32 and Equation 4.33 can be expanded to produce:

$$\mathrm{E}[C_n] = \overline{C}_n + \frac{1}{2}\sum_{i,j,p,q=1}^{E} \frac{\partial^2 C_n}{\partial v_p \partial v_q}\frac{\partial v_p}{\partial Y_i}\frac{\partial v_q}{\partial Y_j}\mathrm{cov}(Y_i, Y_j) + \frac{1}{2}\sum_{i,j,p=1}^{E} \frac{\partial C_n}{\partial v_p}\frac{\partial^2 v_p}{\partial Y_i \partial Y_j}\mathrm{cov}(Y_i, Y_j) \quad \text{Equation 4.34}$$

$$\mathrm{cov}(C_m, C_n) = \sum_{i,j,p,q=1}^{E} \left(\frac{\partial C_m}{\partial v_p}\right)\left(\frac{\partial C_n}{\partial v_q}\right)\left(\frac{\partial v_p}{\partial Y_j}\right)\left(\frac{\partial v_q}{\partial Y_j}\right)\mathrm{cov}(Y_i, Y_j) \quad \text{Equation 4.35}$$

Comparison of these with Equation 4.27 and Equation 4.28 shows that they can be simplified to:

$$\mathrm{E}[C_n] = \overline{C}_n + \frac{1}{2}\sum_{p,q=1}^{E} \frac{\partial^2 C_n}{\partial v_p \partial v_q}\mathrm{cov}(v_p, v_q) + \sum_{p=1}^{E} \frac{\partial C_n}{\partial v_p}\left(\mathrm{E}[v_p] - \overline{v}_p\right) \quad \text{Equation 4.36}$$

$$\mathrm{cov}(C_m, C_n) = \sum_{p,q=1}^{E} \left(\frac{\partial C_m}{\partial v_p}\right)\left(\frac{\partial C_n}{\partial v_q}\right)\mathrm{cov}(v_p, v_q) \quad \text{Equation 4.37}$$

where v_p and v_q are the pore velocities of elements p and q.

The derivative terms in Equation 4.36 and Equation 4.37 can be determined by taking the derivatives of Equation 4.30 with respect to v_i and v_j and rearranging. Note that **A** and **F** are independent of v and hence their derivatives are zero. This leads to:

$$[\mathbf{A} + \omega\Delta t\mathbf{D}]\frac{\partial \mathbf{C}_{t+\Delta t}}{\partial v_p} = [\mathbf{A} - (1-\omega)\Delta t\mathbf{D}]\frac{\partial \mathbf{C}_t}{\partial v_p} - \omega\Delta t\frac{\partial \mathbf{D}}{\partial v_p}\mathbf{C}_{t+\Delta t} - (1-\omega)\Delta t\frac{\partial \mathbf{D}}{\partial v_p}\mathbf{C}_t \quad \text{Equation 4.38}$$

$$\begin{aligned}[\mathbf{A} + \omega\Delta t\mathbf{D}]\frac{\partial^2 \mathbf{C}_{t+\Delta t}}{\partial v_p \partial v_q} = {} & [\mathbf{A} - (1-\omega)\Delta t\mathbf{D}]\frac{\partial^2 \mathbf{C}_t}{\partial v_p \partial v_q} - \omega\Delta t\frac{\partial \mathbf{D}}{\partial v_q}\frac{\partial \mathbf{C}_{t+\Delta t}}{\partial v_p} \\ & - (1-\omega)\Delta t\frac{\partial \mathbf{D}}{\partial v_q}\frac{\partial \mathbf{C}_t}{\partial v_p} - \omega\Delta t\frac{\partial^2 \mathbf{D}}{\partial v_p \partial v_q}\mathbf{C}_{t+\Delta t} - \omega\Delta t\frac{\partial \mathbf{D}}{\partial v_p}\frac{\partial \mathbf{C}_{t+\Delta t}}{\partial v_q} \\ & - (1-\omega)\Delta t\frac{\partial^2 \mathbf{D}}{\partial v_p \partial v_q}\mathbf{C}_t - (1-\omega)\Delta t\frac{\partial \mathbf{D}}{\partial v_p}\frac{\partial \mathbf{C}}{\partial v_q}\end{aligned} \quad \text{Equation 4.39}$$

In Equation 4.38 and Equation 4.39 only the original (deterministic) [**A** + $\omega\Delta t$**D**] matrix appears on the left hand side. This matrix is factorised once, and then used in all forward and backward substitutions. The derivatives of this matrix are only used in multiplication on the right hand side. This is further simplified because they are sparse matrices containing only terms relating to the nodes belonging to the elements for which they are derivatives.

The elements of **D** for element e are given by:

$$D_{i,j} = \sum_{n=1}^{M} \int_V \left(\frac{\partial N_n}{\partial x_i} D_{ij} \frac{\partial N_n}{\partial x_j} + N_n v_e \frac{\partial N_n}{\partial x_j} + N_n \lambda (\rho K_d + \theta) N_n \right) \qquad \text{Equation 4.40}$$

where N_n is the shape function for node n, M is the number of nodes, V is the domain of the problem, D_{ij} is the dispersion coefficient, v_e is the velocity for element e, λ is the solute decay constant, ρ is the bulk density of the aquifer, K_d is the partitioning coefficient of the solute, and θ is volumetric water content of the aquifer (for saturated soil this is the porosity). The results are based on a conservative contaminant and hence λ, the solute decay constant, is set to 1.0. However, the analysis still applies for non-conservative contaminants. Also the results use a value of zero for the dispersion coefficients in each direction. This implies that the effects of diffusion and mechanical dispersion are neglected and the dispersive behaviour in the results is entirely attributable to the variation in hydraulic conductivity. This is being done to highlight the effects of the variation in hydraulic conductivity. The implications of this decision are discussed in Section 7.5.

The derivatives of **D** are determined simply since **D** is a linear function of v:

$$\frac{\partial D_{i,j}}{\partial v_p} = \sum_{n=1}^{M} \int_V \left(\frac{\partial N_n}{\partial x_i} \frac{\partial D_{ij}}{\partial v_e} \frac{\partial N_n}{\partial x_j} + N_n \delta_{e,p} \frac{\partial N_n}{\partial x_j} \right) \qquad \text{Equation 4.41}$$

$$\frac{\partial^2 D_{i,j}}{\partial v_p \partial v_q} = \sum_{n=1}^{M} \int_V \left(\frac{\partial N_n}{\partial x_i} \frac{\partial^2 D_{ij}}{\partial v_p \partial v_q} \frac{\partial N_n}{\partial x_j} \right) \qquad \text{Equation 4.42}$$

where $\delta_{e,i}$ is the Dirac delta function.

These derivatives can be substituted into Equation 4.38 and Equation 4.39 to determine the derivatives of concentration. The concentration derivatives in turn can be substituted into Equation 4.36 and Equation 4.37 to determine the means and covariances of the nodes in the concentration field.

4.3.2 Spatial Moments

Describing a solute plume by the values of the concentration at each point can be quite cumbersome. Therefore it is quite common to describe the plume by its second spatial moments of area. This gives an idea of how big the solute plume is.

The spatial moments are not the boundaries of the plume. Indeed, according to the underlying mathematical model used, at least a very small concentration of the solute spreads over the entire aquifer instantaneously. This is true for both the deterministic and stochastic models. Instead the moments represent the spread of the plume in the way that the standard deviation represents the "spread" in a Gaussian probability density function.

For a deterministic plume the second spatial moment would be calculated using the following equation:

$$M_{ij}(t) = \frac{\int (x_i - X_i)(x_j - X_j) C(x,t) dx}{\int C(x,t) dx} \qquad \text{Equation 4.43}$$

Where $M_{ij}(t)$ is the second spatial moment of the plume in the i and j directions, and X is the centroid of the plume. Note both integrations are over the whole volume of the plume.

In a stochastic problem there are multiple types of second spatial moments.

4.3.2.1 Spatial Second Moments of Mean Concentration

The simplest spatial second moment to calculate is the spatial second moment of the mean values determined in Section 4.3.1. This represents the ensemble second moment of the plumes in all possible realisations, or, in other words, the area that is endangered of becoming contaminated. It is simply calculated using Equation 4.43 using the mean values of concentration.

However, the spatial second moment of the mean concentration does not represent the size of an individual plume, because it also includes the uncertainty about the location of the plume. Commonly an assumption is made that the result of the ensemble is equivalent to the result from a single realisation. This is referred to as the ergodic hypothesis and a situation where this is true as an ergodic result. Essentially the ergodic hypothesis is acceptable when the spread of an individual plume about its centroid is much greater than the uncertainty in the location of the centroid. This is considered to

be the case after the plume has travelled a sufficient distance such that the spread of the plume is significantly larger than the integral scale of the hydraulic conductivity.

4.3.2.2 Spatial Second Moments of Individual Plumes

The next spatial second moment is the average of the spatial second moment of individual plumes. This represents the average amount of spreading of a plume about its centroid as it travels through the aquifer.

Conceptually this is equivalent to performing a Monte-Carlo simulation, determining the spatial second moment for the plume in each realisation using Equation 4.43, and then determining the average of the results. This is the most commonly desired moment as it most accurately represents the spread of a plume in the field.

Rajaram and Gelhar(1993b) show that the average of the spatial second moment can be calculated using:

$$\Sigma_{ij}(t) = \frac{\int (x_{1i} - x_{2i})(x_{1j} - x_{2j}) \mathrm{cov}[C(x_1,t), C(x_1,t)] dx_1 dx_2}{2\left[\int C(x,t) dx\right]^2} \qquad \text{Equation 4.44}$$

4.3.2.3 Uncertainty in Plume Centroid Position

The heterogeneity of the aquifer not only disperses the plume as it travels through the aquifer, it also causes the plume as a whole to deviate from the deterministic position. The size of this effect depends upon the ratio between the size of the plume and the integral scale of the hydraulic conductivity of the aquifer. If the plume is much larger than the integral scale then the effect of the heterogeneity on different parts of the plume will tend to cancel out and the centroid of the plume will tend to follow the

deterministic position. On the other hand if the plume is smaller than the integral scale the effects of the heterogeneity will be highly correlated and the whole of the plume will meander through the aquifer. This is represented as an increase in the uncertainty of the location of the centroid of the plume.

The uncertainty of the location of the centroid can be represented by the second moment of its spatial probability distribution. This can be easily calculated using the following equation:

$$R_{ij}(t) = M_{ij}(t) - \Sigma_{ij}(t) \qquad \text{Equation 4.45}$$

Where $R_{ij}(t)$ is the second moment of the spatial probability distribution of the location of the centroid.

4.4 Original Deterministic Flow Program

The code for the original deterministic flow program (gw1) was obtained from Istok (1989) and is written in FORTRAN.

This code was chosen because it is a simple code and comes with a textbook clearly explaining each part, thus making it easy to develop into a stochastic program.

The code reads nodal point data, element data, material properties, and boundary condition data from an input file, assembles the global conductance matrix from the element conductance matrices, modifies the global conductance matrix for Dirichlet and Neuman boundary conditions, and solves this matrix using the Choleski method for symmetric matrices. The code also calculates the velocities of the groundwater in each element and writes all of the results to an output file.

A few modifications have been made to the code:

1. Only bar, triangle, rectangle, quadrilateral and parallelepiped elements using linear interpolation functions are included.

2. The input reading routines have been changed to accept the flow field dimensions, number of elements in each direction and head values at each end, and to use this data to create the nodal point data, element data and boundary condition data for a rectangular mesh, using quadrilateral elements.

3. All variables are declared explicitly and "implicit none" statements have been added to each subroutine to ensure this.

4. All real variables have been changed to double precision variables.

5. The size of some of the common block arrays have been reduced to compensate for the extra memory required for double precision variables.

6. The variable "E" used as an index for elements has been changed from a global variable in "COMALL" to a local variable in the subroutines that it is used in. This has removed some irritating warning messages.

7. The formatting of the data written to the output files has been changed.

8. The calculation of the inverse Jacobian matrix for three dimensional elements was found to produce the transpose of the correct answer. This has been corrected.

9. The calculation of the velocity in the specialised rectangular element code was found to be incorrect except for special cases. Therefore rectangular elements use the subroutine for quadrilateral elements, which gives correct results.

10. The code was modified to include upstream weighting (see next section).

11. The matrix solving subroutines were replaced with subroutines from the linear algebra package library (LAPACK). LAPACK is a Fortran library designed to run efficiently on shared-memory vector and parallel processors (Anderson et al, 1999).

The code was originally compiled using Microsoft FORTRAN version 5.1 and various test data files were executed on a 486DX2-66 personal computer using Windows 95 operating system.

The output files, which include the input data, are given in Appendix B.

The code was then modified to calculate the derivatives and mean and standard deviations for head, velocity and concentration. The modified code can be found in Appendix A. The results are discussed in Chapters 5 and 6.

It quickly and repeatedly became apparent that the computations were outgrowing the hardware, thus the program has been ported successively to:

- A Dec Alpha Workstation ("Aerin")
- A 12 PA8000 processor Convex ("Jacobi")
- A 20 processor Silicon Graphics Power Challenge ("Napier")

(See Section 4.6)

4.5 Issues with Respect to Upstream Weighting

The finite element method for solute transport has a tendency to create oscillations in the flow field, particularly when the Peclet number is high. The Peclet number is given by

$$Pe = \frac{v_e h}{D} \qquad \text{Equation 4.46}$$

Where Pe is the Peclet number, v_e is the element velocity, h is the grid spacing, and D is the dispersion coefficient.

Figure 4.1 shows the effect of this on the contour plot of a plume in an 80 by 40 mesh. The contours should be circular, however, the effect has distorted them. Also upstream (to the left) of the plume the concentration oscillates from positive to negative (the areas with negative concentration are hatched). Of course there is no physical meaning to negative concentrations, these are purely an artefact of the finite element method.

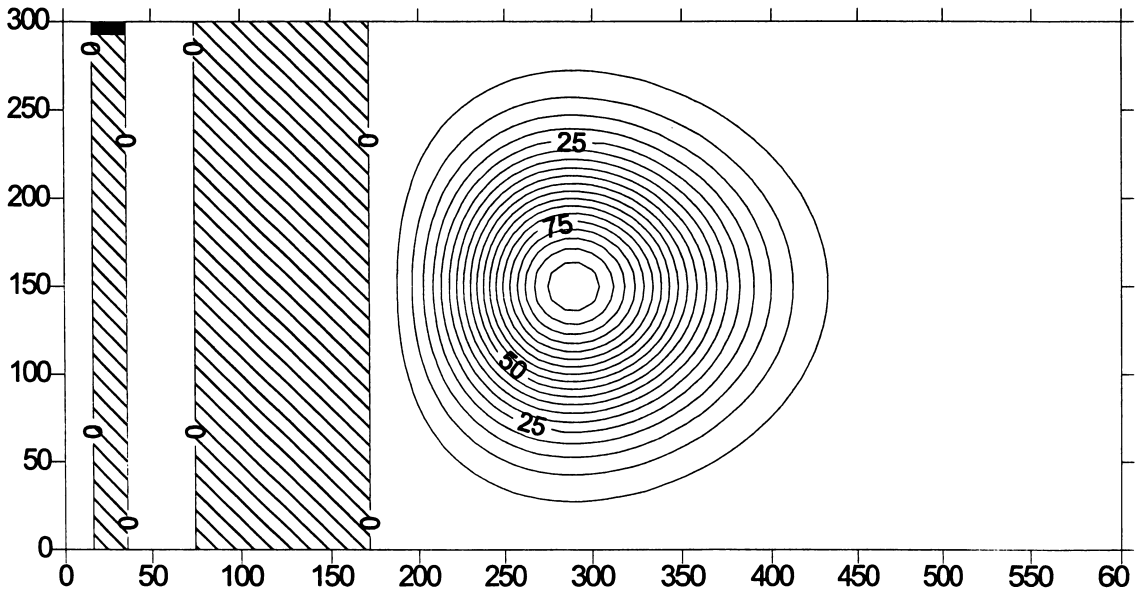


Figure 4.1 Deterministic result for a grid 80 elements wide and 40 elements high.

The average groundwater velocity is from left to right. Hatched areas have negative concentration.

There are two solutions to the problem of these oscillations, either the grid spacing can be reduced, thus reducing the Peclet number or a technique referred to as upstream weighting may be used.

The ability to reduce the grid spacing is limited in this work due to the massive increase in computation time required. Therefore the code was modified to include upstream weighting as an attempt to solve this problem.

Upstream weighting is a technique that changes the shape functions of the finite elements so that the "upstream" part of the element has higher weighting than the "downstream" part. In effect, this reduces the oscillation by creating numerical dispersion.

The standard finite element method used in this work uses linear shape functions. Shape functions based on higher order polynomials can also be used. These give a higher degree of accuracy without requiring more elements (it does introduce more nodes for each element, and hence the computation requirement will increase). However, the increase in accuracy only applies to the model of the concentration field. The accuracy of the model of the random field is strictly a function of the number of elements. Since the point of this work is to model the hydraulic conductivity as a random field it was decided to use elements with linear shape functions for this work as this gives the minimum amount of computational effort per element.

The linear shape functions for the two nodes in a one dimensional element are given by:

$$W_1(x) = 1 - \xi \qquad \text{Equation 4.47}$$

$$W_2(x) = \xi \qquad \text{Equation 4.48}$$

Where:

$$\xi = \left(\frac{x - x_1}{x_2 - x_1} \right)$$

For rectangular elements aligned with the coordinate axes the shape function for each node is given by the product of the one dimensional shape functions in each of the two dimensions direction.

Upstream weighting modifies these shape functions to be the following (Huyakorn and Pinder, 1983)

$$W_1(x) = 1 - \xi + 3\alpha\left(\xi^2 - \xi\right) \qquad \text{Equation 4.49}$$

$$W_2(x) = \xi - 3\alpha\left(\xi^2 - \xi\right) \qquad \text{Equation 4.50}$$

Where x_1 is upstream from x_2 and α is the upstream parameter.

Note that when the upstream parameter is equal to zero the shape functions are simply the linear shape functions given in Equation 4.47 and Equation 4.48.

Once again for rectangular elements aligned with the coordinate axes the shape function for each node is given by the product of the one dimensional shape functions in each of the two dimensions direction, a different value of the upstream parameter will apply in the different directions. For example if the flow is parallel to the x-axis then the upstream parameter in the y direction will be zero. This is the case in this work.

Huyakorn (1976) gives a formula for the optimum amount of upstream weighting to be applied:

$$\alpha_{opt} = \coth\left(\frac{v_e h}{2D}\right) - \frac{2D}{v_e h} \qquad \text{Equation 4.51}$$

Where v_e is the element velocity, h is the grid spacing, and D is the dispersion coefficient.

Given that in the examples used and discussed in Chapter **6** the value of the dispersion coefficient was set to be zero, then the appropriate value for the upstream weighting is equal to one. Figure 4.2 shows the results of applying this amount of upstream weighting to the same plume as shown in Figure 4.1.

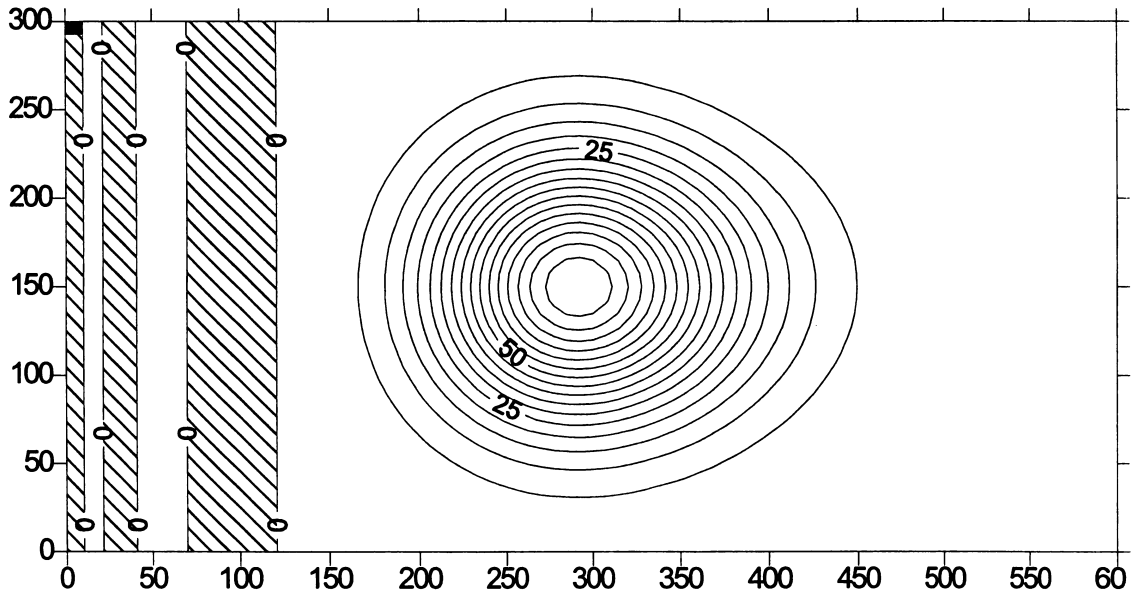


Figure 4.2 Deterministic result for a grid 80 elements wide and 40 elements high including upstream weighting with an upstream weighting parameter of unity. The average groundwater velocity is from left to right. Hatched areas have negative concentration.

Figure 4.2 shows a plume that has been significantly dispersed by the upstream weighting method. In fact the longitudinal second spatial moment has increased from

2500 to 3600, whereas it was relatively unchanged in the case shown in Figure 4.1. Given that the thesis is about measuring the dispersive effect of modelling hydraulic conductivity as a random field, this inaccuracy is a bigger problem than the problem of the oscillations. Therefore the upstream weighting parameter was set to zero in the data sets discussed in Chapter 6. This effectively gives the same answers as if no upstream weighting was applied. However, the functionality to include upstream weighting was not removed from the code, instead the upstream weighting parameter is set to zero in all of the data files used in this research.

4.6 Reducing Computation Time

The stochastic finite element method discussed here is computationally intensive. For example, the meshes used in the examples in Chapters 5 and 6 take only seconds to run for the deterministic case, however they took months to run for the stochastic case (even after the following techniques had been applied). Therefore several techniques have been applied to speed the calculations up.

4.6.1 Order of Computation for Concentrations

Using the straight forward application of the stochastic finite element method to determination of nodal concentrations given in Equation 4.34 and Equation 4.35 implies that the computational time will be approximately proportional to the sixth power of the number of elements in the finite element mesh. By breaking this into two steps, the first one determining the velocity statistics and the second one using Equation 4.36 and Equation 4.37 to determine the concentration statistics this relationship is reduced to the fourth power, a significant improvement for large meshes.

4.6.2 Recalculating First Derivatives

Equation 4.37 contains a large number of first derivatives that are reused many times. They are also used in determining the second derivatives of Equation 4.36. It would appear that saving these numbers would improve the speed. However, for the largest mesh used in this research (approximately 20,000 elements for the groundwater flow problem) the array required to hold all the data would be approximately 6.4GB (6400MB) in size. Such a large array would need constant page swapping to and from the hard disk (the super computer Napier has 2GB of random access memory, which is shared by several users). Therefore the data would need to be stored on disk anyway. It was found that calculating these first derivatives each time they were needed and overwriting with the next was faster than saving them to disk and reading them when required.

Therefore the code as currently implemented calculates a new set of first derivatives for each set of second derivatives.

An interesting side issue about the writing to disk problem was related to the requirement for checkpoint files. Given that the program requires months to execute it is almost certain that the super computer will crash, or for some other reason be restarted before the program finishes executing. Therefore the code was written so that periodically all of its data was written to file so that if the computer crashed the program could start again where it left off. In fact the management of the super computers used was such that this was a necessary prerequisite to using them. Originally the data was written to file one number at a time. Even though binary format was used for this it was found that it took 20 minutes to write all of this data, and another 20 minutes to read it.

By rearranging the data so that it could all be written out as a single array the time spent reading and writing the data to and from disk was reduced to less than one minute.

4.6.3 Use of Symmetry

There are two forms of symmetry that are applicable to this problem. Whilst this may seem to be in conflict with random field modelling because the plume is expected to move asymmetrically, the means and variances will be symmetrical when the initial and boundary conditions etc are symmetrical. The first is the symmetry of the equations and the second is geometrical symmetry.

The second derivatives in Equation 4.16, Equation 4.27 and Equation 4.36 all have symmetry arising from the identity:

$$\frac{\partial^2 z}{\partial x \partial y} = \frac{\partial^2 z}{\partial y \partial x} \qquad \text{Equation 4.52}$$

Similarly the commutative law for multiplication provides symmetry in Equation 4.17, Equation 4.28 and Equation 4.37. Utilising these forms of symmetry cuts the computations required in half.

The specific data sets used in this research all involve geometric symmetry. In a deterministic finite element problem it is possible to get adequate results under these circumstances by halving the flow field and providing appropriate boundary conditions along the introduced boundary. This technique cannot be used in a stochastic finite element problem because the results for each individual derivative are not symmetrical. However, almost half of the derivatives are mirror images of the other half (the discrepancy is that some are their own mirror image). This symmetry exists across the

central flowline for both velocity and concentration calculations, and across the central equipotential line for the velocity calculations.

Utilisation of all of these sets of symmetry can reduce the head and velocity calculations by a factor of eight and the concentration calculations by a factor of four. This has been implemented in the code presented in Appendix A and used in the research.

4.6.4 Dropping Small Covariance Terms

Examination of Equation 4.16, Equation 4.17, Equation 4.27,Equation 4.28, Equation 4.36 and Equation 4.37 (mean and variance equations for head, velocity and concentration) shows that each derivative term is weighted by a covariance. By neglecting those with small covariances less calculations are required. Since the covariance is a decreasing function of the separation distance between the two relevant elements, ignoring pairs of elements with large separation vectors could speed up calculations without loss of too much accuracy.

Figure 4.3 shows the effect on the second derivative term in Equation 4.27 (i.e. the velocity calculations) of the distance between the elements that the derivative is taken with respect to. As the elements become further apart the absolute magnitude of the second derivative decreases, both along the flow field and across it (note this particular effect is unrelated to the autocorrelation function). Thus it would appear that ignoring pairs of elements with large separation vectors might be useful even for highly correlated fields (where all covariances are reasonably large). However, although the derivatives with respect to two widely separated elements are a couple of orders of magnitude lower than those with respect to adjacent elements, there are so many more widely separated pairs of elements that the errors were found to be quite significant.

In general it appears that including all pairs of elements closer together than four integral scales results in the error being no more than one or two percent. It must be remembered that at the relevant small integral scales the accuracy of the result depends heavily on the element size.

This technique of reducing calculation time was implemented but not used because several random fields were considered simultaneously, as discussed in the next section. Some of the random fields had integral scales larger than the size of the flow field and for these cases there were no insignificant covariances.

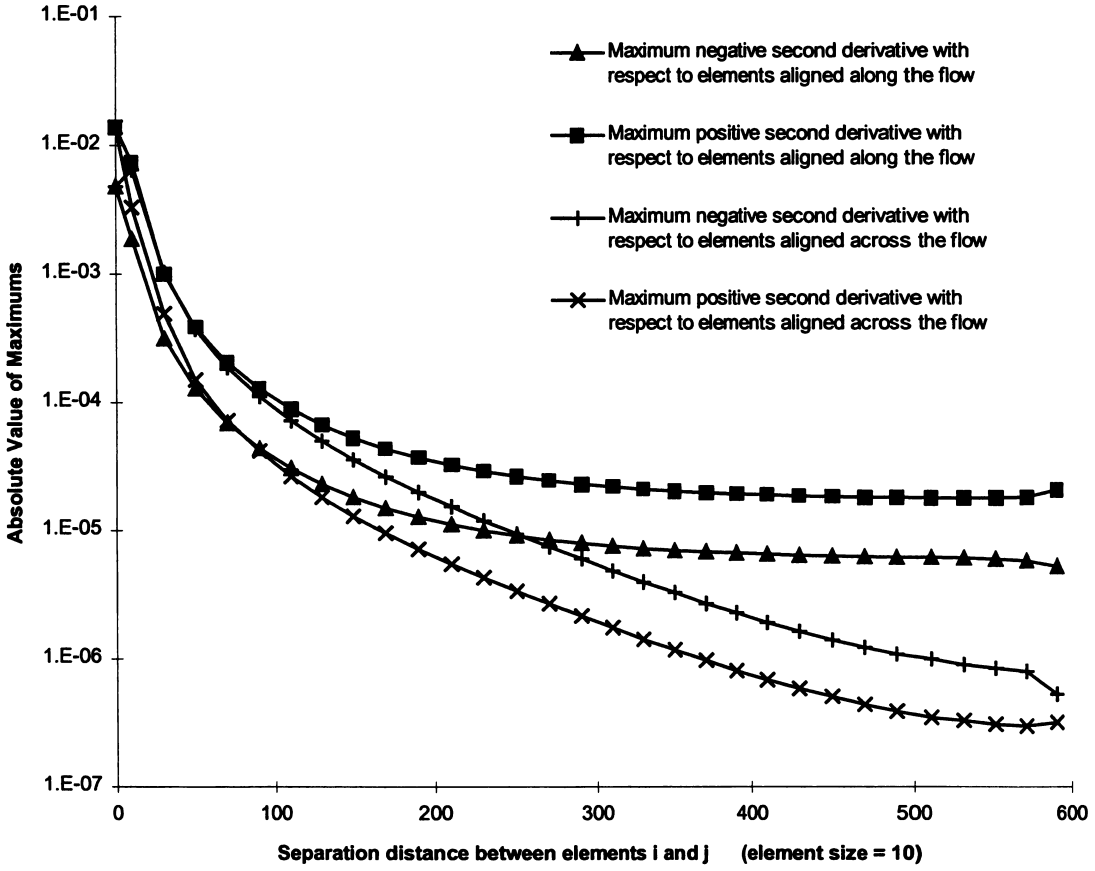


Figure 4.3 Relation between the maximum positive and maximum negative second derivative and the distance between the relevant elements, for element pairs aligned both along and across the mean hydraulic gradient.

Note that for pairs aligned along the mean hydraulic gradient the positive maximum dominates whereas for elements aligned across the flow the negative maximum dominates. Data from 60 by 60 element isotropic square mesh.

4.6.5 Computing Several Correlation Fields Simultaneously

Examination of Equation 4.16, Equation 4.17, Equation 4.27, Equation 4.28, Equation 4.36 and Equation 4.37 (mean and variance equations for head, velocity and concentration) shows that the central calculations involve multiplying one or more derivatives by a covariance. Most of the computational effort goes into the derivatives and these derivatives do not change if the random field is changed. They change only if the geometry, boundary conditions or mesh for the flow field is changed. Unfortunately there are too many derivatives to save to disk for reusing later. However it is quite possible to use each one for several random fields simultaneously. Using ten random fields at once reduces the calculation time for the set of ten by approximately 90%.

A side effect of this technique is that more memory is required to store the data for the total number of sets, thus limiting how many can actually be used.

This technique has been implemented and in general ten data sets were calculated simultaneously. Using more than ten data sets required the computer to spend too much time swapping pages of virtual memory, and hence dramatically increased computation time.

4.6.6 Symmetric Multi Processor (Parallel) Programming

A symmetric multi processor (SMP) computer has multiple central processing units (processors or CPUs), all having access to shared memory. The advantage that this type of computer has over computers with a single processor is that, if the algorithm allows for it, the program can do more than one thing at a time. This can speed up the execution of the program by a factor equal to the number of processors. In practice this speed increase is limited by the overhead in the program required to partition the work amongst the processors, and because not all algorithms (or parts of algorithms) can be broken into parts that can be run simultaneously.

Whilst it is possible to divide many programs into separate parts that can execute on different processors, the simplest way to parallelise a program is to allow the compiler to divide loops up so that different iterations run on different processors. In order to make this work as efficiently as possible (by reducing the proportion of the overhead mentioned above) each iteration of the loop needs to be as long as possible. The difficulty is that this type of parallelism can only be used if the iterations of the parallelised loop do not need to write to the same memory locations (reading is not a problem if there is no writing).

In the program developed for this thesis there are many nested loops. Ideally in the case of nested loops the outermost loop would be parallelised. However, this is not always possible. Therefore to gain maximum improvement from the use of parallelisation it is necessary to arrange the loops in each nesting in such an order that the loop that is parallelised is as close to the outermost loop as possible. This means arranging the code so that iterations of all of the inner loops write to different memory locations. For

example, one way that these locations can be kept separate is to use the loop index as an index into an array. This means that each iteration of the loop will write to a different part of the array and thus that loop can be safely parallelised.

Profiling the code showed that most of the execution time was spent calculating the average of the spatial second moment of individual plumes (Equation 4.44). Careful consideration was therefore placed in ordering the loops so that the outermost loop possible was parallelised. Performance results are shown in Figure 4.4.

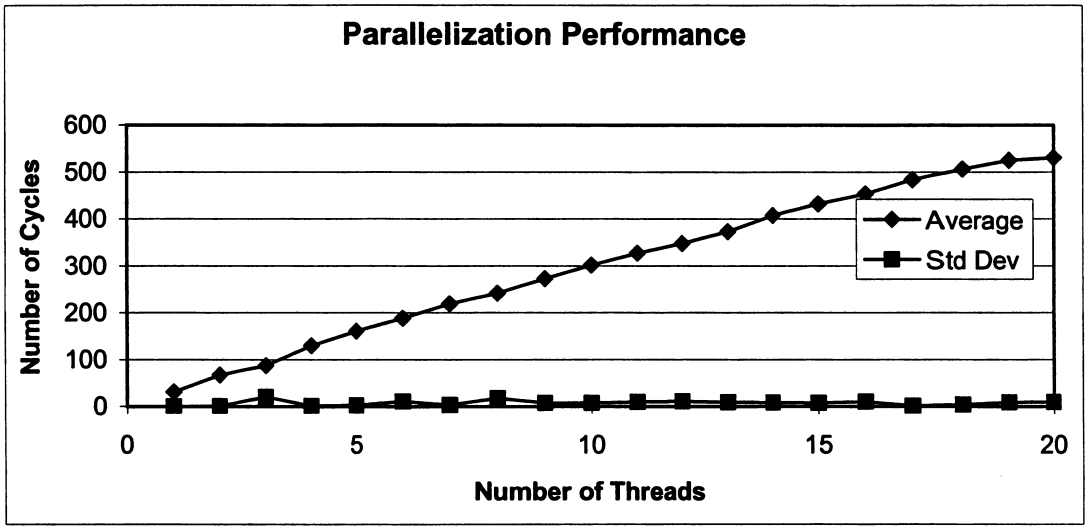


Figure 4.4 Parallelisation performance of the code that implements Equation 4.44. The number of cycles refers to the number of terms in Equation 4.44 calculated in 8 minutes of real time (i.e. 16 minutes of CPU time for 2 processors etc.) This data was averaged over several trials at different times. This data uses a mesh of 60 by 30 elements.

Figure 4.4 shows that excellent speedup results were obtained by using more processors. Unfortunately, several researchers shared the computer and so not all 20 of the processors were available all of the time.

In order to further speed up calculations the part that calculates Equation 4.44 was commented out so that the other data being calculated could be determined using a finer

grid. In this case the code spent most of its time in the function DGBTRS solving Equation 4.38 and Equation 4.39. DGBTRS is a LAPACK routine that solves systems of linear equations given a matrix that has been LU factorised and an arbitrary number of right hand sides. Unfortunately the algorithm for solving a system of linear equations cannot be parallelised. (There is an algorithm for solving systems of linear equations that can be parallelised. However, it is much slower when only a single processor is available and for it to actually speed up the calculations the number of processors needs to be of the order of the number of equations to be solved. Insufficient processors were available for this algorithm to be practical.) However, separate processors can handle the solutions for each different right hand side vector. Therefore the code was arranged to determine the derivatives with respect to the different velocity components to be determined simultaneously. This gave a reasonable increase in speed of the code with up to five processors, see Figure 4.5. The significance of two and five processors in the figure are that there are two right hand side vectors for the first derivative (two dimensions) and five for the second derivative. It would be expected that the performance would level out at four processors, but due to the program being written to handle 3 dimensions there is a gap in the set of two dimensional vectors and so five vectors were calculated instead of four, see Figure 4.6. More than five processors gave some increase in speed from parallelisation of other loops in the code, but the bottleneck in DGBTRS prevented it from being substantial.

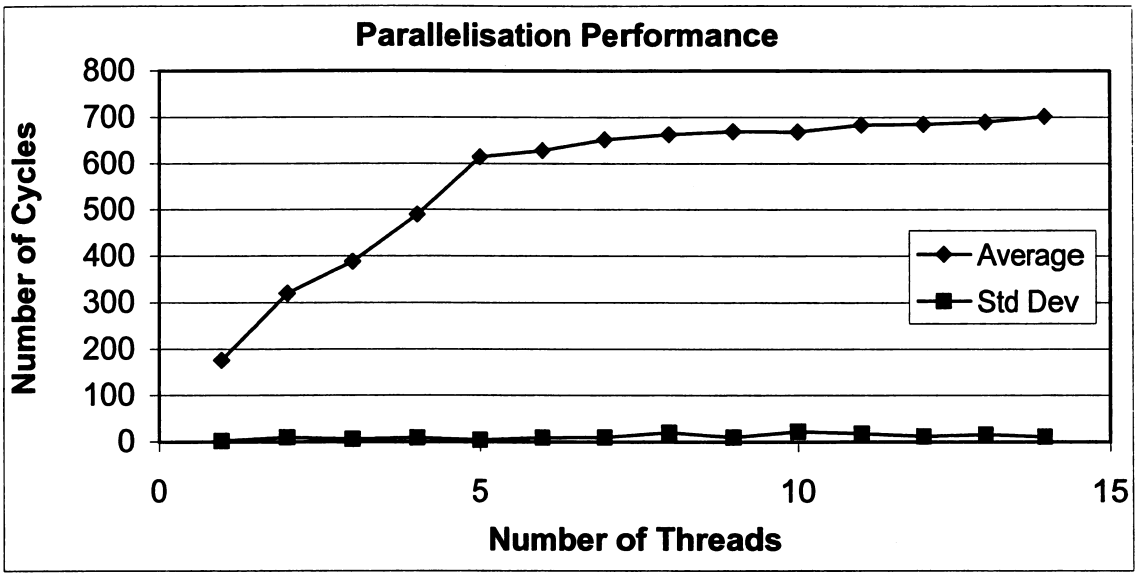


Figure 4.5 Parallelisation performance of the code that implements Equation 4.38 and Equation 4.39.

The number of cycles refers to the number of terms in Equation 4.38 and Equation 4.39 calculated in 8 minutes of real time (i.e. 16 minutes of CPU time for 2 processors etc.) This data was averaged over 7 trials at different times.

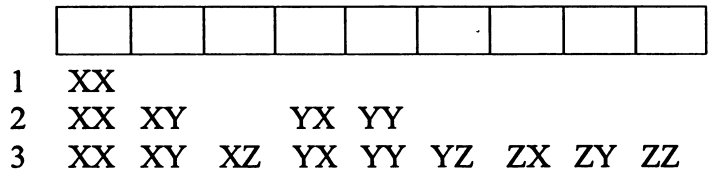


Figure 4.6 Layout of second derivative vectors in the program.

XY refers to the second derivative of concentration with the groundwater velocity in both the X and Y directions.

4.6.7 Alternative Multi Computer Programming

Carrying the reasoning from the previous section further, much better parallelisation could be obtained if each term in Equation 4.36 and Equation 4.37 could be calculated by a different processor. In practice the computer executing the code had limits on how long a program could run for and so it was necessary to save the state of the calculations to file periodically and then restart the program and continue. The saves were arranged to occur between the terms in Equation 4.36 and Equation 4.37. It would be possible therefore to divide the terms amongst several different computers and then add the results. This was not implemented.

5 Velocity Verification and Discussion

5.1 Results

5.1.1 Comparison of Normal-k Model and Lognormal-k Model

The equations governing the normal-*k* model, Equation 4.11 and Equation 4.12, and the lognormal-*k* model, Equation 4.27and Equation 4.28, appear very similar, however, they behave quite differently. In Equation 4.11 the deterministic part, that is the first term on the right hand side, is based on the arithmetic mean of the hydraulic conductivity, whereas in Equation 4.27 it is based on the geometric mean. When the autocorrelation function is isotropic the effective conductivity (see Section 3.2.1) lies between these two means (Gelhar, 1993). Therefore the stochastic part of Equation 4.11, that is the second term on the right hand side, must be negative, whereas in Equation 4.27 it must be positive. This can cause a problem in the normal-*k* model when the variance of the hydraulic conductivity becomes large compared to the mean. If arbitrarily large hydraulic conductivity coefficients of variation are used then negative velocities will be predicted. This problem arises because arbitrarily large coefficients of variation imply that a significant fraction of the probability density fraction is negative and hence violate the assumption that *k* is a non-negative property. It is generally accepted that hydraulic conductivity is lognormally distributed with significantly large coefficients of

variation (Gelhar, 1993). Therefore it is recommended that only the lognormal-*k* method be employed.

5.1.2 Integral Scale and Mesh Resolution

In order to investigate the effect of integral scale a series of numerical experiments were carried out. The experiments involve flow in a square field between two opposing fixed head boundaries, with the other boundaries having no flow conditions, see Figure 5.1.

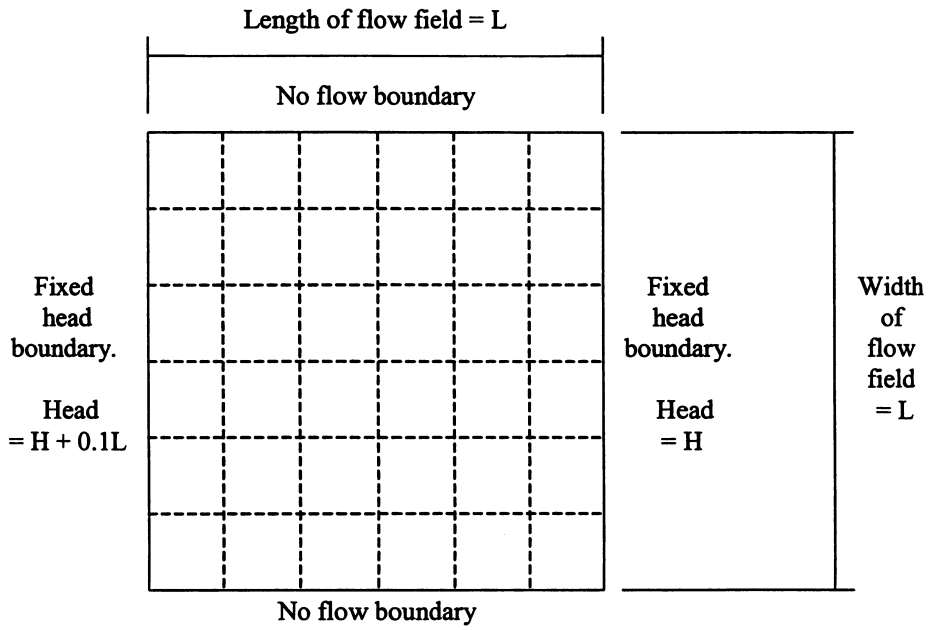


Figure 5.1 Flow field used in all examples.
(The number of elements varied)

The experiments were performed with various ratios between the integral scale and the size of the flow field and over a variety of mesh resolutions. All data points use the same point concentration probability distribution. The mean value of hydraulic conductivity selected was $\ln(0.1/\nabla H)$, thus in a deterministic case where the hydraulic conductivity was equal to this value the x-velocity, v_x, would be 0.1. The standard deviation selected for Y was 0.5. Putting these numbers into the expression for the

mean of a lognormal probability distribution results in the expected velocity in the longitudinal direction for the single random variable model presented in Section 2.2.2 being 0.133. This is different to 0.1 because the lognormal distribution is significantly skewed and so even though $k = e^{Y}$, E[k] is larger than $e^{E[Y]}$.

An exponential autocorrelation function was adopted. Thus covariances were calculated using (Vanmarcke, 1983):

$$\mathrm{cov}(Y_i, Y_j) = \sigma_Y^2 \exp\left(-\sqrt{\left(\frac{x_i - x_j}{I_x}\right)^2 + \left(\frac{y_i - y_j}{I_y}\right)^2} \right) \qquad \text{Equation 5.1}$$

where (x_i, y_i) and (x_j, y_j) are the coordinates of the centres of elements i and j, and I_x and I_y are the integral scales in the x and y directions as defined by Equation 2.10. Unless otherwise noted the integral scales are equal in both directions (i.e. isotropic).

The velocities given were calculated at the centre of each element. Results for mean X velocity and the standard deviation for X and Y velocities at the centre of the grid are given for various element size to flow field ratios to show the effect of mesh resolution. These results are shown in Figure 5.2, Figure 5.3 and Figure 5.4. The plot of mean Y velocity is not given since it always evaluates to zero at the exact centre of the grid. However, it is non-zero at other locations.

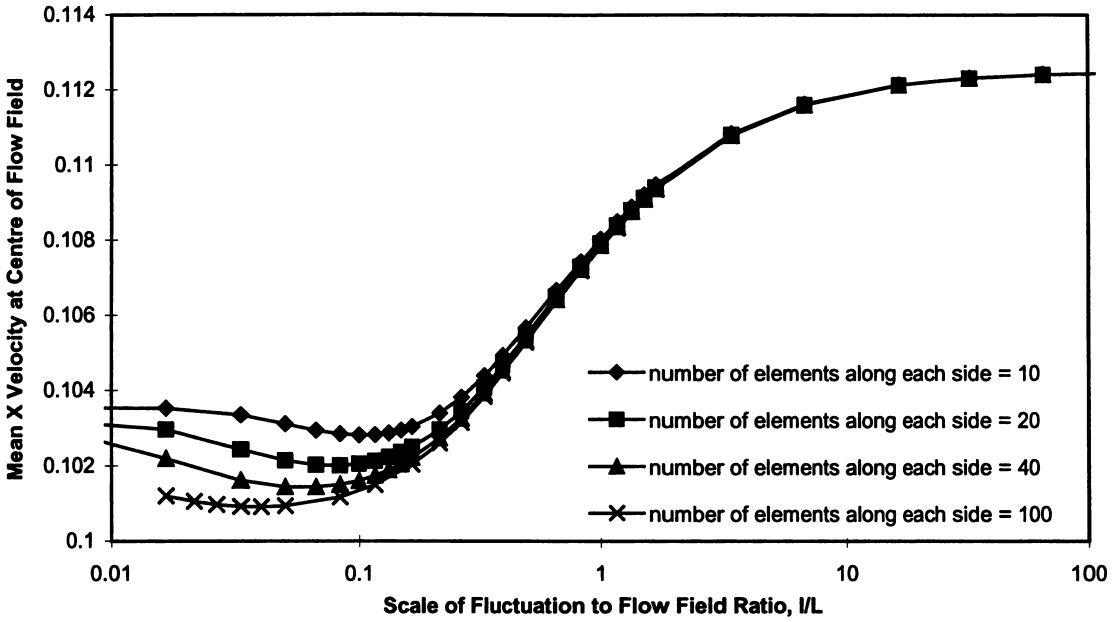


Figure 5.2 Plot of mean X velocity against integral scale

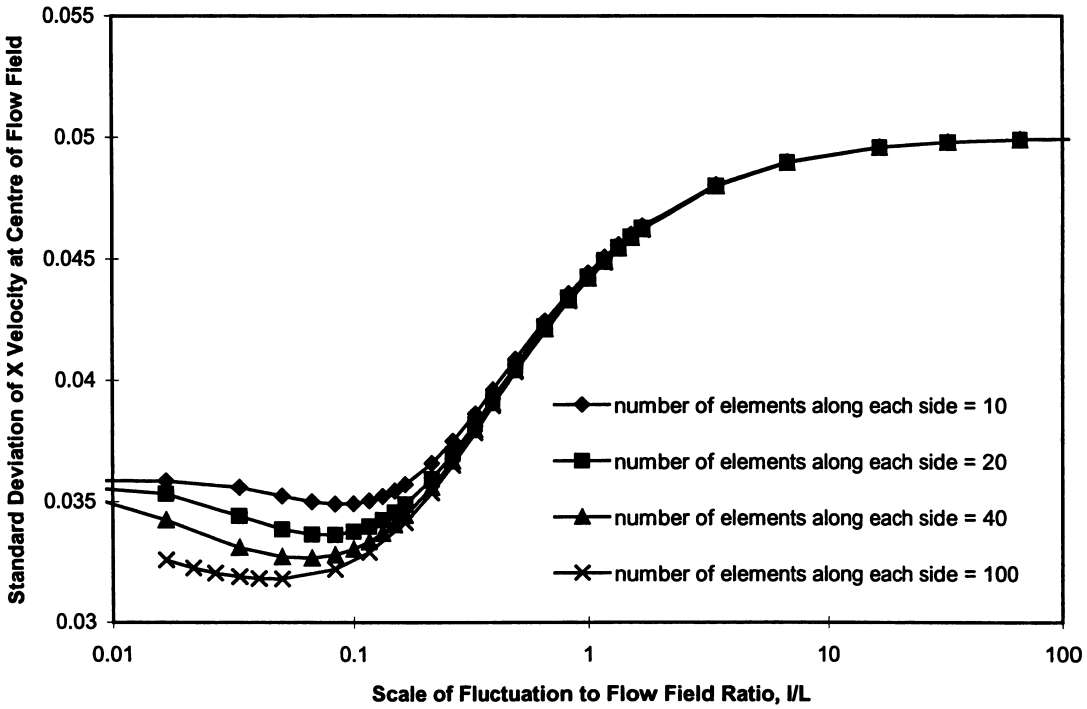


Figure 5.3 Plot of X velocity standard deviation against integral scale.

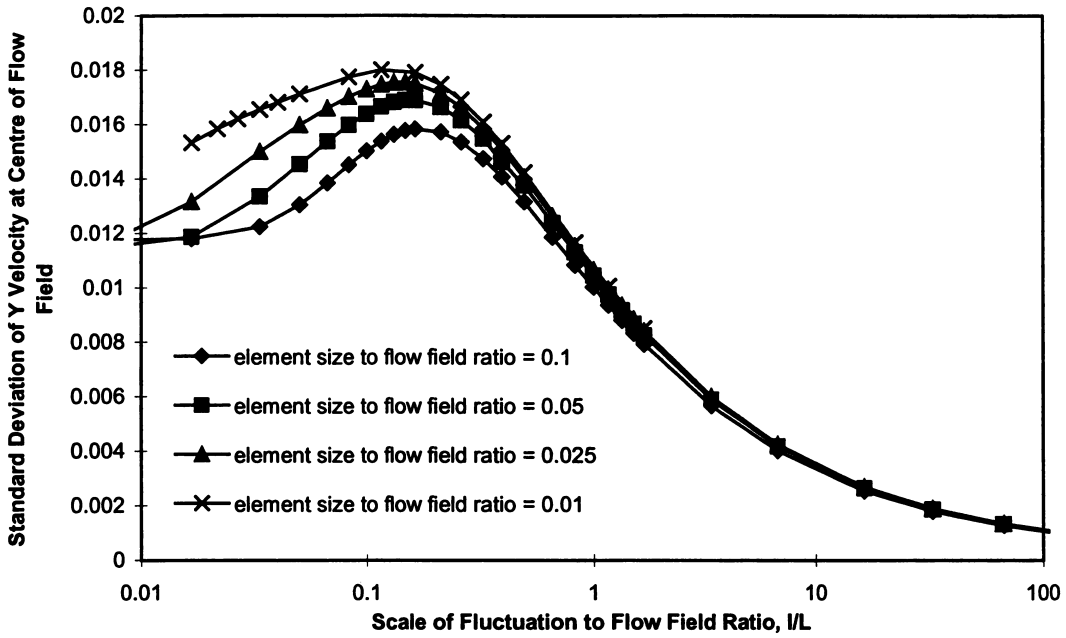


Figure 5.4 Plot of Y velocity standard deviation against integral scale.

Figure 5.2 shows that the mean x velocity (i.e. flow in the direction of mean head gradient) speeds up with increasing correlation. The entire aquifer experiences an increase, although this increase is not uniform across the aquifer. However, there is a mesh resolution problem that breaks this tendency down when the element size becomes a significant fraction of the integral scale, in the left part of the figure. This will be discussed later in this section.

The effect of mean x velocity increasing with increasing integral scale is as expected from the behaviour at the extreme ends of the scale.

Firstly, for small integral scales a small volume in the centre of the flow field is effectively a long way away from the boundaries of the flow field, i.e. many integral scales away. An infinite flow domain is an appropriate model for this. Using the infinite flow domain assumption Gutjahr et al (1978) have shown that for a second order

approximation of the lognormal hydraulic conductivity distribution the effective conductivity in two dimensions is the geometric mean. This would be a value of 0.1 in Figure 5.2. This figure shows that at small integral scales the result tends to the value of 0.1, more so as the mesh becomes finer.

On the other hand, in the case where the integral scale is infinite every point in the flow field is completely correlated to every other point. Therefore, the entire flow field has a uniform value for hydraulic conductivity, albeit an unknown one. In this case Darcy's law can be applied directly to get the velocity and we find that the mean velocity will be the mean hydraulic conductivity times the mean hydraulic gradient. The mean hydraulic conductivity is higher than the geometric mean and it is to be expected that the mean velocity will tend toward this value with increasing correlation. Running a set of data with a very large integral scale gives a velocity very close to 0.1125. This is exactly what would be expected from a first order Taylor series of the mean of k expressed in terms of the variance of Y. In a later section this fact will be used to examine the accuracy of the Taylor series applied in this work.

The standard deviation of the x velocity (i.e. flow in the direction of mean head gradient) chart (Figure 5.3) shows that the standard deviation of the velocity also increases with increasing integral scale. Once again this does not happen with the smallest integral scale. It is expected that this is also an element size/integral scale problem. The standard deviation is lower with smaller integral scales because the process of determining the velocity tends to smooth out the randomness. An area of high or low conductivity is very small and so can only have a limited effect because it locally changes the gradient of the hydraulic head field and this change in gradient has the opposite effect on the velocity to the hydraulic conductivity effect. Thus a local

decrease in hydraulic conductivity tends to create a locally higher hydraulic head gradient, because it is harder for the groundwater to flow through. The higher hydraulic head gradient tends to increase the flow and this counteracts to a limited extent the tendency of the decrease in hydraulic conductivity to decrease the flow. With the largest integral scale the whole aquifer tends to have a high or a low conductivity and so the standard deviation of the velocity is much higher. Once again the result of 0.05 is exactly what would be expected from a first order Taylor series of the standard deviation of k expressed in terms of the variance of Y. This latter effect also causes the entire aquifer to have a more uniform mean velocity value and standard deviation.

The lower end of the scale can be considered to be similar to an unbounded domain. Gelhar (1993) gives an expression for the variance of the specific discharge in an unbounded domain as $3q^2\sigma_{\ln K}^2/8$ where q is the specific discharge. Using this expression for the data in Figure 5.3 gives a result for the standard deviation of 0.0306. Figure 5.3 shows that at small integral scales the result tends to this value, more so as the mesh becomes finer.

On the other hand, Figure 5.4 shows that the standard deviation of the y velocity (i.e. flow perpendicular to the direction of mean head gradient) decreases with increasing integral scale. Once again an element size/integral scale problem seems to reverse this trend at small integral scales. The result here is the opposite of the result for the x velocity since the y velocity represents flow around areas of low local relative hydraulic conductivity. If these areas are large then the change in hydraulic conductivity between them is more gradual and so the water sweeps around them, whereas if they are small the conductivity changes will be much more abrupt and so the water will follow more tortuous paths through the aquifer.

With the largest integral scale there are no areas of lower conductivity since the aquifer is uniform thus the standard deviation of the transverse velocity is zero.

Once again, the lower end of the scale can be considered to be similar to an unbounded domain. Gelhar (1993) gives an expression for the variance of the transverse specific discharge in an unbounded domain as $q^2\sigma_{\ln K}^2/8$ where q is the specific discharge. Using this expression for the data in Figure 5.4 gives a result for the standard deviation of 0.0177. Figure 5.4 shows that at small integral scales the result tends to this value, more so as the mesh becomes finer.

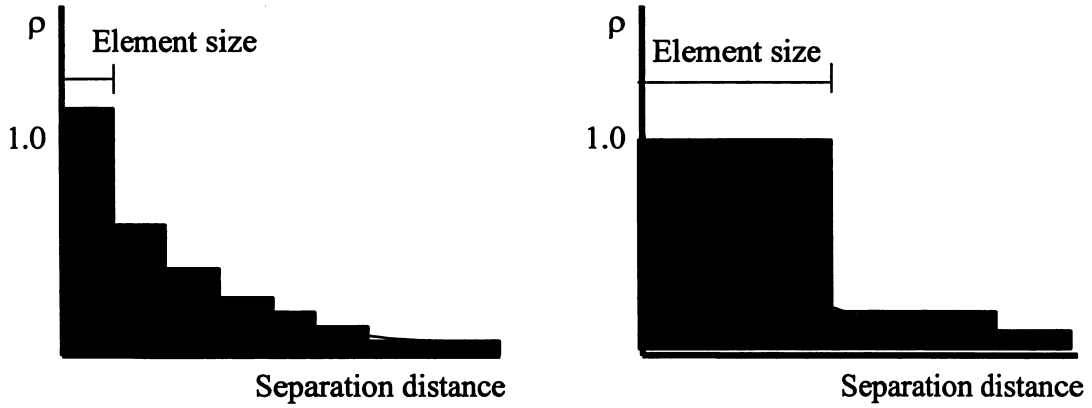


Figure 5.5 Effect of element size to integral scale ratio.

Figure 5.2, Figure 5.3 and Figure 5.4 all show a change in behaviour at small integral scales. That is, the curves change from increasing to decreasing and vice versa. This is a function of the ratio between the element size and the integral scale. For the integral scale to be adequately modelled it is necessary that the integral scale be several times the size of the element. This is shown in Figure 5.5. In the figure on the right the large element size is inadequate for describing the auto correlation function with any accuracy.

5.1.3 Integral Scale Adjustment

Examination of Figure 5.2, Figure 5.3 and Figure 5.4 shows that larger element sizes tend to increase the mean and standard deviation of the X velocity and reduce the standard deviation of the Y velocity. This effect is caused by the greater inaccuracies involved in modelling the random field with larger elements. It is particularly evident at the lower integral scale/element size ratios. If the integral scale is adjusted by adding $\pi^{-0.5}$ times the element length, the results for the various element size to flow field ratios all tend to fall on a single curve (Figure 5.6). This creates a very good fit except at lower integral scales, where the mesh resolution problem, to be discussed in the next section, causes it to break down.

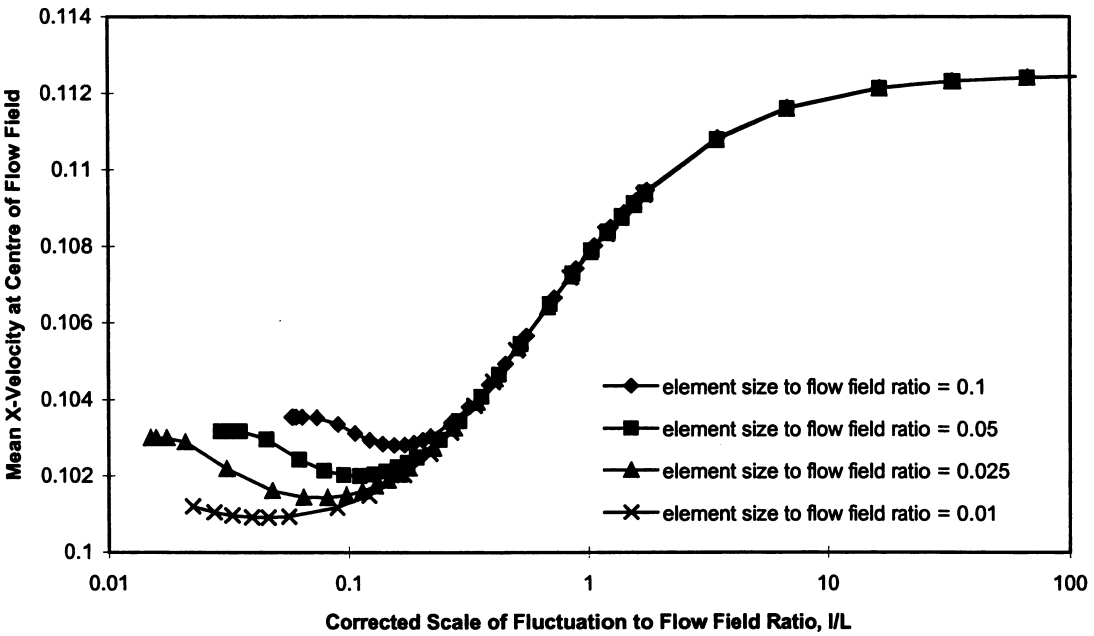


Figure 5.6. Plot of mean X velocity against corrected integral scale.
(The correction involves adding $\pi^{-0.5}$ of the element length to the integral scale. Compare with Figure 5.2.)

Figure 5.7 shows why $\pi^{-0.5}$ times the element size has been added to the integral scale. The integral scale is defined as the integral from zero to infinity of the autocorrelation function. Although Equation 5.1 has been used to determine the covariances the discrete nature of the elements means that the actual autocorrelation function is stepwise because the covariance jumps from one value to another when it crosses element boundaries, see shaded area in Figure 5.7. Therefore the actual integral scale is much closer to the shaded area in Figure 5.7. A second curve has been shown in Figure 5.7 that is the same as the curve representing Equation 5.1 except that it has been shifted right by a distance equal to the "element size". The area under the modelled autocorrelation function is approximately the average of the area under the other two curves. Therefore half of a quantity that represents the "element size" has been added to the integral scale.

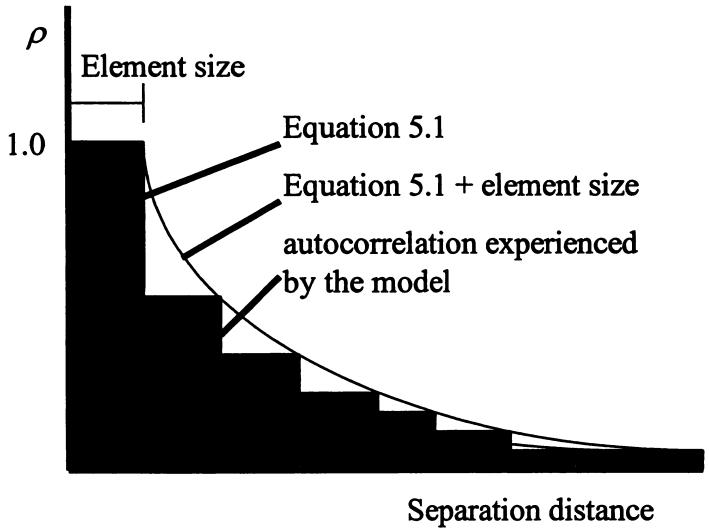


Figure 5.7. Shows why the integral scale is corrected by adding half of the "element size".

A difficulty is that the elements are square and so depending upon the orientation of the element the element size varies from the element length to the element length times the square root of two (along the diagonal). Therefore the question becomes what is the average element size. It appears reasonable to assume that this would be the diameter of

a circle with an equivalent area. If the element size is modelled by the diameter of an equivalent circle then the correction to the integral scale is given by $\pi^{-0.5}$ times the element length. Thus:

$$I_{corrected} = I \times \left(1 + \sqrt{\pi} \times l\right) \qquad \text{Equation 5.2}$$

where $I_{corrected}$ is the value for I used in Figure 5.6, l is the element length, and I is the integral scale value used in Equation 5.1.

Trial and error with the data in Figure 5.6 also indicates that this is the most accurate correction, anything larger tends to overcompensate.

5.1.4 Mesh Resolution

Figure 5.6 shows that when the integral scale becomes a significant fraction of the element size (i.e. on the left part of the chart) the method begins to lose accuracy. This occurs because pairs of elements with highly correlated hydraulic conductivities aligned across the mean hydraulic gradient have the effect of reducing the overall flow and those orientated along the flow tend to increase it, this is discussed in more detail in Section 5.2.2. When the element size becomes a significant fraction of the integral scale these pairs do not have significant weighting because the covariance factor is quite small. Thus the behaviour depends more upon the response of individual elements, for which the covariance is always the variance. These individual elements tend to increase the flow. Therefore the curves in Figure 5.6 rise up at low integral scales.

This means that to get accurate results the integral scale should be several times the element size. Unfortunately the number of elements per integral scale required increases as the integral scale becomes smaller. Examining Figure 5.6 it appears that

for integral scales between one quarter and one half of the flow field length that the number of elements required per integral scale is three times the number of integral scales that fit in the flow field. This relationship may not hold for larger or smaller integral scales.

5.1.5 Anisotropic Conductivity Fields

The previous results have looked at aquifers with isotropic hydraulic conductivity random fields

In order to investigate the effect of anisotropy in the integral scale two examples were run using a integral scale ten times greater along one axis than the other. In one case the larger integral scale was parallel to the mean hydraulic gradient and in the other case it was perpendicular to it. In effect these two examples simulate flow parallel to and flow perpendicular to the bedding planes in an aquifer, see Figure 5.8 and Figure 5.9. As expected the results show that the mean flow is higher when the mean hydraulic gradient is aligned parallel the bedding, and lower when aligned perpendicular to them.

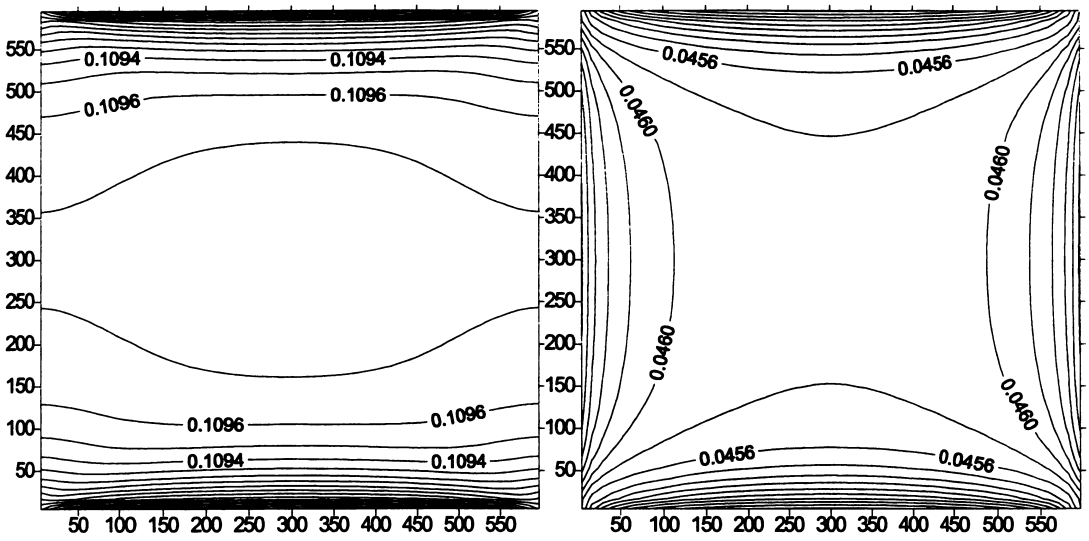


Figure 5.8 Mean x velocity (left) and standard deviation for flow occurring parallel to bedding.

Details as for Figure 5.1 using 60 elements in each direction and the integral scale in the x direction = 100, and in the y direction = 10.

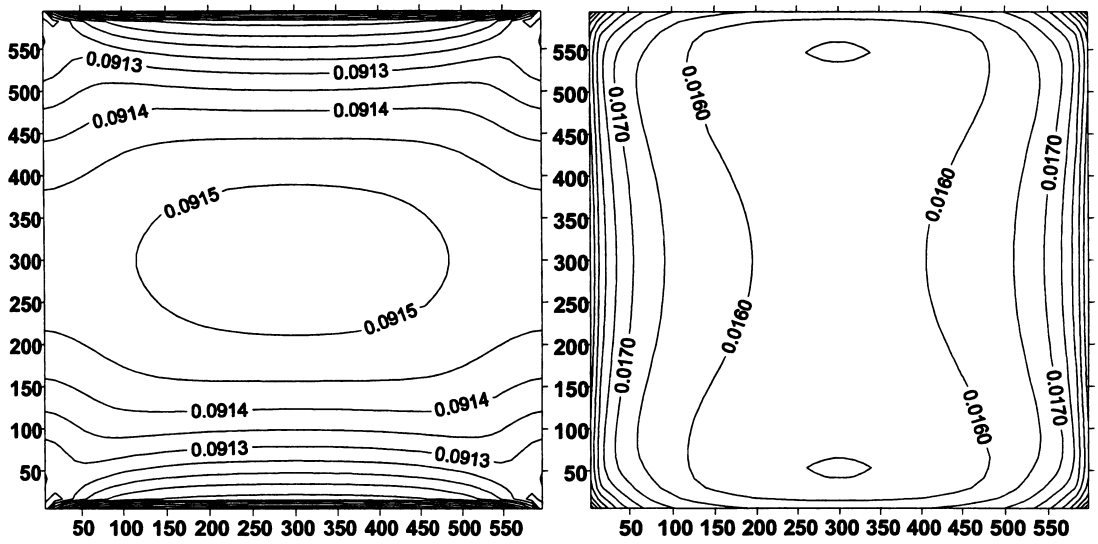


Figure 5.9 Mean x velocity and standard deviations for flow occurring perpendicular to bedding and across them.

Details as for Figure 5.8 except that the integral scale in x direction = 10, y direction = 100.

The higher mean flow along the beds results from the ability of the water to move from the low conductivity sections into the "fast lanes". However, there is a great deal more uncertainty in the flow rates as exhibited by the standard deviation. This is a result of

the uncertainty in the location of the "fast lanes". Each point in the aquifer is either a part of a fast lane or it is not. Therefore the variability is high.

The lower mean flow across the beds occurs because all of the groundwater must alternately pass through both the high and the low conductivity zones. Thus all of the water experiences the impeding effect of the low conductivity zones. However, the uncertainty involved in this case is much less. This is because the total flow through any line of elements stretching from one no flow boundary to the other must be the same as for any other similar line. Since velocity is flow divided by area the velocity does not get the opportunity to vary from one bed to another. Therefore the variability is low.

5.2 Discussion

5.2.1 Validation of Results

Section 5.1.2 gave the results for a series of numerical experiments using a variety of integral scales. It discussed the limiting cases when the integral scale was either zero or infinite and showed that the results of the numerical experiments were consistent with the solutions expected for those end points, except for the mesh resolution problem. It is now necessary to examine the results between these two end points.

Dykaar and Kitanidis (1992a&b) use a spectral approach to investigate the size of an averaging volume required to obtain an effective conductivity equivalent to that given by an unbounded domain solution. In doing so they present corresponding values of conductivity normalised with respect to the theoretical effective conductivity for an unbounded domain (K_G for 2 dimensional problems), versus the ratio between domain

length and integral scale, for a statistically isotropic Y(x) field with $\sigma^2_Y = 1$ and either an exponential or a Gaussian autocorrelation function.

Figure 5.11 shows a comparison of the results from Dykaar and Kitanidis (1992b) with the results shown in this study. These results come from their 2-D exponential covariance curve of their Figure 1 (Shown here as Figure 5.10).

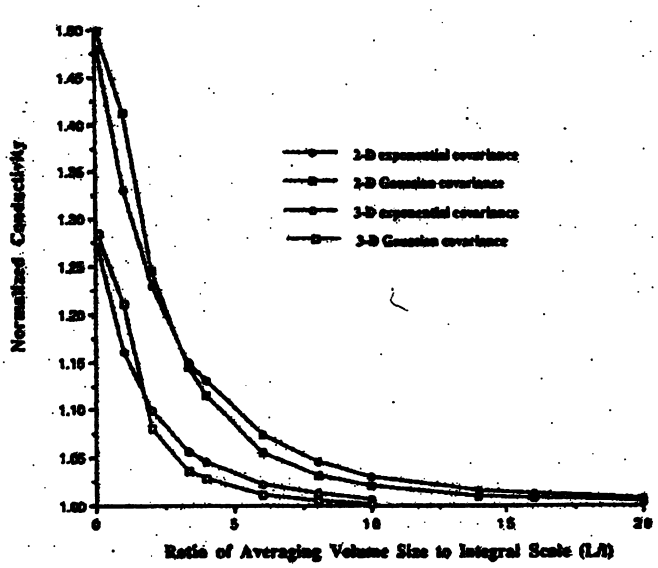


Fig. 1. Asymptotic approach to the analytical infinite-domain solutions, as a function of the averaging volume size L/ℓ, of the normalized effective conductivity computed using the small-perturbation approximation. The two-dimensional conductivity is normalized by the geometric mean, while the three-dimensional conductivity is normalized by $K_G(1+\sigma_f^2/6)$. In two dimensions about 20 integral scales per unit volume are needed to reach the infinite-domain result, while in three dimensions only about 10 are required.

Figure 5.10 Figure 1 from Dykaar and Kitanidis (1992b)

4 curves are provided in Figure 5.10 showing both 2-D and 3-D and both exponential and Gaussian covariances. In the text they give the equations for both of these covariance functions and they point out that the integral scale is equal to the length scale in the exponential case, but equal to the length scale multiplied by $0.5\ \pi^{0.5}$ in the Gaussian case (see their Equations 26 and 27). Interestingly if their integral scales are divided by the above number then their results and mine give virtually perfect

agreement for integral scale to flow field ratios greater than 0.25 (see dotted line in Figure 5.11). For values below this the mesh resolution problem discussed earlier causes my results to be higher than their results, as discussed previously.

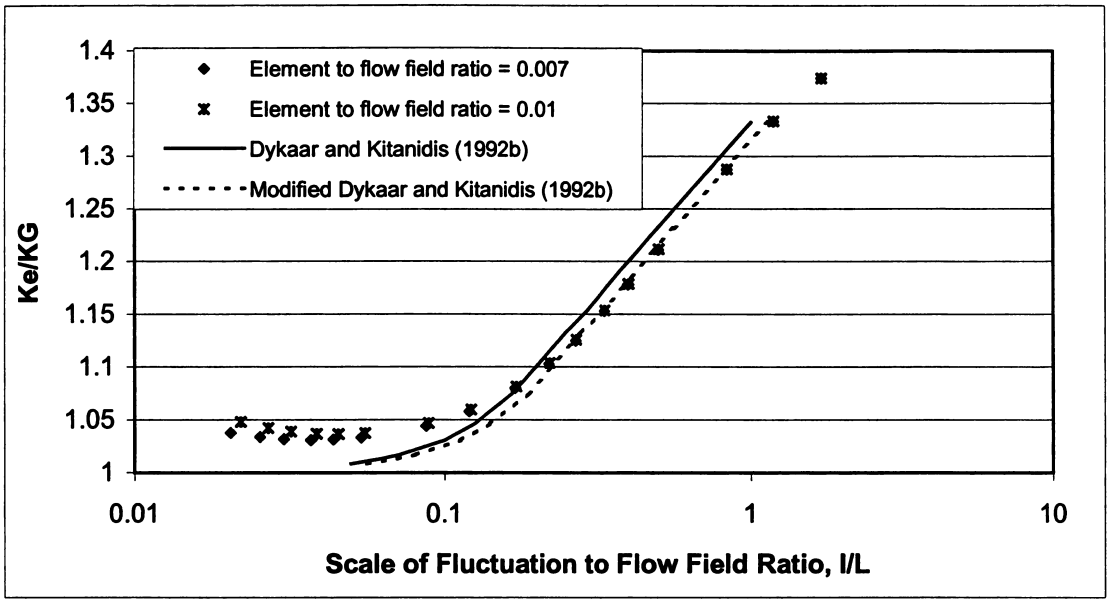


Figure 5.11 Comparison of the results of the present study with those of Dykaar and Kitanidis (1992b)

It should be noted that Dykaar and Kitanidis (1992b) uses a small perturbation solution. This neglects non-linear effects. Including non-linear effects is discussed in Section 7.2.

5.2.2 Mechanics of the Process

The concept of preferential flow path helps us examine the algorithm. The second term of the right hand side of Equation 4.27 has two factors: a second derivative and a covariance. The second derivative can be thought of in terms of its second order finite difference approximation:

$$4\Delta Y^2 \frac{\partial^2 v(Y_i, Y_j)}{\partial Y_i \partial Y_j} = v(Y+\Delta Y, Y+\Delta Y) + v(Y-\Delta Y, Y-\Delta Y) - v(Y+\Delta Y, Y-\Delta Y) - v(Y-\Delta Y, Y+\Delta Y) \quad \text{Equation 5.3}$$

where ΔY is a small perturbation in Y.

The first term on the right hand side of Equation 5.3 involves both hydraulic conductivities being increased. This simulates a preferential flow path through the two elements. The second term on the right hand side of Equation 5.3 involves both hydraulic conductivities being lowered. This simulates a path of low flow through the elements and higher flow around them. The third and fourth terms on the right hand side of Equation 5.3 have one hydraulic conductivity raised and the other lowered. This draws flow away from one element towards the other. These last two terms tend to form mirror images of each other that cancel each other out. Thus, in general, only the first two terms are significant.

When the two elements are aligned along the mean hydraulic gradient, or even diagonally to it, the first term on the right hand side of Equation 5.3 dominates. This simulates a preferential flow path though the elements, see Figure 5.12.

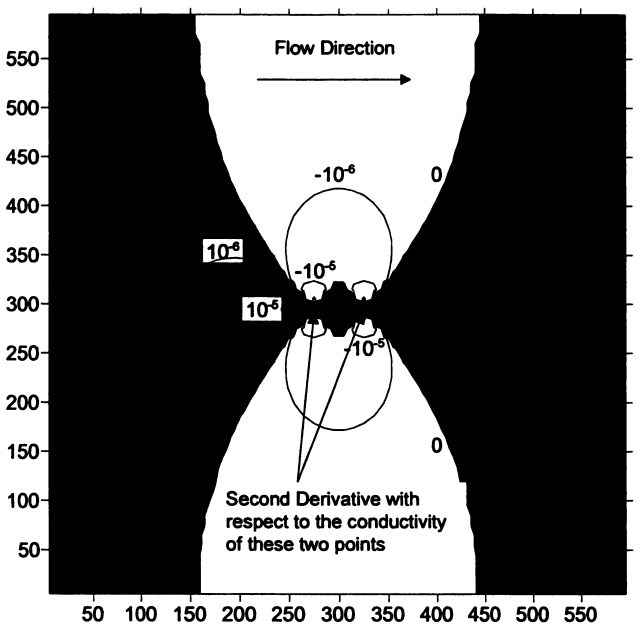


Figure 5.12 Contour plot of the second derivative of ground water velocity in the x direction with respect to the logarithm of the hydraulic conductivities of the elements centred at (275, 295) and (325, 295).

Details as for Figure 5.1 using 60 elements in each direction and average head gradient in the X direction. The shaded area is positive and indicates increased flow. Contours are at 0, +/-10^{-6}, +/-10^{-5}, +/-10^{-4}.

In Figure 5.12 the shaded area represents areas of increased flow and the non-shaded areas represent areas of decreased flow (decreased because the flow has been drawn toward the preferential flow path instead). Overall there is a net increase in flow through the aquifer.

When the two elements are aligned across the mean hydraulic gradient the second term on the right hand side of Equation 5.3 dominates. This simulates an obstruction, reducing the flow through the two elements and causing small local flow increases in certain nearby elements. However the overall effect is to reduce the total flow through the aquifer, see Figure 5.13.

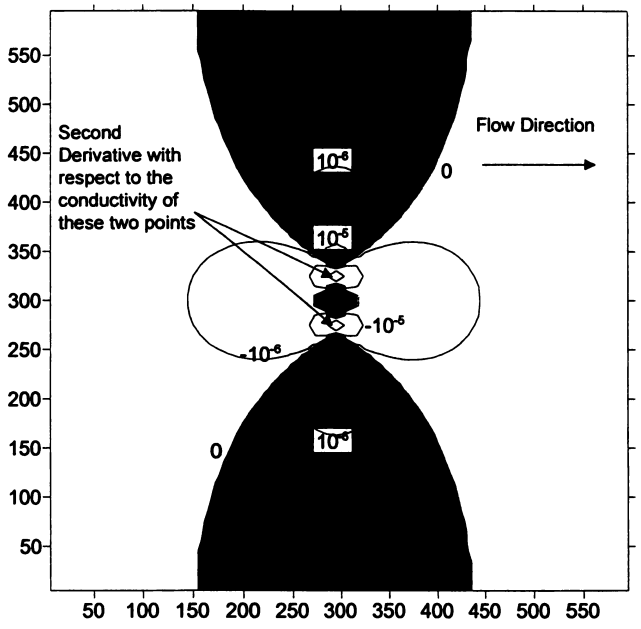


Figure 5.13 As for Figure 5.12 except that the derivative is with respect to the hydraulic conductivities of the elements centred at (295, 275) and (295, 325).

Once again, in Figure 5.13 the shaded area represents areas of increased flow and the non-shaded areas represent areas of decreased flow. There is a decrease in the flow through the two elements that the derivative is being taken with respect to, and a localised corresponding increase in some of the nearby elements that the flow needs to use as an alternative. Overall there is a net decrease in flow through the aquifer.

These effects would exist if the relevant preferential flow path or obstruction existed. They take into account the deterministic part of the problem, i.e. boundary conditions, and perhaps areas of differing mean hydraulic conductivity. The stochastic information indicates whether or not preferential flows paths or obstructions exist, and is contained in the covariance factor of Equation 4.27. This factor contains the information about the structure of the aquifer and determines whether or not a preferential flow path or obstruction exists and, if so, how strong it is. It thus acts as a weighting factor to the preferential flow path effects based on the probability that they exist.

This is most clearly illustrated in the anisotropic examples in Figure 5.8 and Figure 5.9. In Figure 5.8 the higher integral scale in the x direction ensures that the derivatives of pairs of elements that are relatively positioned similar to those in Figure 5.12 have greater weighting and so the overall mean flow is increased. In Figure 5.9 the higher integral scale is in the y direction and so the derivatives of pairs of elements that are relatively positioned similar to those in Figure 5.13 have a greater weighting and the overall mean flow is decreased.

To investigate the isotropic case the second derivatives taken with respect to elements aligned both along and across the flow were examined. Figure 5.14 shows the maximum and minimum for each case plotted as a function of the distance between the two elements.

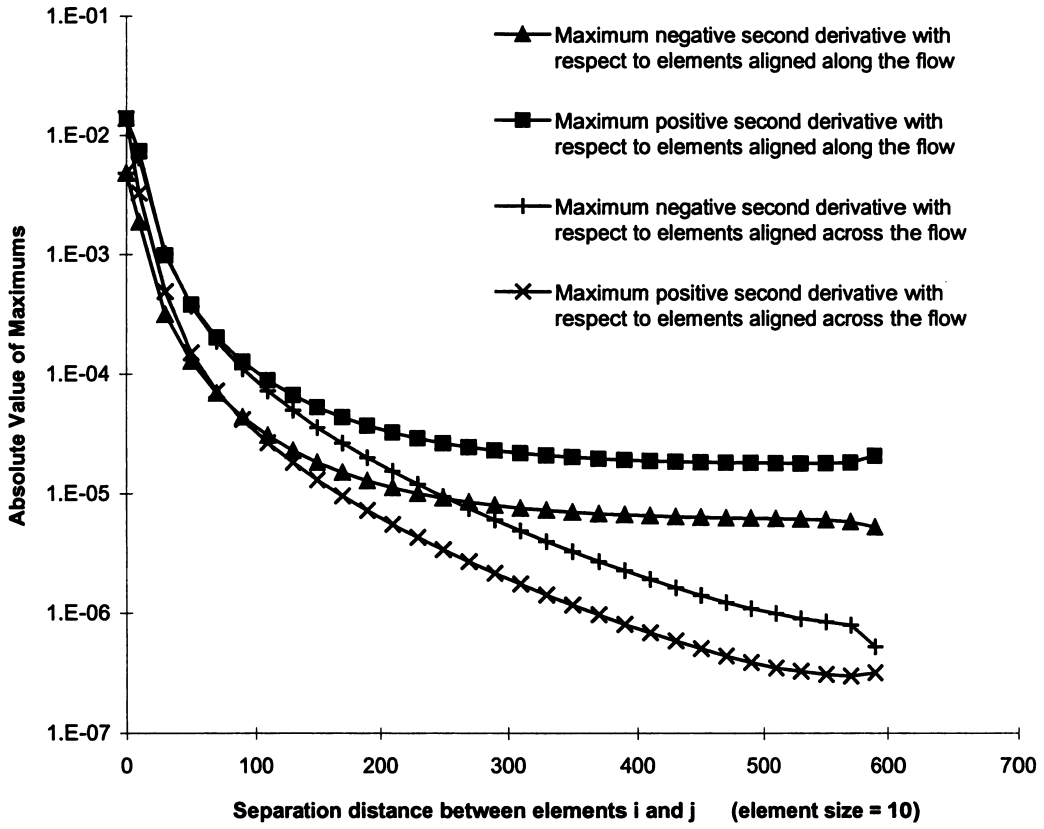


Figure 5.14 Relation between the maximum positive and maximum negative second derivative and the distance between the relevant elements, for element pairs aligned both along and across the mean hydraulic gradient.

Note that for pairs aligned along the mean hydraulic gradient the positive maximum dominates whereas for elements aligned across the flow the negative maximum dominates. Data from 60 by 60 element isotropic square mesh.

The effect in the isotropic case of velocity increasing as the integral scale increases occurs because the greater the correlation the more preferential flow paths are likely to exist. The reducing effect of elements aligned across the mean hydraulic gradient drops off much faster than the increasing effect of elements aligned along the mean hydraulic gradient, see Figure 5.14. Therefore the overall effect is to increase the mean velocity.

6 Concentration

6.1 Results

In order to investigate the behaviour of the model a series of numerical experiments were carried out. These experiments involve flow of a conservative contaminant in a rectangular aquifer. As in the velocity experiments, the boundaries are no flow boundaries along the long sides of the field and fixed head boundaries along the short sides, see Figure 6.1.

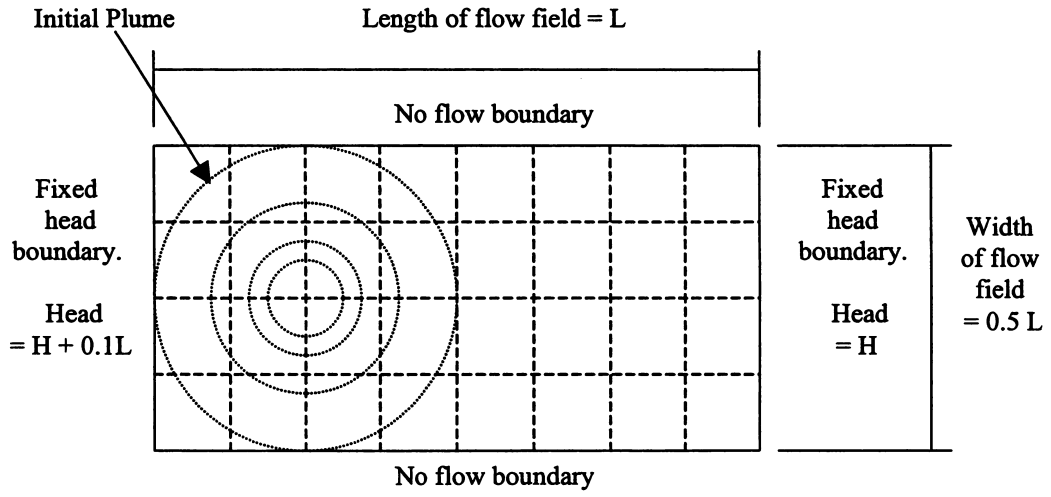


Figure 6.1 Flow field used in all concentration examples. The number of elements varied. However, all elements were square.

The experiments were performed with various ratios between the integral scale and the size of the flow field and over a variety of mesh resolutions. All data points are based on data that gives a deterministic X-velocity result of 0.1 and $\sigma_Y = 0.5$. Thus, in a uniform aquifer, the expected velocity in the x direction would be 0.133. Equation 5.1

was once again used for the autocorrelation function (i.e. exponential). Unless otherwise noted the integral scales are equal in both directions (i.e. isotropic).

The initial concentration conditions consisted of a circular plume with a Gaussian cross section centred along the middle flow line 1/4 of the way down the flow field, see Figure 6.1. The standard deviation for the Gaussian distribution used for the cross section was 1/6 of the width of the field. This means that at least 99.7% of the solute mass will be in the mesh.

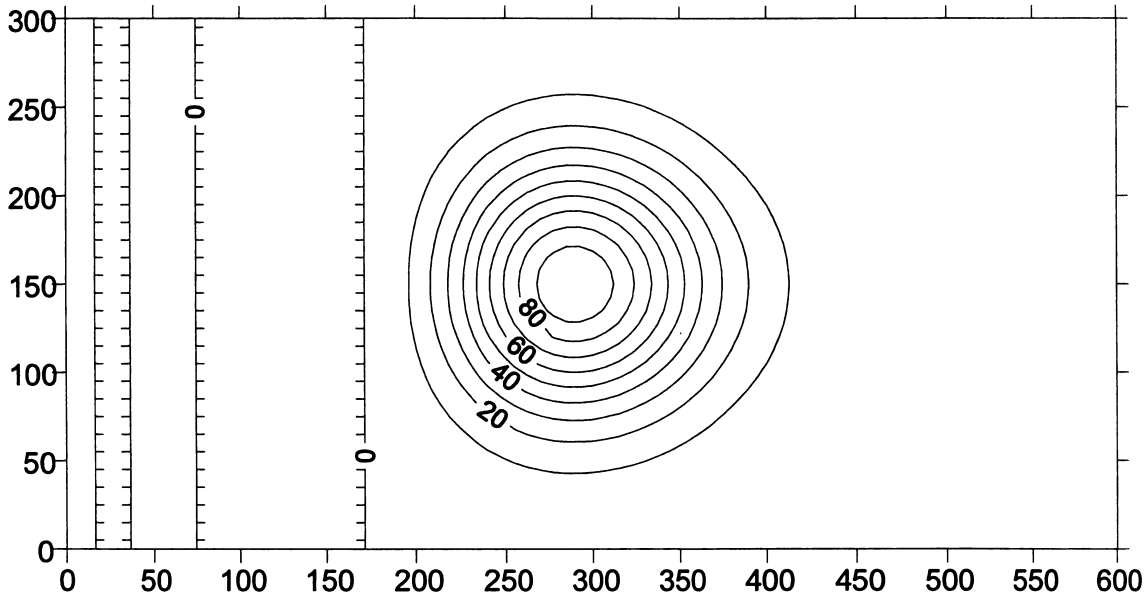


Figure 6.2 Deterministic result for the numerical experiments. Mesh resolution is 80 elements in the x direction and 40 elements in the y direction.

Figure 6.2 shows the deterministic result for the numerical experiments. The porosity for the aquifer was set to 0.5 and the time used was five steps of 150 units each. Thus the deterministic travel distance for the plume is 0.1 * 5 * 150 / 0.5 = 150 units. That is the centre of the plume should be at the centre of the flow field. 80 by 40 is a fairly coarse mesh for this problem so the plume has become quite distorted from its original circular shape and the centre of the plume does not appear to be at (300, 150).

However, integrating the concentration and determining the centroid does result in the centre being at (300, 150). Unfortunately a higher mesh resolution could not be used due to the execution time required by the stochastic part of the code. Using the mesh resolution of 80 elements by 40 elements took approximately 3 months for the code to execute. Most of this time was spent in the stochastic part. The time taken for the algorithm (discussed in Chapter 4) used is $O(n^4)$ in the number of elements. Thus doubling the resolution to 160 by 80, which quadruples the number of elements, will increase the execution time 256 fold. That is it would take 64 years to execute on the 20 processor parallel computer.

Another problem that can be seen in Figure 6.2 is the oscillations that appear upstream of the contaminant plume. These oscillations are highlighted by hachuring the 0 concentration contour. They are severe enough to cause negative concentrations, which obviously do not have a real counter part. The oscillations are a well known artefact of using the advection-dispersion equation with high Peclet numbers (Huyakorn, 1976). In all of the numerical experiments the (local) dispersion is zero, therefore the Peclet number is infinite. As discussed in Section 4.5 Huyakorn (1976) describes how these oscillations can be overcome by using an upstream weighting technique. This technique was implemented in the code used for the numerical experiments. It is governed by a weighting coefficient that ranges from 0 (no upstream weighting) to 1 (full upstream weighting). Huyakorn (1976) gives an expression for determining the ideal value of the weighting coefficient to use depending upon the Peclet number. For an infinite Peclet number the weighting coefficient should be 1.0. The upstream weighting technique works by introducing a small amount of numerical dispersion that smooths over the oscillations. Unfortunately it was found that using a weighting coefficient of 1.0 gave a

lot of dispersion and totally distorted the solute plume. It was therefore decided to accept the oscillations rather than use upstream weighting and so the weighting coefficient was set to zero, effectively disabling the upstream weighting code.

6.1.1 Mean Concentration Results

6.1.1.1 Small Integral scales

The smallest (corrected) integral scale to flow field ratio used was approximately 0.06. This value was chosen using Figure 5.2 because values below this give increasingly erroneous results for the velocity calculations. A contour plot of nodal concentration values is given in Figure 6.3 and a cross section through the centre of this case parallel to the mean groundwater velocity is given in Figure 6.4 showing a comparison with the deterministic result.

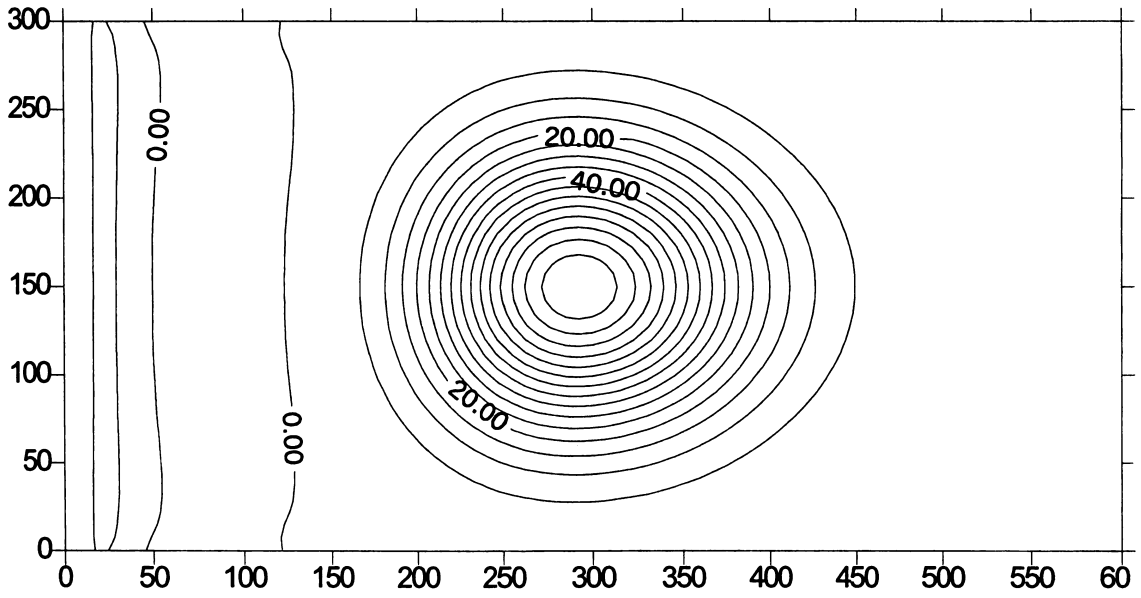


Figure 6.3 Contour plot of mean nodal concentration values for an integral scale to flow field ratio of approximately 0.06.

The plume is moving from left to right.

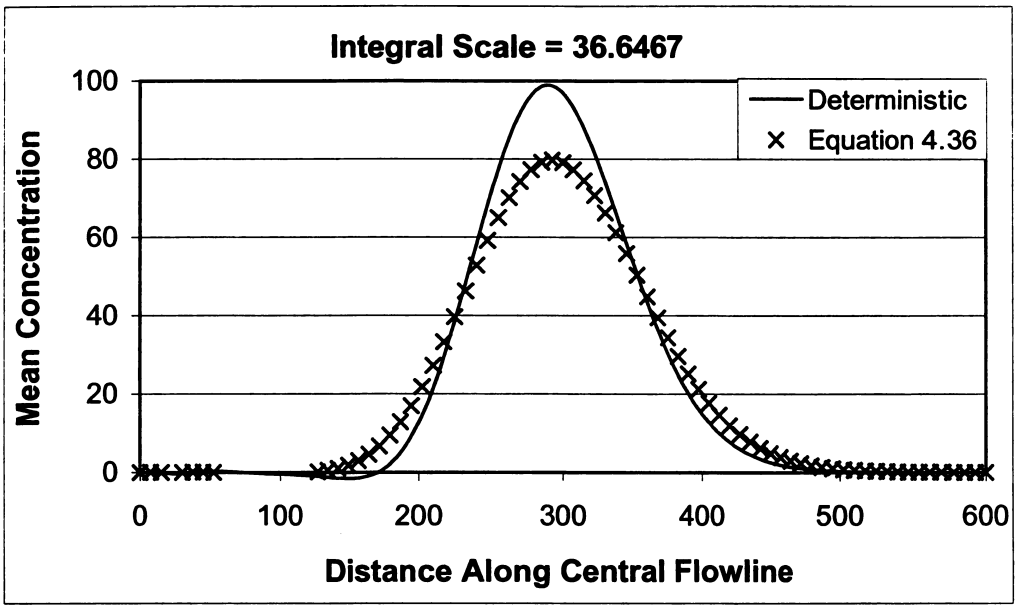


Figure 6.4 Mean concentration profile along the central flowline of the aquifer for an integral scale to flow field ratio of approximately 0.06.

The plume is moving from left to right.

The figure shows that the centroid of the solute plume has travelled about the same distance as it has for the deterministic result. This is as would be expected from Figure 5.2 where the mean velocity along the flow is very close to the deterministic velocity for aquifers with small integral scales compared with the aquifer size.

Furthermore Figure 6.3 shows that the solute plume disperses as it moves to this position relative to the deterministic result. The amount of dispersal will be discussed in the next section.

6.1.1.2 Larger Integral scales

Appendix E gives results for integral scale to flow field ratios of 0.06, 0.11, 0.18, 0.31, 0.51, 1.01, 1.67 and infinity ($I = 10^9$). Three things can be noticed from these diagrams. Firstly, the centroid of the plume travels further as the integral scale to flow field ratio increases. This is expected since Chapter 5 shows that the velocity increases as the

integral scale to flow field ratio increases (see Figure 5.2). Secondly, the plume tends to disperse more as the integral scale to flow field ratio increases. This rate of increase will be discussed in the next section. The final thing to notice is that as the integral scale to flow field ratio increases beyond 0.3 the solute plume has an increasing tendency to become bimodal. This tendency is an artefact of the model and Section 6.1.1.5 resolves the problem. The tendency can be clearly seen in Figure 6.5 where the plume appears to have broken into two lobes. Figure 6.6 shows a cross section through the centre of the plume parallel to the mean groundwater velocity and a comparison with the deterministic result.

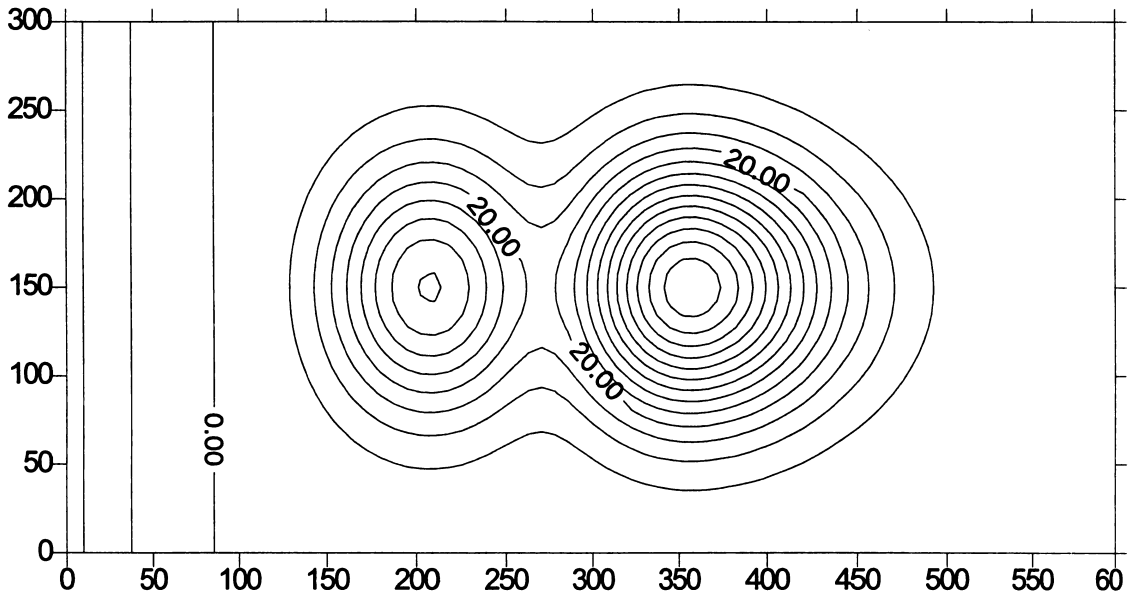


Figure 6.5 Contour plot of mean nodal concentration values for an approximately infinite integral scale to flow field ratio.

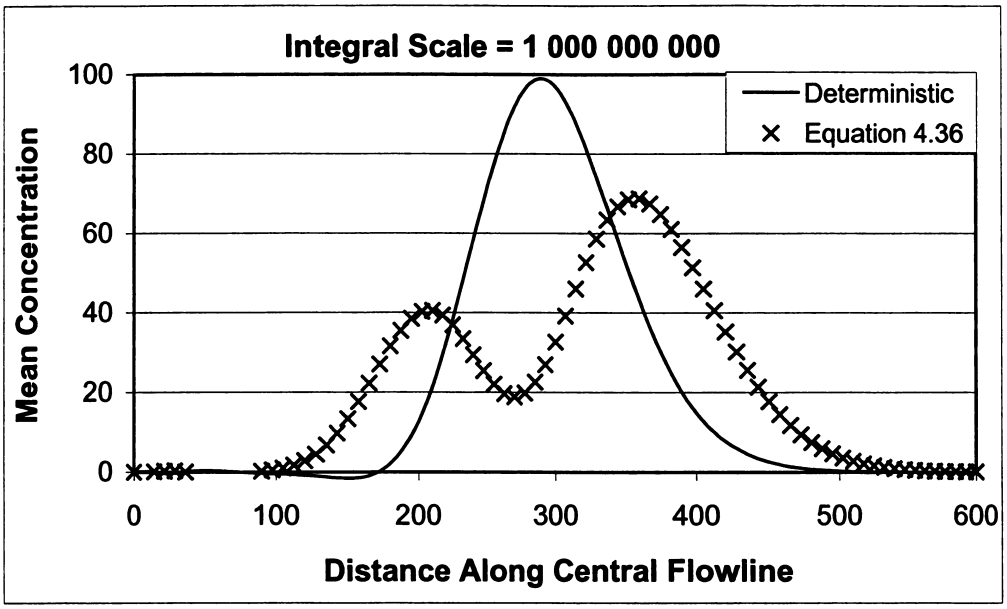


Figure 6.6 Mean concentration profile along the central flowline of the aquifer for an approximately infinite integral scale to flow field ratio.

6.1.1.3 Cause of Bimodal Effect

The cause of the bimodal effect appears to be loss of accuracy in using the Taylor series. The method used involved a Taylor series using derivatives of concentration with respect to the log of permeability (Equation 4.32). This was then expanded to include derivatives of both concentration with respect to the velocity and velocity with respect to the log of permeability. This was then simplified into Equation 4.36. Equation 4.36 is shown here for convenience.

$$\mathrm{E}[C_n] = \overline{C}_n + \frac{1}{2}\sum_{p,q=1}^{E} \frac{\partial^2 C_n}{\partial v_p \partial v_q} \mathrm{cov}(v_p, v_q) + \sum_{p=1}^{E} \frac{\partial C_n}{\partial v_p}\left(\mathrm{E}[v_p] - \overline{v}_p\right) \qquad \text{Equation 4.36}$$

Although Equation 4.36 is written in terms of the velocity statistics it is still a Taylor series based on the log conductivity. There are two differences between Equation 4.36 and a Taylor series based on velocity. The first difference is the first term, which in Equation 4.36 is the deterministic concentration based on the mean log hydraulic

conductivity, whereas in a Taylor series based on velocity it would be the deterministic concentration based on the mean velocity. That these are different can be seen from Figure 5.2, where the mean log conductivity is constant but the mean velocity varies. The second difference is the presence of the third term in Equation 4.36, which would not exist in a Taylor series based on velocity.

$$\overline{C}_{n(v)} - \overline{C}_{n(Y)} = \sum_{p=1}^{E} \frac{\partial C_n}{\partial v_p} \left(\mathrm{E}[v_p] - \overline{v}_{p(Y)} \right) \qquad \text{Equation 6.1}$$

Equation 6.1 shows these two differences, the over bar terms are calculated based on the mean of the respective parenthesised subscript. Equation 6.1 is really a first order Taylor series of the concentration in terms of the velocity. It is postulated that the inaccuracy of this Taylor series is what causes the error in Figure 6.5.

The reason that Equation 4.36 was used instead of a Taylor series based on the mean velocity was execution time. Using Equation 4.36 enabled a single set of derivatives to be used for several different conductivity fields, this took three months of computer time. Using a Taylor series based on the mean velocity would require a new set of derivatives for each conductivity field because the Taylor series would be centred at different velocities. Most of the calculation time was spent determining these derivatives thus calculating separate derivatives for each of the 10 conductivity fields used would take approximately 2.5 years.

6.1.1.4 Anisotropic Scale of Fluctuation

Two cases with anisotropic integral scales were evaluated. The results are also given in Appendix E.

In the first case the integral scale to flow field length ratio is 0.06 along the flow and 0.51 across it. This is analogous to flowing across layers of varying permeability. As expected this retards the movement of the centroid. Also there appears to be no trace of the bimodal effect, see Figure 6.7.

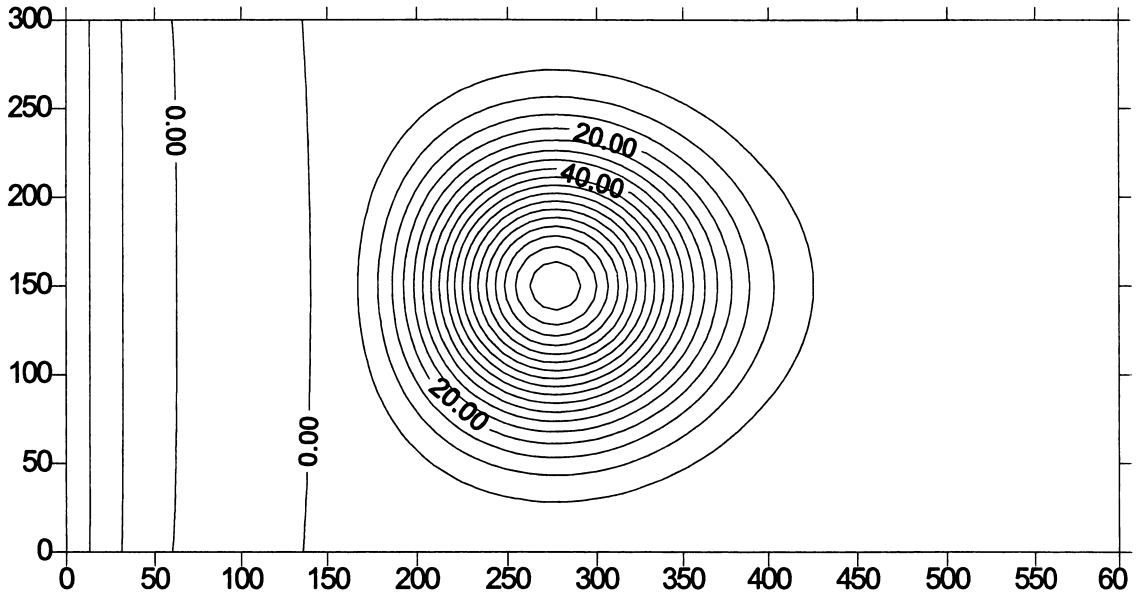


Figure 6.7 Contour plot of mean nodal concentration values for an anisotropic integral scale.

Longitudinal integral scale to flow field ratio is approximately 0.06. Transverse integral scale to flow field ratio is approximately 0.51.

In the second case the integral scale to flow field length ratio is 0.51 along the flow and 0.06 across it. This shows the centroid travelling further and a greater amount of dispersion than the 0.51 isotropic case, see Figure 6.8. Furthermore, the bimodal effect is quite pronounced, approximately equal to the 1.01 isotropic case.

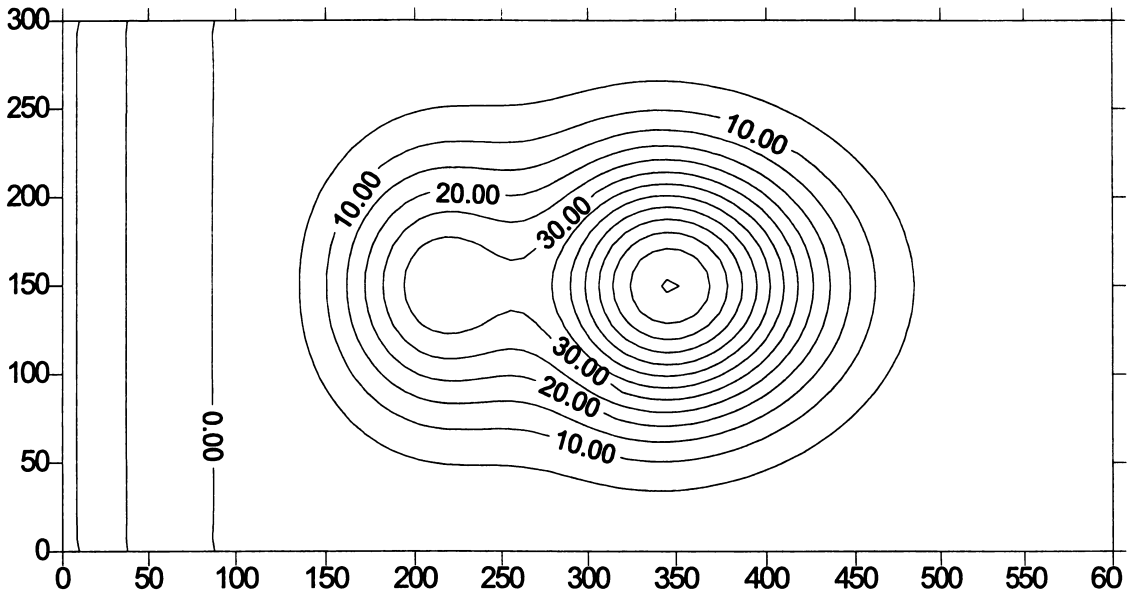


Figure 6.8 Contour plot of mean nodal concentration values for an anisotropic integral scale.

Longitudinal integral scale to flow field ratio is approximately 0.51. Transverse integral scale to flow field ratio is approximately 0.06.

These results give further weight to the hypothesis that it is the increased velocity that causes the bimodal effect since the one that decreases the velocity has no bimodal effect and the one that increases the velocity has an accentuated bimodal effect even beyond that of the isotropic case that uses the larger of the two integral scales used in the anisotropic case.

6.1.1.5 Solution to the Bimodal Effect

The inaccuracy caused by the velocity Taylor series occurs when the ratio between the integral scale and the size of the flow field is greater than 0.3. Under these circumstances most of the variation is contained in some simple geometry that can be ascertained by de-trending. For example the hydraulic conductivity may increase from one end of the aquifer to the other, or from the edges to centre. This trend should be determined and used as the mean conductivity. The remaining variation when modelled

by a random field about this mean will have a much smaller integral scale and thus the inaccuracy being discussed will not be present. It is appropriate to de-trend anyway as large scale aquifer structures tend to move the plume as a whole rather than disperse it (Dagan, 1994b) and neglecting to de-trend the data will give a concentration result with a high degree of artificial variability measured by the concentration variance. This variance will be so large that the mean result will become meaningless. Thus de-trending will give a much more useful result.

Furthermore, Indelman and Rubin (1995) show that when the trend is neither parallel nor perpendicular to the direction of the mean hydraulic head gradient that the mean groundwater velocity, and hence the direction that the centroid of the solute plume travels, will not be in the direction of the mean hydraulic head gradient. Thus de-trending becomes even more important.

6.1.2 Comparison with Analytical Results

6.1.2.1 Spatial Second Moments of Mean Concentration

Dagan (1982b) gives analytical results for the spatial second moments of the mean concentration plume. These assume an infinite two-dimensional homogeneous flow field and an initial point source. They are given in Equation 6.2 and Equation 6.3.

$$\varsigma_x = 2\sigma_y^{\,2} l_y^{\,2}\left\{\frac{3}{4}-\frac{3}{2}E+\frac{L}{l_y}+\frac{3}{2}\left[Ei\left(-\frac{L}{l_y}\right)-In\frac{L}{l_y}+\frac{l_y}{L}e^{\frac{L}{l_y}}\left(1+\frac{l_y}{L}\right)-\frac{l_y^{\,2}}{L^2}\right]\right\} \qquad \text{Equation 6.2}$$

$$\varsigma_y = 2\sigma_y^{\,2} l_y^{\,2}\left\{\frac{3}{2}\left[\frac{l_y^{\,2}}{L^2}-\frac{l_y}{L}\left(1+\frac{l_y}{L}\right)e^{\frac{L}{l_y}}\right]-\frac{1}{2}\left[Ei\left(-\frac{L}{l_y}\right)-In\frac{L}{l_y}\right]-\frac{3}{4}+\frac{1}{2}E\right\} \qquad \text{Equation 6.3}$$

Where ς_x is the second spatial moment of the mean concentration in the x direction, l_y is the integral scale, L is the mean travel distance of the centroid, E = 0.577...is the Euler number, *Ei* is the exponential integral.

In order to compare with these results the mean concentration field has been integrated over the flow field to obtain the second spatial moments.

The two major differences between the assumptions that Dagan (1982b) has used and those adopted here are that Dagan uses a point source, while here a Gaussian source is adopted, and that Dagan uses an infinite flow field, while here a finite flow field is adopted. The Gaussian source plume is adopted to reduce the oscillations seen in Figure 6.2. The finite flow field is a result of using a finite element method and, in addition, no real world aquifer is infinite in extent.

To reconcile the difference in initial plumes, the initial spatial moment of the concentration has been subtracted from the final spatial moments of the mean concentration. The alternative of determining the time when Dagan's result gives the initial plume used here and then determining Dagan's result after an additional period of time is infeasible as the times for the initial plume would be different for the x and y directions. Also Figure 3 in Dagan (1982b) shows that ς_x and ς_y are not linear functions of time so the alternative method would introduce inaccuracies anyway. However, elsewhere in the paper Dagan assumes that the plume is Gaussian as it disperses. This makes it reasonable to assume that the difference between the second moment of the result of using an initial Gaussian source and the second moment of the result of using an initial point source is the second moment of the initial Gaussian source. Indeed, this is the case in the deterministic advection dispersion equation.

It is not possible to reconcile the difference in boundary conditions.

The results can be seen in Figure 6.9 and Figure 6.10.

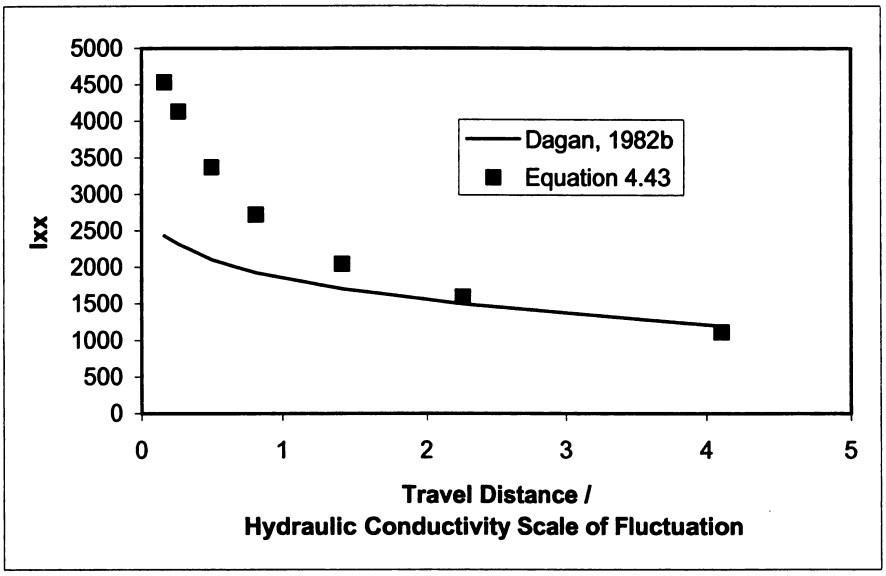


Figure 6.9 Comparison of second moments of the mean concentration plume along the flow.

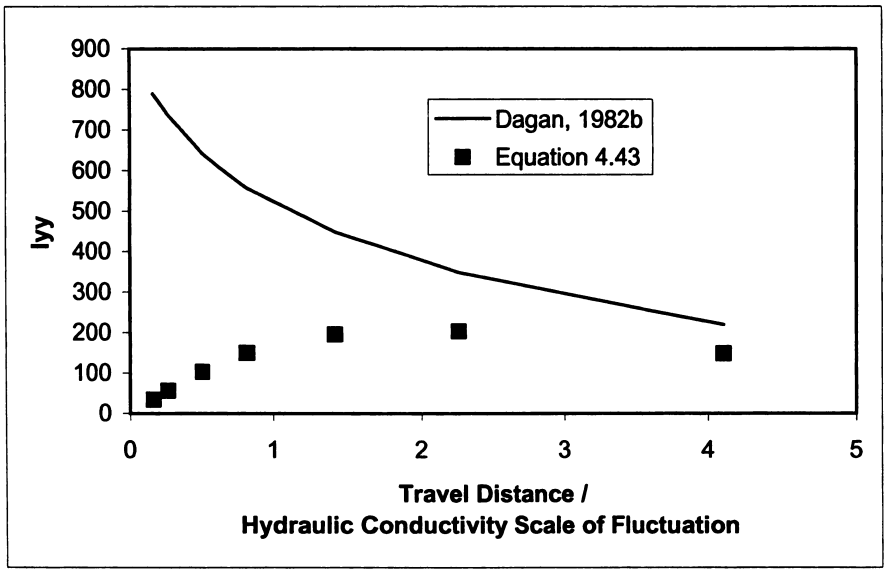


Figure 6.10 Comparison of second moments of the mean concentration plume across the flow.

The x-axis in Figure 6.9 and Figure 6.10 is the travel distance/hydraulic integral scale as this is the major parameter in Equation 6.2 and Equation 6.3. It must be remembered

that the travel distance is more or less constant in the results used here and the integral scale changes. This is important because the integral scale to flow field ratio is therefore also changing.

Figure 6.9 shows the results for the second moment in the direction that the plume is travelling. It shows good agreement for the larger travel distance to hydraulic conductivity integral scale ratios, but poor agreement for the smaller ones. The area of poor agreement corresponds with the area where the ratio between the integral scale and the size of the flow field is larger. As discussed in 6.1.1.5 for these larger integral scales it is more appropriate to de-trend the data and model the trend as the mean conductivity thus obtaining a random field with a much smaller integral scale. It is unknown what effect the boundary conditions have on this result.

Figure 6.10 shows the results for the second moment in the direction perpendicular to the one that the plume is travelling. The agreement is poorer than in Figure 6.9, but still it shows greater reliability for the smaller integral scales. In this case the effects of the boundary conditions are much easier to see. The results have a finite width to the flow field. This limits the ability of the solute to spread transversely to the direction of travel. Therefore even for the smaller integral scales the second moment is considerably smaller than in Dagan's infinite flow field case. Thus the difference probably reflects the current model's ability to model finite boundary conditions.

6.1.2.2 Spatial Second Moments of Individual Plumes

The increase in the spatial second moments in the previous section is a result of two effects. Firstly the plume disperses as it travels. Secondly the plume as a whole meanders as it moves through the aquifer. This second effect occurs when the plume

preferentially flows through or around a sufficiently large area of high or low hydraulic conductivity respectively and can be represented by the movement of the centroid of the plume.

This relationship is commonly expressed as (Rajaram and Gelhar, 1993a):

$$M_{ij}(t) = \Sigma_{ij}(t) + R_{ij}(t) \qquad \text{Equation 6.4}$$

where

M_{ij} is the spatial second moment of the ensemble average concentration, or the size of the mean plume,

Σ_{ij} is the average spatial second moment of the plume, or the mean size of the plume, and

R_{ij} is the spatial second moment of the movement of the centroid of the plume.

The ensemble average concept relates to the Monte Carlo simulation viewpoint. To obtain M_{ij} in a Monte Carlo simulation, first determine the mean concentration at each point, and then determine the spatial second moment of these values. On the other hand to determine Σ_{ij} first determine the spatial second moment for each realisation in the simulation and then determine the mean of these. In a tracer test the spread of the plume measured in the field should be much closer to Σ_{ij} than to M_{ij} since M_{ij} includes movement of the whole plume (R_{ij}) as well as the spreading of the plume.

Rajaram and Gelhar (1993a) discuss the difference between M_{ij} and Σ_{ij} as being related to the motion of one or two particles. M_{ij} is related to the motion of a single particle that could move anywhere in the aquifer depending upon the aquifer characteristics. Σ_{ij} on

the other hand is related to the distance between two particles released into the aquifer. Whilst the two particles could travel anywhere, there will be a tendency for the movement of the particles to be correlated, depending upon the separation of the particles.

Rajaram and Gelhar (1993b) show that

$$\Sigma_{ij}(t) = \frac{n^2}{2m^2} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (x_{1i} - x_{2i})(x_{1j} - x_{2j}) \mathrm{E}[C(x_1,t)C(x_2,t)] d\mathbf{x}_1 d\mathbf{x}_2 \qquad \text{Equation 6.5}$$

These values were calculated in the code. The calculations for Equation 6.5 require an algorithm that is $O(n^6)$. This increased computation time significantly and so the results for Σ_{ij} are based on a mesh of 60 by 30 elements instead of 80 by 40.

Dagan (1991) shows that for a narrow plume aligned along the flow Σ_{ij} grows initially and then remains more or less constant. This is because once the tail of the plume reaches the point where the front started, all of the plume flows through the same flow path. On the other hand a plume that has significant width across the flow disperses because different parts of the plume flow through different parts of the aquifer. Dagan (1994a) gives charts based on analytical results for calculating the dispersion of a thin plume orientated across the flow field.

Dagan (1994a) adopts a finite sized uniform concentration initial plume. As mentioned previously the current work uses a Gaussian initial plume to avoid oscillations. Thus, in order to compare the present results with Dagan (1994a) it has been assumed that the initial width of plume in the present work is twice the square root of the initial spatial second moment of the plume as this implies that the two plumes have the same solute mass. The results are shown in Figure 6.11. Dagan presents his results in non-

dimensional form by dividing the longitudinal dispersion by the variance of the log of the hydraulic conductivity, and by the average groundwater velocity and by the integral scale, and the travel distance by the integral scale. Figure 6.11 uses the same axes

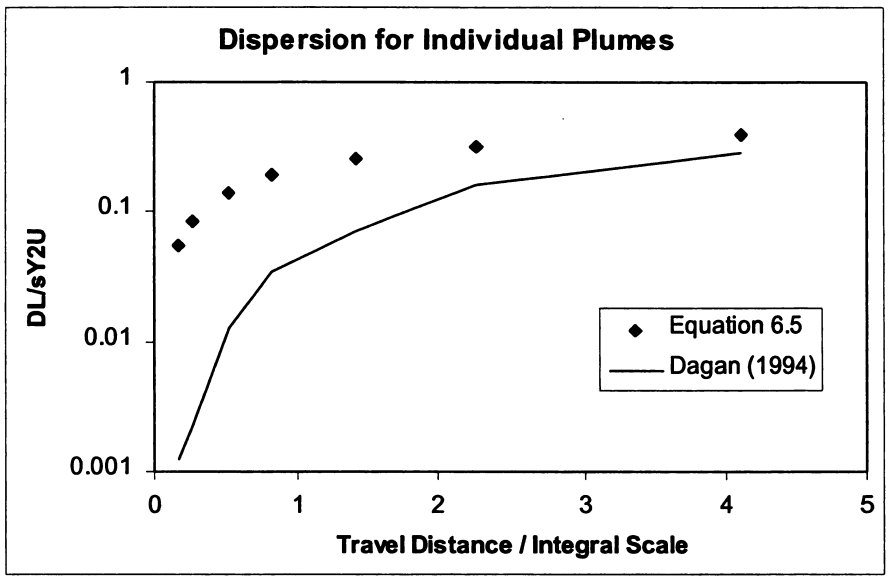


Figure 6.11 Comparison of average longitudinal spatial second moments of the concentration plume.

The results differ in magnitude. This is probably because of the two major differences between the two cases, namely Dagan (1994a) has infinite boundaries and a thin initial plume of constant concentration oriented across the flow, while my results are based on an aquifer with boundaries and an initial plume that is circular with a Gaussian concentration distribution. However, the basic trend of the results is very similar and the poorest fit is in the area of large integral scales where earlier it was suggested that in these cases it would be better to model the dominant variation as a trend rather than a random field.

In particular Zhang et al (1996) investigates the difference between a line source and a cube source in three dimensions. Comparison of their Figures 2 and 3 (given here as Figure 6.12 and Figure 6.13) shows that the effect of the length of the initial plume is to

increase the growth of the second moment ($X_{11} - R_{11}$ is larger in 3a than in 2a for all of the dotted lines). This would help explain why the results of Equation 6.5 in Figure 6.11 are larger than those of Dagan (1994a). Unfortunately Zhang et al (1996) does not present two dimensional results and so a direct comparison is not possible.

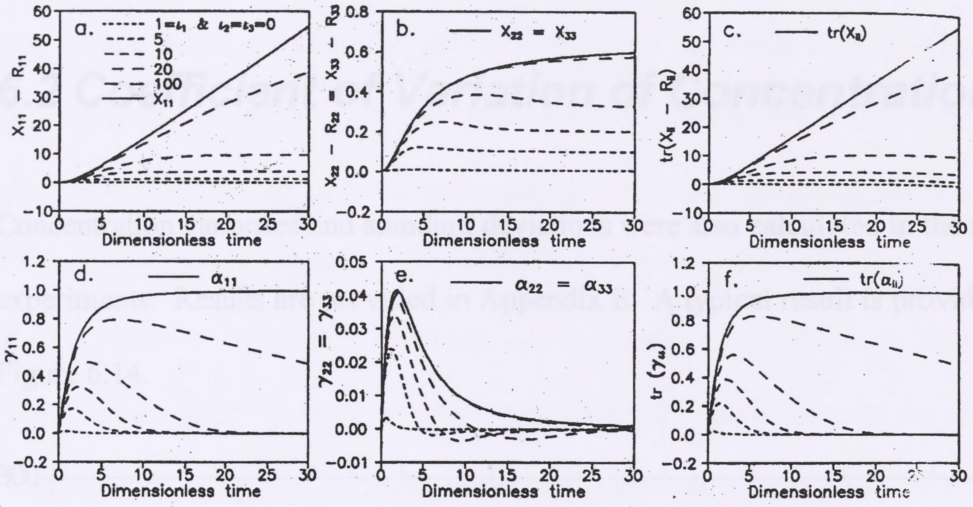


Figure 2. Line sources parallel to mean flow: (a) $X_{11} - R_{11}$, (b) $X_{22} - R_{22} = X_{33} - R_{33}$, (c) trace of $X_{ii} - R_{ii}$, (d) γ_{11}, (e) $\gamma_{22} = \gamma_{33}$, and (f) trace of γ_{ii}.

Figure 6.12 Figure 2 from Zhang et al (1996) – line sources.

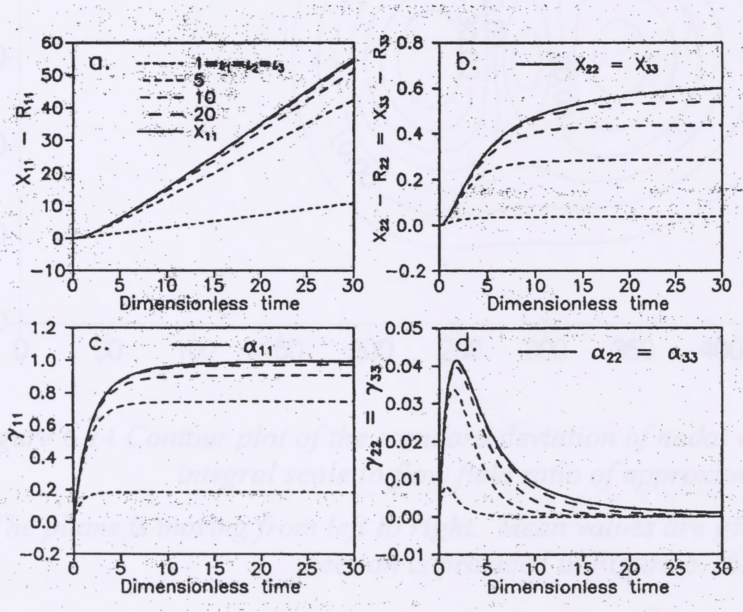


Figure 3. Cube sources: (a) $X_{11} - R_{11}$, (b) $X_{22} - R_{22} = X_{33} - R_{33}$, (c) γ_{11}, and (d) $\gamma_{22} = \gamma_{33}$.

Figure 6.13 Figure 3 from Zhang et al (1996) – cube sources.

Note that this trend is the opposite of that for M_{ij}. At a given travel distance after the plume is released M_{ij} is larger in aquifers with longer integral scales, while Σ_{ij} is smaller. Thus the larger M_{ij} in aquifers with longer integral scales is mainly a result of the uncertainty of where the plume is moving, rather than as a result of the plume dispersing.

6.2 Coefficient of Variation of Concentrations

Concentration variances and standard deviations were also calculated in the numerical experiments. Results are provided in Appendix E. A typical result is provided in Figure 6.14.

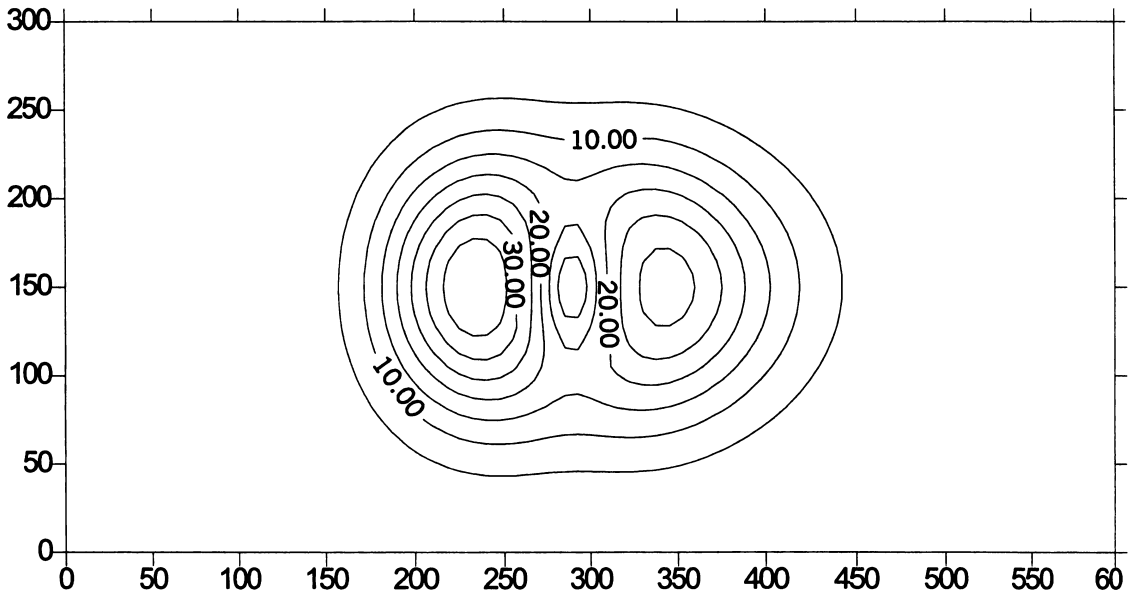


Figure 6.14 Contour plot of the standard deviation of nodal concentration values for an integral scale to flow field ratio of approximately 0.06.

The plume is moving from left to right. Mean values are given in Figure 6.3. A cross section is provided in Figure 6.16.

Typical profiles of the standard deviation show two peaks, one before the main plume and one after. The smaller integral scales even show a depression in the centre of the

plume. Unlike the bimodal effect noted in the mean concentration profiles this is not an error. The variance of the concentration is partially a function of the gradient of the mean concentration. If the solute concentration was completely uniform in an aquifer then entropy would prevent any change in the concentration anywhere and so the variance would be zero. In order for there to be variability in the concentration there must be points in the aquifer that have different concentrations from which particles have a probability of being advected to the location with the variability. This effect is going to be greatest where the concentration gradient is steepest because that is where there are points in near proximity to each other with very different concentrations. On the other hand in the centre of the plume the concentration profile tends to be flat and therefore there is not as much scope for the concentration variances to develop (the initial conditions are that all points have zero concentration variance).

Limited work has focused on the variance of the concentration and even less provides analytical solutions. The most in depth study appears to be Kapoor and Gelhar (1994a&b). Kapoor and Gelhar (1994b) give analytical expressions for the coefficient of variation both with and without local dispersion. The expression that they provide for the coefficient of variation in the no local dispersion case is:

$$\left[\frac{\sigma_C(x,t)}{\overline{C}(x,t)}\right]^2 = \left\{\prod_{i=1}^{N} \frac{(t+t_{0i})}{[t_{0i}(2t+t_{0i})]^{1/2}} \cdot \exp\left[\frac{2t}{2t^2+3tt_0+t_{0i}^2} \cdot \frac{(x_i - v\delta_{i1}t)^2}{4A_{ii}v}\right]\right\} - 1 \quad \text{Equation 6.6}$$

Where t_{0i} is the amount of time that it would take to attain the initial size of the plume in the i direction starting with a point source, δ_{i1} is one if i = the longitudinal dimension (along the flow) and zero otherwise, and A_{ii} is the macrodispersivity in the i direction.

A comparison of the coefficient of variation derived from the results of Equation 4.36 and Equation 4.37 with the coefficient of variation resulting from Equation 6.6 is shown in Figure 6.15. Only the results for the smallest integral scale has been shown here as the mean results tend to indicate that this is the most accurate result.

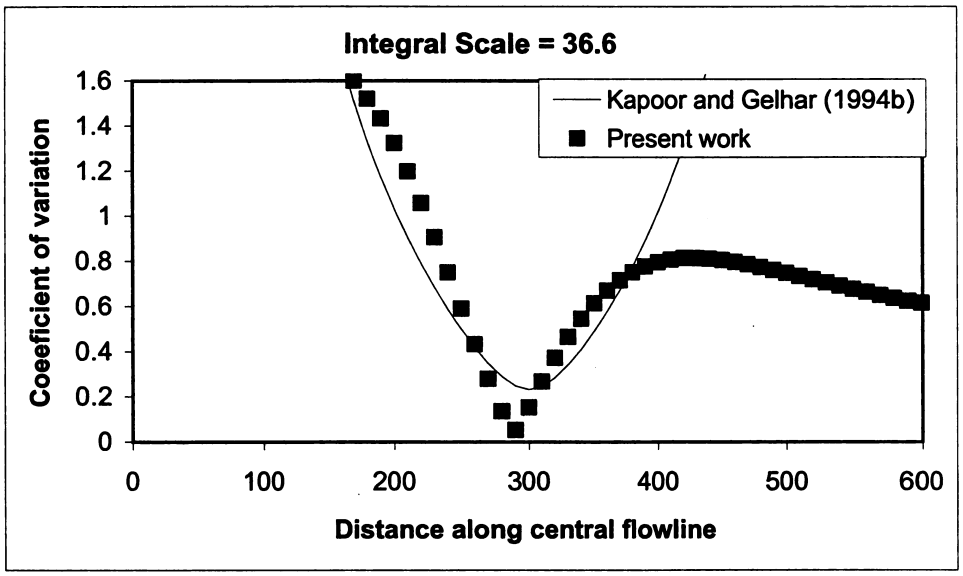


Figure 6.15 Comparison of coefficient of variation of concentration along the plume.

In Figure 6.15 the results from the present work drop off at the downstream end probably because of the downstream boundary condition, which prevents any solute fluctuations returning into the mesh once they drop off the end.

The lower value near the centre of the plume may be due to the no flow side boundary conditions, which reduce the ability of the solute fluctuations to move transversely since the finite element mesh effectively acts as a channel confining the plume, whereas Kapoor and Gelhar (1994b) is based on an unbounded domain. However, t_{02} (the theoretical time that it takes for the transverse size of the plume to reach its initial size starting from a point source) is much larger than t_{01} (the same for longitudinal size), and this disparity may contain this information. Alternatively perhaps the reason for the

lower value near the centre of the plume is because the second derivative for the mean stochastic result is flatter than the top of a Gaussian curve (comparing the second derivatives using a numerical approximation), which is what Kapoor and Gelhar (1994b) is based on.

Figure 6.16 shows the result of converting Kapoor and Gelhar's (1994b) coefficient of variation expression into a standard deviation using the mean concentrations in the present work. (They do not give an expression for standard deviation without local dispersion. The expression that they give with local dispersion has a singularity when the local dispersion is zero.)

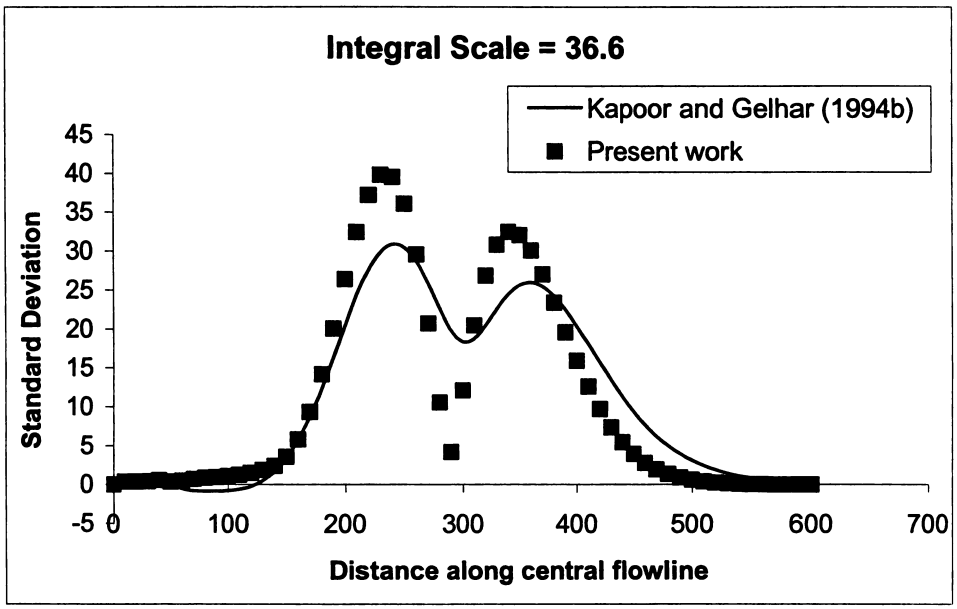


Figure 6.16 Comparison of standard deviation of concentration along the plume.

In Figure 6.16 whilst the curves are not identical they do exhibit the same general shape and the maxima and minimum occur at approximately the same distances along the plume. The fact that both exhibit a larger upstream maximum and that the two maxima are proportional is significant because whilst the continuous line has used the mean

concentration results from this study they are separate from the standard deviation results. It appears that the two methods show good agreement.

7 Further Work

Various avenues are open for extending this work:

- Section 6.1.1.2 suggested a modification to the formulation for concentration. This could be implemented to see if it significantly improves results.
- Higher order moments of the statistics of the velocity and concentration fields could be determined to give more information about their probability distribution.
- The technique of diagonising the covariance matrix can be used to see if it improves computational efficiency.
- Local Averaging can be used to determine the covariance matrix for the hydraulic conductivity to see if convergence is faster.
- The effect of local dispersion can be investigated to compare with effects discovered in analytical solutions.

7.1 Different Formulation for Concentration

Section 6.1.1.2 showed that for larger integral scales the results for the mean concentration plume became increasingly bimodal in profile. Section 6.1.1.3 explained that the cause was the particular formulation for concentration used in Equation 4.36 and suggested that better results would be obtained if instead of using Equation 4.36, a Taylor series centred on the mean velocity was used. This is presented in Equation 7.1.

$$\mathrm{E}[C_n] = \overline{C}_n + \frac{1}{2}\sum_{i,j=1}^{E} \frac{\partial^2 C_n}{\partial v_i \partial v_j} \mathrm{cov}(v_i, v_j) \qquad \text{Equation 7.1}$$

Section 6.1.1.3 also explained that Equation 4.36 was used instead of Equation 7.1 because of computer time constraints. As computers become faster and this becomes less of a problem, this area of further work will become more practical.

7.2 Increasing Accuracy

One of the limitations of the results so far is that they only give information about the first two moments of hydraulic head, velocity and concentration. (Most methods only use the first two moments as in general higher order moments are unknown except in so far as a probability distribution is assumed). This is not enough to determine the probability distributions of these parameters. This can be remedied by extending the work so far to include third and fourth moments. This requires knowledge of the fourth order moments of the log of hydraulic conductivity (the third order moment is zero since the hydraulic conductivity is assumed to be log normally distributed).

$$\mathrm{E}[v_n - \bar{v}_n, v_o - \bar{v}_o, v_p - \bar{v}_p] = \frac{1}{2}\sum_{i,j,l,m=1}^{E}\left(\frac{\partial v_n}{\partial Y_i}\frac{\partial v_o}{\partial Y_j}\frac{\partial^2 v_p}{\partial Y_l \partial Y_m} + \frac{\partial v_n}{\partial Y_i}\frac{\partial^2 v_o}{\partial Y_j \partial Y_l}\frac{\partial v_p}{\partial Y_m} + \frac{\partial^2 v_n}{\partial Y_i \partial Y_j}\frac{\partial v_o}{\partial Y_l}\frac{\partial v_p}{\partial Y_m}\right) \times \mathrm{E}[\Delta Y_i \Delta Y_j \Delta Y_l \Delta Y_m] \qquad \text{Equation 7.2}$$

$$\mathrm{E}[v_n - \bar{v}_n, v_o - \bar{v}_o, v_p - \bar{v}_p, v_q - \bar{v}_q] = \sum_{i,j,k,l=1}^{E} \frac{\partial v_n}{\partial Y_i}\frac{\partial v_o}{\partial Y_j}\frac{\partial v_p}{\partial Y_l}\frac{\partial v_q}{\partial Y_m} \times \mathrm{E}[\Delta Y_i \Delta Y_j \Delta Y_l \Delta Y_m] \qquad \text{Equation 7.3}$$

where:

$$E\left[\Delta Y_i, \Delta Y_j, \Delta Y_l, \Delta Y_m,\right] = \mathrm{cov}\left(Y_i, Y_j\right)\mathrm{cov}\left(Y_l, Y_m\right) + \mathrm{cov}\left(Y_i, Y_l\right)\mathrm{cov}\left(Y_j, Y_m\right) + \mathrm{cov}\left(Y_i, Y_m\right)\mathrm{cov}\left(Y_j, Y_l\right) \quad \text{Equation 7.4}$$

since Y is normally distributed.

Equation 7.2 and Equation 7.3 are Taylor Series (lower order terms are zero). Equation 7.4 is the definition of the fourth moment for a bivariate normal distribution.

Note that Equation 7.3 can be manipulated using Equation 7.4 and Equation 4.28 to give an equation similar to Equation 7.4 but where Y is replaced by v. This does not imply that v is normal. Equation 7.2 and Equation 7.3 are Taylor series and have only limited accuracy. However it does imply that implementing Equation 7.3 will not provide any new information on the probability distribution of v. However Equation 7.2 does provide new information and therefore implementation of it is a potential area of further work as computers become faster and it becomes more practical in terms of computation time.

Another problem is that the results are linear in the covariances and therefore do not adequately represent the behaviour as the variability increases significantly. For example in the problems presented in Section 5.1.2 using the lognormal-k method with a σ_Y of 0.5 and an infinite integral scale would give a result of 0.1125 for the mean of the velocity and 0.05 for the variance, however the correct results in this case are 0.11331 and 0.0603 respectively. Using a fourth order Taylor series would improve the results to 0.11328 and 0.0586. The accuracy of the present results in the case of a finite integral scale is unknown but could be quantified by application of a higher order Taylor Series. It is expected that this will give an idea of the range of σ_k and σ_Y for

which the present Taylor series is accurate in the case of finite integral scale. To use a higher order Taylor series Equation 4.27 and Equation 4.28 needs to be replaced by:

$$\mathrm{E}[v_n] = \bar{v}_n + \frac{1}{2}\sum_{i,j=1}^{E}\frac{\partial^2 v_n}{\partial Y_i \partial Y_j}\mathrm{cov}(Y_i, Y_j) + \frac{1}{24}\sum_{i,j,l,m=1}^{E}\frac{\partial^4 v_n}{\partial Y_i \partial Y_j \partial Y_l \partial Y_m} \times \mathrm{E}[\Delta Y_i \Delta Y_j \Delta Y_l \Delta Y_m] \qquad \text{Equation 7.5}$$

$$\mathrm{cov}[v_n, v_o] = \sum_{i,j=1}^{E}\frac{\partial v_n}{\partial Y_i}\frac{\partial v_o}{\partial Y_j}\mathrm{cov}(Y_i, Y_j) + \frac{1}{4}\sum_{i,j,l,m=1}^{E}\frac{\partial^2 v_n}{\partial Y_i \partial Y_j}\frac{\partial^2 v_o}{\partial Y_l \partial Y_m}\left\{\mathrm{cov}(Y_i, Y_l)\mathrm{cov}(Y_j, Y_m) + \mathrm{cov}(Y_i, Y_m)\mathrm{cov}(Y_j, Y_l)\right\} \qquad \text{Equation 7.6}$$

where the fourth order derivative in Equation 7.5 is given by:

$$\begin{aligned}\frac{\partial^4 v_e}{\partial Y_i \partial Y_j \partial Y_l \partial Y_o} = {} & -\delta_{e,i,j,l,o} k_e \sum_{m=1}^{M}\nabla N_m . H_m - k_e \sum_{m=1}^{M}\nabla N_m . \frac{\partial^4 H_m}{\partial Y_i \partial Y_j \partial Y_l \partial Y_o} \\ & - \delta_{e,j,l,o} k_e \sum_{m=1}^{M}\nabla N_m . \frac{\partial H_m}{\partial Y_i} - \delta_{e,i,l,o} k_e \sum_{m=1}^{M}\nabla N_m . \frac{\partial H_m}{\partial Y_j} \\ & - \delta_{e,i,j,o} k_e \sum_{m=1}^{M}\nabla N_m . \frac{\partial H_m}{\partial Y_l} - \delta_{e,i,j,l} k_e \sum_{m=1}^{M}\nabla N_m . \frac{\partial H_m}{\partial Y_o} \\ & - \delta_{e,l,o} k_e \sum_{m=1}^{M}\nabla N_m . \frac{\partial^2 H_m}{\partial Y_i \partial Y_j} - \delta_{e,j,o} k_e \sum_{m=1}^{M}\nabla N_m . \frac{\partial^2 H_m}{\partial Y_i \partial Y_l} - \delta_{e,j,l} k_e \sum_{m=1}^{M}\nabla N_m . \frac{\partial^2 H_m}{\partial Y_i \partial Y_o} \\ & - \delta_{e,i,o} k_e \sum_{m=1}^{M}\nabla N_m . \frac{\partial^2 H_m}{\partial Y_j \partial Y_l} - \delta_{e,i,l} k_e \sum_{m=1}^{M}\nabla N_m . \frac{\partial^2 H_m}{\partial Y_j \partial Y_o} - \delta_{e,i,j} k_e \sum_{m=1}^{M}\nabla N_m . \frac{\partial^2 H_m}{\partial Y_l \partial Y_o} \\ & - \delta_{e,o} k_e \sum_{m=1}^{M}\nabla N_m . \frac{\partial^3 H_m}{\partial Y_i \partial Y_j \partial Y_l} - \delta_{e,l} k_e \sum_{m=1}^{M}\nabla N_m . \frac{\partial^3 H_m}{\partial Y_i \partial Y_j \partial Y_o} \\ & - \delta_{e,j} k_e \sum_{m=1}^{M}\nabla N_m . \frac{\partial^3 H_m}{\partial Y_i \partial Y_l \partial Y_o} - \delta_{e,i} k_e \sum_{m=1}^{M}\nabla N_m . \frac{\partial^3 H_m}{\partial Y_j \partial Y_l \partial Y_o}\end{aligned} \qquad \text{Equation 7.7}$$

and the third and fourth order derivatives in Equation 7.7 are given by:

$$\begin{aligned}\overline{\mathbf{K}}\frac{\partial^3 \mathbf{H}}{\partial Y_i \partial Y_j \partial Y_l} = {} & -\delta_{i,j,l}\mathbf{K}^i\overline{\mathbf{H}} - \delta_{i,j}\mathbf{K}^i\frac{\partial \mathbf{H}}{\partial Y_l} - \delta_{i,l}\mathbf{K}^i\frac{\partial \mathbf{H}}{\partial Y_j} - \delta_{j,l}\mathbf{K}^j\frac{\partial \mathbf{H}}{\partial Y_i} \\ & - \mathbf{K}^i\frac{\partial^2 \mathbf{H}}{\partial Y_j \partial Y_l} - \mathbf{K}^j\frac{\partial^2 \mathbf{H}}{\partial Y_i \partial Y_l} - \mathbf{K}^l\frac{\partial^2 \mathbf{H}}{\partial Y_i \partial Y_j}\end{aligned} \qquad \text{Equation 7.8}$$

$$\overline{\mathbf{K}}\frac{\partial^4\mathbf{H}}{\partial Y_i\partial Y_j\partial Y_l} = -\delta_{i,j,l,m}\mathbf{K}^i\overline{\mathbf{H}} - \delta_{i,j,l}\mathbf{K}^i\frac{\partial\mathbf{H}}{\partial Y_m} - \delta_{i,j,m}\mathbf{K}^i\frac{\partial\mathbf{H}}{\partial Y_l} - \delta_{i,l,m}\mathbf{K}^i\frac{\partial\mathbf{H}}{\partial Y_j} - \delta_{j,l,m}\mathbf{K}^j\frac{\partial\mathbf{H}}{\partial Y_i}$$
$$- \delta_{i,j}\mathbf{K}^i\frac{\partial^2\mathbf{H}}{\partial Y_l\partial Y_m} - \delta_{i,l}\mathbf{K}^i\frac{\partial^2\mathbf{H}}{\partial Y_j\partial Y_m} - \delta_{i,m}\mathbf{K}^i\frac{\partial^2\mathbf{H}}{\partial Y_j\partial Y_l} \qquad \text{Equation 7.9}$$
$$- \delta_{j,l}\mathbf{K}^i\frac{\partial^2\mathbf{H}}{\partial Y_i\partial Y_m} - \delta_{j,m}\mathbf{K}^i\frac{\partial^2\mathbf{H}}{\partial Y_i\partial Y_l} - \delta_{l,m}\mathbf{K}^i\frac{\partial^2\mathbf{H}}{\partial Y_i\partial Y_j}$$
$$- \mathbf{K}^i\frac{\partial^3\mathbf{H}}{\partial Y_j\partial Y_l\partial Y_m} - \mathbf{K}^j\frac{\partial^3\mathbf{H}}{\partial Y_i\partial Y_l\partial Y_m} - \mathbf{K}^l\frac{\partial^3\mathbf{H}}{\partial Y_i\partial Y_j\partial Y_m} - \mathbf{K}^m\frac{\partial^3\mathbf{H}}{\partial Y_i\partial Y_j\partial Y_l}$$

A simple way to determine if the velocity distribution is lognormal is to calculate:

$$\mathrm{E}\left[\ln\left(v_n-\bar{v}_n\right),\ln\left(v_o-\bar{v}_o\right),\ln\left(v_p-\bar{v}_p\right)\right]$$
$$= \frac{1}{2}\left\{\begin{array}{l} \displaystyle\sum_{i,j,l,m=1}^{E}\frac{\partial v_n}{\partial Y_i}\frac{\partial v_o}{\partial Y_j}\left(\frac{\partial^2 v_p}{\partial Y_l\partial Y_m} - \frac{\dfrac{\partial v_p}{\partial Y_l}\dfrac{\partial v_p}{\partial Y_m}}{\bar{v}_p}\right) \\ \displaystyle+\sum_{i,j,l,m=1}^{E}\frac{\partial v_n}{\partial Y_i}\left(\frac{\partial^2 v_o}{\partial Y_j\partial Y_l} - \frac{\dfrac{\partial v_o}{\partial Y_j}\dfrac{\partial v_o}{\partial Y_l}}{\bar{v}_o}\right)\frac{\partial v_p}{\partial Y_m} \\ \displaystyle+\sum_{i,j,l,m=1}^{E}\left(\frac{\partial^2 v_n}{\partial Y_i\partial Y_j} - \frac{\dfrac{\partial v_n}{\partial Y_i}\dfrac{\partial v_n}{\partial Y_j}}{\bar{v}_n}\right)\frac{\partial v_o}{\partial Y_l}\frac{\partial v_p}{\partial Y_m} \end{array}\right\} \times \frac{\mathrm{E}\left[\Delta Y_i\Delta Y_j\Delta Y_l\Delta Y_m\right]}{\bar{v}_n\bar{v}_o\bar{v}_p} \qquad \text{Equation 7.10}$$

Equation 7.10 will be zero if the groundwater velocity probability density function is log normally distributed.

If it is found that the velocity is log normally distributed in the case of a finite integral scale as it is in the case of an infinite integral scale then the method can be redeveloped to use derivatives of log velocity with respect to log conductivity which would greatly increase the accuracy even with the level of truncation presently used. In this case:

$$E\left[\ln\left(v_n\right)\right] = \ln\left(\bar{v}_n\right) + \frac{1}{2}\sum_{i,j=1}^{E} \frac{\bar{v}_n \dfrac{\partial^2 v_n}{\partial Y_i \partial Y_j} - \dfrac{\partial v_n}{\partial Y_i}\dfrac{\partial v_n}{\partial Y_j}}{\bar{v}_n^2} \operatorname{cov}\left(Y_i, Y_j\right) \qquad \text{Equation 7.11}$$

$$\operatorname{cov}\left[\ln\left(v_n\right), \ln\left(v_o\right)\right] = \sum_{i,j=1}^{E} \frac{\partial v_n}{\partial Y_i}\frac{\partial v_o}{\partial Y_j}\frac{\operatorname{cov}\left(Y_i, Y_j\right)}{\bar{v}_n \bar{v}_o} \qquad \text{Equation 7.12}$$

which can easily be calculated using the results from Equation 4.27 and Equation 4.28.

Similar extensions can be applied to the determination of concentration from groundwater velocity.

7.3 Diagonalisation of Covariance Matrix

Liu et al (1987) show how diagonalising the input covariance matrix can reduce the calculations of the output covariance matrix. To illustrate this Equation 4.28 can be expressed in matrix form as

$$\mathbf{C}_v = \mathbf{M}\mathbf{C}_Y\mathbf{M}^T \qquad \text{Equation 7.13}$$

Where $\mathbf{C}_v$ and $\mathbf{C}_Y$ are the velocity and log hydraulic conductivity covariance matrices and M is the matrix whose elements are given by $m_{ij} = \left(\dfrac{\partial v_i}{\partial Y_j}\right)$.

A diagonal matrix **D** is obtained by solving the eigen problem:

$$\mathbf{C}_Y\mathbf{P} = \mathbf{P}\mathbf{D} \qquad \text{Equation 7.14}$$

Where **P** is a square matrix the same size as $\mathbf{C}_Y$ with the properties that

$$\mathbf{P^T P = P P^T = 1} \qquad \text{Equation 7.15}$$

$$\mathbf{P^T C_Y P = D} \qquad \text{Equation 7.16}$$

Using Equation 7.15 and Equation 7.16, Equation 7.13 can be transformed into:

$$\mathbf{C_v = MPDP^T M^T} \qquad \text{Equation 7.17}$$

Which is the same as

$$\mathbf{C_v = MPD(MP)^T} \qquad \text{Equation 7.18}$$

Which can be rewritten as

$$\operatorname{cov}(v_m, v_n) = \sum_{i=1}^{E} \left(\frac{\partial v_m}{\partial d_i} \right) \left(\frac{\partial v_n}{\partial d_i} \right) \operatorname{var}(d_i) \qquad \text{Equation 7.19}$$

Where the d_i are the diagonal elements of **D**.

Similarly Liu et al (1987) claim that Equation 4.27 can be written as

$$\mathrm{E}[v_n] = \bar{v}_n + \frac{1}{2} \sum_{i,j=1}^{E} \frac{\partial^2 v_n}{\partial^2 d_i} \operatorname{var}(d_i) \qquad \text{Equation 7.20}$$

Similar reasoning applies to the process of determining the concentrations from the velocities (Equation 4.36 and Equation 4.37).

Section 4.6.1 points out that the computational time for Equation 4.27 and Equation 4.28 is proportional to the fourth power of the number of elements. It would appear that using Equation 7.19 and Equation 7.20 could reduce this to the third power. As the computational time is a major limitation this would be a fruitful area of further work,

particularly to determine if the savings in time outweighs the extra work required to determine the derivatives with respect to the d_i.

7.4 Local Averages

In Section 3.4.4 it was explained that the work presented here uses midpoint discretisation to determine the covariance matrix instead of local averaging, which is the other method that is commonly used. The reason for choosing midpoint discretisation was that macrodispersion of solute is caused by the magnitude and frequency of the random fluctuations in aquifer properties rather than a summation of those properties. Vanmarcke (1983) points out that local averaging has the effect of increasing the integral scale and decreasing the standard deviation depending on the element size, and thus changes the magnitude and frequency of the random fluctuations.

However, Fenton and Griffiths (1993) found that local averaging converges faster than midpoint discretisation when determining effective hydraulic conductivities of aquifers. Effective hydraulic conductivity is a different phenomenon to macrodispersion and depends on the summation of the aquifer properties over a region rather than the size and frequency of the random fluctuations. Therefore the Fenton and Griffiths (1993) result might not be relevant to macrodispersion.

Whilst the argument presented for using midpoint discretisation instead of local averages is plausible it would still be worthwhile doing some comparative modelling to determine which method does converge faster. It may be that the increase in integral scale and decrease in standard deviation interact with other factors relating to the size of the element to make local averaging converge faster than midpoint discretisation. If so,

this would allow better accuracy with finite element meshes containing fewer elements (and thus less computation time).

7.5 Inclusion of Local Dispersion

The results presented in Chapter 6 assume that the local dispersion coefficient derived from diffusion and mechanical dispersion is zero. In nature local dispersion is not zero, it is just very small compared to the macrodispersion. Even so it does perform important functions.

Kapoor and Gelhar (1994a & b) investigate the fluctuations in the concentration field. The size of these fluctuations at a point is proportional to the standard deviation of the concentration at that point. They show that the processes that cause macrodispersion create these fluctuations, for example as a preferential flow path carries solute into a solute free area it creates a fluctuation or difference between the high concentration in the flow path and the much lower (or zero) concentration outside it. They also show that the only process that can destroy these fluctuations is local dispersion, for example the solute will diffuse and/or mechanically disperse from the area of high concentration to low concentration. They give analytical expressions for this creation and destruction.

A useful area of further work would be to add a finite amount of local dispersion to determine the result and compare with Kapoor and Gelhar (1994a & b). The code in Appendix A already has the capability of modelling the effect of local dispersion. However, it was not done due to the large amount of calculation time required.

7.6 Moore's Law

Moore (1965) observed that the most economical number of components on a silicon chip increases exponentially in time and the cost of those components correspondingly decreases. This has led to extraordinary growth in the speed of computers in the last 4 decades. Many of the ideas presented in Sections 7.1 – 7.5 may not have been feasible whilst the research presented in this thesis was carried out because of the execution time required. However, if the exponential growth in the speed of computers continues, this will not always be a hindrance and more of these avenues will be open for further work.

8 Conclusion

This thesis has applied a stochastic numerical perturbation method to the groundwater flow and solute transport problems. Heterogeneity of the hydraulic conductivity field has been modelled using a random field approach. Both normal and log normal hydraulic conductivity distributions have been investigated. The algorithms required for determining the hydraulic heads, groundwater velocities, and solute concentrations were presented as well as algorithms for determining the spatial second moments of overall mean concentration, individual plume size, and solute plume centroid position.

A computer code was written to implement these algorithms.

It was shown how the method could incorporate upstream weighting. Methods of reducing computation time were discussed and in general implemented, such as, the order of performing the computations, use of symmetry, the dropping of small covariance terms, the computing of several fields simultaneously, the use of parallel computation, and whether to save results that are too large for random access memory to disk or recalculate them.

It was found that models based on normal (Gaussian) hydraulic conductivity fields are flawed because typically in this model the stochastic groundwater velocity is less than the deterministic result and that for large variations in hydraulic conductivity this would create negative velocities.

Comparison was made between the lognormal model for determining groundwater velocities and Dykaar and Kitanidis (1992b) and found to produce very good results,

except were mesh resolution became a problem. Anisotropic random fields were also investigated and plausible results obtained. The mechanics of how the perturbation method is able to obtain these results was discussed.

Results of concentration calculations gave reasonable agreement with analytical results for practical scenarios. Differences were discussed and improvements were suggested to obtain a better fit with mean concentration values.

Further areas of work were suggested including extending the analysis to use fourth moments of hydraulic conductivity, applying diagonalisation of the covariance matrix, using local averages, and the inclusion of local dispersion.

The work presented in this thesis builds on previous work in that it uses a stochastic finite element method and so avoids the problems of analytical solutions, namely restricted geometries and limited types of solute source characteristics. The work builds on other research that uses stochastic finite element methods in that it determines the statistics of the groundwater velocity field in the groundwater flow problem, determines the statistics of the concentration field in the solute transport problem given the statistics of the hydraulic conductivity field, and uses a second order method for the mean values, thus avoiding the problem of using the same result as obtained in the deterministic case.

9 References

Ababou, R., D. McLaughlin, Gelhar L. W. and Tompson A. F. B., 1989, "Numerical simulation of three-dimensional saturated flow in randomly heterogeneous porous media", *Transport in Porous Media*, v4, pp549-565.

Anderson, E., Bai, Z. and Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A. and Sorensen, D., 1999, *LAPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA.

Andricevic, R., 1998, "Effects of local dispersion and sampling volume on the evolution of concentration fluctuations in aquifers", *Water Resources Research*, v34, n5, pp1115-1129.

Ashby, S. F. and Falgout, R. D., 1996, "A parallel multigrid preconditioned conjugate gradient algorithm for groundwater flow simulations", *Nuclear Science and Engineering*, v124, pp145-159.

Attinger S., Neuweiler, I. and Kinzelbach W., 2001, "Macrodispersion in a radially diverging flow field with finite Peclet numbers. 1 Homogenization theory approach", *Water Resources Research*, v37, n3, pp495-505.

Bakr, A. A., Gelhar L. W., Gutjahar A. L., and MacMillam J. R., 1978, "Stochastic analysis of spatial variability in subsurface flows, 1, Comparison of one- and three-dimensional flows", *Water Resources Research*, v14 n2, pp263-271.

Barry, D. A., Coves, J., and Sposito, G., 1988, "On the Dagan model of solute transport in groundwater: application to the Borden site" *Water Resources Research*, v24 n10, pp1805-1817.

Beckie, R., Aldama A. and Wood E. F., 1994, "The universal structure of the groundwater flow equations", *Water Resources Research*, v30 n5, pp1407-1419.

Beckie, R., Aldama A. and Wood E. F., 1996, "Modeling the large-scale dynamics of saturated groundwater flow using spatial-filtering theory: 1. Theoretical development", *Water Resources Research*, v32 n5, pp1269-1280.

Beckie, R., 1996, "Measurement scale, network sampling scale, and groundwater model parameters", *Water Resources Research*, v32, n1, pp65-76.

Bellin, A. and Rubin, Y., 1994, "Eulerian-Lagranian approach for modeling of flow and transport in heterogeneous geological formations", *Water Resources Research*, v30, n11, pp2913-2924.

Bellin, A., Salandin P. and Rinaldo A., 1992, "Simulation of dispersion in heterogeneous porous formations: statistics, first-order theories, convergence of computations", *Water Resources Research*, v28, n9, pp2211-2227.

Burr, D.T., Sudicky, E.A. and Naff, R.L., 1994, "Nonreactive and reactive solute transport in three dimensional heterogeneous porous media: mean displacement, plume spreading, and uncertainty", *Water Resources Research*, v30, n3, pp731-815.

Chan, T. P. and Govindaraju, R. S., "Interval computing method for analysising field-scale solute transport", *Journal of Hydrologic Engineering,* Nov, pp480-489.

Chowdhury, R. N. and Zhang, S., 1988, "Prediction of Critical Slip Surfaces", *Fifth Australia-New Zealand Conference on Geomechanics*, pp451-455.

Crane, M. J. and Blunt, M..J., 1999, "Streamline-based simulation of solute transport", *Water Resources Research*, v35, n10, pp3061-3078.

Cvetkovic, V., Shapiro, A. M., and Dagan, G., 1992, "A solute flux approach to transport in heterogeneous formations 1. Uncertainty analysis", *Water Resources Research*, v28, n5, pp1377-1388.

Dagan, G., 1979, "Models of groundwater flow in statistically homogeneous porous formations", *Water Resources Research*, v15, n1, pp47-63.

Dagan, G., 1982a, "Stochastic modelling of groundwater flow by unconditional and conditional probabilities 1. Conditional simulation and direct problem", *Water Resources Research*, v18, n4, pp813-833.

Dagan, G., 1982b, "Stochastic modelling of groundwater flow by unconditional and conditional probabilities 2. The solute transport", *Water Resources Research*, v18, n4, pp835-848.

Dagan, G., 1984, "Solute transport in heterogeneous porous formations", *Journal of Fluid Mechanics*, v145, pp151-177.

Dagan, G., 1985, "A note on higher-order corrections of the head covariances in steady aquifer flow", *Water Resources Research*, v21, n4, pp573-578.

Dagan, G., 1989, *Flow and Transport in Porous Formations*, Springer-Verlag, Berlin.

Dagan, G., 1990, "Transport in heterogeneous porous formations: spatial moments, ergodicity and effective dispersion ", *Water Resources Research*, v26, n6, pp1281-1290.

Dagan, G., 1991, "Dispersion of a passive solute in non-ergodic transport by steady velocity fields in heterogeneous formations", *Journal of Fluid Mechanics,* v233, pp197-210.

Dagan, G., 1994a, "Transport by two-dimensional random velocity fields: effective dispersion coefficients of a finite plume", *Stochastic and statistical methods in hydrology and environmental engineering,* v2, pp113-126.

Dagan, G., 1994b, "The significance of heterogeneity of evolving scales to transport in porous formations", *Water Resources Research*, v30, n12, pp3327-3336.

Dagan, G., 1994c, "An exact nonlinear correction to transverse macrodispersivity for transport in heterogeneous formations", *Water Resources Research*, v30, n10, pp2699-2705.

Dagan, G., Bellin, A. and Rubin, Y., 1996, "Lagrangian analysis of transport in heterogeneous formations under transient flow", *Water Resources Research*, v28, n5, pp1369-1376.

Dagan, G., Cvetkovic, V. and Shapiro, A., 1992, "A solute flux approach to transport in heterogeneous formations 1. The general framework", *Water Resources Research*, v28, n5, pp1369-1376.

Dagan, G. and Nguyen, V., 1989, "A comparison of travel time and concentration approaches to modeling transport by groundwater", *Journal of Contaminant Hydrology,* v4, pp79-91.

Dentz, M., Kinzelbach, H., Attinger, S. and Kinzelbach, H, 2000, "Temporal behaviour of a solute cloud in a heterogeneous porous medium 1. Point-like injection" *Water Resources Research*, v36, n12, pp3591-3604.

Dentz, M., Kinzelbach, H., Attinger, S. and Kinzelbach, H, 2000, "Temporal behaviour of a solute cloud in a heterogeneous porous medium 1. Spatially extended injection" *Water Resources Research*, v36, n12, pp3605-3614.

Deodatis, G., 1991, "Weighted integral method. 1: stochastic stiffness matrix", *Journal of Engineering Mechanics,* v117, n8, pp1851-1864.

Deodatis, G. and Sinozuka M., 1991a, "The weighted integral method for calculating the stochastic stiffness matrix of stochastic systems", *Mechanics Computing in the 1990's and Beyond*, ASCE, New York, pp183-187.

Deodatis, G. and Sinozuka M., 1991b, "Response variability and reliability of stochastic systems using the weighted integral method", *Mechanics Computing in the 1990's and Beyond*, ASCE, New York, pp263-267.

Deodatis, G. and Sinozuka M., 1991c, "Weighted integral method. II: Response Variability and reliability", *Journal of Engineering Mechanics,* v117, n8, pp1865-1877.

Der Kiureghan, A. and De Stefano M., 1991, "An efficient algorithm for second-order finite element reliability analysis", *Mechanics in the 1990's and beyond,* v1, Hojjat Adeli and Robert L Sierakowski (eds), ASCE, New York pp248-252.

Der Kiureghan, A. and. Ke J., 1988, " The finite element in structural reliability", *Probabilistic Engineering Mechanics,* v3, n2, pp83-91.

Destouni, G. and Graham, W., 1995, "Solute transport through an integrated heterogeneous soil-groundwater system" *Water Resources Research*, v31, n8, pp1935-1944.

Dettinger, M. D. and Wilson, J. L., 1981, "First order analysis of uncertainty in numerical models of groundwater flow. Part 1. Mathematical models" *Water Resources Research*, v17, n1, pp149-161.

Dietrich, C. R. and Newsam, G. N., 1993, "A fast and exact method for multidimensional gaussian stochastic simulations" *Water Resources Research*, v29, n8, pp2861-2869.

Di Fredrico, V. and Zhang, Y., 1999, "Solute transport in heterogeneous porous media with long-range correlations", *Water Resources Research*, v35, n10, pp3185-3191.

Dykaar, B. B., and Kitanidis P. K., 1992a, "Determination of the effective hydraulic conductivity for heterogeneous porous media using a numerical spectral approach: 1 method." *Water Resources Research*, v28 n4, pp1155-1166.

Dykaar, B. B., and Kitanidis P. K., 1992b, "Determination of the effective hydraulic conductivity for heterogeneous porous media using a numerical spectral approach: 2 results." *Water Resources Research,* v28 n4, pp1167-1178.

Dykaar, B. B., and Kitanidis P. K., 1993, "Tranmissivity of a heterogeneous formation" *Water Resources Research*, v29 n4, pp985-1001.

El-Kadi, A. I. And Williams, S. A., 2000, "Generating two-dimensional fields of autocorrelated, normally distributed parameters by the matrix decomposition technique", *Ground Water,* v38, n4, pp530-523.

Fenton, G.A. and Griffiths, G.A., 1993, " Statistics of block conductivity through a simple bounded stochastic medium" *Resources Research*, v29 n6, pp1825-1830.

Feyen, L., Beven, K. J., De Smedt, F. and Freer, J., 2001, "Stochastic capture zone delineation within the generalized likelihood uncertainty estimation methodology: Conditioning on head observations", *Water Resources Research*, v37, n3, pp625-638.

Fiori, A., 2001a, "On the influence of local dispersion in solute transport through formations with evolving scales of heterogeneity", *Water Resources Research*, v37, n2, pp235-242.

Fiori, A., 2001b, "The Lagrangian concentration approach for determining dilution in aquifer transport: Theoretical analysis and comparison with field experiments", *Water Resources Research*, v37, n12, pp3105-3114.

Fu, G. and Moses F., 1992, "Multimodal stimulation method for system reliability analysis", *Journal of Engineering Mechanics,* v119, n6, pp1173-1179.

Freeze, R. A., 1975, "A stochastic-conceptual analysis of one-dimensional groundwater flow in nonuniform homogeneous media", *Water Resources Research*, v11 n6, pp725-741.

Gelhar, L. W., 1993, *Stochastic Subsurface Hydrology,* Prentice Hall, London.

Gelhar, L. W. and Axness C. L., 1983, "Three-dimensional stochastic analysis of macrodispersion in aquifers", *Water Resources Research*, v19, n1, pp161-180.

Ghanem, R., 1998, "scales of fluctuation and the propagation of uncertainty in random porous media", *Water Resources Research,* v34 n9, pp2123-2136.

Goode, D., 1990, "Particle velocity interpolation in block-centered finite difference groundwater flow models" *Water Resources Research,* v26 n5, pp925-940.

Graham, W. and McLaughlin, D., 1989a, "Stochastic analysis of nonstationary subsurface solute transport. 1. Unconditional moments", *Water Resources Research,* v25 n2, pp215-232.

Graham, W. and McLaughlin, D., 1989b, "Stochastic analysis of nonstationary subsurface solute transport. 2. Conditional moments", *Water Resources Research,* v25 n11, pp2331-2355.

Guadagnini, A. and Neuman, S. P., 1999a, "Nonlocal and localized analyses of conditional mean steady state flow in bounded, randomly nonuniform domains. 1. Theory and computational approach", *Water Resources Research*, v35, n10, pp2999-3018.

Guadagnini, A. and Neuman, S. P., 1999a, "Nonlocal and localized analyses of conditional mean steady state flow in bounded, randomly nonuniform domains. 1. Computational examples", *Water Resources Research*, v35, n10, pp3019-3039.

Gutjahar, A. L., 1984, "Stochastic models of subsurface flow: log linearized Gaussian models are "exact" for covariances", *Water Resources Research*, v20, n12, pp1909-1912.

Gutjahar, A. L. and Gelhar L. W., 1981, "Stochastic models of subsurface flow: Infinite versus finite domains and stationarity", *Water Resources Research,* v17, n2, pp337-350.

Gutjahar, A. L., Gelhar L. W., Bakr A. A., and MacMillam J. R., 1978, "Stochastic analysis of spatial variability in subsurface flows, 2, evaluation and application", *Water Resources Research*, v14 n5, pp953-959.

Hamed, M. M., 1995, "Probabliistic screening tool for ground-water contamination assessment", *Journal of Environmental Engineering,* Nov, pp767-775.

Hantush, M. M. and Marino, M. A., 1995, "Continuous time stochastic analysis of groundwater flow in heterogeneous aquifers", *Water Resources Research*, v31, n3, pp565-575.

Hassan, A. E., Cushman, J. H. and Delleur, J .W., 1998, "A Monte Carlo assessment of eulerian flow and transport perturbation models" *Water Resources Research*, v34 n5, pp1143-1163.

Hisada, T. and Nakagiri S., 1981, "Stochastic finite element method developed for structural safety and reliability", *Proceedings of the 3rd International Conference Structural Safety & Reliability*, pp395-408.

Hsu, K, and Neuman, S.P., 1997, "Second-order expressions for velocity moments in two and three dimensional statistically anisotropic media" *Water Resources Research*, v33, n4, pp625-637.

Huyakorn, P.S., 1976, *An upwind finite element scheme for improved solution of the convection-diffusion equation*, Dept of Civil Engineering, Princeton University, Princeton.

Huyakorn, P.S. and Pinder G. F., 1983, *Computational methods in subsurface flow*, Academic Press, London.

Indelman, P., Fiori A. and Dagan G., 1986, "Steady flow toward wells in heterogeneous formations: Mean head and equivalent conductivity", *Water Resources Research*, v32, n7, pp1975-1983.

Indelman, P. and Rubin, Y., 1995, "Flow in heterogeneous media displaying a linear trend in the log conductivity", *Water Resources Research*, v31, n5, pp1257-1265.

Istok, J., 1989, *Groundwater modelling by the finite element method*, American Geophysical Union, Washington DC.

Jury, W. A. and Scotter, D. R., 1994, " A unified approach to stochastic-convective transport problems", *Soil Science Society America Journal*, v58, pp1327-1336.

Jussel, P., Stauffer, F. and Dracos, T., 1994a, "Transport modelling in heterogeneous aquifers: 1. statistical description and numerical generation of gravel deposits", *Water Resources Research*, v30, n6, pp1803-1817.

Jussel, P., Stauffer, F. and Dracos, T., 1994b, "Transport modelling in heterogeneous aquifers: 2. Three dimensional transport model and stochastic numerical tracer experiments", *Water Resources Research*, v30, n6, pp1819-1831.

Kapoor, and Gelhar, 1994a, "Transport in three-dimensionally heterogeneous aquifers. 1. Dynamics of concentration fluctuations", *Water Resources Research*, v30, n6, pp1775-1788.

Kapoor, and Gelhar, 1994b, "Transport in three-dimensionally heterogeneous aquifers. 2. Predictions and observations of concentration fluctuations", *Water Resources Research*, v30, n6, pp1789-1801.

Karakas, A. and Kavvas, M. L., 2000, "Conservation equations for ground-water velocity in general conditions", *Journal of Hydrologic Engineering,* April, pp206-215.

Katafygiotis, L. S and Beck J. L., 1995, "A very efficient moment calculation method for uncertain linear dynamic problems", *Probabilistic Engineering Mechanics,* v10, n2, pp117-128.

Kitanidis, P. K., 1994, "The concept of dilution index", *Water Resources Research*, v30, n7, pp2011-2026.

Kunstman, H, Kinzelbach, W, and Siegfied, T, 2002, "Conditional first-order second-moment method and its application to the quantification of uncertainty in groundwater modeling", *Water Resources Research*, v38, n4, pp6.1-6.15.

Lessoff, S. C., Indelman, P. and Dagan G., 2000, "A note on the influence of a constant velocity boundary condition on flow and transport in heterogeneous formations", *Water Resources Research*, v36, n10, pp3095-3101.

Li, S. and McLaughlin D., 1991, "A nonstationary spectral method for solving stochastic groundwater problems: unconditional analysis", *Water Resources Research*, v27 n7, pp1589-1605.

Li, S. and McLaughlin D., 1995, "Using the nonstationary spectral method to analyse flow through heterogeneous trending media", *Water Resources Research*, v31 n3, pp541-551.

Liu, W. K., Belytschko A. and Mani A., 1986a, "Random field finite elements", *International Journal for Numerical Methods in Engineering, v23, pp1831-1845.*

Liu, W. K., Belytschko T. and Mani A., 1986b, "Probabilistic finite elements for nonlinear structural dynamics", *Computer Method in Applied Mechanics and Engineering,* v56, n1, pp61-81.

Liu, W. K., Belytschko T. and Mani A., 1987, "Applications of probabilistic finite elements methods in elastic/plastic dynamics", *Journal of Engineering for Industry,* v109, n1, pp2-8

Liu, W. K., Besterfield, G. and Belytschko, T. 1988, "Transient probabilistic systems", *Computer Methods in Applied Mechanics and Engineering,* v67 pp27-54.

Liu, P. L and Der Kiureghan A., 1991a, "Finite element reliability of geometrically nonlinear structures", *Journal of Engineering Mechanics,* v117, n8, pp1806-1825.

Liu, P. L and Der Kiureghan A., 1991b, "Optimisation algorithms for structural reliability", *Structural Safety,* v9, n3, pp161-177.

Mantoglou, A. and Wilson J. L., 1982, "The turning bands method for simulation of random fields using line generation by a spectral method", *Water Resources Research,* v18, n5, pp1379-1394.

Melchers, R. E., 1990, "Radial importance sampling for structural reliability", *Journal of Engineering Mechanics,* v116, n1, pp189-203.

Meyer, P. D., Valocchi A. J., Ashby S. F. and Saylor, P.E., 1989, "A numerical investigation of the conjugate gradient method as applied to three dimensional groundwater flow problems in randomly heterogeneous porous media", *Water Resources Research,* v25, n6, pp1440-1446.

Moore, G., 1965, "Cramming more components onto integrated circuits", *Electronics*, v38, n8, pp 114-117.

Moreno, L. and Tsang, C., 1994, "Flow channeling in strong heterogeneous porous media: a numerical study", *Water Resources Research*, v30, n5, pp1421-1430.

Naff, R. L., 1992, "Arrival times and temporal moments of breakthrough curves for an imperfectly stratified aquifer", *Water Resources Research*, v28, n1, pp53-68.

Naff, R.L., 1994, "An Eulerian scheme for second order approximation of subsurface transport moments", *Water Resources Research*, v30, n5, pp1439-1455.

Naff, R.L., Haley, D.F. and Sudicky, E. A., 1998, "High-resolution Monte Carlo simulation of flow and conservative transport in heterogeneous porous media 1. Methodology and flow results", *Water Resources Research*, v34, n4, pp663-677.

Naff, R.L., Haley, D.F. and Sudicky, E. A., 1998, "High-resolution Monte Carlo simulation of flow and conservative transport in heterogeneous porous media 1. Transport results", *Water Resources Research*, v34, n4, pp679-697.

Nakagiri, S., 1987, "Fluctuation of structural response, why and how", *JSME International Journal,* v30, n261, pp369-374.

Neuman, S.P., 1993, "Eulerian- Lagrangian theory of transport in space time nonstationary velocity fields: exact nonlocal formalism by conditional moments and weak approximation" *Water Resources Research*, v29, n3, pp633-645.

Neuman, S.P. and Orr, S., 1993, "Prediction of a steady state flow in nonuniform geologic media by conditional moments: exact nonlocal formalism, effective

conductivities, and weak approximation." *Water Resources Research*, v29, n2, pp341-364.

Neuman, S.P., Winter C. L. and Newman C. M., 1987, "Stochastic theory of field-scale Fickina Dispersion in anisotropic porous media" *Water Resources Research*, v29, n3, pp633-645.

Neuman, S.P. and Yakowitz, S., 1979, "A statistical approach to the inverse problem of aquifer hydrology 1. Theory." *Water Resources Research*, v15, n4, pp845-860.

Neuweiler, I., Attinger S. and Kinzelbach W., 2001, "Macrodispersion in a radially diverging flow field with finite Peclet numbers. 1 Perturbation theory approach", *Water Resources Research*, v37, n3, pp481-493.

Paleologos, E. K., Neuman S. P. and Tartakovsky D., 1996, "Effective hydraulic conductivity of bounded, strongly hetrogeneous porous media", *Water Resources Research*, v23, n5, pp1333-1341.

Rajaram, H. and Gelhar, L.W., 1993a, "Plume scale-dependent dispersion in heterogeneous aquifers. 1. Lagrangian analysis in a stratified aquifer", *Water Resources Research*, v29, n9, pp3249-3260.

Rajaram, H. and Gelhar, L.W., 1993b, "Plume scale-dependent dispersion in heterogeneous aquifers. 2. Eulerian analysis and three dimensional aquifers", *Water Resources Research*, v29, n9, pp3261-3276.

Rajaram, H. and Gelhar, L.W., 1995, "Plume scale-dependent dispersion in aquifers with a wide range of heterogeneity", *Water Resources Research*, v31, n10, pp2469-2482.

Rehfeldt, K.R. and Gelhar, L.W., 1992, "Stochastic analysis of dispersion in unsteady flow in heterogeneous aquifers", *Water Resources Research*, v28, n8, pp2085-2099.

Rubin, Y., 1990, "Stochastic modeling of macrodispersion in heterogeneous porous media", *Water Resources Research*, v26, n1, pp133-141.

Rubin, Y., 1991, "Transport in heterogeneous porous media: prediction and uncertainty", *Water Resources Research*, v27, n7, pp1723-1738.

Rubin, Y., "On the concept of block effective macrodispersivity", *Calibration and Reliability in Ground Water Modelling* (Proceedings of the ModelCARE 99 Conference Held at Zurich, Switzerland, September 1999), IAHS, n265, pp137-140.

Rubin, Y. and Dagan, G., 1992, "A note on head and velocity covariances in three dimentional flow through heterogeneous anistropic porous media", *Water Resources Research*, v28, n5, pp1463-1470.

Rubin, Y. and Dagan, G., 1992, "Conditional estimation of solute travel time in heterogeneous formations: impact of transmissivity measurements", *Water Resources Research*, v28, n4, pp1033-1040.

Rushton, K. R. and Redshaw, S. C., 1979, *Seepage and Groundwater Flow*, Wiley, Chichester.

Russo, D., 1984, "A geostatistical approach to solute transport in heterogeneous fields and its application to salinity management" ", *Water Resources Research*, v20, n9, pp1260-1270.

Russo, D., 1995, "On the velocity covariance and transport modeling in heterogeneous anisotrophic porous formations. 1. Saturated flow", *Water Resources Research*, v31, n1, pp129-137.

Sagar, B, 1978, "Galerkin finite element procedure for analyzing flow through random media", *Water Resources Research*, v14, n6, pp1035-1044.

Salandin, P. and Fiorotto, V., 1998, "Solute transport in highly heterogeneous aquifers", *Water Resources Research*, v34, n5, pp949-961.

Scheibe, T. D. and Cole, C.R., 1994, "Non Gaussian particle tracking: application to scaling of transport processes in heterogeneous porous media", *Water Resources Research*, v30, n7, pp2077-2039.

Seroos, J, 1995, "Temporal moments for nonergodic solute transport in heterogeneous aquifers" *Water Resources Research*, v31, n7, pp1705-1712.

Serrano, S. E., "Forecasting scale-dependent dispersion from spills in heterogeneous aquifers", *Journal of Hydrology,* v169, pp151-169.

Serrano, S. E., 1996, "Hydrologic theory of dispersion in heterogeneous aquifers", *Journal of Hydrologic Engineering,* Oct, pp144-151.

Serrano, S. E., 1997, "Non-Fickian transport in heterogeneous saturated porous media", *Journal of Engineering Mechanics,* Jan, pp70-76.

Shapiro, A. and Cvetkovic, V., 1988, "Stochastic analysis of solute arrival time in heterogeneous porous media", *Water Resources Research*, v24, n10, pp1711-1718.

Shrestha, S. P. and Loganathan, G. V., 1994, "Monte Carlo simulation and effective medium approximation in subsurface flow modeling", *Ground Water*, v32, n6, pp929-936.

Smith, L. and Freeze R. A., 1979a, "Stochastic analysis of steady state groundwater flow in a bounded domain, 1, One-dimensional simulations", *Water Resources Research*, v15 n3, pp521-528.

Smith, L. and Freeze R. A., 1979b, "Stochastic analysis of steady state groundwater flow in a bounded domain, 2, Two-dimensional simulations", *Water Resources Research*, v15 n6, pp1543-1559.

Takada, T., 1990, "Weighted integral method in stochastic finite element analysis", *Probabilistic Engineering Mechanics*, v5, n3, pp146-156.

Takada, T. and Masanobu M., 1990, "Local integration method in stochastic finite element analysis", *5th International Conference on Structural Safety and Reliability*, pp1073-1080.

Tartakovsky, D. M., Guadagnini, A. and Guadagnini, L., 2000, "Effective hydraulic conductivity and transmissivity for heterogeneous aquifers", *Mathematical Geology*, v32, n6, pp751-757.

Tartakovsky, D. M. and Neuman S. P., 1998, "Transient flow in bounded randomly heterogeneous domains: 1. Exact conditional moment equations and recursive approximations", *Water Resources Research,* v34, n1, pp1-12.

Tompkins, J. A., Gan, K. C., Wheater, H. S. and Hirano, F., 1994, "Prediction of solute dispersion in heterogeneous porous media: effects of erodicity and hydraulic conductivity discretisation", *Journal of Hydrology,* v159, pp105-123.

Tompson, A. F. B. and Gelhar, L.W., 1990, "Numerical simulation of solute transport in three-dimensional, randomly heterogeneous porous media", *Water Resources Research,* v26, n10, pp2541-2562.

Tompson, A. F. B. and Gelhar, L.W., 1990, "Numerical simulation of solute transport in three-dimensional, randomly heterogeneous porous media", *Water Resources Research,* v26, n10, pp2541-2562.

Toride, N. and Feike, J. L., 1996, "Convective-dispersive stream tube model for field-scale solute transport: I. Moment analysis", *Soil Society America Journal*, v60, pp342-352.

Toride, N. and Feike, J. L., 1996, "Convective-dispersive stream tube model for field-scale solute transport: II. Examples and calibration", *Soil Society America Journal*, v60, pp352-361.

Winter, C. L., Tartakovsky, D. M. and Guadini, A., 2002, "Numerical solutions of moment equations for flow in heterogeneous composite aquifers", *Water Resources Research,* v38, n5, pp13.1-13.8.

Van Lent, T. and Kitanidis P.K., 1986, "Effects of first-order approximations on head and specific discharge covariances in high-contrast log conductivity", *Water Resources Research,* v32, n5, pp1197-1207.

Vanderborght, J., 2001, "Concentration variance and spatial covariance in second-order stationary heterogeneous conductivity fields", *Water Resources Research,* v37, n7, pp1893-1912.

Vanmarke, E. H., 1983, Random Fields: Analysis and Synthesis, MIT Press, Cambridge.

Vanmarke, E. H., 1994, "Stochastic finite elements and experimental measurements", *Probabilistic Engineering Mechanics*, v9, n1-2, pp103-114.

Vanmarcke, E. H. and Hachich, W., 1983 "Probabilistic updating of pore pressure fields", *Journal of Geotechnical Engineering – ASCE*, v109, n3, pp373-387.

Vanmarke, E. H., Shinozuka, M., Nakagiri, S., Schueller, G. I., and Grigoriu, M., 1996, "Random fields and stochastic finite elements", *Structural Safety,* v3, pp143-166.

Vladimir, C. and Dagan, G., 1994, "Transport of kinetically sorbing solute by steady random velocity in heterogeneous porous formations", *Journal of Fluid Mechanics,* v265, pp189-215.

Yaacob, I., 1991, "An adaptive importance sampling strategy", *Mechanics in the 1990's and Beyond*, v1, Hojjat Adeli and Robert L Sierakowski (eds), ASCE, New York pp253-255.

Yamazaki, F., and Shinozuka M., 1990, "Stimulation of stochastic fields by statistical preconditioning", *Journal of Engineering Mechanics,* v116, n2, pp268-278.

Zhang, D., Andricevic, R., Sun, A. Y., Hu, X., and He, G., 2000, "Solute flux approach to transport through spatially nonstationary flow in porous media", *Water Resources Research*, v36, n8, pp2107-2120.

Zhang, D. and Lu, Z., 2002, "Stochastic analysis of flow in a heterogeneous unsaturated-saturated system", *Water Resources Research,* v38 n2, pp10.1-10.15.

Zhang, D. and Neuman, S., 1996, "Effect of local dispersion on solute transport in randomly heterogeneous media", *Water Resources Research,* v32 n9, pp2715-2723.

Zhang, Y., 1997, "On the variances of second spatial moments of a nonergodic plume in heterogeneous aquifers", *Water Resources Research*, v33, n8, pp1893-1900.

Zhang, Y.and Di Fredrico, V., 1998, "Solute transport in three-dimensional heterogeneous media with a gaussian covariance of log hydraulic conductivity", *Water Resources Research*, v34, n8, pp1929-1934.

Zhang, Y. and Zhang, D., 1997, "Time-dependent dispersion of nonergodic plumes in two-dimensional heterogeneous aquifiers", *Journal of Hydrologic Engineering,* April, pp91-94.

Zhang, Y., Zhang, D. and Lin, J., 1996, "Nonergodic solute transport in three-dimensional heterogeneous isotropic aquifers", *Water Resources Research*, v32, n9, pp2955-2963.

Zhu, W. Q., Ren Y. J. and Wu W. Q., 1992, "Stochastic FEM based on local averages of random vector fields", *Journal of Engineering Mechanics,* v118, n3, pp496-511.

Appendix A Program Listing

```
      PROGRAM GW3
C*********************************************************************
C
C     THIS PROGRAM SOLVES STEADY-STATE, SATURATED
C     GROUNDWATER FLOW / CONTAMINANT TRANSPORT PROBLEMS.
C     THE ORIGINAL PROGRAM COMES FROM ISTOK, 1989.
C     THIS VERSION HAS BEEN MODIFIED TO MODEL THE HYDRAULIC
C     CONDUCTIVITY FIELD AS A RANDOM FIELD.  THE COVARIANCES
C     OF THE HYDRAULIC CONDUCTIVITY IN VARIOUS ELEMENTS HAVE
C     BEEN DETERMINED USING THE MID-POINT METHOD.  THE HEAD,
C     VELOCITY AND CONCENTRATION FIELDS ARE CALCULATED USING
C     TAYLOR SERIES.
C     IN ADDITION THIS VERSION HAS BEEN MODIFIED TO TAKE
C     ADVANTAGE OF SYMMETRY IN THE COVARIANCE MATRIX AND
C     TO TAKE ADVANTAGE OF LONGITUDINAL SYMETRY IN THE HEAD,
C     VELOCITY AND CONCENTRATION FIELDS AND TRANSVERSE SYMMETRY
C     IN THE HEAD AND VELOCITY FIELDS.
C
C     THE PROGRAM IS WRITTEN SO THAT IT CAN EXECUTE FOR A WHILE,
C     WRITE A CHECKPOINT FILE, TERMINATE AND THEN START AGAIN
C     TAKING UP WHERE IT LEFT OFF.  THIS BEHAVIOUR IS CONTROLLED
C     BY THE VARIABLE STATUS.  THE VALUE OF STATUS HAS THESE
C     MEANINGS.
C       0 = PROGRAM FIRST STARTING.  HYDRAULIC INPUT DATA NEEDS
C           TO BE ECHOED TO RESULTS FILE.
C       1 = PROGRAM IS IN THE HYDRAULIC CALCULATIONS STAGE.
C           HYDRAULIC INPUT DOES NOT NEED TO BE ECHOED TO
C           RESULTS FILE.
C       2 = HYDRAULIC CALCULATIONS FINISHED.  CONCENTRATION
C           INPUT NEEDS TO BE ECHOED TO RESULTS FILE.
C       3 = UNUSED.
C       4 = PROGRAM IS IN THE CONCENTRATION CALCULATIONS STAGE.
C           CONCENTRATION DATA DOES NOT NEED TO BE ECHOED TO
C           RESULTS FILE.
C     THE VALUE OF STATUS IS INITIALLY DETERMINED IN SETUP.
C     IF THE CHECKPOINT FILES HAVE NOT BEEN CREATED IT IS 0.
C     IF THE CHECKPOINT FILES EXIST BUT DONEFILE DOES NOT IT IS 1.
C     IF DONEFILE EXISTS IT IS READ FROM DONEFILE.
C     ITS VALUE IS MODIFIED AS THE PROGRAM COMPLETES EACH SECTION.
C     COMPLETION OF EXECUTION IS INDICATED BY THE PRESENCE OF
C     THE FILE ENDFILE.  THE BATCH FILE THAT RUNS THE PROGRAM SHOULD
C     CHECK THAT THIS FILE DOES NOT EXIST BEFORE RUNNING THE PROGRAM.
C     FOR DETAILS ON COMMAND LINE ARGUMENTS SEE SUBROUTINE SETUP.
C
C*********************************************************************

      INCLUDE 'COMALL'
      DOUBLE PRECISION DNDX(MAX3,MAX2,3)
      INTEGER I, J, STATUS, ERROR
      CHARACTER*80 TITLE
C
C  0. SET SYSTEM UP FOR DEALING WITH TERMINATION SIGNAL
C
      INTEGER SIGNAL
      EXTERNAL CHKPNT
      I = SIGNAL(15,CHKPNT,-1)
      CONTROL = .FALSE.
C
C  1. DETERMINE THE NAMES OF AND OPEN THE INPUT AND OUTPUT FILES
C
      CALL SETUP(STATUS)
```

```
C
C  2. CREATE GEOMETRIC, MATERIAL AND BOUNDARY DATA
C
      LABEL1 = '   HYDRAULIC HEAD'
      LABEL2 = '  GROUNDWATER FLOW'
      CALL CREATE (STATUS .GE. 1)
C
C  5. ASSEMBLE AND MODIFY THE GLOBAL SYSTEM OF EQUATIONS
C
      CALL ASMBK
C
C  6. SOLVE THE SYSTEM OF EQUATIONS
C     THIS SECTION HAS BEEN CHANGED TO USE THE LAPACK
C     ROUTINES DPBTRF AND DPBTRS, WHICH RESPECTIVELY FACTORISE
C     AND SOLVE THE MATRIX CREATED IN ASMBK
C
      CALL DPBTRF('UPPER',NDOF,SBW,M1,MAX6+1,ERROR)
      IF (ERROR .NE. 0) THEN
        PRINT *,'ERROR IN FACTORISATION',ERROR,M1(SBW+1,ABS(ERROR))
      ENDIF
      CALL DPBTRS('UPPER',NDOF,SBW,1,M1,MAX6+1,B,MAX0,ERROR)
      IF (ERROR .NE. 0) THEN
        PRINT *,'ERROR IN SOLVING',ERROR,M1(SBW+1,-ERROR)
      ENDIF
C
C  7. WRITE OUT COMPUTED HYDRAULIC HEAD VALUES
C     DTERMINISTIC HYDRAULIC HEAD VALUES ARE NOW WRITTEN OUT
C     AT THE END OF STOCH, TOGETHER WITH THE STOCHASTIC VALUES.
C     THIS SECTION SETS UP THE HYDRAULIC HEAD VECTOR REQUIRED
C     FOR DETERMINING THE VELOCITIES
C
      J = 0
      DO 40 I = 1, NUMNOD
        IF (ICH(I) .EQ. 0) THEN
          J = J+1
          X(I) = B(J)
        ENDIF
 40   CONTINUE
C
C  8. COMPUTE GROUNDWATER VELOCITIES FOR EACH ELEMENT
C
      CALL VELOCITY(DNDX)
C
C  9. DETERMINE MEANS AND (CO)VARIANCES OF HEADS AND VELOCITIES.
C     NORMALLY THE PROGRAM WILL EXECUTE STOCH FOR A WHILE,
C     SAVE A CHECKPOINT FILE AND TERMINATE.  IT WILL THEN NEED
C     TO START AGAIN TO TAKE UP WHERE IT LEFT OFF.
C     IF STATUS IS 2 OR GREATER THEN THE PROGRAM HAS PREVIOUSLY
C     COMPLETED ALL OF THESE CALCULATIONS AND CAN SKIP TO THE
C     CONCENTRATION CALCULATIONS (10).
C
      IF (STATUS .LE. 1) THEN
        CALL STOCH(DNDX, STATUS .EQ. 1)
        STATUS = 2
        OPEN (UNIT = DONEF, FILE = DONEFILE, STATUS = 'UNKNOWN')
        WRITE (DONEF,*) 2,'   Heads and velocities complete'
        CLOSE (UNIT = DONEF, STATUS = 'KEEP')
      ENDIF
C
C 10. BEGIN SOLUTE TRANSPORT PROBLEM
C
      SYMM = .FALSE.
      LABEL1 = 'SOLUTE CONCENTRATION'
      LABEL2 = '    SOLUTE FLUX'
C
C 11. INPUT MATERIAL PROPERTIES, BOUNDARY CONDITIONS
C     AND INITIAL CONDITIONS FOR EACH ELEMENT
C
```

```
      CALL CREAT2(STATUS .GE. 3)
C
C 12. CARRY OUT CONCENTRATION CALCULATIONS, BOTH DETERMINISTIC
C     AND STOCHASTIC.
C     LIKE STOCH THIS SUBROUTINE EXECUTES FOR A WHILE, WRITES
C     A CHECKPOINT FILE, TERMINATES AND THE PROGRAM STARTS AGAIN.
C
      IF (STATUS .LE. 4) THEN
        CALL CONCEN (STATUS .GE. 4)
      ENDIF
C
C 13. CREATE A FILE TO ENSURE THAT THE PROGRAM DOES NOT START AGAIN.
C     THIS FILE IS CHECKED FOR IN THE BATCHFILE THAT RUNS THE PROGRAM.
C
      OPEN (UNIT = ENDF, FILE = ENDFILE, STATUS = 'UNKNOWN')
      WRITE (ENDF,*) '  CONCENTRATIONS COMPLETE'
      CLOSE (UNIT = ENDF, STATUS = 'KEEP')
      END

      SUBROUTINE CHKPNT
C*****************************************************************
C
C     THIS SUBROUTINE ENABLES THE USER TO HALT THE PROGRAM.
C     THE INVOCATION OF SIGNAL AT THE START OF THE MAIN
C     PROGRAM SETS THIS SUBROUTINE UP AS THE SIGNAL HANDLER
C     FOR THE "TERM" SIGNAL (SIGNAL 15).  THE MAJOR TIME
C     CONSUMING LOOPS OF THE PROGRAM (IN SUBROUTINES STOCH
C     AND CONCEN) CHECK THE VARIABLE CONTROL AND IF IT IS
C     TRUE A CHECKPOINT FILE IS WRITTEN OUT OF THE PROGRAM'S
C     CURRENT STATUS AND THE PROGRAM EXITS.
C     CONTROL IS SET TO FALSE AT THE BEGINNING OF THE PROGRAM.
C     USERS CAN CREATE A TERM SIGNAL USING THE COMMAND
C           kill -15 PID
C     WHERE PID IS THE PROCESS NUMBER OF THE PROGRAM.
C
C*****************************************************************
      INCLUDE 'COMALL'
      CONTROL = .TRUE.
      PRINT *,'I HAVE BEEN TOLD TO FINISH. SIGNAL 15'
      RETURN
      END

      SUBROUTINE SETUP(STATUS)
C*****************************************************************
C
C     THIS SUBROUTINE DETERMINES THE NAMES OF THE FILES USED
C     AND THE LOCATION IN THE PROGRAM TO START (ASSUMING THAT
C     CHECKPOINT FILES ARE AVAILABLE).  THIS DATA IS READ FROM
C     THE COMAND LINE.  IF ANY OF THESE NAMES ARE NOT PRESENT
C     ON THE COMMAND LINE THEN DEFAULTS ARE ASSUMED.
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER STATUS
      LOGICAL PRIOR
      CHARACTER*80 INFILE, OUTFILE, TITLE
C
C     GETARG IS A UNIX SYSTEM CALL THAT OBTAINS THE ARGUMENTS
C     IN THE COMMAND LINE
C
      EXTERNAL GETARG
C
C     GET NAME OF INPUT DATA FILE
C
      CALL GETARG(1,INFILE)
      IF (INFILE .NE. ' ')  GOTO 15
      WRITE (*,10) ' Enter the name of the input data file: '
 10   FORMAT (A)
```

```
      READ (*,10) INFILE
 15   OPEN (INF, FILE=INFILE, STATUS = 'OLD')
C
C     GET NAME OF OUTPUT FILE
C
      CALL GETARG(2,OUTFILE)
      IF (OUTFILE .NE. ' ')  GOTO 25
 20   WRITE (*,10) ' Enter the name of the output file: '
      READ (*,10) OUTFILE
      INQUIRE (FILE = OUTFILE, EXIST = PRIOR)
      IF (PRIOR) THEN
        WRITE (*,10) ' File exists.  Overwrite?'
        READ (*,10) TITLE
        IF (TITLE .NE. 'Y' .AND. TITLE .NE. 'y') GOTO 20
      ENDIF
 25   OPEN (OUTF, FILE=OUTFILE, STATUS='UNKNOWN')
C
C     GET NAME OF VELOCITY DATA CHECKPOINT FILE.
C     THIS WILL BE A BINARY FILE DUE TO THE LARGE
C     SIZE OF THE VELOCITY COVARIANCE MATRIX.
C
      CALL GETARG(3, VELFILE)
      IF (VELFILE .EQ. ' ') THEN
        VELFILE = 'AUTOSAVE'
      ENDIF
C
C     GET NAME OF HEAD AND CONCENTRATION CHECKPOINT FILE
C     THIS WILL BE AN ASCII FILE.
C
      CALL GETARG(4, HCFILE)
      IF (HCFILE .EQ. ' ') THEN
        HCFILE = 'HEADSAVE'
      ENDIF
C
C     GET NAME OF STATUS FILE AND DETERMINE STATUS.
C     THIS IS AN ASCII FILE WHOSE EXISTANCE INDICATES THAT
C     HYDRAULIC CALCULATIONS ARE FINISHED AND WHOSE CONTENTS
C     IF CONCENTARTION INPUT DATA HAS BEEN ECHOED TO THE
C     RESULTS FILE
C
      CALL GETARG(5,DONEFILE)
      IF (DONEFILE .EQ. ' ') THEN
        DONEFILE = 'DONE'
      ENDIF
      INQUIRE (FILE = DONEFILE, EXIST = PRIOR)
      IF (.NOT. PRIOR) THEN
        INQUIRE (FILE = VELFILE, EXIST = PRIOR)
        IF (PRIOR) THEN
          STATUS = 1
        ELSE
          STATUS = 0
        ENDIF
      ELSE
        OPEN (UNIT = DONEF,FILE = DONEFILE, STATUS = 'UNKNOWN')
        READ (DONEF,*) STATUS
        CLOSE (UNIT = DONEF, STATUS = 'KEEP')
      ENDIF
C
C     GET NAME OF ENDFILE.
C     EXISTANCE OF THIS FILE INDICATES THAT THE PROGRAM IS FINISHED.
C
      CALL GETARG(6, ENDFILE)
      IF (ENDFILE .EQ. ' ') THEN
        ENDFILE = 'FINISHED'
      ENDIF
C
C     GET NAME OF CHECKPOINT PROBLEM FILE.
C     THIS FILE IS CREATED BEFORE A CHECKPOINT BEGINS AND
```

```
C     DELETED WHEN IT IS FINISHED.  THUS, IF IT IS PRESENT
C     BEFORE EXECUTION BEGINS THEN THE CHECKPOINT FILE IS
C     CORRUPTED.
C     THE BATCHFILE EXECUTING THE PROGRAM USUALLY MAKES
C     A BACKUP OF THE CHECKPOINT FILE.  THUS IF THIS FILE
C     EXISTS THEN THE BACKUP OF THE CHECKPOINT FILES SHOULD
C     BE USED INSTEAD OF THE MAIN COPY.
C
      CALL GETARG(7, PROBFILE)
      IF (PROBFILE .EQ. ' ') THEN
        PROBFILE = 'PROBLEM'
      ENDIF
      RETURN
      END

      SUBROUTINE STOCH(DNDX,PRIOR)
C*********************************************************************
C
C 12.1  PURPOSE:
C         SUBROUTINE STOCH DETERMINES THE SECOND DERIVATIVES OF EACH
C         NODAL HYDRAULIC HEAD VALUE AND ELEMENTAL VELOCITY
C         COMPONENT WITH RESPECT TO THE HYDRAULIC CONDUCTIVITY
C         OF EACH ELEMENT.  THESE VALUES AND THE VALUES FOR THE
C         RESPECTIVE FIRST DERIVATIVES ARE USED IN DETERMINING THE
C         MEANS AND COVARIANCES OF THE NODAL HYDRAULIC HEAD VALUES
C         AND ELEMENTAL VELOCITY COMPONENTS
C
C  8.2  INPUT:
C         NONE
C
C  8.3  OUTPUT:
C         THE MEANS AND VARIANCES OF THE NODAL HYDRAULIC HEAD
C         VALUES AND ELEMENTAL VELOCITY COMPONENTS ARE WRITTEN
C         TO THE USER-DEFINED FILE ASSIGNED TO UNIT "OUTF"
C
C
C 12.4  DEFINITIONS OF VARIABLES:
C                B(I) = RIGHT HAND SIDE VECTOR
C                     = [dKe/dke] [X]
C           DHDK(I,E) = DERIVATIVE OF HEAD AT NODE I WITH RESPECT TO
C                       HYDRAULIC CONDUCTIVITY OF ELEMENT E
C                   E = ELEMENT NUMBER
C          ELEMTYP(I) = ELEMENT TYPE FOR ELEMENT I
C              ICH(I) = 1 IF THE VALUE OF THE FIELD VARIABLE IS
C                       SPECIFIED FOR NODE I
C                     = 0 OTHERWISE
C             KE(I,J) = CONDUCTANCE MATRIX FOR ELEMENT E IN FULL
C                       MATRIX STORAGE
C              LCH(I) = ICH(I) + ICH(I-1) + ICH(I-2) + ...
C                       THE ARRAYS ICH AND LCH ARE USED TO MODIFY
C                       GLOBAL SYSTEM OF EQUATIONS IN SUBROUTINES
C                       ASMBK, ASMBKC, AND ASMBAD
C               M(IJ) = MODIFIED GLOBAL CONDUCTANCE MATRIX IN
C                       VECTOR STORAGE
C                NDOF = NUMBER OF DEGREES OF FREEDOM
C                       (UNSPECIFIED NODES)
C          NODETBL(I) = NUMBER OF NODES IN ELEMENT TYPE I
C              NUMELM = NUMBER OF ELEMENTS IN MESH
C                 SBW = SEMI-BANDWIDTH OF MODIFIED GLOBAL
C                       CONDUCTANCE MATRIX
C               X(I) = VALUE OF THE FIELD VARIABLE AT NODE I
C
C 12.5  USAGE:
C         DHDK IS DETERMINED USING:
C                M * DHDK = - DMDK * H
C         D2HDK2 IS DETERMINED USING:
C                M * D2HDK2(I,J) = - DMDK(I) * DHDK(J)
C                                  - DMDK(J) * DHDK(I)
```

```
C
C       SUBROUTINES CALLED:   (IF I HAVE BOTHERED TO INCLUDE THEM)
C         KBAR2,KBAR3,KBAR4,KTRI3,KREC4,KQUA4,KQUA8,KQUA12,KPAR8,
C         KPAR20,KPAR32,KTRI3A,KREC4A,KREP8
C         LOC
C
C*********************************************************************
      INCLUDE 'COMALL'
      INTEGER E, E2, E3, E4, I, J, K, L, J5, J6, J7
      INTEGER J0, J1, J2, J3, J4, K2, N(3), REFLEC(0:3,4)
      DOUBLE PRECISION KE(MAX3,MAX3),SUM(3),SQUARE(3),THETA
      DOUBLE PRECISION DNDX(MAX3,MAX2,3), KCOR(MAX4), XC(MAX2,3)
      DOUBLE PRECISION R, MOMENT, TEMP, TEMP1, LIMIT, LIMIT2
      REAL TARRAY(2), SECS, ETIME
      LOGICAL PRIOR
      CHARACTER Z(3)
      EXTERNAL ETIME
      Z(1) = 'X'
      Z(2) = 'Y'
      Z(3) = 'Z'
      IF (PRIOR) THEN
        CALL VLOAD
        CALL HLOAD(E3,E4)
        DO 10 E = 1, E3
          CALL KCORR(E, 1, XC, KCOR)
 10     CONTINUE
      ELSE
        DO 20 J = 1, NDOF
          DO 20 L = 1, NUMMAT
            MEAN(J,L) = 0.0
            VAR(J,L)  = 0.0
 20     CONTINUE
        DO 30 E = 1, NUMELM / 4
          DO 30 K = 1, DIM
            DO 30 L = 1, NUMMAT
              VMEAN(L,K,E) = 0.0
 30     CONTINUE
        DO 35 E = 1, NUMELM * (NUMELM / 4 + 1) / 2
          DO 35 K = 1, DIM
            DO 35 K2 = 1, DIM
              DO 35 L = 1, NUMMAT
                COVAR(L,K2,K,E) = 0.0
 35     CONTINUE
        E3 = 1
        E4 = 0
      ENDIF
C     THE NEXT TWO LOOPS MUST NOT BE PARALLELISED SO AUTOSAVING FUNCTIONS
      DO 170 E = E3, NUMELM
        PRINT *,'BEGINNING FLOW FOR ELEMENT',E,'.  NOR2 =',NOR2,(E-1)**2
        CALL HDER1(E,KE,1)
          CALL VDER1(E,DNDX,1)
        REFLEC(0,1) =  2
        REFLEC(0,2) = -2
        REFLEC(0,3) = -2
        REFLEC(0,4) =  2
        REFLEC(1,1) =  2
        REFLEC(1,2) =  2
        REFLEC(1,3) =  2
        REFLEC(1,4) =  2
        REFLEC(2,1) =  2
        REFLEC(2,2) = -2
        REFLEC(2,3) =  2
        REFLEC(2,4) = -2
        IF (E .LE. NUMELM / 4) THEN
          LIMIT = E
        ELSEIF (E .LE. NUMELM / 2) THEN
          LIMIT = NUMELM / 2 + 1 - E
        ELSEIF (E .LE. 3 * NUMELM / 4) THEN
```

```
          LIMIT = E - NUMELM / 2
        ELSE
          LIMIT = NUMELM + 1 - E
        ENDIF
        DO 160 E2 = E4 + 1, LIMIT
          IF (E2 .EQ. LIMIT) THEN
C           SYMMETRY OF DERIVATIVE DOES NOT DOUBLE ANSWER
            DO 42 J = 0, DIM
              DO 42 J2 = 1, 4
                REFLEC(J,J2) = REFLEC(J,J2) / 2
 42         CONTINUE
          ENDIF
          CALL KCORR(E,E2,XC,KCOR)
C         THESE LINES AND THE IF RESTRICT THE VALUES TO "CLOSE" PAIRS
          R = 0
          DO 50 K = 1, DIM
            R = R + (XC(E,K) - XC(E2,K)) ** 2
 50       CONTINUE
          IF (R .LE. R2 * R2) THEN
            NOR2 = NOR2 + REFLEC(1,1) + REFLEC(1,2)
     1                  + REFLEC(1,3) + REFLEC(1,4)
            CALL HDER2(E,E2,KE)
            DO 60 J = 1, NDOF
              J4 = (J+NDN/2-1) / (NDN/2) * NDN/2 - MOD(J-1, NDN/2)
              J3 = NDOF + 1 - J
              J2 = NDOF + 1 - J4
              TEMP = 0.5 * ( REFLEC(0,1) * D2HDK2(J)
     1                     + REFLEC(0,2) * D2HDK2(J2)
     2                     + REFLEC(0,3) * D2HDK2(J3)
     3                     + REFLEC(0,4) * D2HDK2(J4))
              TEMP1 = ABS(REFLEC(0,1)) * DHDK(J,1) * DHDK(J,2)
     1              + ABS(REFLEC(0,2)) * DHDK(J2,1) * DHDK(J2,2)
     2              + ABS(REFLEC(0,3)) * DHDK(J3,1) * DHDK(J3,2)
     3              + ABS(REFLEC(0,4)) * DHDK(J4,1) * DHDK(J4,2)
              DO 60 L = 1, NUMMAT
                MEAN(J,L) = MEAN(J,L) + TEMP * KCOR(L)
                VAR(J,L) = VAR(J,L) + TEMP1 * KCOR(L)
 60         CONTINUE
C     END OF HYDRAULIC HEAD CALCULATIONS.
        CALL VDER2(E,E2,DNDX)
        CALL VDER1(E2,DNDX,2)
            DO 125 J = 1, NUMELM / 4
              J2 = NUMELM / 2 + 1 - J
              J3 = NUMELM / 2 + J
              J4 = NUMELM + 1 - J
              DO 125 K = 1, DIM
                TEMP = 0.5 * ( REFLEC(K,1) * D2VDK2(J ,K)
     1                       + REFLEC(K,2) * D2VDK2(J2,K)
     2                       + REFLEC(K,3) * D2VDK2(J3,K)
     3                       + REFLEC(K,4) * D2VDK2(J4,K) )
                DO 125 L = 1, NUMMAT
                  VMEAN(L,K,J) = VMEAN(L,K,J) + TEMP * KCOR(L)
125           CONTINUE
C$DOACROSS LOCAL (J,J0,J1,J2,J3,J4,J5,J6,J7,TEMP,LIMIT2)
C$&       , MP_SCHEDTYPE=INTERLEAVE
            DO 130 J = 1, NUMELM
              IF (J .LE. NUMELM / 4) THEN
                LIMIT2 = J
                J5 = NUMELM / 2 + 1 - J
                J6 = NUMELM / 2 + J
                J7 = NUMELM + 1 - J
              ELSEIF (J .LE. NUMELM / 2) THEN
                LIMIT2 = NUMELM / 2 + 1 - J
                J5 = NUMELM / 2 + 1 - J
                J6 = NUMELM / 2 + J
                J7 = NUMELM + 1 - J
              ELSEIF (J .LE. 3 * NUMELM / 4) THEN
                LIMIT2 = J - NUMELM / 2
```

```
              J5 = 3 * NUMELM / 2 + 1 - J
              J6 = J - NUMELM / 2
              J7 = NUMELM + 1 - J
            ELSE
              LIMIT2 = NUMELM + 1 - J
              J5 = 3 * NUMELM / 2 + 1 - J
              J6 = J - NUMELM / 2
              J7 = NUMELM + 1 - J
            ENDIF
            DO 120 J1 = 1, LIMIT2
              IF (J .LE. NUMELM / 4) THEN
                J0 = J * (J - 1) / 2 + J1
              ELSEIF (J .LE. NUMELM / 2) THEN
                J0 = NUMELM / 2 * (NUMELM / 4 + 1) / 2
     1             - (NUMELM/2-J+1) * (NUMELM/2-J+2) / 2 + J1
              ELSEIF (J .LE. 3 * NUMELM / 4) THEN
                J0 = NUMELM / 2 * (NUMELM / 4 + 1) / 2
     1             + (J-NUMELM/2) * (J-NUMELM/2-1) / 2 + J1
              ELSE
                J0 = NUMELM / 2 * (NUMELM / 4 + 1)
     1             - (NUMELM-J+1) * (NUMELM-J+2) / 2 + J1
              ENDIF
              J2 = NUMELM / 2 + 1 - J1
              J3 = NUMELM / 2 + J1
              J4 = NUMELM + 1 - J1
              DO 115 K = 1, DIM
                DO 115 K2 = 1, DIM
                  TEMP =(REFLEC(K,1) * REFLEC(K2,1) / REFLEC(1,1)
     1                     * ( DVDK(J,K,1) * DVDK(J1,K2,2)
     1                       + DVDK(J,K,2) * DVDK(J1,K2,1) )
     2                   + REFLEC(K,2) * REFLEC(K2,2) / REFLEC(1,2)
     3                     * ( DVDK(J5,K,1) * DVDK(J2,K2,2)
     3                       + DVDK(J5,K,2) * DVDK(J2,K2,1) )
     4                   + REFLEC(K,3) * REFLEC(K2,3) / REFLEC(1,3)
     5                     * ( DVDK(J6,K,1) * DVDK(J3,K2,2)
     5                       + DVDK(J6,K,2) * DVDK(J3,K2,1) )
     6                   + REFLEC(K,4) * REFLEC(K2,4) / REFLEC(1,4)
     7                     * ( DVDK(J7,K,1) * DVDK(J4,K2,2)
     7                       + DVDK(J7,K,2) * DVDK(J4,K2,1) )) / 2
                  DO 115 L = 1, NUMMAT
                    COVAR(L,K2,K,J0) = COVAR(L,K2,K,J0)
     1                               + TEMP * KCOR(L)
115           CONTINUE
120         CONTINUE
130       CONTINUE
        ENDIF
        SECS = ETIME(TARRAY)
C       THIS ALLOWS UP TO TWELVE THREADS!
        IF (SECS .GT. 2400 .OR. CONTROL) THEN
          PRINT *,'EXUCUTION TIME = ',SECS,' SECONDS.'
          CALL HSAVE(E,E2)
          CALL VSAVE
          SECS = ETIME(TARRAY)
          PRINT *,'STOPPING AT TIME = ',ETIME(TARRAY)
          STOP
        ENDIF
160     CONTINUE
        E4 = 0
170   CONTINUE
      CALL VSAVE
      REFLEC(0,1) =  1
      REFLEC(0,2) = -1
      REFLEC(0,3) = -1
      REFLEC(0,4) =  1
      REFLEC(1,1) =  1
      REFLEC(1,2) =  1
      REFLEC(1,3) =  1
      REFLEC(1,4) =  1
```

```
      REFLEC(2,1) =  1
      REFLEC(2,2) = -1
      REFLEC(2,3) =  1
      REFLEC(2,4) = -1
      DO 260 L = 1, NUMMAT
        WRITE(OUTF, 180)L,LABEL1,LABEL1,'MEAN'
180     FORMAT(//1X,74('*')//,23X,'RESULTS FOR MATERIAL SET',I3
     1         //18X,'COMPUTED VALUES OF ',A/
     2         18X,36('-')//
     3         1X,'NODE NO.',A,2X,A,8X,
     3       'FIRST_ORDER',4X,'TRUE_MEAN',7X,'X',6X,'Y',6X,'Z'/)
        J = 0
        DO 210 I = 1, NUMNOD
          IF (ICH(I) .EQ. 0) THEN
            J = J + 1
            IF (VAR(J,L) .GE. 0.0) THEN
              WRITE(OUTF,190) I,X(I),MEAN(J,L),SQRT(VAR(J,L))
     3                      ,X(I) + MEAN(J,L)
     1                            ,X1(I),X2(I),X3(I)
            ELSE
              WRITE(OUTF,190) I,X(I),MEAN(J,L),SQRT(-VAR(J,L))
     3                     ,X(I) + MEAN(J,L)
     1                            ,X1(I),X2(I),X3(I)
            ENDIF
          ELSE
            WRITE(OUTF,190) I,X(I),0.0,0.0,X(I),X1(I),X2(I),X3(I)
          ENDIF
190       FORMAT (I5,1X,4F16.12,3F7.2)
200       FORMAT (1X,I5,3X,F22.18,15X,A)
210     CONTINUE
        DO 260 K = 1, DIM
          WRITE (OUTF,270) Z(K)
          DO 220 E = 1, NUMELM
            IF (E .LE. NUMELM / 4) THEN
              E2 = E
              E3 = E2 * (E2 + 1) / 2
              E4 = 1
            ELSEIF (E .LE. NUMELM / 2) THEN
              E2 = NUMELM / 2 - E + 1
              E3 = E2 * (E2 + 1) / 2
              E4 = 2
            ELSEIF (E .LE. 3 * NUMELM / 4) THEN
              E2 = E - NUMELM / 2
              E3 = E2 * (E2 + 1) / 2
              E4 = 3
            ELSE
              E2 = NUMELM - E + 1
              E3 = E2 * (E2 + 1) / 2
              E4 = 4
            ENDIF
            IF (COVAR(L,K,K,E3) .GE. 0.0) THEN
              WRITE (OUTF,190) E,V(E,K),REFLEC(K,E4)*VMEAN(L,K,E2),
     1               SQRT(COVAR(L,K,K,E3)),
     3               V(E,K) + REFLEC(K,E4)*VMEAN(L,K,E2)
     2                         ,XC(E,1),XC(E,2),XC(E,3)
            ELSE
              WRITE (OUTF,190) E,V(E,K),REFLEC(K,E4)*VMEAN(L,K,E2),
     1                 -SQRT(-COVAR(L,K,K,E3))
     3                 ,V(E,K) + REFLEC(K,E4)*VMEAN(L,K,E2)
     2                         ,XC(E,1),XC(E,2),XC(E,3)
            ENDIF
270     FORMAT(/1X,74('*')//7X,'COMPUTED VALUES OF APPARENT ',
     1        'GROUNDWATER VELOCITY IN ',A,' DIRECTION'/7X,63('-')/2X,
     2        'ELEMENT',3X,'VELOCITY',10X,'MEAN',8X,
     3        'FIRST_ORDER',6X,'TRUE_MEAN',6X,
     4        'X',6X,'Y',6X,'Z'/)
220       CONTINUE
          DO 240 K2 = 1, DIM
```

```
            N(K2) = 0
            SUM(K2) = 0.0
            SQUARE(K2) = 0.0
240       CONTINUE
          DO 250 E = 1, NUMELM
            IF (E .LE. NUMELM / 4) THEN
              LIMIT = E
            ELSEIF (E .LE. NUMELM / 2) THEN
              LIMIT = NUMELM / 2 + 1 - E
            ELSEIF (E .LE. 3 * NUMELM / 4) THEN
              LIMIT = E - NUMELM / 2
            ELSE
              LIMIT = NUMELM + 1 - E
            ENDIF
            DO 250 E2 = 1, LIMIT
              DO 310 K2 = 1, DIM
                IF (ABS(XC(E,MOD(K2,3)+1) - XC(E2,MOD(K2,3)+1))
     1              .LT.1E-9 .AND. ABS(XC(E,MOD(K2+1,3)+1)
     2               - XC(E2,MOD(K2+1,3)+1)) .LT. 1E-9 .AND.
     3              ABS(XC(E,K2) - XC(E2,K2)) .GT. 1E-9) THEN
                  MOMENT = ABS(XC(E2,K2) - XC(E,K2))
                  IF (E .LE. NUMELM / 4) THEN
                    J1 = E
                    J2 = J1 * (J1 + 1) / 2
                    J3 = J1 * (J1 - 1) / 2 + E2
                    J1 = E2 * (E2 + 1) / 2
                  ELSEIF (E .LE. NUMELM / 2) THEN
                    J1 = NUMELM / 2 - E + 1
                    J2 = J1 * (J1 + 1) / 2
                    J3 = NUMELM * (NUMELM / 4 + 1) / 4
     1                 - J2 + E2
                    J1 = E2 * (E2 + 1) / 2
                  ELSEIF (E .LE. 3 * NUMELM / 4) THEN
                    J1 = E - NUMELM / 2
                    J2 = J1 * (J1 + 1) / 2
                    J3 = NUMELM * (NUMELM / 4 + 1) / 4
     1                 + J1 * (J1 - 1) / 2 + E2
                    J1 = E2 * (E2 + 1) / 2
                  ELSE
                    J1 = NUMELM - E + 1
                    J2 = J1 * (J1 + 1) / 2
                    J3 = NUMELM * (NUMELM / 4 + 1) / 2
     1                 - J2 + E2
                    J1 = E2 * (E2 + 1) / 2
                  ENDIF
                  THETA = SQRT(ABS(COVAR(L,K,K,J1)
     1                        * COVAR(L,K,K,J2)))
                  THETA = (COVAR(L, K, K, J3)) / THETA
                  IF (THETA .GT. 0.0) THEN
                    THETA = - MOMENT / LOG(THETA)
                    N(K2) = N(K2) + 1
                    SUM(K2) = SUM(K2) + THETA
                    SQUARE(K2) = SQUARE(K2) + THETA ** 2
                  ENDIF
                ENDIF
310           CONTINUE
250       CONTINUE
          DO 260 K2 = 1, DIM
            WRITE (OUTF,320) Z(K2),Z(K),SUM(K2)/N(K2),
     1            SQRT(SQUARE(K2)*N(K2) - SUM(K2)**2) / (N(K2)-1)
260     CONTINUE
        WRITE (OUTF,280)
280     FORMAT(/1X,74('*'))
320     FORMAT (/1X,'SCALE OF FLUCTUATION IN ',A,' DIRECTION OF ',A,
     1  ' VELOCITY:  MEAN ',F19.3,', STANDARD DEVIATION ',F19.3)
        write (*,280)

c       print *,'Their is a stop in n.f at the end of STOCH'
```

```
c       stop

        RETURN
        END

        SUBROUTINE KCORR(E,E2,XC,KCOR)
C*****************************************************************
        INCLUDE 'COMALL'
        DOUBLE PRECISION XC(MAX2,3), MOMENT, KCOR(MAX4), THETA(3)
        INTEGER E, E2, J, L, NODETBL(13)
        DATA NODETBL/2,3,4,3,4,4,8,12,8,20,32,3,4/
        IF (E2 .EQ. 1) THEN
          DO 10 J = 1, DIM
            XC(E,J) = 0.0
 10       CONTINUE
          DO 20 J = 1, NODETBL(ELEMTYP(E))
            XC(E,1) = XC(E,1) + X1(IN(E,J))
            XC(E,2) = XC(E,2) + X2(IN(E,J))
            XC(E,3) = XC(E,3) + X3(IN(E,J))
 20       CONTINUE
          DO 30 J = 1, DIM
            XC(E,J) = XC(E,J) / NODETBL(ELEMTYP(E))
 30       CONTINUE
        ENDIF
        DO 80 L = 1, NUMMAT
          MOMENT = 0.0
          DO 70 J = 1, DIM
            THETA(J) = PROP(L,4+J)
            IF (THETA(J) .NE. 0.0) THEN
              MOMENT = MOMENT + ((XC(E,J) - XC(E2,J)) / THETA(J)) ** 2
            ELSE
              PRINT *,' You have a theta of zero.  Check no of props.'
            ENDIF
 70       CONTINUE
          KCOR(L) = EXP(-SQRT(MOMENT)) * PROP(L,4) ** 2
 80     CONTINUE
        RETURN
        END

SUBROUTINE ASMBK
C*****************************************************************
C
C 12.1  PURPOSE:
C         SUBROUTINE ASMBK ASSEMBLES THE GLOBAL CONDUCTANCE MATRIX
C         AND THE GLOBAL SPECIFIED FLOW MATRIX.  THE GLOBAL MATRICES
C         ARE MODIFIED DURING THE ASSEMBLY PROCESS TO ACCOUNT FOR
C         SPECIFIED VALUES OF THE FIELD VARIABLE AND GROUNDWATER
C         FLOW DURING THE ASSEMBLY PROCESS.  THE GLOBAL CONDUCTANCE
C         MATRIX IS ASSEMBLED AND MODIFIED IN VECTOR STORAGE.  ASMBK
C         ALSO COMPUTES THE SEMI-BANDWIDTH AND THE NUMBER OF DEGREES
C         OF FREEDOM FOR THE MODIFIED GLOBAL CONDUCTANCE MATRIX.
C
C 12.2  INPUT:
C         NONE
C
C 12.3  OUTPUT:
C         SEMI-BANDWIDTH AND NUMBER OF DEGREES OF FREEDOM WRITTEN TO
C         THE USER-DEFINED FILE "OUTF"
C
C 12.4  DEFINITIONS OF VARIABLES:
C               B(I) = MODIFIED SPECIFIED FLOW MATRIX
C                  E = ELEMENT NUMBER
C         ELEMTYP(I) = ELEMENT TYPE FOR ELEMENT I
C            FLUX(I) = SPECIFIED VALUE OF GROUNDWATER FLOW AT NODE I
C             ICH(I) = 1 IF THE VALUE OF THE FIELD VARIABLE IS
C                      SPECIFIED FOR NODE I
C                    = 0 OTHERWISE
C           IJSIZE = LENGTH OF ARRAY M
```

```
C             KE(I,J) = CONDUCTANCE MATRIX FOR ELEMENT E IN FULL
C                       MATRIX STORAGE
C              LCH(I) = ICH(I) + ICH(I-1) + ICH(I-2) + ...
C                       THE ARRAYSICH AND LCH ARE USED TO MODIFY
C                       GLOBAL SYSTEM OF EQUATIONS IN SUBROUTINES
C                       ASMBK, ASMBKC, AND ASMBAD
C               M(IJ) = MODIFIED GLOBAL CONDUCTANCE MATRIX IN
C                       VECTOR STORAGE
C                 NDOF = NUMBER OF DEGREES OF FREEDOM (UNSPECIFIED NODES)
C          NODETBL(I) = NUMBER OF NODES IN ELEMENT TYPE I
C              NUMELM = NUMBER OF ELEMENTS IN MESH
C                 SBW = SEMI-BANDWIDTH OF MODIFIED GLOBAL
C                       CONDUCTANCE MATRIX
C                X(I) = VALUE OF THE FIELD VARIABLE AT NODE I
C
C 12.5  USAGE:
C         THE SEMI-BANDWIDTH OF THE GLOBAL CONDUCTANCE MATRIX IS
C         COMPUTED FIRST.  THEN THE ENTRIES OF THE ELEMENT
C         CONDUCTANCE MATRIX ARE COMPUTED IN A SET OF SUBROUTINES,
C         ONE SUBROUTINE FOR EACH ELEMENT TYPE.  THE GLOBAL
C         CONDUCTANCE MATRIX FOR THE MESH IS ASSEMBLED BY ADDING
C         THE CORRESPONDING ENTRIES OF THE ELEMENT CONDUCTANCE
C         MATRICES TO THE GLOBAL CONDUCTANCE MATRIX.  DURING THE
C         ASSEMBLY PROCESS THE GLOBAL CONDUCTANCE MATRIX IS MODIFIED
C         FOR SPECIFIED VALUES OF HEAD.  SPECIFIED VALUES OF
C         GROUNDWATER FLOW ARE ADDED TO THE GLOBAL FLOW MATRIX.
C
C       SUBROUTINES CALLED:   (IF I HAVE BOTHERED TO INCLUDE THEM)
C         KBAR2,KBAR3,KBAR4,KTRI3,KREC4,KQUA4,KQUA8,KQUA12,KPAR8,
C         KPAR20,KPAR32,KTRI3A,KREC4A,KREP8
C         LOCATE
C
C***********************************************************************
        INCLUDE 'COMALL'
        DOUBLE PRECISION KE(MAX3,MAX3)
        INTEGER NODETBL(14), E, I, KI, II, J, KJ, JJ
        DATA NODETBL/2,3,4,3,4,4,8,12,8,20,32,3,4,8/

C       INITIALISE ENTRIES OF GLOBAL CONDUCTANCE MATRIX TO ZERO
        IF (NDOF .GT. MAX0) STOP'** EXCEEDS MAX DEGREES OF FREEDOM**'
        DO 10 J = 1, NDOF
          DO 10 I = 1, SBW + 1
            M1(I,J) = 0.0
 10     CONTINUE
        DO 60 I = 1, NUMNOD
          IF (ICH(I) .EQ. 0) B(I-LCH(I)) = FLUX(I)
 60     CONTINUE
C       LOOP ON THE NUMBER OF ELEMENTS
        DO 90 E = 1, NUMELM
C       COMPUTE THE ELEMENT CONDUCTANCE MATRIX FOR THIS ELEMENT TYPE
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        IF (ELEMTYP(E) .EQ. 1) THEN
C         ELEMENT IS A LINEAR BAR
          CALL KBAR2(E,KE)
        ELSEIF (ELEMTYP(E) .EQ. 4) THEN
C         ELEMENT IS A LINEAR TRIANGLE
          CALL KTRI3(E,KE)
        ELSEIF (ELEMTYP(E) .EQ. 5) THEN
C         ELEMENT IS A LINEAR RECTANGLE
          CALL KREC4(E,KE)
        ELSEIF (ELEMTYP(E) .EQ. 6) THEN
C         ELEMENT IS A LINEAR QUADTILATERAL
          CALL KQUA4(E,KE)
        ELSEIF (ELEMTYP(E) .EQ. 9) THEN
C         ELEMENT IS A THREE DIMENSIONAL LINEAR PARALLELEPIPED
          CALL KPAR8(E,KE)
        ENDIF
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
C       ADD THE ELEMENT CONDUCTANCE MATRIX TO THE GLOBAL MATRIX
        DO 80 I = 1, NODETBL(ELEMTYP(E))
          KI = IN(E,I)
          IF (ICH(KI) .EQ. 0) THEN
            II = KI - LCH(KI)
            DO 70 J = 1, NODETBL(ELEMTYP(E))
              KJ = IN(E,J)
              IF (ICH(KJ) .NE. 0) THEN
                B(II) = B(II) - KE(I,J) * X(KJ)
              ELSEIF (KJ .GE. KI) THEN
c              ELSEIF (J .GE. I) THEN
               JJ = KJ - LCH(KJ)
               M1(SBW+1+II-JJ,JJ) = M1(SBW+1+II-JJ,JJ) + KE(I,J)
C               IF (I .EQ. J .AND. KE(I,J) .LT. 0.0) THEN
C                 PRINT *,' There is a problem with element',E,', that'
C                 PRINT *,' causes a negative entry for node',IN(E,I)
C               ENDIF
              ENDIF
 70         CONTINUE
          ENDIF
 80     CONTINUE
 90     CONTINUE
        RETURN
        END

        SUBROUTINE KBAR2(E,KE)
C***********************************************************************
C
C       PURPOSE:
C         TO COMPUTE THE ELEMENT CONDICTANCE MATRIX FOR A
C         ONE-DIMENSIONAL, LINEAR BAR ELEMENT
C
C       DEFINITIONS OF VARIABLES:
C               E = ELEMENT NUMBER
C         KE(I,J) = ELEMENT CONDUCTANCE MATRIC
C             KXE = HYDRAULIC CONDUCTIVITY IN X COOREDINATE DIRECTION
C              LE = ELEMENT LENGTH
C
C***********************************************************************
        INCLUDE 'COMALL'
        INTEGER E
        DOUBLE PRECISION KE(MAX3,MAX3),KXE,LE

        KXE = PROP(MATSET(E),1)
        LE = ABS(X1(IN(E,2)) - X1(IN(E,1)))
        KE(1,1) =  KXE / LE
        KE(1,2) = -KE(1,1)
        KE(2,1) = -KE(1,1)
        KE(2,2) =  KE(1,1)
        RETURN
        END

        SUBROUTINE KTRI3(E,KE)
C***********************************************************************
C
C         PURPOSE:
C           TO COMPUTE THE ELEMENT CONDUCTANCE MATRIX FOR TWO
C           DIMENSIONAL, LINEAR TRIANGLE ELEMENT
C
C         DEFINITIONS OF VARIABLES:
C               AE4 = FOUR TIMES ELEMENT AREA
C                 E = ELEMENT NUMBER
C           IN(I,J) = NODE NUMBER J FOR ELEMENT I
C           KE(I,J) = ELEMENT CONDUCTANCE MATRIX
C               KXE = HYDRAULIC CONDUCTIVITY IN X DIRECTION
C               KYE = HYDRAULIC CONDUCTIVITY IN Y DIRECTION
C
C***********************************************************************
```

```
      INCLUDE 'COMALL'
      INTEGER E, I, J
      DOUBLE PRECISION KE(MAX3,MAX3),KXE,KYE,BE(3),CE(3),AE4

      KXE = PROP(MATSET(E),1)
      KYE = PROP(MATSET(E),2)
      BE(1) = X2(IN(E,2)) - X2(IN(E,3))
      BE(2) = X2(IN(E,3)) - X2(IN(E,1))
      BE(3) = X2(IN(E,1)) - X2(IN(E,2))
      CE(1) = X1(IN(E,3)) - X1(IN(E,2))
      CE(2) = X1(IN(E,1)) - X1(IN(E,3))
      CE(3) = X1(IN(E,2)) - X1(IN(E,1))
C  CHECK THIS FORMULA!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      AE4 = ( X1(IN(E,2)) * X2(IN(E,3)) + X1(IN(E,1)) * X2(IN(E,2)) +
     1        X2(IN(E,1)) * X1(IN(E,3)) - X2(IN(E,3)) * X1(IN(E,1)) -
     2        X1(IN(E,3)) * X2(IN(E,2)) - X1(IN(E,2)) * X2(IN(E,1)) )*2
      DO 20 I = 1,3
        DO 10 J = 1,3
          KE(I,J) = (KXE * BE(I) * BE(J) + KYE * CE(I) * CE(J)) / AE4
 10     CONTINUE
 20   CONTINUE
      RETURN
      END

        SUBROUTINE KREC4(E,KE)
C**********************************************************************
C
C         PURPOSE:
C           TO COMPUTE THE ELEMENT CONDUCTANCE MATRIX FOR TWO
C           DIMENSIONAL, LINEAR RECTANGLE ELEMENT
C
C         DEFINITIONS OF VARIABLES:
C                 E = ELEMENT NUMBER
C           IN(I,J) = NODE NUMBER J FOR ELEMENT I
C           KE(I,J) = ELEMENT CONDUCTANCE MATRIX
C               KXE = HYDRAULIC CONDUCTIVITY IN X DIRECTION
C               KYE = HYDRAULIC CONDUCTIVITY IN Y DIRECTION
C
C**********************************************************************
      INCLUDE 'COMALL'
      INTEGER E
      DOUBLE PRECISION KE(MAX3,MAX3),KXE,KYE, AE, BE, CX, CY

      KXE = PROP(MATSET(E),1)
      KYE = PROP(MATSET(E),2)
      AE = ABS(X2(IN(E,1)) - X2(IN(E,3))) / 2
      BE = ABS(X1(IN(E,1)) - X1(IN(E,3))) / 2
      CX = KXE * AE / (6.0 * BE)
      CY = KYE * BE / (6.0 * AE)
      KE(1,1) =  2.0 * CX + 2.0 * CY
      KE(1,2) = -2.0 * CX + CY
      KE(1,3) = -CX - CY
      KE(1,4) =  CX - 2.0 * CY
      KE(2,1) =  KE(1,2)
      KE(2,2) =  KE(1,1)
      KE(2,3) =  KE(1,4)
      KE(2,4) =  KE(1,3)
      KE(3,1) =  KE(1,3)
      KE(3,2) =  KE(2,3)
      KE(3,3) =  KE(1,1)
      KE(3,4) =  KE(1,2)
      KE(4,1) =  KE(1,4)
      KE(4,2) =  KE(2,4)
      KE(4,3) =  KE(3,4)
      KE(4,4) =  KE(1,1)
      RETURN
      END
```

```
      SUBROUTINE KQUA4(E,KE)
C*********************************************************************
C
C       PURPOSE:
C           TO COMPUTE THE ELEMENT CONDUCTANCE MATRIX FOR TWO
C           DIMENSIONAL, LINEAR QUADRILATERAL ELEMENT
C
C         DEFINITIONS OF VARIABLES:
C               DETJAC = DETERMINANT OF JACOBIAN MATRIX
C             DNDXI(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                        FUNCTION WITH RESPECT TO XI AT NODE I
C              DNDX(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                        FUNCTION WITH RESPECT TO X AT NODE I
C            DNDETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                        FUNCTION WITH RESPECT TO ETA AT NODE I
C              DNDY(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                        FUNCTION WITH RESPECT TO Y AT NODE I
C                   E = ELEMENT NUMBER
C               ETA(I) = LOCATION OF GAUSS POINT IN ETA
C                        COORDINATE DIRECTION
C             JAC(I,J) = JACOBIAN MATRIX
C           JACINV(I,J) = INVERSE OF JACOBIAN MATRIX
C               KE(I,J) = ELEMENT CONDUCTANCE MATRIX
C                   KXE = HYDRAULIC CONDUCTIVITY IN X DIRECTION
C                   KYE = HYDRAULIC CONDUCTIVITY IN Y DIRECTION
C                  W(I) = WEIGHT FOR GAUSS POINT I
C            X1(IN(E,I) = X COORDINATE FOR NODE I, ELEMENT E
C            X2(IN(E,I) = Y COORDINATE FOR NODE I, ELEMENT E
C                 XI(I) = LOCATION OF GAUSS POINT IN XI COORDINATE
C                         DIRECTION
C
C*********************************************************************
      INCLUDE 'COMALL'
      DOUBLE PRECISION JAC(2,2), JACINV(2,2), KE(MAX3,MAX3), DNDXI(4)
      DOUBLE PRECISION DNDX(4), DNDETA(4), DNDY(4), W(2), XI(2)
      DOUBLE PRECISION ETA(2), SIGN1(4), SIGN2(4), KXE, KYE, DETJAC
      INTEGER E, I, J, K, N
      DATA SIGN1/-1.0, 1.0, 1.0, -1.0/
      DATA SIGN2/-1.0, -1.0, 1.0, 1.0/

      XI(1) = 1.0 / SQRT(3D0)
      XI(2) = -XI(1)
      ETA(1) = XI(1)
      ETA(2) = XI(2)
      W(1) = 1.0
      W(2) = 1.0
      KXE = PROP(MATSET(E),1)
      KYE = PROP(MATSET(E),2)

      DO 30 K = 1, 4
        DO 20 N = 1, 4
          KE(K,N) = 0.0
 20     CONTINUE
 30   CONTINUE

      DO 120 I = 1, 2
        DO 110 J = 1, 2

          DO 50 K = 1, 2
            DO 40 N = 1, 2
              JAC(K,N) = 0.0
 40         CONTINUE
 50       CONTINUE

          DO 60 N = 1,4
            DNDXI(N)  = 0.25 * SIGN1(N) * (1.0 + SIGN2(N) * ETA(J))
            DNDETA(N) = 0.25 * SIGN2(N) * (1.0 + SIGN1(N) * XI(I))
 60       CONTINUE
```

```
          DO 70 N = 1,4
            JAC(1,1) = JAC(1,1) + DNDXI(N) * X1(IN(E,N))
            JAC(1,2) = JAC(1,2) + DNDXI(N) * X2(IN(E,N))
            JAC(2,1) = JAC(2,1) + DNDETA(N) * X1(IN(E,N))
            JAC(2,2) = JAC(2,2) + DNDETA(N) * X2(IN(E,N))
 70       CONTINUE
          DETJAC = JAC(1,1) * JAC(2,2) - JAC(1,2) * JAC(2,1)
          JACINV(1,1) =  JAC(2,2) / DETJAC
          JACINV(1,2) = -JAC(1,2) / DETJAC
          JACINV(2,1) = -JAC(2,1) / DETJAC
          JACINV(2,2) =  JAC(1,1) / DETJAC
          DO 80 N = 1,4
            DNDX(N) = JACINV(1,1) * DNDXI(N) + JACINV(1,2) * DNDETA(N)
            DNDY(N) = JACINV(2,1) * DNDXI(N) + JACINV(2,2) * DNDETA(N)
 80       CONTINUE
          DO 100 K = 1, 4
            DO 90 N = 1, 4
              KE(K,N) = KE(K,N) + W(I) * W(J) * (KXE * DNDX(K) *
     1                  DNDX(N) + KYE * DNDY(K) * DNDY(N)) * DETJAC
 90         CONTINUE
 100      CONTINUE
 110    CONTINUE
 120  CONTINUE
      RETURN
      END

      SUBROUTINE KPAR8(E,KE)
C*****************************************************************
C
C         PURPOSE:
C           TO COMPUTE THE ELEMENT CONDUCTANCE MATRIX FOR THREE
C           DIMENSIONAL, LINEAR PARALLELEPIPED ELEMENT
C
C         DEFINITIONS OF VARIABLES:
C                DETJAC = DETERMINANT OF JACOBIAN MATRIX
C              DNDXI(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                         FUNCTION WITH RESPECT TO XI AT NODE I
C               DNDX(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                         FUNCTION WITH RESPECT TO X AT NODE I
C             DNDETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                         FUNCTION WITH RESPECT TO ETA AT NODE I
C               DNDY(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                         FUNCTION WITH RESPECT TO Y AT NODE I
C            DNDZETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                         FUNCTION WITH RESPECT TO ZETA AT NODE I
C               DNDZ(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                         FUNCTION WITH RESPECT TO Z AT NODE I
C                     E = ELEMENT NUMBER
C                ETA(I) = LOCATION OF GAUSS POINT IN ETA
C                         COORDINATE DIRECTION
C               IN(I,J) = NODE NUMBER J FOR ELEMENT I
C              JAC(I,J) = JACOBIAN MATRIX
C            JACINV(I,J) = INVERSE OF JACOBIAN MATRIX
C                KE(I,J) = ELEMENT CONDUCTANCE MATRIX
C                    KXE = HYDRAULIC CONDUCTIVITY IN X DIRECTION
C                    KYE = HYDRAULIC CONDUCTIVITY IN Y DIRECTION
C                    KZE = HYDRAULIC CONDUCTIVITY IN Z DIRECTION
C                   W(I) = WEIGHT FOR GAUSS POINT I
C             X1(IN(E,I) = X COORDINATE FOR NODE I, ELEMENT E
C             X2(IN(E,I) = Y COORDINATE FOR NODE I, ELEMENT E
C                  XI(I) = LOCATION OF GAUSS POINT IN XI COORDINATE
C                         DIRECTION
C               ZETA(I) = LOCATION OF GAUSS POINT IN ZETA COORDINATE
C                         DIRECTION
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I, J, K, L, N
```

```
      DOUBLE PRECISION JAC(3,3), JACINV(3,3), KE(MAX3,MAX3)
      DOUBLE PRECISION DNDXI(8), DNDX(8), DNDETA(8), DNDY(8)
      DOUBLE PRECISION DNDZETA(8), DNDZ(8), W(2), XI(2), ETA(2)
      DOUBLE PRECISION ZETA(2), SIGN1(8), SIGN2(8), SIGN3(8)
      DOUBLE PRECISION KXE, KYE, KZE, DETJAC
      DATA SIGN1/-1.0, 1.0, 1.0,-1.0,-1.0, 1.0, 1.0,-1.0/
      DATA SIGN2/-1.0,-1.0, 1.0, 1.0,-1.0,-1.0, 1.0, 1.0/
      DATA SIGN3/-1.0,-1.0,-1.0,-1.0, 1.0, 1.0, 1.0, 1.0/
      XI(1) = 1.0 / SQRT(3D0)
      XI(2) = -XI(1)
      ETA(1) = XI(1)
      ETA(2) = XI(2)
      ZETA(1) = XI(1)
      ZETA(2) = XI(2)
      W(1) = 1.0
      W(2) = 1.0
      KXE = PROP(MATSET(E),1)
      KYE = PROP(MATSET(E),2)
      KZE = PROP(MATSET(E),3)
      DO 20 K = 1, 8
        DO 10 N = 1, 8
          KE(K,N) = 0.0
10      CONTINUE
20    CONTINUE
      DO 120 I = 1, 2
        DO 110 J = 1, 2
          DO 100 K = 1, 2
            DO 40 L = 1, 3
              DO 30 N = 1, 3
                JAC(L,N) = 0.0
30            CONTINUE
40          CONTINUE
            DO 50 N = 1, 8
              DNDXI(N)   = 0.125 * SIGN1(N) * (1.0 + SIGN2(N) * ETA(J))
     1                     * (1.0 + SIGN3(N) * ZETA(K))
              DNDETA(N)  = 0.125 * SIGN2(N) * (1.0 + SIGN1(N) * XI(I))
     1                     * (1.0 + SIGN3(N) * ZETA(K))
              DNDZETA(N) = 0.125 * SIGN3(N) * (1.0 + SIGN1(N) * XI(I))
     1                     * (1.0 + SIGN2(N) * ETA(J))
50          CONTINUE
            DO 60 N = 1, 8
              JAC(1,1) = JAC(1,1) + DNDXI(N) * X1(IN(E,N))
              JAC(1,2) = JAC(1,2) + DNDXI(N) * X2(IN(E,N))
              JAC(1,3) = JAC(1,3) + DNDXI(N) * X3(IN(E,N))
              JAC(2,1) = JAC(2,1) + DNDETA(N) * X1(IN(E,N))
              JAC(2,2) = JAC(2,2) + DNDETA(N) * X2(IN(E,N))
              JAC(2,3) = JAC(2,3) + DNDETA(N) * X3(IN(E,N))
              JAC(3,1) = JAC(3,1) + DNDZETA(N) * X1(IN(E,N))
              JAC(3,2) = JAC(3,2) + DNDZETA(N) * X2(IN(E,N))
              JAC(3,3) = JAC(3,3) + DNDZETA(N) * X3(IN(E,N))
60          CONTINUE
            DETJAC = JAC(1,1) * (JAC(2,2)*JAC(3,3) - JAC(3,2)*JAC(2,3))
     1             - JAC(1,2) * (JAC(2,1)*JAC(3,3) - JAC(3,1)*JAC(2,3))
     2             - JAC(1,3) * (JAC(2,1)*JAC(3,2) - JAC(3,1)*JAC(2,2))
C           INVERSE JACOBIAN MATRIX FORMULA HAS BEEN TRANSPOSED STEVE 9/7/96
            JACINV(1,1) = ( JAC(2,2) * JAC(3,3) - JAC(2,3) * JAC(3,2))
     1                    / DETJAC
            JACINV(2,1) = (-JAC(2,1) * JAC(3,3) + JAC(2,3) * JAC(3,1))
     1                    / DETJAC
            JACINV(3,1) = ( JAC(2,1) * JAC(3,2) - JAC(3,1) * JAC(2,2))
     1                    / DETJAC
            JACINV(1,2) = (-JAC(1,2) * JAC(3,3) + JAC(1,3) * JAC(3,2))
     1                    / DETJAC
            JACINV(2,2) = ( JAC(1,1) * JAC(3,3) - JAC(1,3) * JAC(3,1))
     1                    / DETJAC
            JACINV(3,2) = (-JAC(1,1) * JAC(3,2) + JAC(1,2) * JAC(3,1))
     1                    / DETJAC
            JACINV(1,3) = ( JAC(1,2) * JAC(2,3) - JAC(1,3) * JAC(2,2))
```

```
     1                    / DETJAC
            JACINV(2,3) = (-JAC(1,1) * JAC(2,3) + JAC(1,3) * JAC(2,1))
     1                    / DETJAC
            JACINV(3,3) = ( JAC(1,1) * JAC(2,2) - JAC(1,2) * JAC(2,1))
     1                    / DETJAC
            DO 70 N = 1, 8
              DNDX(N) = JACINV(1,1) * DNDXI(N) + JACINV(1,2) *
     1                  DNDETA(N) + JACINV(1,3) * DNDZETA(N)
              DNDY(N) = JACINV(2,1) * DNDXI(N) + JACINV(2,2) *
     1                  DNDETA(N) + JACINV(2,3) * DNDZETA(N)
              DNDZ(N) = JACINV(3,1) * DNDXI(N) + JACINV(3,2) *
     1                  DNDETA(N) + JACINV(3,3) * DNDZETA(N)
 70         CONTINUE
            DO 90 L = 1, 8
              DO 80 N = 1, 8
                KE(L,N) = KE(L,N) + W(I) * W(J) * W(K) * DETJAC *
     1                    (KXE * DNDX(L) * DNDX(N) +
     2                     KYE * DNDY(L) * DNDY(N) +
     3                     KZE * DNDZ(L) * DNDZ(N) )
 80           CONTINUE
 90         CONTINUE
100       CONTINUE
110     CONTINUE
120   CONTINUE
      RETURN
      END



SUBROUTINE ASMBK
C*****************************************************************
C
C 12.1  PURPOSE:
C         SUBROUTINE ASMBK ASSEMBLES THE GLOBAL CONDUCTANCE MATRIX
C         AND THE GLOBAL SPECIFIED FLOW MATRIX.  THE GLOBAL MATRICES
C         ARE MODIFIED DURING THE ASSEMBLY PROCESS TO ACCOUNT FOR
C         SPECIFIED VALUES OF THE FIELD VARIABLE AND GROUNDWATER
C         FLOW DURING THE ASSEMBLY PROCESS.  THE GLOBAL CONDUCTANCE
C         MATRIX IS ASSEMBLED AND MODIFIED IN VECTOR STORAGE.  ASMBK
C         ALSO COMPUTES THE SEMI-BANDWIDTH AND THE NUMBER OF DEGREES
C         OF FREEDOM FOR THE MODIFIED GLOBAL CONDUCTANCE MATRIX.
C
C 12.2  INPUT:
C         NONE
C
C 12.3  OUTPUT:
C         SEMI-BANDWIDTH AND NUMBER OF DEGREES OF FREEDOM WRITTEN TO
C         THE USER-DEFINED FILE "OUTF"
C
C 12.4  DEFINITIONS OF VARIABLES:
C                B(I) = MODIFIED SPECIFIED FLOW MATRIX
C                   E = ELEMENT NUMBER
C          ELEMTYP(I) = ELEMENT TYPE FOR ELEMENT I
C             FLUX(I) = SPECIFIED VALUE OF GROUNDWATER FLOW AT NODE I
C              ICH(I) = 1 IF THE VALUE OF THE FIELD VARIABLE IS
C                       SPECIFIED FOR NODE I
C                     = 0 OTHERWISE
C              IJSIZE = LENGTH OF ARRAY M
C             KE(I,J) = CONDUCTANCE MATRIX FOR ELEMENT E IN FULL
C                       MATRIX STORAGE
C              LCH(I) = ICH(I) + ICH(I-1) + ICH(I-2) + ...
C                       THE ARRAYSICH AND LCH ARE USED TO MODIFY
C                       GLOBAL SYSTEM OF EQUATIONS IN SUBROUTINES
C                       ASMBK, ASMBKC, AND ASMBAD
C               M(IJ) = MODIFIED GLOBAL CONDUCTANCE MATRIX IN
C                       VECTOR STORAGE
C                NDOF = NUMBER OF DEGREES OF FREEDOM (UNSPECIFIED NODES)
C          NODETBL(I) = NUMBER OF NODES IN ELEMENT TYPE I
```

```
C              NUMELM = NUMBER OF ELEMENTS IN MESH
C                 SBW = SEMI-BANDWIDTH OF MODIFIED GLOBAL
C                       CONDUCTANCE MATRIX
C               X(I) = VALUE OF THE FIELD VARIABLE AT NODE I
C
C 12.5  USAGE:
C         THE SEMI-BANDWIDTH OF THE GLOBAL CONDUCTANCE MATRIX IS
C         COMPUTED FIRST.  THEN THE ENTRIES OF THE ELEMENT
C         CONDUCTANCE MATRIX ARE COMPUTED IN A SET OF SUBROUTINES,
C         ONE SUBROUTINE FOR EACH ELEMENT TYPE.  THE GLOBAL
C         CONDUCTANCE MATRIX FOR THE MESH IS ASSEMBLED BY ADDING
C         THE CORRESPONDING ENTRIES OF THE ELEMENT CONDUCTANCE
C         MATRICES TO THE GLOBAL CONDUCTANCE MATRIX.  DURING THE
C         ASSEMBLY PROCESS THE GLOBAL CONDUCTANCE MATRIX IS MODIFIED
C         FOR SPECIFIED VALUES OF HEAD.  SPECIFIED VALUES OF
C         GROUNDWATER FLOW ARE ADDED TO THE GLOBAL FLOW MATRIX.
C
C       SUBROUTINES CALLED:   (IF I HAVE BOTHERED TO INCLUDE THEM)
C         KBAR2,KBAR3,KBAR4,KTRI3,KREC4,KQUA4,KQUA8,KQUA12,KPAR8,
C         KPAR20,KPAR32,KTRI3A,KREC4A,KREP8
C         LOCATE
C
C*****************************************************************
        INCLUDE 'COMALL'
        DOUBLE PRECISION KE(MAX3,MAX3)
        INTEGER NODETBL(14), E, I, KI, II, J, KJ, JJ
        DATA NODETBL/2,3,4,3,4,4,8,12,8,20,32,3,4,8/

C       INITIALISE ENTRIES OF GLOBAL CONDUCTANCE MATRIX TO ZERO
        IF (NDOF .GT. MAX0) STOP'** EXCEEDS MAX DEGREES OF FREEDOM**'
        DO 10 J = 1, NDOF
          DO 10 I = 1, SBW + 1
            M1(I,J) = 0.0
 10     CONTINUE
        DO 60 I = 1, NUMNOD
          IF (ICH(I) .EQ. 0) B(I-LCH(I)) = FLUX(I)
 60     CONTINUE
C       LOOP ON THE NUMBER OF ELEMENTS
        DO 90 E = 1, NUMELM
C       COMPUTE THE ELEMENT CONDUCTANCE MATRIX FOR THIS ELEMENT TYPE
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        IF (ELEMTYP(E) .EQ. 1) THEN
C         ELEMENT IS A LINEAR BAR
          CALL KBAR2(E,KE)
        ELSEIF (ELEMTYP(E) .EQ. 4) THEN
C         ELEMENT IS A LINEAR TRIANGLE
          CALL KTRI3(E,KE)
        ELSEIF (ELEMTYP(E) .EQ. 5) THEN
C         ELEMENT IS A LINEAR RECTANGLE
          CALL KREC4(E,KE)
        ELSEIF (ELEMTYP(E) .EQ. 6) THEN
C         ELEMENT IS A LINEAR QUADTILATERAL
          CALL KQUA4(E,KE)
        ELSEIF (ELEMTYP(E) .EQ. 9) THEN
C         ELEMENT IS A THREE DIMENSIONAL LINEAR PARALLELEPIPED
          CALL KPAR8(E,KE)
        ENDIF
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C       ADD THE ELEMENT CONDUCTANCE MATRIX TO THE GLOBAL MATRIX
        DO 80 I = 1, NODETBL(ELEMTYP(E))
          KI = IN(E,I)
          IF (ICH(KI) .EQ. 0) THEN
            II = KI - LCH(KI)
            DO 70 J = 1, NODETBL(ELEMTYP(E))
              KJ = IN(E,J)
              IF (ICH(KJ) .NE. 0) THEN
                B(II) = B(II) - KE(I,J) * X(KJ)
              ELSEIF (KJ .GE. KI) THEN
```

```
c              ELSEIF (J .GE. I) THEN
                JJ = KJ - LCH(KJ)
                M1(SBW+1+II-JJ,JJ) = M1(SBW+1+II-JJ,JJ) + KE(I,J)
C                IF (I .EQ. J .AND. KE(I,J) .LT. 0.0) THEN
C                  PRINT *,' There is a problem with element',E,', that'
C                  PRINT *,' causes a negative entry for node',IN(E,I)
C                ENDIF
              ENDIF
 70         CONTINUE
          ENDIF
 80       CONTINUE
 90       CONTINUE
          RETURN
          END

          SUBROUTINE KBAR2(E,KE)
C*****************************************************************
C
C         PURPOSE:
C           TO COMPUTE THE ELEMENT CONDICTANCE MATRIX FOR A
C           ONE-DIMENSIONAL, LINEAR BAR ELEMENT
C
C         DEFINITIONS OF VARIABLES:
C                  E = ELEMENT NUMBER
C           KE(I,J) = ELEMENT CONDUCTANCE MATRIC
C               KXE = HYDRAULIC CONDUCTIVITY IN X COOREDINATE DIRECTION
C                LE = ELEMENT LENGTH
C
C*****************************************************************
          INCLUDE 'COMALL'
          INTEGER E
          DOUBLE PRECISION KE(MAX3,MAX3),KXE,LE

          KXE = PROP(MATSET(E),1)
          LE = ABS(X1(IN(E,2)) - X1(IN(E,1)))
          KE(1,1) =  KXE / LE
          KE(1,2) = -KE(1,1)
          KE(2,1) = -KE(1,1)
          KE(2,2) =  KE(1,1)
          RETURN
          END

          SUBROUTINE KTRI3(E,KE)
C*****************************************************************
C
C           PURPOSE:
C             TO COMPUTE THE ELEMENT CONDUCTANCE MATRIX FOR TWO
C             DIMENSIONAL, LINEAR TRIANGLE ELEMENT
C
C           DEFINITIONS OF VARIABLES:
C                AE4 = FOUR TIMES ELEMENT AREA
C                  E = ELEMENT NUMBER
C             IN(I,J) = NODE NUMBER J FOR ELEMENT I
C             KE(I,J) = ELEMENT CONDUCTANCE MATRIX
C                 KXE = HYDRAULIC CONDUCTIVITY IN X DIRECTION
C                 KYE = HYDRAULIC CONDUCTIVITY IN Y DIRECTION
C
C*****************************************************************
         INCLUDE 'COMALL'
         INTEGER E, I, J
         DOUBLE PRECISION KE(MAX3,MAX3),KXE,KYE,BE(3),CE(3),AE4

         KXE = PROP(MATSET(E),1)
         KYE = PROP(MATSET(E),2)
         BE(1) = X2(IN(E,2)) - X2(IN(E,3))
         BE(2) = X2(IN(E,3)) - X2(IN(E,1))
         BE(3) = X2(IN(E,1)) - X2(IN(E,2))
         CE(1) = X1(IN(E,3)) - X1(IN(E,2))
```

```
      CE(2) = X1(IN(E,1)) - X1(IN(E,3))
      CE(3) = X1(IN(E,2)) - X1(IN(E,1))
C  CHECK THIS FORMULA!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      AE4 = ( X1(IN(E,2)) * X2(IN(E,3)) + X1(IN(E,1)) * X2(IN(E,2)) +
     1        X2(IN(E,1)) * X1(IN(E,3)) - X2(IN(E,3)) * X1(IN(E,1)) -
     2        X1(IN(E,3)) * X2(IN(E,2)) - X1(IN(E,2)) * X2(IN(E,1)) )*2
      DO 20 I = 1,3
        DO 10 J = 1,3
          KE(I,J) = (KXE * BE(I) * BE(J) + KYE * CE(I) * CE(J)) / AE4
 10     CONTINUE
 20   CONTINUE
      RETURN
      END

        SUBROUTINE KREC4(E,KE)
C*****************************************************************
C
C         PURPOSE:
C           TO COMPUTE THE ELEMENT CONDUCTANCE MATRIX FOR TWO
C           DIMENSIONAL, LINEAR RECTANGLE ELEMENT
C
C         DEFINITIONS OF VARIABLES:
C                 E = ELEMENT NUMBER
C           IN(I,J) = NODE NUMBER J FOR ELEMENT I
C           KE(I,J) = ELEMENT CONDUCTANCE MATRIX
C               KXE = HYDRAULIC CONDUCTIVITY IN X DIRECTION
C               KYE = HYDRAULIC CONDUCTIVITY IN Y DIRECTION
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E
      DOUBLE PRECISION KE(MAX3,MAX3),KXE,KYE, AE, BE, CX, CY

      KXE = PROP(MATSET(E),1)
      KYE = PROP(MATSET(E),2)
      AE = ABS(X2(IN(E,1)) - X2(IN(E,3))) / 2
      BE = ABS(X1(IN(E,1)) - X1(IN(E,3))) / 2
      CX = KXE * AE / (6.0 * BE)
      CY = KYE * BE / (6.0 * AE)
      KE(1,1) =  2.0 * CX + 2.0 * CY
      KE(1,2) = -2.0 * CX + CY
      KE(1,3) = -CX - CY
      KE(1,4) =  CX - 2.0 * CY
      KE(2,1) =  KE(1,2)
      KE(2,2) =  KE(1,1)
      KE(2,3) =  KE(1,4)
      KE(2,4) =  KE(1,3)
      KE(3,1) =  KE(1,3)
      KE(3,2) =  KE(2,3)
      KE(3,3) =  KE(1,1)
      KE(3,4) =  KE(1,2)
      KE(4,1) =  KE(1,4)
      KE(4,2) =  KE(2,4)
      KE(4,3) =  KE(3,4)
      KE(4,4) =  KE(1,1)
      RETURN
      END

      SUBROUTINE KQUA4(E,KE)
C*****************************************************************
C
C       PURPOSE:
C           TO COMPUTE THE ELEMENT CONDUCTANCE MATRIX FOR TWO
C           DIMENSIONAL, LINEAR QUADRILATERAL ELEMENT
C
C         DEFINITIONS OF VARIABLES:
C               DETJAC = DETERMINANT OF JACOBIAN MATRIX
C             DNDXI(I) = PARTIAL DERIVATIVE OF INTERPOLATION
```

```
C                          FUNCTION WITH RESPECT TO XI AT NODE I
C               DNDX(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                          FUNCTION WITH RESPECT TO X AT NODE I
C             DNDETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                          FUNCTION WITH RESPECT TO ETA AT NODE I
C               DNDY(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                          FUNCTION WITH RESPECT TO Y AT NODE I
C                     E = ELEMENT NUMBER
C                ETA(I) = LOCATION OF GAUSS POINT IN ETA
C                          COORDINATE DIRECTION
C              JAC(I,J) = JACOBIAN MATRIX
C            JACINV(I,J) = INVERSE OF JACOBIAN MATRIX
C                KE(I,J) = ELEMENT CONDUCTANCE MATRIX
C                    KXE = HYDRAULIC CONDUCTIVITY IN X DIRECTION
C                    KYE = HYDRAULIC CONDUCTIVITY IN Y DIRECTION
C                   W(I) = WEIGHT FOR GAUSS POINT I
C             X1(IN(E,I) = X COORDINATE FOR NODE I, ELEMENT E
C             X2(IN(E,I) = Y COORDINATE FOR NODE I, ELEMENT E
C                  XI(I) = LOCATION OF GAUSS POINT IN XI COORDINATE
C                          DIRECTION
C
C*********************************************************************
      INCLUDE 'COMALL'
      DOUBLE PRECISION JAC(2,2), JACINV(2,2), KE(MAX3,MAX3), DNDXI(4)
      DOUBLE PRECISION DNDX(4), DNDETA(4), DNDY(4), W(2), XI(2)
      DOUBLE PRECISION ETA(2), SIGN1(4), SIGN2(4), KXE, KYE, DETJAC
      INTEGER E, I, J, K, N
      DATA SIGN1/-1.0, 1.0, 1.0, -1.0/
      DATA SIGN2/-1.0, -1.0, 1.0, 1.0/

      XI(1) = 1.0 / SQRT(3D0)
      XI(2) = -XI(1)
      ETA(1) = XI(1)
      ETA(2) = XI(2)
      W(1) = 1.0
      W(2) = 1.0
      KXE = PROP(MATSET(E),1)
      KYE = PROP(MATSET(E),2)

      DO 30 K = 1, 4
        DO 20 N = 1, 4
          KE(K,N) = 0.0
 20     CONTINUE
 30   CONTINUE

      DO 120 I = 1, 2
        DO 110 J = 1, 2

          DO 50 K = 1, 2
            DO 40 N = 1, 2
              JAC(K,N) = 0.0
 40         CONTINUE
 50       CONTINUE

          DO 60 N = 1,4
            DNDXI(N)  = 0.25 * SIGN1(N) * (1.0 + SIGN2(N) * ETA(J))
            DNDETA(N) = 0.25 * SIGN2(N) * (1.0 + SIGN1(N) * XI(I))
 60       CONTINUE
          DO 70 N = 1,4
            JAC(1,1) = JAC(1,1) + DNDXI(N) * X1(IN(E,N))
            JAC(1,2) = JAC(1,2) + DNDXI(N) * X2(IN(E,N))
            JAC(2,1) = JAC(2,1) + DNDETA(N) * X1(IN(E,N))
            JAC(2,2) = JAC(2,2) + DNDETA(N) * X2(IN(E,N))
 70       CONTINUE
          DETJAC = JAC(1,1) * JAC(2,2) - JAC(1,2) * JAC(2,1)
          JACINV(1,1) =  JAC(2,2) / DETJAC
          JACINV(1,2) = -JAC(1,2) / DETJAC
          JACINV(2,1) = -JAC(2,1) / DETJAC
```

```
          JACINV(2,2) =  JAC(1,1) / DETJAC
          DO 80 N = 1,4
            DNDX(N) = JACINV(1,1) * DNDXI(N) + JACINV(1,2) * DNDETA(N)
            DNDY(N) = JACINV(2,1) * DNDXI(N) + JACINV(2,2) * DNDETA(N)
 80       CONTINUE
          DO 100 K = 1, 4
            DO 90 N = 1, 4
              KE(K,N) = KE(K,N) + W(I) * W(J) * (KXE * DNDX(K) *
     1                  DNDX(N) + KYE * DNDY(K) * DNDY(N)) * DETJAC
 90         CONTINUE
 100      CONTINUE
 110    CONTINUE
 120  CONTINUE
      RETURN
      END

      SUBROUTINE KPAR8(E,KE)
C*****************************************************************
C
C         PURPOSE:
C           TO COMPUTE THE ELEMENT CONDUCTANCE MATRIX FOR THREE
C           DIMENSIONAL, LINEAR PARALLELEPIPED ELEMENT
C
C         DEFINITIONS OF VARIABLES:
C               DETJAC = DETERMINANT OF JACOBIAN MATRIX
C             DNDXI(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                        FUNCTION WITH RESPECT TO XI AT NODE I
C              DNDX(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                        FUNCTION WITH RESPECT TO X AT NODE I
C            DNDETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                        FUNCTION WITH RESPECT TO ETA AT NODE I
C              DNDY(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                        FUNCTION WITH RESPECT TO Y AT NODE I
C           DNDZETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                        FUNCTION WITH RESPECT TO ZETA AT NODE I
C              DNDZ(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                        FUNCTION WITH RESPECT TO Z AT NODE I
C                    E = ELEMENT NUMBER
C               ETA(I) = LOCATION OF GAUSS POINT IN ETA
C                        COORDINATE DIRECTION
C              IN(I,J) = NODE NUMBER J FOR ELEMENT I
C             JAC(I,J) = JACOBIAN MATRIX
C          JACINV(I,J) = INVERSE OF JACOBIAN MATRIX
C              KE(I,J) = ELEMENT CONDUCTANCE MATRIX
C                  KXE = HYDRAULIC CONDUCTIVITY IN X DIRECTION
C                  KYE = HYDRAULIC CONDUCTIVITY IN Y DIRECTION
C                  KZE = HYDRAULIC CONDUCTIVITY IN Z DIRECTION
C                W(I) = WEIGHT FOR GAUSS POINT I
C           X1(IN(E,I) = X COORDINATE FOR NODE I, ELEMENT E
C           X2(IN(E,I) = Y COORDINATE FOR NODE I, ELEMENT E
C                XI(I) = LOCATION OF GAUSS POINT IN XI COORDINATE
C                        DIRECTION
C              ZETA(I) = LOCATION OF GAUSS POINT IN ZETA COORDINATE
C                        DIRECTION
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I, J, K, L, N
      DOUBLE PRECISION JAC(3,3), JACINV(3,3), KE(MAX3,MAX3)
      DOUBLE PRECISION DNDXI(8), DNDX(8), DNDETA(8), DNDY(8)
      DOUBLE PRECISION DNDZETA(8), DNDZ(8), W(2), XI(2), ETA(2)
      DOUBLE PRECISION ZETA(2), SIGN1(8), SIGN2(8), SIGN3(8)
      DOUBLE PRECISION KXE, KYE, KZE, DETJAC
      DATA SIGN1/-1.0, 1.0, 1.0,-1.0,-1.0, 1.0, 1.0,-1.0/
      DATA SIGN2/-1.0,-1.0, 1.0, 1.0,-1.0,-1.0, 1.0, 1.0/
      DATA SIGN3/-1.0,-1.0,-1.0,-1.0, 1.0, 1.0, 1.0, 1.0/
      XI(1) = 1.0 / SQRT(3D0)
      XI(2) = -XI(1)
```

```
      ETA(1) = XI(1)
      ETA(2) = XI(2)
      ZETA(1) = XI(1)
      ZETA(2) = XI(2)
      W(1) = 1.0
      W(2) = 1.0
      KXE = PROP(MATSET(E),1)
      KYE = PROP(MATSET(E),2)
      KZE = PROP(MATSET(E),3)
      DO 20 K = 1, 8
        DO 10 N = 1, 8
          KE(K,N) = 0.0
 10     CONTINUE
 20   CONTINUE
      DO 120 I = 1, 2
        DO 110 J = 1, 2
          DO 100 K = 1, 2
            DO 40 L = 1, 3
              DO 30 N = 1, 3
                JAC(L,N) = 0.0
 30           CONTINUE
 40         CONTINUE
            DO 50 N = 1, 8
              DNDXI(N)    = 0.125 * SIGN1(N) * (1.0 + SIGN2(N) * ETA(J))
     1                      * (1.0 + SIGN3(N) * ZETA(K))
              DNDETA(N)   = 0.125 * SIGN2(N) * (1.0 + SIGN1(N) * XI(I))
     1                      * (1.0 + SIGN3(N) * ZETA(K))
              DNDZETA(N)  = 0.125 * SIGN3(N) * (1.0 + SIGN1(N) * XI(I))
     1                      * (1.0 + SIGN2(N) * ETA(J))
 50         CONTINUE
            DO 60 N = 1, 8
              JAC(1,1) = JAC(1,1) + DNDXI(N) * X1(IN(E,N))
              JAC(1,2) = JAC(1,2) + DNDXI(N) * X2(IN(E,N))
              JAC(1,3) = JAC(1,3) + DNDXI(N) * X3(IN(E,N))
              JAC(2,1) = JAC(2,1) + DNDETA(N) * X1(IN(E,N))
              JAC(2,2) = JAC(2,2) + DNDETA(N) * X2(IN(E,N))
              JAC(2,3) = JAC(2,3) + DNDETA(N) * X3(IN(E,N))
              JAC(3,1) = JAC(3,1) + DNDZETA(N) * X1(IN(E,N))
              JAC(3,2) = JAC(3,2) + DNDZETA(N) * X2(IN(E,N))
              JAC(3,3) = JAC(3,3) + DNDZETA(N) * X3(IN(E,N))
 60         CONTINUE
            DETJAC = JAC(1,1) * (JAC(2,2)*JAC(3,3) - JAC(3,2)*JAC(2,3))
     1             - JAC(1,2) * (JAC(2,1)*JAC(3,3) - JAC(3,1)*JAC(2,3))
     2             - JAC(1,3) * (JAC(2,1)*JAC(3,2) - JAC(3,1)*JAC(2,2))
C           INVERSE JACOBIAN MATRIX FORMULA HAS BEEN TRANSPOSED STEVE 9/7/96
            JACINV(1,1) = ( JAC(2,2) * JAC(3,3) - JAC(2,3) * JAC(3,2))
     1                    / DETJAC
            JACINV(2,1) = (-JAC(2,1) * JAC(3,3) + JAC(2,3) * JAC(3,1))
     1                    / DETJAC
            JACINV(3,1) = ( JAC(2,1) * JAC(3,2) - JAC(3,1) * JAC(2,2))
     1                    / DETJAC
            JACINV(1,2) = (-JAC(1,2) * JAC(3,3) + JAC(1,3) * JAC(3,2))
     1                    / DETJAC
            JACINV(2,2) = ( JAC(1,1) * JAC(3,3) - JAC(1,3) * JAC(3,1))
     1                    / DETJAC
            JACINV(3,2) = (-JAC(1,1) * JAC(3,2) + JAC(1,2) * JAC(3,1))
     1                    / DETJAC
            JACINV(1,3) = ( JAC(1,2) * JAC(2,3) - JAC(1,3) * JAC(2,2))
     1                    / DETJAC
            JACINV(2,3) = (-JAC(1,1) * JAC(2,3) + JAC(1,3) * JAC(2,1))
     1                    / DETJAC
            JACINV(3,3) = ( JAC(1,1) * JAC(2,2) - JAC(1,2) * JAC(2,1))
     1                    / DETJAC
            DO 70 N = 1, 8
              DNDX(N) = JACINV(1,1) * DNDXI(N) + JACINV(1,2) *
     1                  DNDETA(N) + JACINV(1,3) * DNDZETA(N)
              DNDY(N) = JACINV(2,1) * DNDXI(N) + JACINV(2,2) *
     1                  DNDETA(N) + JACINV(2,3) * DNDZETA(N)
```

```
              DNDZ(N) = JACINV(3,1) * DNDXI(N) + JACINV(3,2) *
     1                  DNDETA(N) + JACINV(3,3) * DNDZETA(N)
 70         CONTINUE
            DO 90 L = 1, 8
              DO 80 N = 1, 8
                KE(L,N) = KE(L,N) + W(I) * W(J) * W(K) * DETJAC *
     1                    (KXE * DNDX(L) * DNDX(N) +
     2                     KYE * DNDY(L) * DNDY(N) +
     3                     KZE * DNDZ(L) * DNDZ(N) )
 80           CONTINUE
 90         CONTINUE
100       CONTINUE
110     CONTINUE
120   CONTINUE
      RETURN
      END

      SUBROUTINE HDER1(E,KE,S1)
C*****************************************************************
C
C       PURPOSE:
C         SUBROUTINE HDER DETERMINES THE FIRST DERIVATIVE OF
C         EACH NODAL HYDRAULIC HEAD VALUE WITH RESPECT TO THE LOG OF THE
C         HYDRAULIC CONDUCTIVITY OF ELEMENT E.  THESE VALUES
C         ARE THEN STORED IN THE TEMPORARY FILE "DHDK1.UNF".
C
C       INPUT:
C         NONE
C
C       OUTPUT:
C         NONE
C
C       DEFINITIONS OF VARIABLES:
C                B(I) = RIGHT HAND SIDE VECTOR
C                     = [dKe/dke] [X]
C           DHDK(I,1) = DERIVATIVE OF HEAD AT NODE I WITH RESPECT TO
C                       HYDRAULIC CONDUCTIVITY OF ELEMENT E
C                   E = ELEMENT NUMBER
C          ELEMTYP(E) = ELEMENT TYPE FOR ELEMENT E
C              ICH(I) = 1 IF THE VALUE OF THE FIELD VARIABLE IS
C                       SPECIFIED FOR NODE I
C                     = 0 OTHERWISE
C             KE(I,J) = CONDUCTANCE MATRIX FOR ELEMENT E IN FULL
C                       MATRIX STORAGE
C              LCH(I) = ICH(I) + ICH(I-1) + ICH(I-2) + ...
C                       THE ARRAYS ICH AND LCH ARE USED TO MODIFY
C                       GLOBAL SYSTEM OF EQUATIONS IN SUBROUTINES
C                       ASMBK, ASMBKC, AND ASMBAD
C               M(IJ) = MODIFIED GLOBAL CONDUCTANCE MATRIX IN
C                       VECTOR STORAGE
C                NDOF = NUMBER OF DEGREES OF FREEDOM (UNSPECIFIED NODES)
C           NODETBL(I) = NUMBER OF NODES IN ELEMENT TYPE I
C              NUMELM = NUMBER OF ELEMENTS IN MESH
C                 SBW = SEMI-BANDWIDTH OF MODIFIED GLOBAL
C                       CONDUCTANCE MATRIX
C               X(I) = VALUE OF THE FIELD VARIABLE AT NODE I
C
C       USAGE:
C         DHDK IS DETERMINED USING:
C                M * DHDK = - DMDK * H
C         D2HDK2 IS DETERMINED USING:
C                M * D2HDK2(I,J) = - DMDK(I) * DHDK(J)
C                                  - DMDK(J) * DHDK(I)
C
C       SUBROUTINES CALLED:   (IF I HAVE BOTHERED TO INCLUDE THEM)
C         KBAR2,KBAR3,KBAR4,KTRI3,KREC4,KQUA4,KQUA8,KQUA12,KPAR8,
C         KPAR20,KPAR32,KTRI3A,KREC4A,KREP8
C         LOC
```

```
C
C*********************************************************************
      INCLUDE 'COMALL'
      DOUBLE PRECISION KE(MAX3,MAX3)
      INTEGER E, I, II, J, KI, KJ, NODETBL(13), S1, ERROR
      DATA NODETBL/2,3,4,3,4,4,8,12,8,20,32,3,4/
C     COMPUTE THE ELEMENT CONDUCTANCE MATRIX FOR THIS ELEMENT TYPE
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      IF (ELEMTYP(E) .EQ. 1) THEN
C       ELEMENT IS A LINEAR BAR
        CALL KBAR2(E,KE)
      ELSEIF (ELEMTYP(E) .EQ. 4) THEN
C       ELEMENT IS A LINEAR TRIANGLE
        CALL KTRI3(E,KE)
      ELSEIF (ELEMTYP(E) .EQ. 5) THEN
C       ELEMENT IS A LINEAR RECTANGLE
        CALL KREC4(E,KE)
      ELSEIF (ELEMTYP(E) .EQ. 6) THEN
C       ELEMENT IS A LINEAR QUADRILATERAL
        CALL KQUA4(E,KE)
      ELSEIF (ELEMTYP(E) .EQ. 9) THEN
C       ELEMENT IS A THREE DIMENSIONAL LINEAR PARALLELEPIPED
        CALL KPAR8(E,KE)
      ENDIF
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     DETERMINE FIRST DERIVATIVES
C     MULTIPLY THE DERIVATIVE OF THE ELEMENT CONDUCTANCE MATRIX
C     BY THE HYDRAULIC HEAD VECTOR TO GET THE RIGHT HAND SIDE
      DO 10 I = 1, NDOF
        B(I) = 0.0
 10   CONTINUE
      DO 30 I = 1, NODETBL(ELEMTYP(E))
        KI = IN(E,I)
        IF (ICH(KI) .EQ. 0) THEN
          II = KI - LCH(KI)
          DO 20 J = 1, NODETBL(ELEMTYP(E))
            KJ = IN(E,J)
            B(II) = B(II) - KE(I,J) * X(KJ)
 20       CONTINUE
        ENDIF
 30   CONTINUE
      CALL DPBTRS ('UPPER',NDOF,SBW,1,M1,MAX6+1,B,MAX0,ERROR)
      IF (ERROR .NE. 0) THEN
        PRINT *, 'ERROR IN SOLVING 1ST DERIVE',ERROR,'ELEMENT=',E
      ELSE
        DO 40 I = 1, NDOF
          DHDK(I,S1) = B(I)
 40     CONTINUE
      ENDIF
      RETURN
      END

      SUBROUTINE HDER2(E,E2,KE)
C*********************************************************************
C
C       PURPOSE:
C         SUBROUTINE HDER DETERMINES THE FIRST DERIVATIVE OF
C         EACH NODAL HYDRAULIC HEAD VALUE WITH RESPECT TO THE LOG OF THE
C         HYDRAULIC CONDUCTIVITY OF ELEMENT E.  THESE VALUES
C         ARE THEN STORED IN THE TEMPORARY FILE "DHDK1.UNF".
C
C       INPUT:
C         NONE
C
C       OUTPUT:
C         NONE
C
C       DEFINITIONS OF VARIABLES:
```

```
C               B(I) = RIGHT HAND SIDE VECTOR
C                    = [dKe/dke] [X]
C          DHDK(I,1) = DERIVATIVE OF HEAD AT NODE I WITH RESPECT TO
C                      HYDRAULIC CONDUCTIVITY OF ELEMENT E
C                  E = ELEMENT NUMBER
C         ELEMTYP(E) = ELEMENT TYPE FOR ELEMENT E
C             ICH(I) = 1 IF THE VALUE OF THE FIELD VARIABLE IS
C                      SPECIFIED FOR NODE I
C                    = 0 OTHERWISE
C           KE(I,J) = CONDUCTANCE MATRIX FOR ELEMENT E IN FULL
C                      MATRIX STORAGE
C             LCH(I) = ICH(I) + ICH(I-1) + ICH(I-2) + ...
C                      THE ARRAYS ICH AND LCH ARE USED TO MODIFY
C                      GLOBAL SYSTEM OF EQUATIONS IN SUBROUTINES
C                      ASMBK, ASMBKC, AND ASMBAD
C              M(IJ) = MODIFIED GLOBAL CONDUCTANCE MATRIX IN
C                      VECTOR STORAGE
C               NDOF = NUMBER OF DEGREES OF FREEDOM (UNSPECIFIED NODES)
C         NODETBL(I) = NUMBER OF NODES IN ELEMENT TYPE I
C             NUMELM = NUMBER OF ELEMENTS IN MESH
C                SBW = SEMI-BANDWIDTH OF MODIFIED GLOBAL
C                      CONDUCTANCE MATRIX
C              X(I) = VALUE OF THE FIELD VARIABLE AT NODE I
C
C       USAGE:
C         DHDK IS DETERMINED USING:
C                M * DHDK = - DMDK * H
C         D2HDK2 IS DETERMINED USING:
C                M * D2HDK2(I,J) = - DMDK(I) * DHDK(J)
C                                  - DMDK(J) * DHDK(I)
C
C       SUBROUTINES CALLED:    (IF I HAVE BOTHERED TO INCLUDE THEM)
C         KBAR2,KBAR3,KBAR4,KTRI3,KREC4,KQUA4,KQUA8,KQUA12,KPAR8,
C         KPAR20,KPAR32,KTRI3A,KREC4A,KREP8
C         LOC
C
C*****************************************************************
      INCLUDE 'COMALL'
      DOUBLE PRECISION KE(MAX3,MAX3), KE2(MAX3,MAX3)
      INTEGER E, E2, I, II, J, KI, KJ, NODETBL(13), ERROR
      DATA NODETBL/2,3,4,3,4,4,8,12,8,20,32,3,4/
      IF (E .EQ. E2) THEN
C       E2 = E =>  KE2 = KE  ETC.
        DO 20 I = 1, NODETBL(ELEMTYP(E))
          DO 10 J = 1, NODETBL(ELEMTYP(E))
            KE2(I,J) = KE (I,J)
 10       CONTINUE
 20     CONTINUE
        DO 30 I = 1, NDOF
          B(I) = 0.0
          DHDK(I,2) = DHDK(I,1)
 30     CONTINUE
      ELSE
        CALL HDER1(E2,KE2,2)
        DO 40 I = 1, NDOF
          B(I) = 0.0
 40     CONTINUE
      ENDIF
C     MULTIPLY THE DERIVATIVE OF THE ELEMENT CONDUCTANCE MATRIX
C     WITH RESPECT TO ELEMENT E BY THE DERIVATIVE OF THE
C     HYDRAULIC HEAD VECTOR WITH RESPECT TO ELEMENT E2 TO GET
C     THE RIGHT HAND SIDE:
      DO 80 I = 1, NODETBL(ELEMTYP(E))
        KI = IN(E,I)
        IF (ICH(KI) .EQ. 0) THEN
          II = KI - LCH(KI)
          DO 70 J = 1, NODETBL(ELEMTYP(E))
            KJ = IN(E,J)
```

```
            IF (ICH(KJ) .EQ. 0) THEN
              B(II) = B(II) - KE(I,J) * DHDK(KJ-LCH(KJ),2)
            ENDIF
 70       CONTINUE
        ENDIF
 80   CONTINUE
      IF (E .EQ. E2 ) THEN
        DO 90 I = 1, NDOF
          B(I) = 2.0 * B(I)
 90     CONTINUE
        DO 110 I = 1, NODETBL(ELEMTYP(E))
          KI = IN(E,I)
          IF (ICH(KI) .EQ. 0) THEN
            II = KI - LCH(KI)
            DO 100 J = 1, NODETBL(ELEMTYP(E))
              KJ = IN(E,J)
              B(II) = B(II) - KE(I,J) * X(KJ)
100         CONTINUE
          ENDIF
110     CONTINUE
      ELSE
C       SWITCH E AND E2:
        DO 130 I = 1, NODETBL(ELEMTYP(E2))
          KI = IN(E2,I)
          IF (ICH(KI) .EQ. 0) THEN
            II = KI - LCH(KI)
            DO 120 J = 1, NODETBL(ELEMTYP(E2))
              KJ = IN(E2,J)
              IF (ICH(KJ) .EQ. 0) THEN
                B(II) = B(II) - KE2(I,J) * DHDK(KJ-LCH(KJ),1)
              ENDIF
120         CONTINUE
          ENDIF
130     CONTINUE
      ENDIF
      CALL DPBTRS ('UPPER',NDOF,SBW,1,M1,MAX6+1,B,MAX0,ERROR)
      IF (ERROR .NE. 0) THEN
        PRINT *, 'ERROR IN SOLVING 1ST DERIVE',ERROR,'ELEMENT=',E
      ELSE
        DO 140 I = 1, NDOF
          D2HDK2(I) = B(I)
140     CONTINUE
      ENDIF
      RETURN
      END

      SUBROUTINE VDER1(I,DNDX,S1)
C*****************************************************************
C
C       PURPOSE:
C         TO DETERMINE THE FIRST DERIVATIVES OF THE APPARENT
C         GROUNDWATER VELOCITY OF ELEMENT E WITH RESPECT TO
C         THE LOG OF THE HYDRRAULIC CONDUCTIVITY OF ELEMENT I
C
C       DEFINITIONS OF VARIABLES:
C         DNDX(J,E,K) = PARTIAL DERIVATIVE OF INTERPOLATION
C                       FUNCTION WITH RESPECT TO DIRECTION K
C                       AT NODE J OF ELEMENT E
C           DHDK(L,1) = DERIVATIVE OF HEAD AT NODE L WITH RESPECT TO
C                       HYDRAULIC CONDUCTIVITY OF ELEMENT I
C         DVDK(E,K,1) = DERIVATIVE OF VELOCITY IN K DIRECTION OF
C                       ELEMENT E WITH RESPECT TO THE HYDRAULIC
C                       CONDUCTIVITY OF ELEMENT I
C                   E = ELEMENT NUMBER FOR THE ELEMENT THAT THE VELOCITY
C                       AND DERIVATIVES ARE BEING DETERMINED FOR
C                   I = ELEMENT THAT THE DERIVATIVE IS BEING TAKEN
C                       WITH RESPECT TO
C                 KXE = HYDRAULIC CONDUCTIVITY IN X COORDINATE DIRECTION
```

```
C                  LE = ELEMENT LENGTH
C           X(IN(E,I)) = COMPUTED HEAD FOR NODE I, ELEMENT E
C          X1(IN(E,I)) = X COORDINATE FOR NODE I, ELEMENT E
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I, J, K, L, NODETBL(13),S1
      DOUBLE PRECISION KXE, DNDX(MAX3,MAX2,3)
      DATA NODETBL/2,3,4,3,4,4,8,12,8,20,32,3,4/
      DO 50 E = 1, NUMELM
        KXE = PROP(MATSET(E),1)
        DO 10 K = 1, DIM
            DVDK(E,K,S1) = 0.0
 10     CONTINUE
C       DETERMINE PARTIAL DERIVATIVES OF VE WRT KI - DERIVATIVES OF H
        DO 40 J = 1, NODETBL(ELEMTYP(E))
          IF (ICH(IN(E,J)) .EQ. 0) THEN
            L = IN(E,J) - LCH(IN(E,J))
            DO 20 K = 1, DIM
              DVDK(E,K,S1) = DVDK(E,K,S1) - KXE * DNDX(J,E,K)*DHDK(L,S1)
 20         CONTINUE
          ELSE
C           THIS ELSE SECTION ENABLES LOOP 50 TO PARALLELISE
            DO 30 K = 1, DIM
              DVDK(E,K,S1) = DVDK(E,K,S1)
 30         CONTINUE
          ENDIF
 40     CONTINUE
 50   CONTINUE
C     DETERMINE PARTIAL DERIVATIVES - DERIVATIVE OF K (= 1 FOR I=E)
      KXE = PROP(MATSET(I),1)
      DO 70 J = 1, NODETBL(ELEMTYP(I))
        DO 60 K = 1, DIM
          DVDK(I,K,S1) = DVDK(I,K,S1) - KXE * DNDX(J,I,K) * X(IN(I,J))
 60     CONTINUE
 70   CONTINUE
      RETURN
      END

      SUBROUTINE VDER2(I,I2,DNDX)
C*****************************************************************
C
C       PURPOSE:
C         TO DETERMINE THE SECOND DERIVATIVES OF THE APPARENT
C         GROUNDWATER VELOCITY OF ELEMENT E WITH RESPECT TO
C         THE HYDRRAULIC CONDUCTIVITIES OF ELEMENTS I AND I2
C
C       DEFINITIONS OF VARIABLES:
C         DNDX(J,E,K) = PARTIAL DERIVATIVE OF INTERPOLATION
C                       FUNCTION WITH RESPECT TO DIRECTION K
C                       AT NODE J OF ELEMENT E
C           DHDK(L,1) = DERIVATIVE OF HEAD AT NODE L WITH RESPECT TO
C                        HYDRAULIC CONDUCTIVITY OF ELEMENT I
C           DHDK(L,2) = DERIVATIVE OF HEAD AT NODE L WITH RESPECT TO
C                       HYDRAULIC CONDUCTIVITY OF ELEMENT I2
C           D2HDK2(L) = DERIVATIVE OF HEAD AT NODE L WITH RESPECT TO
C                       HYDRAULIC CONDUCTIVITIES OF ELEMENTS I AND I2
C         D2VDK2(E,K) = DERIVATIVE OF VELOCITY IN K DIRECTION OF
C                       ELEMENT E WITH RESPECT TO THE HYDRAULIC
C                       CONDUCTIVITIES OF ELEMENTS I AND I2
C                   E = ELEMENT NUMBER FOR THE ELEMENT THAT THE VELOCITY
C                       AND DERIVATIVES ARE BEING DETERMINED FOR
C               I, I2 = ELEMENTS THAT THE DERIVATIVES ARE BEING TAKEN
C                       WITH RESPECT TO
C                 KXE = HYDRAULIC CONDUCTIVITY IN X COORDINATE DIRECTION
C                  LE = ELEMENT LENGTH
C           X(IN(E,I)) = COMPUTED HEAD FOR NODE I, ELEMENT E
C          X1(IN(E,I)) = X COORDINATE FOR NODE I, ELEMENT E
```

```
C
C*********************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I, I2, J, K, L, NODETBL(13)
      DOUBLE PRECISION KXE, DNDX(MAX3,MAX2,3)
      DATA NODETBL/2,3,4,3,4,4,8,12,8,20,32,3,4/
      DO 50 E = 1, NUMELM
        KXE = PROP(MATSET(E),1)
        DO 10 K = 1, DIM
          D2VDK2(E,K) = 0.0
 10     CONTINUE
C       DETERMINE PARTIAL DERIVATIVES OF VE WRT KI - DERIVATIVES OF H
        DO 40 J = 1, NODETBL(ELEMTYP(E))
          IF (ICH(IN(E,J)) .EQ. 0) THEN
            L = IN(E,J) - LCH(IN(E,J))
            DO 20 K = 1, DIM
              D2VDK2(E,K) = D2VDK2(E,K) - KXE * DNDX(J,E,K) * D2HDK2(L)
 20         CONTINUE
          ELSE
C           THIS ELSE SECTION ENABLES LOOP 50 TO PARALLELISE
            DO 30 K = 1, DIM
              D2VDK2(E,K) = D2VDK2(E,K)
 30         CONTINUE
          ENDIF
 40     CONTINUE
 50   CONTINUE
C     DETERMINE PARTIAL DERIVATIVES - DERIVATIVE OF K (= 1 FOR I=E)
      KXE = PROP(MATSET(I),1)
      DO 70 J = 1, NODETBL(ELEMTYP(I))
        IF (ICH(IN(I,J)) .EQ. 0) THEN
          L = IN(I,J) - LCH(IN(I,J))
          DO 60 K = 1, DIM
            D2VDK2(I,K) = D2VDK2(I,K) - KXE * DNDX(J,I,K) * DHDK(L,2)
 60       CONTINUE
        ENDIF
 70   CONTINUE
      KXE = PROP(MATSET(I2),1)
      DO 90 J = 1, NODETBL(ELEMTYP(I2))
        IF (ICH(IN(I2,J)) .EQ. 0) THEN
          L = IN(I2,J) - LCH(IN(I2,J))
          DO 80 K = 1, DIM
            D2VDK2(I2,K) = D2VDK2(I2,K) - KXE * DNDX(J,I2,K) * DHDK(L,1)
 80       CONTINUE
        ENDIF
 90   CONTINUE
      IF (I .EQ. I2) THEN
        KXE = PROP(MATSET(I),1)
        DO 110 J = 1, NODETBL(ELEMTYP(I))
          DO 100 K = 1, DIM
            D2VDK2(I,K) = D2VDK2(I,K) - KXE * DNDX(J,I,K) * X(IN(I,J))
100       CONTINUE
110     CONTINUE
      ENDIF
      RETURN
      END

      SUBROUTINE HSAVE(E,E2)
C*********************************************************************
C
C       PURPOSE:
C         SUBROUTINE HVSAVE SAVES THE CURRENT VALUES OF THE
C         VARIABLES INTO A FILE CALLED 'AUTOSAVE' SO THAT THE
C         PROGRAM CAN BE SHUT DOWN AND RESTARTED WHERE IT LEFT
C         OFF.
C
C*********************************************************************
      INCLUDE 'COMALL'
      INTEGER E, E2, J, L
```

```
      OPEN (UNIT = PROBF, FILE = PROBFILE,
     1      FORM = 'FORMATTED', STATUS = 'NEW',
     2      ACCESS = 'SEQUENTIAL')
      OPEN (UNIT = HCF, FORM = 'FORMATTED',
     1      FILE = HCFILE,
     2      STATUS = 'UNKNOWN', ACCESS = 'SEQUENTIAL')
      WRITE (HCF,*) R1, R2, NOR1, NOR2
      WRITE (HCF,*) E, E2
      DO 20 L = 1, NUMMAT
        DO 10 J = 1, NDOF
          WRITE (HCF,*) MEAN(J,L), VAR(J,L)
 10     CONTINUE
 20   CONTINUE
      WRITE (HCF,*) 'FILE ENDS'
      CLOSE (UNIT = HCF, STATUS = 'KEEP')
      CLOSE (UNIT = PROBF, STATUS = 'KEEP')
C     FILE 'PROBLEM' (UNIT PROBF) IS DELETED IN VSAVE.
      PRINT *,'DATA SAVED AT E, E2 = ',E, E2
      PRINT *,NOR2,' SECOND DERIVATIVES COMPLETED'
      RETURN
      END

      SUBROUTINE HLOAD(E,E2)
C*********************************************************************
C
C       PURPOSE:
C         SUBROUTINE HVSAVE SAVES THE CURRENT VALUES OF THE
C         VARIABLES INTO A FILE CALLED 'AUTOSAVE' SO THAT THE
C         PROGRAM CAN BE SHUT DOWN AND RESTARTED WHERE IT LEFT
C         OFF.
C
C*********************************************************************
      INCLUDE 'COMALL'
      INTEGER E, E2, J, L
      OPEN (UNIT = HCF, FORM = 'FORMATTED',
     1      FILE = HCFILE,
     2      STATUS = 'UNKNOWN', ACCESS = 'SEQUENTIAL')
      READ (HCF,*) R1, R2, NOR1, NOR2
      READ (HCF,*) E, E2
      DO 20 L = 1, NUMMAT
        DO 10 J = 1, NDOF
          READ (HCF,*) MEAN(J,L), VAR(J,L)
 10     CONTINUE
 20   CONTINUE
      CLOSE (UNIT = HCF, STATUS = 'KEEP')
      PRINT *,'DATA RESTORED AT E, E2 = ',E, E2
      RETURN
      END

      SUBROUTINE VSAVE
C*********************************************************************
C
C       PURPOSE:
C         SUBROUTINE VSAVE SAVES THE CURRENT VALUES OF VMEAN
C         AND COVAR IN A FILE ('AUTOSAVE' BY DEFAULT) THIS
C         IS USED AS A CHECKPOINT FOR STOPPING AND RESTARTING
C         THE PROGRAM AND FOR FREEING UP MEMORY SPACE FOR
C         THE CONCENTRATION PART OF THE PROGRAM.  TO DO THIS
C         AS FAST AS POSSIBLE THE VECTOR VMC IS EQUIVALENCED
C         TO THE REQUIRED DATA AND IS WRITTEN TO A FILE IN
C         A SINGLE WRITE.  THE FILE IS A DIRECT ACCESS,
C         UNFORMATTED FILE TO BOTH SPEED UP DATA TRANSFER
C         AND TO ENABLE PIECEMEAL READING FOR THE CONCENTRATION
C         PART OF THE PROGRAM.
C
C       DEFINITIONS OF VARIABLES:
C         VMC     = VECTOR CONTAINING ALL OF THE ELEMENTS OF
C                   VMEAN AND COVAR AND SO CAN BE QUICKLY
```

```
C                   CHECKPOINTED
C         PROBLEM = FILE WHOSE EXISTENCE INDICATES THAT THE
C                   LAST CHECKPOINT SAVE WAS NOT COMPLETED
C
C*********************************************************************
      INCLUDE 'COMALL'
      DOUBLE PRECISION VM0((VSTEP3 - 1) / 10 )
      DOUBLE PRECISION VM1((VSTEP3 - 1) / 10 )
      DOUBLE PRECISION VM2((VSTEP3 - 1) / 10 )
      DOUBLE PRECISION VM3((VSTEP3 - 1) / 10 )
      DOUBLE PRECISION VM4((VSTEP3 - 1) / 10 )
      DOUBLE PRECISION VM5((VSTEP3 - 1) / 10 )
      DOUBLE PRECISION VM6((VSTEP3 - 1) / 10 )
      DOUBLE PRECISION VM7((VSTEP3 - 1) / 10 )
      DOUBLE PRECISION VM8((VSTEP3 - 1) / 10 )
      DOUBLE PRECISION VM9((VSTEP3 - 1) / 10 )
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 0),VM0)
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 1),VM1)
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 2),VM2)
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 3),VM3)
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 4),VM4)
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 5),VM5)
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 6),VM6)
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 7),VM7)
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 8),VM8)
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 9),VM9)

      OPEN (UNIT = PROBF, FILE = PROBFILE,
     1      FORM = 'FORMATTED', STATUS = 'UNKNOWN',
     2      ACCESS = 'SEQUENTIAL')
      WRITE (PROBF,*) 'ABOUT TO SAVE COVARIANCE DATA'
      OPEN (UNIT = VELF, FORM = 'UNFORMATTED',
     1      FILE = VELFILE, RECL = 8 * (VSTEP3 - 1) / 10,
     2      STATUS = 'UNKNOWN', ACCESS = 'DIRECT')
      WRITE (VELF,REC=1) VM0
      WRITE (VELF,REC=2) VM1
      WRITE (VELF,REC=3) VM2
      WRITE (VELF,REC=4) VM3
      WRITE (VELF,REC=5) VM4
      WRITE (VELF,REC=6) VM5
      WRITE (VELF,REC=7) VM6
      WRITE (VELF,REC=8) VM7
      WRITE (VELF,REC=9) VM8
      WRITE (VELF,REC=10) VM9
      CLOSE (UNIT = VELF, STATUS = 'KEEP')
      CLOSE (UNIT = PROBF, STATUS = 'DELETE')
      RETURN
      END

      SUBROUTINE VLOAD
C*********************************************************************
C
C       PURPOSE:
C         SUBROUTINE VLOAD RELOADS THE DATA SAVED IN A CHECKPOINT
C         BY VSAVE.  SEE VSAVE FOR MORE INFORMATION.
C
C*********************************************************************
      INCLUDE 'COMALL'
      DOUBLE PRECISION VM0((VSTEP3 - 1) / 10 )
      DOUBLE PRECISION VM1((VSTEP3 - 1) / 10 )
      DOUBLE PRECISION VM2((VSTEP3 - 1) / 10 )
      DOUBLE PRECISION VM3((VSTEP3 - 1) / 10 )
      DOUBLE PRECISION VM4((VSTEP3 - 1) / 10 )
      DOUBLE PRECISION VM5((VSTEP3 - 1) / 10 )
      DOUBLE PRECISION VM6((VSTEP3 - 1) / 10 )
      DOUBLE PRECISION VM7((VSTEP3 - 1) / 10 )
      DOUBLE PRECISION VM8((VSTEP3 - 1) / 10 )
      DOUBLE PRECISION VM9((VSTEP3 - 1) / 10 )
```

```
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 0),VM0)
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 1),VM1)
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 2),VM2)
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 3),VM3)
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 4),VM4)
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 5),VM5)
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 6),VM6)
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 7),VM7)
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 8),VM8)
      EQUIVALENCE (BLOCK(1 + ((VSTEP3 - 1) / 10) * 9),VM9)

      OPEN (UNIT = VELF, FORM = 'UNFORMATTED',
     1      FILE = VELFILE, RECL = 8 * (VSTEP3 - 1) / 10,
     2      STATUS = 'UNKNOWN', ACCESS = 'DIRECT')
      READ (VELF,REC=1) VM0
      READ (VELF,REC=2) VM1
      READ (VELF,REC=3) VM2
      READ (VELF,REC=4) VM3
      READ (VELF,REC=5) VM4
      READ (VELF,REC=6) VM5
      READ (VELF,REC=7) VM6
      READ (VELF,REC=8) VM7
      READ (VELF,REC=9) VM8
      READ (VELF,REC=10) VM9
      CLOSE (UNIT = VELF, STATUS = 'KEEP')
      RETURN
      END

      SUBROUTINE VLOAD3
C*********************************************************************
C
C       PURPOSE:
C         SUBROUTINE VLOAD RELOADS THE DATA SAVED IN A CHECKPOINT
C         BY VSAVE.  SEE VSAVE FOR MORE INFORMATION.
C
C*********************************************************************
      INCLUDE 'COMALL'
      DOUBLE PRECISION VMC(VSTEP3 - 1)
      EQUIVALENCE (VMEAN,VMC)

      OPEN (UNIT = VELF, FORM = 'UNFORMATTED',
     1      FILE = VELFILE, RECL = 8 * (VSTEP3 - VSTEP1),
     2      STATUS = 'UNKNOWN', ACCESS = 'DIRECT')
      READ (VELF,REC=1) VMC
      CLOSE (UNIT = VELF, STATUS = 'KEEP')
      RETURN
      END

SUBROUTINE VELOCITY(DNDX)
C*********************************************************************
C
C 14.1  PURPOSE:
C         TO COMPUTE THE COMPONENTS OF APPARENT GROUNDWATER
C         VELOCITY FOR EACH ELEMENT IN THE MESH AND THE PARTIAL
C         DERIVATIVES OF THE VELOCITY OF EACH ELEMENT WITH RESPECT
C         TO THE HYDRAULIC CONDUCTIVITY OF EACH ELEMENT
C
C 14.2  INPUT:
C         NONE
C
C 14.3  OUTPUT:
C         THE COMPONENTS OF APPARENT GROUNDWATER VELOCITY AND
C         THE DERIVATIVES OF THE VELOCITY OF EACH ELEMENT WITH
C         RESPECT TO THE HYDRAULIC CONDUCTIVITY OF EACH ELEMENT ARE
C         WRITTEN TO THE USER DEFINED FILE ASSIGNED TO UNIT "OUTF".
C
C 14.4  DEFINITIONS OF VARIABLES:
C               DIM = COORDINATE SYSTEM TYPE
```

```
C          DHDK(I,E) = DERIVATIVE OF HEAD AT NODE I WITH RESPECT TO
C                      HYDRAULIC CONDUCTIVITY OF ELEMENT E
C        DVDK(I,E,J) = DERIVATIVE OF VELOCITY IN J DIRECTION OF
C                      ELEMENT I WITH RESPECT TO THE
C                      HYDRAULIC CONDUCTIVITY OF ELEMENT E
C                E = ELEMENT NUMBER
C         ELEMTYP(I) = ELEMENT TYPE FOR ELEMENT I
C             NUMELM = NUMBER OF ELEMENTS IN THE MESH
C              V1(I) = APPARENT VELOCITY IN X DIRECTION
C              V2(I) = Y DIRECTION
C              V3(I) = Z DIRECTION
C
C 14.5  USAGE:
C         THE COMPONENTS OF APPARENT GROUNDWATER VELOCITY ARE
C         COMPUTED IN A SET OF SUBROUTINES, ONE SUBROUTINE
C         FOR EACH ELEMENT TYPE.
C
C       SUBROUTINES CALLED:
C         ALL THE V-----'S
C
C***********************************************************************
      INCLUDE 'COMALL'
      INTEGER E
      DOUBLE PRECISION DNDX(MAX3,MAX2,3)
C     COMPUTE THE COMPONENTS OF APPARENT GROUNDWATER VELOCITY
C     FOR EACH ELEMENT
      DO 10 E = 1, NUMELM
        IF (ELEMTYP(E) .EQ. 1) THEN
C         ELEMENT IS A LINEAR BAR
          CALL VBAR2(E,DNDX)
        ELSEIF (ELEMTYP(E) .EQ. 2) THEN
C         ELEMENT IS A QUADRATIC BAR
          CALL VBAR3(E,DNDX)
        ELSEIF (ELEMTYP(E) .EQ. 3) THEN
C         ELEMENT IS A CUBIC BAR
          CALL VBAR4(E,DNDX)
        ELSEIF (ELEMTYP(E) .EQ. 4) THEN
C         ELEMENT IS A LINEAR TRIANGLE
          CALL VTRI3(E,DNDX)
        ELSEIF (ELEMTYP(E) .EQ. 5) THEN
C         ELEMENT IS A LINEAR RECTANGLE
          CALL VREC4(E,DNDX)
        ELSEIF (ELEMTYP(E) .EQ. 6) THEN
C         ELEMENT IS A LINEAR QUADRILATERAL
          CALL VQUA4(E,DNDX)
        ELSEIF (ELEMTYP(E) .EQ. 7) THEN
C         ELEMENT IS A QUADRATIC QUADRILATERAL
          CALL VQUA8(E,DNDX)
        ELSEIF (ELEMTYP(E) .EQ. 8) THEN
C         ELEMENT IS A CUBIC QUADRILATERAL
          CALL VQUA12(E,DNDX)
        ELSEIF (ELEMTYP(E) .EQ. 9) THEN
C         ELEMENT IS A LINEAR PARALLELEPIPED
          CALL VPAR8(E,DNDX)
        ELSEIF (ELEMTYP(E) .EQ. 10) THEN
C         ELEMENT IS A QUADRATIC PARALLELEPIPED
          CALL VPAR20(E,DNDX)
        ELSEIF (ELEMTYP(E) .EQ. 11) THEN
C         ELEMENT IS A CUBIC PARALLELEPIPED
          CALL VPAR32(E,DNDX)
        ENDIF
 10   CONTINUE
      RETURN
      END

      SUBROUTINE VBAR2(E,DNDX)
C***********************************************************************
C
```

```
C       PURPOSE:
C         TO COMPUTE APPARENT GROUNDWATER VELOCITY FOR A
C         ONE-DIMENSIONAL, LINEAR BAR ELEMENT
C
C       DEFINITIONS OF VARIABLES:
C              DHDX = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO X
C        DNDX(J,E,1) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                      OF NODE J OF ELEMENT E WITH RESPECT TO X
C                E = ELEMENT NUMBER
C               KXE = HYDRAULIC CONDUCTIVITY IN X COORDINATE DIRECTION
C                LE = ELEMENT LENGTH
C             V1(E) = APPARENT GROUND WATER VELOCITY IN
C                      X COORDINATE DIRECTION
C         X(IN(E,I)) = COMPUTED HEAD FOR NODE I, ELEMENT E
C        X1(IN(E,I)) = X COORDINATE FOR NODE I, ELEMENT E
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E
      DOUBLE PRECISION KXE, LE, DHDX, DNDX(MAX3,MAX2,3)
      KXE = PROP(MATSET(E),1)
      LE = X1(IN(E,2)) - X1(IN(E,1))
      DNDX(1,E,1) = -1.0 / LE
      DNDX(2,E,1) =  1.0 / LE
      DHDX = (X(IN(E,2)) - X(IN(E,1))) / LE
      V1(E) = -KXE * DHDX
      RETURN
      END

      SUBROUTINE VBAR3(E,DNDX)
C*****************************************************************
C
C       PURPOSE:
C         TO COMPUTE APPARENT GROUNDWATER VELOCITY FOR A
C         ONE-DIMENSIONAL, QUADRATIC BAR ELEMENT
C
C       DEFINITIONS OF VARIABLES:
C              DHDX = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO X
C        DNDX(I,E,1) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                      OF NODE I OF ELEMENT E WITH RESPECT TO X
C           DNDXI(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                      FUNCTION WITH RESPECT TO XI FOR NODE I
C                E = ELEMENT NUMBER FOR THE ELEMENT THAT THE VELOCITY
C                      AND DERIVATIVES ARE BEING DETERMINED FOR
C             I, I2 = ELEMENTS THAT THE DERIVATIVES ARE BEING TAKEN
C                      WITH RESPECT TO
C               JAC = JACOBIAN MATRIX
C            JACINV = INVERSE OF JACOBIAN MATRIX
C               KXE = HYDRAULIC CONDUCTIVITY IN X COORDINATE DIRECTION
C             V1(E) = APPARENT GROUND WATER VELOCITY IN
C                      X COORDINATE DIRECTION
C         X(IN(E,I)) = COMPUTED HEAD FOR NODE I, ELEMENT E
C        X1(IN(E,I)) = X COORDINATE FOR NODE I, ELEMENT E
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I
      DOUBLE PRECISION KXE, DHDX, DNDX(MAX3,MAX2,3)
      DOUBLE PRECISION DNDXI(3), JAC, JACINV

      KXE = PROP(MATSET(E),1)
      DNDXI(1) = -0.5
      DNDXI(2) =  0.0
      DNDXI(3) =  0.5
      JAC = 0.0
      DO 10 I = 1, 3
        JAC = JAC + DNDXI(I) * X1(IN(E,I))
 10   CONTINUE
```

```
      JACINV = 1.0 / JAC
      DNDX(1,E,1) = JACINV * DNDXI(1)
      DHDX = 0.0
      DO 20 I = 1, 3
        DNDX(I,E,1) = JACINV * DNDXI(I)
        DHDX = DHDX + DNDX(I,E,1) * X(IN(E,I))
 20   CONTINUE
      V1(E) = -KXE * DHDX
      RETURN
      END

      SUBROUTINE VBAR4(E,DNDX)
C*****************************************************************
C
C       PURPOSE:
C         TO COMPUTE APPARENT GROUNDWATER VELOCITY FOR A
C         ONE-DIMENSIONAL, CUBIC BAR ELEMENT
C
C       DEFINITIONS OF VARIABLES:
C              DHDX = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO X
C         DHDK(I,E) = DERIVATIVE OF HEAD AT NODE I WITH RESPECT TO
C                     HYDRAULIC CONDUCTIVITY OF ELEMENT E
C        DNDX(I,E,1) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                     OF NODE I OF ELEMENT E WITH RESPECT TO X
C           DNDXI(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                     FUNCTION WITH RESPECT TO XI FOR NODE I
C        DVDK(E,I,J) = DERIVATIVE OF VELOCITY IN J DIRECTION OF
C                     ELEMENT E WITH RESPECT TO THE
C                     HYDRAULIC CONDUCTIVITY OF ELEMENT I
C                 E = ELEMENT NUMBER FOR THE ELEMENT THAT THE VELOCITY
C                     AND DERIVATIVES ARE BEING DETERMINED FOR
C              I, I2 = ELEMENTS THAT THE DERIVATIVES ARE BEING TAKEN
C                     WITH RESPECT TO
C                JAC = JACOBIAN MATRIX
C             JACINV = INVERSE OF JACOBIAN MATRIX
C                KXE = HYDRAULIC CONDUCTIVITY IN X COORDINATE DIRECTION
C              V1(E) = APPARENT GROUND WATER VELOCITY IN
C                     X COORDINATE DIRECTION
C         X(IN(E,I)) = COMPUTED HEAD FOR NODE I, ELEMENT E
C        X1(IN(E,I)) = X COORDINATE FOR NODE I, ELEMENT E
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I
      DOUBLE PRECISION KXE, DHDX, DNDX(MAX3,MAX2,3)
      DOUBLE PRECISION DNDXI(4), JAC, JACINV
      KXE = PROP(MATSET(E),1)
      DNDXI(1) =   1.0 / 16.0
      DNDXI(2) = -27.0 / 16.0
      DNDXI(3) = -DNDXI(2)
      DNDXI(4) = -DNDXI(1)
      JAC = 0.0
      DO 10 I = 1, 4
        JAC = JAC + DNDXI(I) * X1(IN(E,I))
 10   CONTINUE
      JACINV = 1.0 / JAC
      DHDX = 0.0
      DO 20 I = 1, 4
        DNDX(I,E,1) = JACINV * DNDXI(I)
        DHDX = DHDX + DNDX(I,E,1) * X(IN(E,I))
 20   CONTINUE
      V1(E) = -KXE * DHDX
      RETURN
      END

      SUBROUTINE VTRI3(E,DNDX)
C*****************************************************************
C
```

```
C       PURPOSE:
C         TO COMPUTE COMPONENTS OF APPARENT GROUNDWATER
C         VELOCITY FOR A TWO-DIMENSIONAL, LINEAR TRIANGLE ELEMENT
C
C       DEFINITIONS OF VARIABLES:
C               DHDX = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO X
C               DHDY = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO Y
C        DNDX(J,E,1) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                     OF NODE J OF ELEMENT E WITH RESPECT TO X
C        DNDX(J,E,2) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                     OF NODE J OF ELEMENT E WITH RESPECT TO Y
C                 E = ELEMENT NUMBER
C               KXE = HYDRAULIC CONDUCTIVITY IN X COORDINATE DIRECTION
C               KYE = HYDRAULIC CONDUCTIVITY IN Y COORDINATE DIRECTION
C             V1(E) = APPARENT GROUND WATER VELOCITY IN
C                     X COORDINATE DIRECTION
C             V2(E) = APPARENT GROUND WATER VELOCITY IN
C                     Y COORDINATE DIRECTION
C         X(IN(E,I)) = COMPUTED HEAD FOR NODE I, ELEMENT E
C        X1(IN(E,I)) = X COORDINATE FOR NODE I, ELEMENT E
C        X2(IN(E,I)) = Y COORDINATE FOR NODE I, ELEMENT E
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I
      DOUBLE PRECISION DNDX(MAX3,MAX2,3) ,KXE, KYE, AE2
      DOUBLE PRECISION DHDX, DHDY
      KXE = PROP(MATSET(E),1)
      KYE = PROP(MATSET(E),2)
      AE2 = X1(IN(E,2)) * X2(IN(E,3)) + X1(IN(E,1)) * X2(IN(E,2)) +
     1      X2(IN(E,1)) * X1(IN(E,3)) - X2(IN(E,3)) * X1(IN(E,1)) -
     2      X1(IN(E,3)) * X2(IN(E,2)) - X1(IN(E,2)) * X2(IN(E,1))
      DNDX(1,E,1) = (X2(IN(E,2)) - X2(IN(E,3))) / AE2
      DNDX(2,E,1) = (X2(IN(E,3)) - X2(IN(E,1))) / AE2
      DNDX(3,E,1) = (X2(IN(E,1)) - X2(IN(E,2))) / AE2
      DNDX(1,E,2) = (X1(IN(E,3)) - X1(IN(E,2))) / AE2
      DNDX(2,E,2) = (X1(IN(E,1)) - X1(IN(E,3))) / AE2
      DNDX(3,E,2) = (X1(IN(E,2)) - X1(IN(E,1))) / AE2
      DHDX = 0.0
      DHDY = 0.0
      DO 20 I = 1, 3
        DHDX = DHDX + DNDX(I,E,1) * X(IN(E,I))
        DHDY = DHDY + DNDX(I,E,2) * X(IN(E,I))
 20   CONTINUE
      V1(E) = -KXE * DHDX
      V2(E) = -KYE * DHDY
      RETURN
      END

      SUBROUTINE VREC4(E,DNDX)
C*****************************************************************
C
C       PURPOSE:
C         TO COMPUTE COMPONENTS OF APPARENT GROUNDWATER
C         VELOCITY FOR A TWO-DIMENSIONAL, LINEAR RECTANGLE ELEMENT
C
C       DEFINITIONS OF VARIABLES:
C               DHDX = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO X
C               DHDY = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO Y
C        DNDX(J,E,1) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                     OF NODE J OF ELEMENT E WITH RESPECT TO X
C        DNDX(J,E,2) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                     OF NODE J OF ELEMENT E WITH RESPECT TO Y
C                 E = ELEMENT NUMBER
C               KXE = HYDRAULIC CONDUCTIVITY IN X COORDINATE DIRECTION
C               KYE = HYDRAULIC CONDUCTIVITY IN Y COORDINATE DIRECTION
C             V1(E) = APPARENT GROUND WATER VELOCITY IN
C                     X COORDINATE DIRECTION
```

```
C              V2(E) = APPARENT GROUND WATER VELOCITY IN
C                      Y COORDINATE DIRECTION
C          X(IN(E,I)) = COMPUTED HEAD FOR NODE I, ELEMENT E
C         X1(IN(E,I)) = X COORDINATE FOR NODE I, ELEMENT E
C         X2(IN(E,I)) = Y COORDINATE FOR NODE I, ELEMENT E
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I
      DOUBLE PRECISION DNDX(MAX3,MAX2,3) ,KXE, KYE, AE, BE
      DOUBLE PRECISION DHDX, DHDY
      KXE = PROP(MATSET(E),1)
      KYE = PROP(MATSET(E),2)
      AE = 0.25 / (X2(IN(E,3))-X2(IN(E,1)))
      BE = 0.25 / (X1(IN(E,3))-X1(IN(E,1)))
      DNDX(1,E,1) = -BE
      DNDX(2,E,1) =  BE
      DNDX(3,E,1) =  BE
      DNDX(4,E,1) = -BE
      DNDX(1,E,2) = -AE
      DNDX(2,E,2) = -AE
      DNDX(3,E,2) =  AE
      DNDX(4,E,2) =  AE
      DHDX = 0.0
      DHDY = 0.0
      DO 10 I = 1, 4
        DHDX = DHDX + DNDX(I,E,1) * X(IN(E,I))
        DHDY = DHDY + DNDX(I,E,2) * X(IN(E,I))
 10   CONTINUE
      V1(E) = -KXE * DHDX
      V2(E) = -KYE * DHDY
      RETURN
      END

      SUBROUTINE VQUA4(E,DNDX)
C*****************************************************************
C
C       PURPOSE:
C         TO COMPUTE COMPONENTS OF APPARENT GROUNDWATER
C         VELOCITY FOR A TWO-DIMENSIONAL, LINEAR QUADRILATERAL ELEMENT
C
C       DEFINITIONS OF VARIABLES:
C             DETJAC = DETERMINANT OF JACOBIAN MATRIX
C               DHDX = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO X
C               DHDY = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO Y
C        DNDX(I,E,1) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                      OF NODE I OF ELEMENT E WITH RESPECT TO X
C        DNDX(I,E,2) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                      OF NODE I OF ELEMENT E WITH RESPECT TO Y
C           DNDXI(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                      FUNCTION WITH RESPECT TO XI FOR NODE I
C          DNDETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                      FUNCTION WITH RESPECT TO ETA FOR NODE I
C                  E = ELEMENT NUMBER
C           JAC(I,J) = JACOBIAN MATRIX
C        JACINV(I,J) = INVERSE OF JACOBIAN MATRIX
C                KXE = HYDRAULIC CONDUCTIVITY IN X COORDINATE DIRECTION
C                KYE = HYDRAULIC CONDUCTIVITY IN Y COORDINATE DIRECTION
C              V1(E) = APPARENT GROUND WATER VELOCITY IN
C                      X COORDINATE DIRECTION
C              V2(E) = APPARENT GROUND WATER VELOCITY IN
C                      Y COORDINATE DIRECTION
C          X(IN(E,I)) = COMPUTED HEAD FOR NODE I, ELEMENT E
C         X1(IN(E,I)) = X COORDINATE FOR NODE I, ELEMENT E
C         X2(IN(E,I)) = Y COORDINATE FOR NODE I, ELEMENT E
C
C*****************************************************************
      INCLUDE 'COMALL'
```

```
      INTEGER E, I, J
      DOUBLE PRECISION JAC(2,2),JACINV(2,2),DNDXI(4),DNDETA(4)
      DOUBLE PRECISION DNDX(MAX3,MAX2,3) ,KXE, KYE, SIGN1(4),SIGN2(4)
      DOUBLE PRECISION DHDX, DHDY, DETJAC
      DATA SIGN1/-1.0, 1.0, 1.0,-1.0/
      DATA SIGN2/-1.0,-1.0, 1.0, 1.0/
      KXE = PROP(MATSET(E),1)
      KYE = PROP(MATSET(E),2)
      DO 20 I = 1, 2
        DO 10 J = 1, 2
          JAC(I,J) = 0.0
 10     CONTINUE
 20   CONTINUE
      DO 30 I = 1, 4
        DNDXI(I)  = 0.25 * SIGN1(I)
        DNDETA(I) = 0.25 * SIGN2(I)
 30   CONTINUE
      DO 40 I = 1, 4
        JAC(1,1) = JAC(1,1) + DNDXI(I)  * X1(IN(E,I))
        JAC(1,2) = JAC(1,2) + DNDXI(I)  * X2(IN(E,I))
        JAC(2,1) = JAC(2,1) + DNDETA(I) * X1(IN(E,I))
        JAC(2,2) = JAC(2,2) + DNDETA(I) * X2(IN(E,I))
 40   CONTINUE
      DETJAC = JAC(1,1) * JAC(2,2) - JAC(1,2) * JAC(2,1)
      JACINV(1,1) =  JAC(2,2) / DETJAC
      JACINV(1,2) = -JAC(1,2) / DETJAC
      JACINV(2,1) = -JAC(2,1) / DETJAC
      JACINV(2,2) =  JAC(1,1) / DETJAC
      DO 50 I = 1, 4
        DNDX(I,E,1) = JACINV(1,1) * DNDXI(I) + JACINV(1,2) * DNDETA(I)
        DNDX(I,E,2) = JACINV(2,1) * DNDXI(I) + JACINV(2,2) * DNDETA(I)
 50   CONTINUE
      DHDX = 0.0
      DHDY = 0.0
      DO 60 I = 1, 4
        DHDX = DHDX + DNDX(I,E,1) * X(IN(E,I))
        DHDY = DHDY + DNDX(I,E,2) * X(IN(E,I))
 60   CONTINUE
      V1(E) = -KXE * DHDX
      V2(E) = -KYE * DHDY
      RETURN
      END

      SUBROUTINE VQUA8(E,DNDX)
C*****************************************************************
C
C       PURPOSE:
C         TO COMPUTE COMPONENTS OF APPARENT GROUNDWATER
C         VELOCITY FOR A TWO-DIMENSIONAL, QUADRATIC QUADRILATERAL
C         ELEMENT
C
C       DEFINITIONS OF VARIABLES:
C              DETJAC = DETERMINANT OF JACOBIAN MATRIX
C                DHDX = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO X
C                DHDY = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO Y
C        DNDX(I,E,1) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                      OF NODE I OF ELEMENT E WITH RESPECT TO X
C        DNDX(I,E,2) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                      OF NODE I OF ELEMENT E WITH RESPECT TO Y
C            DNDXI(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                       FUNCTION WITH RESPECT TO XI FOR NODE I
C           DNDETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                       FUNCTION WITH RESPECT TO ETA FOR NODE I
C                   E = ELEMENT NUMBER
C             JAC(I,J) = JACO8IAN MATRIX
C          JACINV(I,J) = INVERSE OF JACOBIAN MATRIX
C                 KXE = HYDRAULIC CONDUCTIVITY IN X COORDINATE DIRECTION
C                 KYE = HYDRAULIC CONDUCTIVITY IN Y COORDINATE DIRECTION
```

```
C               V1(E) = APPARENT GROUNDWATER VELOCITY IN X
C                       COORDINATE DIRECTION
C               V2(E) = APPARENT GROUNDWATER VELOCITY IN Y
C                       COORDINATE DIRECTION
C          X(IN(E,I)) = COMPUTED HEAD FOR NODE I, ELEMENT E
C         X1(IN(E,I)) = X COORDINATE FOR NODE I, ELEMENT E
C         X2(IN(E,I)) = Y COORDINATE FOR NODE I, ELEMENT E
C         ISTOK,J D. GROUNDWATER FLOW AND SOLUTE TRANSPORT
C         MODELING BY THE FINITE ELEMENT METHOD, FIGURE 4.11,
C         EQUATION 6.14A, 6.14B, 6.17, 6.22A, AND 6.22B.
C*********************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I, J
      DOUBLE PRECISION JAC(2,2), JACINV(2,2), DNDX(MAX3,MAX2,3)
      DOUBLE PRECISION DNDXI(8), DNDETA(8), SIGN1(8), SIGN2(8)
      DOUBLE PRECISION KXE, KYE, DETJAC, DHDX, DHDY
      DATA SIGN1/-1.0, 0.0, 1.0, 1.0, 1.0, 0.0,-1.0,-1.0/
      DATA SIGN2/-1.0,-1.0,-1.0, 0.0, 1.0, 1.0, 1.0, 0.0/
      KXE = PROP (MATSET(E),1)
      KYE = PROP (MATSET(E),2)
      DO 20 I = 1, 2
        DO 10 J = 1, 2
          JAC(I,J) = 0.0
 10     CONTINUE
 20   CONTINUE
      DO 30 I = 1, 8
        IF ((I .EQ. 1) .OR. (I .EQ. 3) .OR.
     1      (I .EQ. 5) .OR. (I .EQ. 7)) THEN
          DNDXI(I) = 0.0
          DNDETA(I) = 0.0
        ELSEIF ((I .EQ. 2) .OR. (I .EQ. 6)) THEN
          DNDXI(I) = 0.0
          DNDETA(I) = 0.5 * SIGN2(I)
        ELSEIF ((I .EQ. 4) .OR. (I .EQ. 8)) THEN
          DNDXI(I) = 0.5 * SIGN1(I)
          DNDETA(I) = 0.0
        ENDIF
 30   CONTINUE
      DO 40 I = 1, 8
        JAC(1,1) = JAC(1,1) + DNDXI(I) * X1(IN(E,I))
        JAC(1,2) = JAC(1,2) + DNDXI(I) * X2(IN(E,I))
        JAC(2,1) = JAC(2,1) + DNDETA(I) * X1(IN(E,I))
        JAC(2,2) = JAC(2,2) + DNDETA(I) * X2(IN(E,I))
 40   CONTINUE
      DETJAC = JAC(1,1) * JAC(2,2) - JAC(1,2) * JAC(2,1)
      JACINV(1,1) =  JAC(2,2) / DETJAC
      JACINV(1,2) = -JAC(1,2) / DETJAC
      JACINV(2,1) = -JAC(2,1) / DETJAC
      JACINV(2,2) =  JAC(1,1) / DETJAC
      DO 50 I = 1, 8
        DNDX(I,E,1) = JACINV(1,1) * DNDXI(I) + JACINV(1,2) * DNDETA(I)
        DNDX(I,E,2) = JACINV(2,1) * DNDXI(I) + JACINV(2,2) * DNDETA(I)
 50   CONTINUE
      DHDX = 0.0
      DHDY = 0.0
      DO 60 I = 1, 8
        DHDX = DHDX + DNDX(I,E,1) * X(IN(E,I))
        DHDY = DHDY + DNDX(I,E,2) * X(IN(E,I))
 60   CONTINUE
      V1(E) = -KXE * DHDX
      V2(E) = -KYE * DHDY
      RETURN
      END

      SUBROUTINE VQUA12(E,DNDX)
C*********************************************************************
C
C       PURPOSE:
```

```
C         TO COMPUTE COMPONENTS OF APPARENT GROUNDWATER
C         VELOCITY FOR A TWO-DIMENSIONAL, CUBIC QUADRILATERAL
C         ELEMENT
C
C       DEFINITIONS OF VARIABLES:
C             DETJAC = DETERMINANT OF JACOBIAN MATRIX
C               DHDX = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO X
C               DHDY = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO Y
C        DNDX(I,E,1) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                      OF NODE I OF ELEMENT E WITH RESPECT TO X
C        DNDX(I,E,2) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                      OF NODE I OF ELEMENT E WITH RESPECT TO Y
C           DNDXI(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                      FUNCTION WITH RESPECT TO XI FOR NODE I
C          DNDETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                      FUNCTION WITH RESPECT TO ETA FOR NODE I
C                  E = ELEMENT NUMBER
C            JAC(I,J) = JACO8IAN MATRIX
C         JACINV(I,J) = INVERSE OF JACOBIAN MATRIX
C                KXE = HYDRAULIC CONDUCTIVITY IN X COORDINATE DIRECTION
C                KYE = HYDRAULIC CONDUCTIVITY IN Y COORDINATE DIRECTION
C              V1(E) = APPARENT GROUNDWATER VELOCITY IN X
C                      COORDINATE DIRECTION
C              V2(E) = APPARENT GROUNDWATER VELOCITY IN Y
C                      COORDINATE DIRECTION
C          X(IN(E,I)) = COMPUTED HEAD FOR NODE I, ELEMENT E
C         X1(IN(E,I)) = X COORDINATE FOR NODE I, ELEMENT E
C         X2(IN(E,I)) = Y COORDINATE FOR NODE I, ELEMENT E
C
C*********************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I, J
      DOUBLE PRECISION JAC(2,2), JACINV(2,2), DNDX(MAX3,MAX2,3)
      DOUBLE PRECISION DNDXI(12), DNDETA(12), SIGN1(12), SIGN2(12)
      DOUBLE PRECISION KXE, KYE, DETJAC, DHDX, DHDY
      DATA SIGN1/-1.0,-1.0, 1.0, 1.0, 1.0, 1.0,
     1           1.0, 1.0,-1.0,-1.0,-1.0,-1.0/
      DATA SIGN2/-1.0,-1.0,-1.0,-1.0,-1.0, 1.0,
     1           1.0, 1.0, 1.0, 1.0, 1.0,-1.0/
      KXE = PROP (MATSET(E),1)
      KYE = PROP (MATSET(E),2)
      DO 20 I = 1, 2
        DO 10 J = 1, 2
          JAC(I,J) = 0.0
 10     CONTINUE
 20   CONTINUE
      DO 30 I = 1, 12
      IF ((I .EQ. 1) .OR. (I .EQ. 4) .OR.
     1    (I .EQ. 7) .OR. (I .EQ. 10)) THEN
        DNDXI(I)  = -(10.0 / 32.0) * SIGN1(I)
        DNDETA(I) = -(10.0 / 32.0) * SIGN2(I)
      ELSEIF ((I .EQ. 2) .OR. (I .EQ. 3) .OR.
     1        (I .EQ. 8) .OR. (I .EQ. 9)) THEN
        DNDXI(I)  = (27.0 / 32.0) * SIGN1(I)
        DNDETA(I) = (9.0 / 32.0) * SIGN2(I)
      ELSEIF ((I .EQ. 5) .OR. (I .EQ. 6) .OR.
     1        (I .EQ. 11) .OR. (I .EQ. 12)) THEN
        DNDXI(I)  = (9.0 / 32.0) * SIGN1(I)
        DNDETA(I) = (27.0 / 32.0) * SIGN2(I)
      ENDIF
 30   CONTINUE
      DO 40 I = 1, 12
        JAC(1,1) = JAC(1,1) + DNDXI(I) * X1(IN(E,I))
        JAC(1,2) = JAC(1,2) + DNDXI(I) * X2(IN(E,I))
        JAC(2,1) = JAC(2,1) + DNDETA(I) * X1(IN(E,I))
        JAC(2,2) = JAC(2,2) + DNDETA(I) * X2(IN(E,I))
 40   CONTINUE
      DETJAC = JAC(1,1) * JAC(2,2) - JAC(1,2) * JAC(2,1)
```

```
      JACINV(1,1) =  JAC(2,2) / DETJAC
      JACINV(1,2) = -JAC(1,2) / DETJAC
      JACINV(2,1) = -JAC(2,1) / DETJAC
      JACINV(2,2) =  JAC(1,1) / DETJAC
      DO 50 I = 1, 8
        DNDX(I,E,1) = JACINV(1,1) * DNDXI(I) + JACINV(1,2) * DNDETA(I)
        DNDX(I,E,2) = JACINV(2,1) * DNDXI(I) + JACINV(2,2) * DNDETA(I)
 50   CONTINUE
      DHDX = 0.0
      DHDY = 0.0
      DO 60 I = 1, 12
        DHDX = DHDX + DNDX(I,E,1) * X(IN(E,I))
        DHDY = DHDY + DNDX(I,E,2) * X(IN(E,I))
 60   CONTINUE
      V1(E) = -KXE * DHDX
      V2(E) = -KYE * DHDY
      RETURN
      END

      SUBROUTINE VPAR8(E,DNDX)
C*********************************************************************
C
C         PURPOSE:
C           TO COMPUTE COMPONENTS OF APPARENT GROUNDWATER
C           VELOCITY FOR A THREE DIMENSIONAL, LINEAR
C           PARALLELEPIPED ELEMENT
C
C         DEFINITIONS OF VARIABLES:
C                DETJAC = DETERMINANT OF JACOBIAN MATRIX
C                  DHDX = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO X
C                  DHDY = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO Y
C                  DHDZ = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO Z
C            DNDX(I,E,1) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                          OF NODE I OF ELEMENT E WITH RESPECT TO X
C            DNDX(I,E,2) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                          OF NODE I OF ELEMENT E WITH RESPECT TO Y
C            DNDX(I,E,3) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                          OF NODE I OF ELEMENT E WITH RESPECT TO Z
C               DNDXI(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                          FUNCTION WITH RESPECT TO XI AT NODE I
C              DNDETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                          FUNCTION WITH RESPECT TO ETA AT NODE I
C             DNDZETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                          FUNCTION WITH RESPECT TO ZETA AT NODE I
C                      E = ELEMENT NUMBER
C                 IN(I,J) = NODE NUMBER J FOR ELEMENT I
C                JAC(I,J) = JACOBIAN MATRIX
C              JACINV(I,J) = INVERSE OF JACOBIAN MATRIX
C                      KXE = HYDRAULIC CONDUCTIVITY IN X DIRECTION
C                      KYE = HYDRAULIC CONDUCTIVITY IN Y DIRECTION
C                      KZE = HYDRAULIC CONDUCTIVITY IN Z DIRECTION
C                    V1(E) = APPARENT GROUNDWATER VELOCITY IN X DIRECTION
C                    V2(E) = APPARENT GROUNDWATER VELOCITY IN Y DIRECTION
C                      VZE = APPARENT GROUNDWATER VELOCITY IN Z DIRECTION
C               X(IN(E,I)  = COMPUTED HEAD FOR NODE I, ELEMENT E
C              X1(IN(E,I)  = X COORDINATE FOR NODE I, ELEMENT E
C              X2(IN(E,I)  = Y COORDINATE FOR NODE I, ELEMENT E
C              X3(IN(E,I)  = Z COORDINATE FOR NODE I, ELEMENT E
C
C*********************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I, J
      DOUBLE PRECISION JAC(3,3), JACINV(3,3), DNDX(MAX3,MAX2,3)
      DOUBLE PRECISION DNDXI(8), DNDETA(8)
      DOUBLE PRECISION DNDZETA(8), DETJAC
      DOUBLE PRECISION SIGN1(8), SIGN2(8), SIGN3(8)
      DOUBLE PRECISION  KXE, KYE, KZE
      DOUBLE PRECISION DHDX, DHDY, DHDZ
```

```
      DATA SIGN1/-1.0, 1.0, 1.0,-1.0,-1.0, 1.0, 1.0,-1.0/
      DATA SIGN2/-1.0,-1.0, 1.0, 1.0,-1.0,-1.0, 1.0, 1.0/
      DATA SIGN3/-1.0,-1.0,-1.0,-1.0, 1.0, 1.0, 1.0, 1.0/
      KXE = PROP(MATSET(E),1)
      KYE = PROP(MATSET(E),2)
      KZE = PROP(MATSET(E),3)
      DO 20 I = 1, 2
        DO 10 J = 1, 2
          JAC(I,J) = 0.0
 10     CONTINUE
 20   CONTINUE
      DO 30 I = 1, 8
        DNDXI(I)   = 0.125 * SIGN1(I)
        DNDETA(I)  = 0.125 * SIGN2(I)
        DNDZETA(I) = 0.125 * SIGN3(I)
 30   CONTINUE
      DO 40 I = 1, 8
        JAC(1,1) = JAC(1,1) + DNDXI(I) * X1(IN(E,I))
        JAC(1,2) = JAC(1,2) + DNDXI(I) * X2(IN(E,I))
        JAC(1,3) = JAC(1,3) + DNDXI(I) * X3(IN(E,I))
        JAC(2,1) = JAC(2,1) + DNDETA(I) * X1(IN(E,I))
        JAC(2,2) = JAC(2,2) + DNDETA(I) * X2(IN(E,I))
        JAC(2,3) = JAC(2,3) + DNDETA(I) * X3(IN(E,I))
        JAC(3,1) = JAC(3,1) + DNDZETA(I) * X1(IN(E,I))
        JAC(3,2) = JAC(3,2) + DNDZETA(I) * X2(IN(E,I))
        JAC(3,3) = JAC(3,3) + DNDZETA(I) * X3(IN(E,I))
 40   CONTINUE
      DETJAC = JAC(1,1) * (JAC(2,2)*JAC(3,3) - JAC(3,2)*JAC(2,3))
     1       - JAC(1,2) * (JAC(2,1)*JAC(3,3) - JAC(3,1)*JAC(2,3))
     2       - JAC(1,3) * (JAC(2,1)*JAC(3,2) - JAC(3,1)*JAC(2,2))
      IF (DETJAC .EQ. 0.0 ) STOP 'DETERMINANT IS ZERO !!!!!'
C     INVERSEJACOBIAN MATRIX FORMULA HAS BEEN TRANSPOSEDED - STEVE 9/7/96
            JACINV(1,1) = ( JAC(2,2) * JAC(3,3) - JAC(2,3) * JAC(3,2))
     1                    / DETJAC
            JACINV(2,1) = (-JAC(2,1) * JAC(3,3) + JAC(2,3) * JAC(3,1))
     1                    / DETJAC
            JACINV(3,1) = ( JAC(2,1) * JAC(3,2) - JAC(3,1) * JAC(2,2))
     1                    / DETJAC
            JACINV(1,2) = (-JAC(1,2) * JAC(3,3) + JAC(1,3) * JAC(3,2))
     1                    / DETJAC
            JACINV(2,2) = ( JAC(1,1) * JAC(3,3) - JAC(1,3) * JAC(3,1))
     1                    / DETJAC
            JACINV(3,2) = (-JAC(1,1) * JAC(3,2) + JAC(1,2) * JAC(3,1))
     1                    / DETJAC
            JACINV(1,3) = ( JAC(1,2) * JAC(2,3) - JAC(1,3) * JAC(2,2))
     1                    / DETJAC
            JACINV(2,3) = (-JAC(1,1) * JAC(2,3) + JAC(1,3) * JAC(2,1))
     1                    / DETJAC
            JACINV(3,3) = ( JAC(1,1) * JAC(2,2) - JAC(1,2) * JAC(2,1))
     1                    / DETJAC
      DO 50 I = 1, 8
        DNDX(I,E,1) = JACINV(1,1) * DNDXI(I) + JACINV(1,2) *
     1                DNDETA(I) + JACINV(1,3) * DNDZETA(I)
        DNDX(I,E,2) = JACINV(2,1) * DNDXI(I) + JACINV(2,2) *
     1                DNDETA(I) + JACINV(2,3) * DNDZETA(I)
        DNDX(I,E,3) = JACINV(3,1) * DNDXI(I) + JACINV(3,2) *
     1                DNDETA(I) + JACINV(3,3) * DNDZETA(I)
 50   CONTINUE
      DHDX = 0.0
      DHDY = 0.0
      DHDZ = 0.0
      DO 60 I = 1, 8
        DHDX = DHDX + DNDX(I,E,1) * X(IN(E,I))
        DHDY = DHDY + DNDX(I,E,2) * X(IN(E,I))
        DHDZ = DHDZ + DNDX(I,E,3) * X(IN(E,I))
 60   CONTINUE
      V1(E) = -KXE * DHDX
      V2(E) = -KYE * DHDY
```

```
      V3(E) = -KZE * DHDZ
      RETURN
      END

      SUBROUTINE VPAR20(E,DNDX)
C*****************************************************************
C
C         PURPOSE:
C           TO COMPUTE COMPONENTS OF APPARENT GROUNDWATER
C           VELOCITY FOR A THREE DIMENSIONAL, QUADRATIC
C           PARALLELEPIPED ELEMENT
C
C         DEFINITIONS OF VARIABLES:
C              DETJAC = DETERMINANT OF JACOBIAN MATRIX
C                DHDX = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO X
C                DHDY = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO Y
C                DHDZ = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO Z
C           DNDX(I,E,1) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                         OF NODE I OF ELEMENT E WITH RESPECT TO X
C           DNDX(I,E,2) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                         OF NODE I OF ELEMENT E WITH RESPECT TO Y
C           DNDX(I,E,3) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                         OF NODE I OF ELEMENT E WITH RESPECT TO Z
C              DNDXI(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                         FUNCTION WITH RESPECT TO XI AT NODE I
C             DNDETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                         FUNCTION WITH RESPECT TO ETA AT NODE I
C            DNDZETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                         FUNCTION WITH RESPECT TO ZETA AT NODE I
C                     E = ELEMENT NUMBER
C               IN(I,J) = NODE NUMBER J FOR ELEMENT I
C              JAC(I,J) = JACOBIAN MATRIX
C            JACINV(I,J) = INVERSE OF JACOBIAN MATRIX
C                    KXE = HYDRAULIC CONDUCTIVITY IN X DIRECTION
C                    KYE = HYDRAULIC CONDUCTIVITY IN Y DIRECTION
C                    KZE = HYDRAULIC CONDUCTIVITY IN Z DIRECTION
C                  V1(E) = APPARENT GROUNDWATER VELOCITY IN X DIRECTION
C                  V2(E) = APPARENT GROUNDWATER VELOCITY IN Y DIRECTION
C                    VZE = APPARENT GROUNDWATER VELOCITY IN Z DIRECTION
C             X(IN(E,I)) = COMPUTED HEAD FOR NODE I, ELEMENT E
C            X1(IN(E,I)) = X COORDINATE FOR NODE I, ELEMENT E
C            X2(IN(E,I)) = Y COORDINATE FOR NODE I, ELEMENT E
C            X3(IN(E,I)) = Z COORDINATE FOR NODE I, ELEMENT E
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I, J
      DOUBLE PRECISION JAC(3,3), JACINV(3,3), DNDX(MAX3,MAX2,3)
      DOUBLE PRECISION DNDXI(20), DNDETA(20)
      DOUBLE PRECISION DNDZETA(20), DETJAC
      DOUBLE PRECISION SIGN1(20), SIGN2(20), SIGN3(20)
      DOUBLE PRECISION KXE, KYE, KZE
      DOUBLE PRECISION DHDX, DHDY, DHDZ
      DATA SIGN1/-1.0, 0.0, 1.0, 1.0, 1.0, 0.0,-1.0,-1.0,-1.0, 1.0,
     1           1.0,-1.0,-1.0, 0.0, 1.0, 1.0, 1.0, 0.0,-1.0,-1.0/
      DATA SIGN2/-1.0,-1.0,-1.0, 0.0, 1.0, 1.0, 1.0, 0.0,-1.0,-1.0,
     1           1.0, 1.0,-1.0,-1.0,-1.0, 0.0, 1.0, 1.0, 1.0, 0.0/
      DATA SIGN3/-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0, 0.0, 0.0,
     1           0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0/
      KXE = PROP(MATSET(E),1)
      KYE = PROP(MATSET(E),2)
      KZE = PROP(MATSET(E),3)
      DO 20 I = 1, 3
        DO 10 J = 1, 3
          JAC(I,J) = 0.0
 10     CONTINUE
 20   CONTINUE
      DO 30 I = 1, 20
```

```
      IF ((I .EQ.  1) .OR. (I .EQ.  3) .OR. (I .EQ.  5) .OR.
     1    (I .EQ.  7) .OR. (I .EQ. 13) .OR. (I .EQ. 15) .OR.
     2    (I .EQ. 17) .OR. (I .EQ. 19)) THEN
        DNDXI(I)   = -0.125 * SIGN1(I)
        DNDETA(I)  = -0.125 * SIGN2(I)
        DNDZETA(I) = -0.125 * SIGN3(I)
      ELSEIF ((I .EQ.  2) .OR. (I .EQ.  6) .OR.
     1        (I .EQ. 14) .OR. (I .EQ. 18)) THEN
        DNDXI(I)   = 0.0
        DNDETA(I)  = 0.25 * SIGN2(I)
        DNDZETA(I) = 0.25 * SIGN3(I)
      ELSEIF ((I .EQ.  4) .OR. (I .EQ.  8) .OR.
     1        (I .EQ. 16) .OR. (I .EQ. 20)) THEN
        DNDXI(I)   = 0.25 * SIGN1(I)
        DNDETA(I)  = 0.0
        DNDZETA(I) = 0.25 * SIGN3(I)
      ELSEIF ((I .GE. 9) .AND. (I .LE. 12)) THEN
        DNDXI(I)   = 0.25 * SIGN1(I)
        DNDETA(I)  = 0.25 * SIGN2(I)
        DNDZETA(I) = 0.0
      ENDIF
 30   CONTINUE
      DO 40 I = 1, 20
        JAC(1,1) = JAC(1,1) + DNDXI(I) * X1(IN(E,I))
        JAC(1,2) = JAC(1,2) + DNDXI(I) * X2(IN(E,I))
        JAC(1,3) = JAC(1,3) + DNDXI(I) * X3(IN(E,I))
        JAC(2,1) = JAC(2,1) + DNDETA(I) * X1(IN(E,I))
        JAC(2,2) = JAC(2,2) + DNDETA(I) * X2(IN(E,I))
        JAC(2,3) = JAC(2,3) + DNDETA(I) * X3(IN(E,I))
        JAC(3,1) = JAC(3,1) + DNDZETA(I) * X1(IN(E,I))
        JAC(3,2) = JAC(3,2) + DNDZETA(I) * X2(IN(E,I))
        JAC(3,3) = JAC(3,3) + DNDZETA(I) * X3(IN(E,I))
 40   CONTINUE
      DETJAC = JAC(1,1) * (JAC(2,2)*JAC(3,3) - JAC(3,2)*JAC(2,3))
     1       - JAC(1,2) * (JAC(2,1)*JAC(3,3) - JAC(3,1)*JAC(2,3))
     2       - JAC(1,3) * (JAC(2,1)*JAC(3,2) - JAC(3,1)*JAC(2,2))
      IF (DETJAC .EQ. 0.0 ) STOP 'DETERMINANT IS ZERO !!!!!'
C     INVERSEJACOBIAN MATRIX FORMULA HAS BEEN TRANSPOSEDED - STEVE 9/7/96
           JACINV(1,1) = ( JAC(2,2) * JAC(3,3) - JAC(2,3) * JAC(3,2))
     1                   / DETJAC
           JACINV(2,1) = (-JAC(2,1) * JAC(3,3) + JAC(2,3) * JAC(3,1))
     1                   / DETJAC
           JACINV(3,1) = ( JAC(2,1) * JAC(3,2) - JAC(2,2) * JAC(3,1))
     1                   / DETJAC
           JACINV(1,2) = (-JAC(1,2) * JAC(3,3) + JAC(1,3) * JAC(3,2))
     1                   / DETJAC
           JACINV(2,2) = ( JAC(1,1) * JAC(3,3) - JAC(1,3) * JAC(3,1))
     1                   / DETJAC
           JACINV(3,2) = (-JAC(1,1) * JAC(3,2) + JAC(1,2) * JAC(3,1))
     1                   / DETJAC
           JACINV(1,3) = ( JAC(1,2) * JAC(2,3) - JAC(1,3) * JAC(2,2))
     1                   / DETJAC
           JACINV(2,3) = (-JAC(1,1) * JAC(2,3) + JAC(1,3) * JAC(2,1))
     1                   / DETJAC
           JACINV(3,3) = ( JAC(1,1) * JAC(2,2) - JAC(1,2) * JAC(2,1))
     1                   / DETJAC
      DO 50 I = 1, 20
        DNDX(I,E,1) = JACINV(1,1) * DNDXI(I) + JACINV(1,2) *
     1                DNDETA(I) + JACINV(1,3) * DNDZETA(I)
        DNDX(I,E,2) = JACINV(2,1) * DNDXI(I) + JACINV(2,2) *
     1                DNDETA(I) + JACINV(2,3) * DNDZETA(I)
        DNDX(I,E,3) = JACINV(3,1) * DNDXI(I) + JACINV(3,2) *
     1                DNDETA(I) + JACINV(3,3) * DNDZETA(I)
 50   CONTINUE
      DHDX = 0.0
      DHDY = 0.0
      DHDZ = 0.0
      DO 60 I = 1, 20
```

```
        DHDX = DHDX + DNDX(I,E,1) * X(IN(E,I))
        DHDY = DHDY + DNDX(I,E,2) * X(IN(E,I))
        DHDZ = DHDZ + DNDX(I,E,3) * X(IN(E,I))
 60   CONTINUE
      V1(E) = -KXE * DHDX
      V2(E) = -KYE * DHDY
      V3(E) = -KZE * DHDZ
      RETURN
      END

      SUBROUTINE VPAR32(E,DNDX)
C*****************************************************************
C
C         PURPOSE:
C           TO COMPUTE COMPONENTS OF APPARENT GROUNDWATER
C           VELOCITY FOR A THREE DIMENSIONAL, CUBIC
C           PARALLELEPIPED ELEMENT
C
C         DEFINITIONS OF VARIABLES:
C               DETJAC = DETERMINANT OF JACOBIAN MATRIX
C                 DHDX = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO X
C                 DHDY = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO Y
C                 DHDZ = PARTIAL DERIVATIVE OF HEAD WITH RESPECT TO Z
C           DNDX(I,E,1) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                         OF NODE I OF ELEMENT E WITH RESPECT TO X
C           DNDX(I,E,2) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                         OF NODE I OF ELEMENT E WITH RESPECT TO Y
C           DNDX(I,E,3) = PARTIAL DERIVATIVE OF INTERPOLATION FUNCTION
C                         OF NODE I OF ELEMENT E WITH RESPECT TO Z
C              DNDXI(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                         FUNCTION WITH RESPECT TO XI AT NODE I
C             DNDETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                         FUNCTION WITH RESPECT TO ETA AT NODE I
C            DNDZETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                         FUNCTION WITH RESPECT TO ZETA AT NODE I
C                     E = ELEMENT NUMBER
C               IN(I,J) = NODE NUMBER J FOR ELEMENT I
C              JAC(I,J) = JACOBIAN MATRIX
C            JACINV(I,J) = INVERSE OF JACOBIAN MATRIX
C                   KXE = HYDRAULIC CONDUCTIVITY IN X DIRECTION
C                   KYE = HYDRAULIC CONDUCTIVITY IN Y DIRECTION
C                   KZE = HYDRAULIC CONDUCTIVITY IN Z DIRECTION
C                 V1(E) = APPARENT GROUNDWATER VELOCITY IN X DIRECTION
C                 V2(E) = APPARENT GROUNDWATER VELOCITY IN Y DIRECTION
C                   VZE = APPARENT GROUNDWATER VELOCITY IN Z DIRECTION
C             X(IN(E,I) = COMPUTED HEAD FOR NODE I, ELEMENT E
C            X1(IN(E,I) = X COORDINATE FOR NODE I, ELEMENT E
C            X2(IN(E,I) = Y COORDINATE FOR NODE I, ELEMENT E
C            X3(IN(E,I) = Z COORDINATE FOR NODE I, ELEMENT E
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I, J
      DOUBLE PRECISION JAC(3,3), JACINV(3,3), DNDX(MAX3,MAX2,3)
      DOUBLE PRECISION DNDXI(32), DNDETA(32)
      DOUBLE PRECISION DNDZETA(32), DETJAC
      DOUBLE PRECISION SIGN1(32), SIGN2(32), SIGN3(32)
      DOUBLE PRECISION KXE, KYE, KZE
      DOUBLE PRECISION DHDX, DHDY, DHDZ
      DATA SIGN1/ 2*-1.0, 6* 1.0, 5*-1.0, 2* 1.0,
     1            2*-1.0, 2* 1.0, 3*-1.0, 6* 1.0, 4*-1.0/
      DATA SIGN2/ 5*-1.0, 6* 1.0, 3*-1.0, 2* 1.0,
     1            2*-1.0, 2* 1.0, 5*-1.0, 6* 1.0,   -1.0/
      DATA SIGN3/ 16*-1.0, 16*1.0/
      KXE = PROP(MATSET(E),1)
      KYE = PROP(MATSET(E),2)
      KZE = PROP(MATSET(E),3)
      DO 20 I = 1, 3
```

```
        DO 10 J = 1, 3
          JAC(I,J) = 0.0
 10     CONTINUE
 20    CONTINUE
       DO 30 I = 1, 32
         IF ((I .EQ.  1) .OR. (I .EQ.  4) .OR. (I .EQ.  7) .OR.
     1       (I .EQ. 10) .OR. (I .EQ. 21) .OR. (I .EQ. 24) .OR.
     2       (I .EQ. 27) .OR. (I .EQ. 30)) THEN
           DNDXI(I)   = (-19.0 / 64.0) * SIGN1(I)
           DNDETA(I)  = (-19.0 / 64.0) * SIGN2(I)
           DNDZETA(I) = (-19.0 / 64.0) * SIGN3(I)
         ELSEIF ((I .EQ.  2) .OR. (I .EQ.  3) .OR. (I .EQ.  8) .OR.
     1           (I .EQ.  9) .OR. (I .EQ. 22) .OR. (I .EQ. 23) .OR.
     2           (I .EQ. 28) .OR. (I .EQ. 29)) THEN
           DNDXI(I)   = (27.0 / 64.0) * SIGN1(I)
           DNDETA(I)  = ( 9.0 / 64.0) * SIGN2(I)
           DNDZETA(I) = ( 9.0 / 64.0) * SIGN3(I)
         ELSEIF ((I .EQ.  5) .OR. (I .EQ.  6) .OR. (I .EQ. 11) .OR.
     1           (I .EQ. 12) .OR. (I .EQ. 25) .OR. (I .EQ. 26) .OR.
     2           (I .EQ. 31) .OR. (I .EQ. 32)) THEN
           DNDXI(I)   = ( 9.0 / 64.0) * SIGN1(I)
           DNDETA(I)  = (27.0 / 64.0) * SIGN2(I)
           DNDZETA(I) = ( 9.0 / 64.0) * SIGN3(I)
         ELSEIF ((I .GE. 13) .AND. (I .LE. 20)) THEN
           DNDXI(I)   = ( 9.0 / 64.0) * SIGN1(I)
           DNDETA(I)  = ( 9.0 / 64.0) * SIGN2(I)
           DNDZETA(I) = (27.0 / 64.0) * SIGN3(I)
         ENDIF
 30    CONTINUE
       DO 40 I = 1, 32
         JAC(1,1) = JAC(1,1) + DNDXI(I) * X1(IN(E,I))
         JAC(1,2) = JAC(1,2) + DNDXI(I) * X2(IN(E,I))
         JAC(1,3) = JAC(1,3) + DNDXI(I) * X3(IN(E,I))
         JAC(2,1) = JAC(2,1) + DNDETA(I) * X1(IN(E,I))
         JAC(2,2) = JAC(2,2) + DNDETA(I) * X2(IN(E,I))
         JAC(2,3) = JAC(2,3) + DNDETA(I) * X3(IN(E,I))
         JAC(3,1) = JAC(3,1) + DNDZETA(I) * X1(IN(E,I))
         JAC(3,2) = JAC(3,2) + DNDZETA(I) * X2(IN(E,I))
         JAC(3,3) = JAC(3,3) + DNDZETA(I) * X3(IN(E,I))
 40    CONTINUE
       DETJAC = JAC(1,1) * (JAC(2,2)*JAC(3,3) - JAC(3,2)*JAC(2,3))
     1        - JAC(1,2) * (JAC(2,1)*JAC(3,3) - JAC(3,1)*JAC(2,3))
     2        - JAC(1,3) * (JAC(2,1)*JAC(3,2) - JAC(3,1)*JAC(2,2))
       IF (DETJAC .EQ. 0.0 ) STOP 'DETERMINANT IS ZERO !!!!!'
C      INVERSEJACOBIAN MATRIX FORMULA HAS BEEN TRANSPOSEDED - STEVE 9/7/96
             JACINV(1,1) = ( JAC(2,2) * JAC(3,3) - JAC(2,3) * JAC(3,2))
     1                     / DETJAC
             JACINV(2,1) = (-JAC(2,1) * JAC(3,3) + JAC(2,3) * JAC(3,1))
     1                     / DETJAC
             JACINV(3,1) = ( JAC(2,1) * JAC(3,2) - JAC(2,2) * JAC(3,1))
     1                     / DETJAC
             JACINV(1,2) = (-JAC(1,2) * JAC(3,3) + JAC(1,3) * JAC(3,2))
     1                     / DETJAC
             JACINV(2,2) = ( JAC(1,1) * JAC(3,3) - JAC(1,3) * JAC(3,1))
     1                     / DETJAC
             JACINV(3,2) = (-JAC(1,1) * JAC(3,2) + JAC(1,2) * JAC(3,1))
     1                     / DETJAC
             JACINV(1,3) = ( JAC(1,2) * JAC(2,3) - JAC(1,3) * JAC(2,2))
     1                     / DETJAC
             JACINV(2,3) = (-JAC(1,1) * JAC(2,3) + JAC(1,3) * JAC(2,1))
     1                     / DETJAC
             JACINV(3,3) = ( JAC(1,1) * JAC(2,2) - JAC(1,2) * JAC(2,1))
     1                     / DETJAC
       DO 50 I = 1, 32
         DNDX(I,E,1) = JACINV(1,1) * DNDXI(I) + JACINV(1,2) *
     1                 DNDETA(I) + JACINV(1,3) * DNDZETA(I)
         DNDX(I,E,2) = JACINV(2,1) * DNDXI(I) + JACINV(2,2) *
     1                 DNDETA(I) + JACINV(2,3) * DNDZETA(I)
```

```
        DNDX(I,E,3) = JACINV(3,1) * DNDXI(I) + JACINV(3,2) *
     1                DNDETA(I) + JACINV(3,3) * DNDZETA(I)
 50   CONTINUE
      DHDX = 0.0
      DHDY = 0.0
      DHDZ = 0.0
      DO 60 I = 1, 32
        DHDX = DHDX + DNDX(I,E,1) * X(IN(E,I))
        DHDY = DHDY + DNDX(I,E,2) * X(IN(E,I))
        DHDZ = DHDZ + DNDX(I,E,3) * X(IN(E,I))
 60   CONTINUE
      V1(E) = -KXE * DHDX
      V2(E) = -KYE * DHDY
      V3(E) = -KZE * DHDZ
      RETURN
      END

SUBROUTINE CONCEN(PRIOR)
C**********************************************************************
C
C 19.1  PURPOSE
C         TO DETERMINE THE CONCENTRATIONS AT EACH NODAL POINT
C         AND THE DERIVATIVES THE DERIVATIVES OF EACH NODAL
C         CONCENTRATION VALUE WITH RESPECT TO THE VELOCITY
C         COMPONENTS OF EACH ELEMENT.  THESE VALUES ARE
C         STORED IN UNFORMATTED FILES.
C
C 19.2  INPUT
C         NONE
C
C 19.3  OUTPUT
C         VALUES OF
C
C 19.4  DEFINITIONS OF VARIABLES:
C                      AE(I,J) = SORPTION MATRIX FOR ELEMENT E IN FULL
C                                MATRIX STORAGE
C                         B(I) = RHS USED TO DETERMINE DERIVATIVES
C                     DDE(I,J) = DERIVATIVE OF THE ADVECTION-DISPERSION
C                                MATRIX FOR ELEMENT E IN FULL MATRIX STORAGE
C                            E = ELEMENT NUMBER
C                   ELEMTYP(E) = ELEMENT TYPE FOR ELEMENT E
C                      FLUX(I) = SPECIFIED VALUE OF SOLUTE FLUX AT NODE I
C                       ICH(I) = 1 IF THE VALUE OF THE FIELD VARIABLE IS
C                                SPECIFIED AT NODE I
C                              = 0 OTHERWISE
C                      IJSIZE = LENGTH OF ARRAY ADGLOBAL
C                      LCH(I) = ICH(I) + ICH(I-1) + ...
C                       M(IJ) = MODIFIED, COMBINED GLOBAL SORPTION AND
C                                ADVECTION-DISPERSION MATRIX IN VECTOR
C                                STORAGE
C                        NDOF = NUMBER OF NODES WHERE THE VALUE OF
C                                THE FIELD VARIABLE IS UNKNOWN
C         NODETBL(ELEMTYP(E)) = NUMBER OF NODES IN ELEMENT TYPE E
C                      NUMELM = NUMBER OF ELEMENTS IN THE MESH
C                         SBW = SEMI-BANDWIDTH
C                        X(I) = VALUE OF SOLUTE CONCENTRATION AT NODE I
C
C 19.5  USAGE:
C         SIMILAR TO DHDK, DCDV IS DETERMINED USING:
C
C               M * DCDV = DFDV - DDDV * C
C
C         NOW
C           F = (A - (1-w)dt * D) Ct-1 + dt((1-w)Ft-1 + wFt-1)
C
C         AND ONLY C AND D ARE FUNCTIONS OF V
C         THEREFORE
C
```

```
C           DFDV = (A - (1-w)dt * D) * DCDVt-1
C                      + (1-w)dt * DDDV * Ct-1
C
C
C         SIMILAR TO D2HDK2, D2CDK2 IS DETERMINED USING:
C               M * D2CDK2(I,J) = D2FDK2(I,J)
C                                 - DDDK(I) * DCDK(J)
C                                 - DDDK(J) * DCDK(I)
C                                 - D2DDK2(I,J) * C
C
C         AS ABOVE
C
C           DF2DV2(I,J) = (A - (1-w)dt * D) D2CDV2(I,J)t-1
C                         + (1-w)dt * DDDV(I) * DCDV(J)t-1
C                         + (1-w)dt * DDDV(J) * DCDV(I)t-1
C                         + (1-w)dt * D2DDV2(I,J) * Ct-1
C
C       SUBROUTINES CALLED:
C         HEAPS, INCLUDING LOCATE AND DD~
C
C*************************************************************************
      INCLUDE 'COMALL'
      DOUBLE PRECISION DDE(MAX3,MAX3,3,0:3), C(MAX1,2)
      DOUBLE PRECISION TEMP(3,3), IPIV(MAX0),TEMPS(3,3,MAX0)
      DOUBLE PRECISION DDE2(MAX3,MAX3,3,0:0), Y(MAX0,3)
      DOUBLE PRECISION TEMP1, TEMP2, TEMP3, TEMP4, MASS
      INTEGER ISTEP, ISTART, OLD, NEW, REFLEC(2,4), INFO
      INTEGER NODETBL(13), I, I2, J, J2, J3, J4, J5, K, K2, L
      INTEGER E, E2, E3, E4, LIMIT, RECORD, K3, K4
      LOGICAL PRIOR, RESTART, BEGIN
      LOGICAL INCSIG
      REAL TARRAY(2), SECS, ETIME
      DATA NODETBL/2,3,4,3,4,4,8,12,8,20,32,3,4/

C     INCSIG INCLUDES THE SIGMA CALCULATIONS.
      INCSIG = .TRUE.
      BEGIN = .TRUE.
      IF (PRIOR) THEN
        CALL CLOAD(E3,E4)
      ELSE
        IDT = 0
        DO 30 ISTEP = 1, MXSTEP
          IF (ISTEP .EQ. 1 .OR. ISTEP .GT. DTSTEP(IDT)) THEN
            IDT = IDT + 1
            DO 20 L = 1, NUMMAT
              DO 10 J = 1, NDOF
                CMEAN(J,IDT,L) = 0.0
                CVAR(J,IDT,L)  = 0.0
 10           CONTINUE
              DO 20 K = 1, DIM
                DO 20 K2 = 1, K
                  SIGMA(K,K2,IDT,L) = 0.0
 20         CONTINUE
          ENDIF
 30     CONTINUE
        E3 = 1
        E4 = 0
      ENDIF
      IDT = 0
      DO 40 I = 1, NUMNOD
        IF (ICH(I) .EQ. 0) THEN
          Y(I-LCH(I),1) = X1(I)
          Y(I-LCH(I),2) = X2(I)
          Y(I-LCH(I),3) = X3(I)
        ENDIF
 40   CONTINUE
C     THIS LOOP DETERMINES THAT THE FIRST TIMESTEP EQUALS THE LAST
      IDT = 0
```

```
      DO 45 ISTEP = 1, MXSTEP
        IF (ISTEP .EQ. 1 .OR. ISTEP .GT. DTSTEP(IDT)) THEN
          IDT = IDT + 1
        ENDIF
 45   CONTINUE
      RESTART = DELTAT(1) .NE. DELTAT(IDT)
      CALL VLOAD
      print *,'vmean read'
      REFLEC(1,1) =  1
      REFLEC(1,2) =  1
      REFLEC(1,3) =  1
      REFLEC(1,4) =  1
      REFLEC(2,1) =  1
      REFLEC(2,2) = -1
      REFLEC(2,3) =  1
      REFLEC(2,4) = -1
C     THE NEXT TWO LOOPS MUST NOT BE PARALLELISED SO AUTOSAVING FUNCTIONS
      DO 270 E = E3, NUMELM
        PRINT *,'BEGINNING CONCENTRATION FOR ELEMENT',E
        IF (E .LE. NUMELM / 4) THEN
          LIMIT = E
          J5 = 1
        ELSEIF (E .LE. NUMELM / 2) THEN
          LIMIT = E
          J5 = 2
        ELSEIF (E .LE. 3 * NUMELM / 4) THEN
          LIMIT = NUMELM + 1 - E
          J5 = 3
        ELSE
          LIMIT = NUMELM + 1 - E
          J5 = 4
        ENDIF
        DO 260 E2 = E4 + 1, LIMIT
C         OBTAIN VELOCITY COVARIANCE DATA
          IF (E .LE. NUMELM / 4) THEN
            J = E * (E - 1) / 2 + E2
          ELSEIF (E .LE. NUMELM / 2) THEN
            IF (E2 .LE. NUMELM / 2 + 1 - E) THEN
              J = NUMELM / 4 * (NUMELM / 4 + 1)
     1          - (NUMELM/2-E+1) * (NUMELM/2-E+2) / 2 + E2
            ELSEIF(E2 .LE. NUMELM / 4) THEN
              J  = NUMELM / 2 + 1 - E2
              J2 = NUMELM / 2 + 1 - E
              J  = NUMELM / 2 * (NUMELM / 4 + 1) / 2
     1          - (NUMELM/2-J+1) * (NUMELM/2-J+2) / 2 + J2
            ELSE
              J  = NUMELM / 2 + 1 - E2
              J2 = NUMELM / 2 + 1 - E
              J  = J * (J - 1) / 2 + J2
            ENDIF
          ELSEIF (E .LE. 3 * NUMELM / 4) THEN
            IF (E2 .LE. E - NUMELM / 2) THEN
              J = NUMELM / 2 * (NUMELM / 4 + 1) / 2
     1          + (E-NUMELM/2) * (E-NUMELM/2-1) / 2 + E2
            ELSEIF(E2 .LE. NUMELM / 4) THEN
              J  = E2 + NUMELM / 2
              J2 = E - NUMELM / 2
              J = NUMELM / 2 * (NUMELM / 4 + 1) / 2
     1          + (J-NUMELM/2) * (J-NUMELM/2-1) / 2 + J2
            ELSE
              J  = NUMELM / 2 + E2
              J2 = E - NUMELM / 2
              J = NUMELM - J + 1
              J = J * (J + 1) / 2
              J = NUMELM * (NUMELM / 4 + 1) / 2 - J + J2
            ENDIF
          ELSE
            J = NUMELM - E + 1
```

```
            J2 = J * (J + 1) / 2
            J  = NUMELM * (NUMELM / 4 + 1) / 2 - J2 + E2
          ENDIF
C         SINCE THE WHOLE ARRAY IS BEING WRITTEN OUT REGARDLESS
C         OF EMPTY SPACES IT IS NECESARRY TO USE ABSOLUTE
C         REFERENCES HERE.  THIS SPEEDS UP THE EARLIER
C         CHECKPOINTING OF VERY LARGE FILES.
C         ** RECORD = NUMMAT * DIM * (DIM * (J - 1) + NUMELM / 4)
          RECORD = MAX4 * 2 * (2 * (J - 1) + MAX2 / 4)
          IF ((E .LE. NUMELM / 2 .AND. E2 .GT. NUMELM / 2 + 1 - E) .OR.
     1        (E .GT. NUMELM / 2 .AND. E2 .GT. E - NUMELM / 2)) THEN
            DO 60 K = 1, DIM
              DO 60 K2 = 1, DIM
                DO 50 L = 1, NUMMAT
                  RECORD = RECORD + 1
                  VCOR(K2,K,L) = BLOCK(RECORD)
 50               CONTINUE
c                  RECORD = RECORD + MAX4 - 1
 60           CONTINUE
            IF (E .LE. NUMELM / 2) THEN
              DO 70 L = 1, NUMMAT
                VCOR(2,1,L) = - VCOR(2,1,L)
                VCOR(1,2,L) = - VCOR(1,2,L)
 70           CONTINUE
            ENDIF
          ELSE
            DO 90 K = 1, DIM
              DO 90 K2 = 1, DIM
                DO 80 L = 1, NUMMAT
                  RECORD = RECORD + 1
                  VCOR(K,K2,L) = BLOCK(RECORD)
 80               CONTINUE
c                  RECORD = RECORD + MAX4 - 1
 90           CONTINUE
          ENDIF

C         INITIALISE COUNTERS

          IDT = 0
          IGT = 1
          IGTDT = 1
          T = 0.0
          J = 0
          OLD = 1
          NEW = 2
          DO 120 I = 1, NUMNOD
            IF (ICH(I) .EQ. 0) THEN
              J = J + 1
              C(J,OLD) = X(I)
              DO 110 K = 1, DIM
                DO 100 J2 = 1,2
                  DO 100 J3 = 1,2
                    DCDV(J,J2,K,J3) = 0.0
 100              CONTINUE
                DO 110 K2 = 1, DIM
                  D2CDV2(J,K,K2) = 0.0
 110            CONTINUE
            ENDIF
 120      CONTINUE
C
C         FOR EACH TIME STEP. . .
C
          DO 250 ISTEP = 1, MXSTEP
            IF (ISTEP .EQ. 1 .OR. ISTEP .GT. DTSTEP(IDT)) THEN
              IDT = IDT + 1
C             IF TIMESTEP OR ELEMENTS CHANGE COMPUTE THE
C             DERIVATIVE OF THE ELEMENT ADVECTION-
C             DISPERSION MATRIX FOR THESE ELEMENT TYPES
```

```
              IF (ISTEP .EQ. 1
     1            .OR. DELTAT(IDT) .NE. DELTAT(IDT-1)) THEN
                IF (ELEMTYP(E) .EQ. 1) THEN
C                 ELEMENT IS A ONE DIMENSIONAL, LINEAR BAR
                  CALL DDBAR2(E,DDE)
                ELSEIF (ELEMTYP(E) .EQ. 4) THEN
C                 ELEMENT IS A TWO DIMENSIONAL, LINEAR TRIANGLE
                  CALL DDTRI3(E,DDE,1)
                ELSEIF (ELEMTYP(E) .EQ. 5) THEN
C                 ELEMENT IS A TWO DIMENSIONAL, LINEAR RECTANGLE
                  CALL DDREC4(E,DDE,1)
                ELSEIF (ELEMTYP(E) .EQ. 6) THEN
C                 ELEMENT IS A TWO DIMENSIONAL, LINEAR QUADRILATERAL
                  CALL DDQUA4(E,DDE,1)
                ELSEIF (ELEMTYP(E) .EQ. 9) THEN
C                 ELEMENT IS A THREE DIMENSIONAL, LINEAR PARALLELEPIPED
                  CALL DDPAR8(E,DDE,1)
                ENDIF
                IF (ELEMTYP(E2) .EQ. 1) THEN
C                 ELEMENT IS A ONE DIMENSIONAL, LINEAR BAR
                  CALL DDBAR2(E2,DDE2)
                ELSEIF (ELEMTYP(E2) .EQ. 4) THEN
C                 ELEMENT IS A TWO DIMENSIONAL, LINEAR TRIANGLE
                  CALL DDTRI3(E2,DDE2,0)
                ELSEIF (ELEMTYP(E2) .EQ. 5) THEN
C                 ELEMENT IS A TWO DIMENSIONAL, LINEAR RECTANGLE
                  CALL DDREC4(E2,DDE2,0)
                ELSEIF (ELEMTYP(E2) .EQ. 6) THEN
C                 ELEMENT IS A TWO DIMENSIONAL, LINEAR QUADRILATERAL
                  CALL DDQUA4(E2,DDE2,0)
                ELSEIF (ELEMTYP(E2) .EQ. 9) THEN
C                 ELEMENT IS A THREE DIMENSIONAL, LINEAR PARALLELEPIPED
                  CALL DDPAR8(E2,DDE2,0)
                ENDIF
              ENDIF
C             IF SIZE OF TIMESTEP CHANGES REASSEMBLE GLOBAL MATRICES
              IF ((ISTEP .EQ. 1 .AND. RESTART) .OR.
     1                BEGIN .OR. (ISTEP .NE. 1 .AND.
     2                DELTAT(IDT) .NE. DELTAT(IDT-1))) THEN

                CALL ASMBAD
                CALL DGBTRF (NDOF,NDOF,SBW,SBW,M2,3*MAX6+1,IPIV,INFO)
                IF (INFO .NE. 0) PRINT *,'ERROR FACTORISING',INFO
                BEGIN = .FALSE.
              ENDIF
            ENDIF

C           SOLVE FOR CONCENTRATION
            CALL RHS0(C,OLD,NEW)
            CALL DGBTRS('NotTransposed',NDOF,SBW,SBW,1,
     1                  M2,3*MAX6+1,IPIV,C(1,NEW),MAX0,INFO)
C           SOLVE FOR FIRST DERIVATIVES
            DO 130 K = 1, DIM
              CALL RHS1(C,DDE,E,K,1,OLD,NEW)
              CALL RHS1(C,DDE2,E2,K,2,OLD,NEW)
 130        CONTINUE
            CALL DGBTRS('NotTransposed',NDOF,SBW,SBW,2*DIM,
     1                  M2,3*MAX6+1,IPIV,DCDV(1,1,1,NEW),MAX0,INFO)
            IF (ISTEP .EQ. DTSTEP(IDT) .AND. E2 .EQ. 1) THEN
              DO 150 K = 1, DIM
                IF (E .LE. NUMELM / 4) THEN
                  J2 = E
                ELSEIF (E .LE. NUMELM / 2) THEN
                  J2 = NUMELM / 2 - E + 1
                ELSEIF (E .LE. 3 * NUMELM / 4) THEN
                  J2 = E - NUMELM / 2
                ELSE
                  J2 = NUMELM - E + 1
```

```
                ENDIF
                DO 140 J = 1, NDOF
                  DO 140 L = 1, NUMMAT
                    CMEAN(J,IDT,L) = CMEAN(J,IDT,L)
     1              + DCDV(J,1,K,NEW) * REFLEC(K,J5) * VMEAN(L,K,J2)
 140              CONTINUE
 150            CONTINUE
              ENDIF

C             SOLVE FOR SECOND DERIVATIVES
              IF (E .EQ. E2) THEN
                CALL RHS2SAME(C,DDE,E,OLD,NEW)
              ELSE
                CALL RHS2DIFF(DDE,DDE2,E,E2,OLD,NEW)
              ENDIF
C             THIS NEXT LINE HAS THE PROBLEM THAT WHEN DIM = 2 IT SOLVES
C             FOR FIVE RHS INSTEAD OF FOUR.  THIS IS BECAUSE THERE IS
C             A GAP BETWEEN TWO PAIRS OF TWO.
              CALL DGBTRS('NotTransposed',NDOF,SBW,SBW,(DIM-1)*3+DIM,
     1                    M2,3*MAX6+1,IPIV,D2CDV2,MAX0,INFO)
C             ADD INCREMENTS TO RESULTS
              IF (ISTEP .EQ. DTSTEP(IDT)) THEN
                DO 170 J = 1, NDOF
                  DO 170 K = 1, DIM
                    DO 170 K2 = 1, DIM
                      IF (E2 .EQ. NUMELM + 1 - E) THEN
                        TEMP1 = 0.5 * D2CDV2(J,K,K2)
                        TEMP2 = DCDV(J,1,K,NEW) * DCDV(J,2,K2,NEW)
                      ELSE
                        J3 = (J + NDN - 1) / NDN * NDN - MOD(J-1, NDN)
                        TEMP1 = 0.5 * (D2CDV2(J,K,K2)+D2CDV2(J3,K,K2))
                        TEMP2 = DCDV(J, 1,K,NEW) * DCDV(J, 2,K2,NEW)
     1                        + DCDV(J3,1,K,NEW) * DCDV(J3,2,K2,NEW)
                      ENDIF
                      IF (E .NE. E2) THEN
                        TEMP1 = 2 * TEMP1
                        TEMP2 = 2 * TEMP2
                      ENDIF
                      DO 170 L = 1, NUMMAT
                        CMEAN(J,IDT,L) = CMEAN(J,IDT,L)
     1                                 + TEMP1 * VCOR(K,K2,L)
                        CVAR(J,IDT,L) = CVAR(J,IDT,L)
     1                                + TEMP2 * VCOR(K,K2,L)
 170            CONTINUE

                IF (INCSIG) THEN
                  DO 240 K = 1, DIM
                    DO 240 K2 = 1, DIM
                      DO 180 J = 1, NDOF
                        DO 180 K3 = 1, DIM
                          DO 180 K4 = 1, DIM
                            TEMPS(K4,K3,J) = 0D0
 180                  CONTINUE
                      DO 200 J = 1, NDOF
                        DO 190 J2 = 1, NDOF
                          IF (E2 .EQ. NUMELM + 1 - E) THEN
                            TEMP1 = DCDV(J,1,K,NEW) * DCDV(J2,2,K2,NEW)
C                           J5 IS SET TO ZERO TO HELP PARALLELISE
                            J5 = 0
                          ELSE
                            J5 = (J + NDN - 1) / NDN * NDN - MOD(J-1, NDN)
                            J4 = (J2+NDN-1) / NDN * NDN - MOD(J2-1,NDN)
                            TEMP1 = DCDV(J, 1,K,NEW) * DCDV(J2,2,K2,NEW)
     1                            + DCDV(J5,1,K,NEW) * DCDV(J4,2,K2,NEW)
                          ENDIF
                          IF (E .NE. E2) THEN
                            TEMP1 = 2 * TEMP1
                          ENDIF
```

```
                        DO 190 K4 = 1, DIM
                          DO 190 K3 = 1, K4
                            TEMPS(K4,K3,J) = TEMPS(K4,K3,J) + TEMP1 *
     1                       (Y(J,K4) - Y(J2,K4)) * (Y(J,K3) - Y(J2,K3))
 190                    CONTINUE
 200                  CONTINUE
                      DO 210 K3 = 1, DIM
                        DO 210 K4 = 1, DIM
                          TEMP(K3,K4) = 0D0
 210                  CONTINUE
ccccccccccC$DOACROSS NEST(K3, K4)
                      DO 220 K3 = 1, DIM
                        DO 220 K4 = 1, DIM
                          DO 220 J = 1, NDOF
                            TEMP(K3,K4) = TEMP(K3,K4) + TEMPS(K3,K4,J)
 220                  CONTINUE
                      DO 230 L = 1, NUMMAT
                        DO 230 K3 = 1, DIM
                          DO 230 K4 = 1, K3
                            SIGMA(K3,K4,IDT,L) = SIGMA(K3,K4,IDT,L)
     1                                       + VCOR(K,K2,L) * TEMP(K3,K4)
 230                  CONTINUE
 240                CONTINUE
C                 THIS ENDIF IS IF INCSIG
                  ENDIF
                ENDIF
                OLD = MOD(OLD, 2) + 1
                NEW = MOD(NEW, 2) + 1
 250          CONTINUE
              SECS = ETIME(TARRAY)
C             5200 => 2 threads
              IF (SECS .GT. 2400 .OR. CONTROL ) THEN
                print *,'EXECUTION TIME ',SECS,' SECONDS.'
                CALL CSAVE(E, E2)
                print *,'STOPPING AT TIME = ',SECS,' SECONDS.'
                STOP
              ENDIF
 260        CONTINUE
            E4 = 0
 270      CONTINUE

 333      print *,'finished stochastic concentration calcs',c(1,old)

          MASS = 0.0
          TEMP1 = 0.0
          TEMP2 = 0.0
          TEMP3 = 0.0
          DO 280 I = 1, NUMNOD
            MASS = MASS + X(I)
            TEMP1 = TEMP1 + X1(I) * X(I)
            TEMP2 = TEMP2 + X2(I) * X(I)
            TEMP3 = TEMP3 + X3(I) * X(I)
 280      CONTINUE
          WRITE (OUTF,290) MASS * ABS(X1(IN(1,3)) - X1(IN(1,1)))
         1                    * ABS(X2(IN(1,3)) - X2(IN(1,1))),
         2                 TEMP1 / MASS, TEMP2 / MASS
 290      FORMAT(/6X, 'INITIAL TOTAL MASS',
         1        6X, 'INITIAL X CENTER OF MASS',
         2        6X, 'INITIAL Y CENTER OF MASS'
         3       /9X,F12.3,13X,F12.3,18X,F12.3)
          TEMP(1,1) = 0.0
C         IF (DIM .GE. 2) THEN
            TEMP(2,1) = 0.0
            TEMP(2,2) = 0.0
C           IF (DIM .EQ. 3) THEN
C             TEMP(3,1) = 0.0
C             TEMP(3,2) = 0.0
C             TEMP(3,2) = 0.0
```

```
C        ENDIF
C      ENDIF
       DO 300 I = 1, NUMNOD
         TEMP(1,1) = TEMP(1,1) + X(I) * X1(I) * X1(I)
C         IF (DIM .GE. 2) THEN
           TEMP(2,1) = TEMP(2,1) + X(I) * X2(I) * X1(I)
           TEMP(2,2) = TEMP(2,2) + X(I) * X2(I) * X2(I)
C           IF (DIM .EQ. 3) THEN
C             TEMP(3,1) = TEMP(3,1) + X(I) * X3(I) * X1(I)
C             TEMP(3,2) = TEMP(3,2) + X(I) * X3(I) * X2(I)
C             TEMP(3,3) = TEMP(3,3) + X(I) * X3(I) * X3(I)
C           ENDIF
C         ENDIF
 300   CONTINUE
       WRITE(OUTF,310) SQRT(TEMP(1,1) / MASS - (TEMP1 / MASS) ** 2),
     1             SQRT(TEMP(2,1) / MASS - TEMP1*TEMP2/MASS**2),
     2             SQRT(TEMP(2,2) / MASS - (TEMP2 / MASS) ** 2)
 310   FORMAT(/31X,'INITIAL RADII OF GYRATION',
     1       /7X,'XX',F12.3,11X,'XY',F12.3,16X,'YY',F12.3)

       DO 400 L = 1, NUMMAT
         IDT = 0
         IGT = 1
         IGTDT = 1
         J = 0
         T = 0.0
         ISTART = 1
         OLD = 1
         NEW = 2
         DO 320 I = 1, NUMNOD
           IF (ICH(I) .EQ. 0) THEN
             C(I-LCH(I),OLD) = X(I)
           ENDIF
 320     CONTINUE
         DO 400 ISTEP = 1, MXSTEP
           IF (ISTEP .EQ. 1 .OR. ISTEP .GT. DTSTEP(IDT)) THEN
             IDT = IDT + 1
             IF ((ISTEP .EQ. 1 .AND. RESTART) .OR.
     1               BEGIN .OR. (ISTEP .NE. 1 .AND.
     2               DELTAT(IDT) .NE. DELTAT(IDT-1))) THEN
       print*,'re-assembling AD  matrix'
               CALL ASMBAD
       print *,'AD matrix re-assembled'
               CALL DGBTRF (NDOF,NDOF,SBW,SBW,M2,3*MAX6+1,IPIV,INFO)
               IF (INFO .NE. 0) PRINT *,'ERROR FACTORISING',INFO
       print *,'AD matrix re-decomposed'
             ENDIF
             DO 330 I = ISTART, DTSTEP(IDT)
               CALL RHS0(C,OLD,NEW)
               CALL DGBTRS('NotTransposed',NDOF,SBW,SBW,1,
     1                     M2,3*MAX6+1,IPIV,C(1,NEW),MAX0,INFO)
               OLD = MOD(OLD, 2) + 1
               NEW = MOD(NEW, 2) + 1
 330         CONTINUE
             ISTART = DTSTEP(IDT) + 1
             WRITE(OUTF, 340)L,T,LABEL1,'MEAN'
 340         FORMAT(//1X,74('*')//,23X,'RESULTS FOR MATERIAL SET',I3
     1                          //,23X,'AT TIME = ',F17.3
     1          //18X,'COMPUTED VALUES OF ',A/
     2          18X,36('-')//
     3          1X,'NODE_NO ','SOLUTE_CONCENTRATION',2X,A,8X,
     3        'FIRST_ORDER',4X,'TRUE_MEAN',7X,'X',6X,'Y',6X,'Z'/)
             J = 0
             DO 360 I = 1, NUMNOD
               IF (ICH(I) .EQ. 0) THEN
                 J = J + 1
                 IF (VAR(J,L) .GE. 0.0) THEN
                   WRITE(OUTF,350) I,C(J,OLD),CMEAN(J,IDT,L),
```

```
     3                    SQRT(CVAR(J,IDT,L)),
     1                   C(J,OLD) + CMEAN(J,IDT,L),X1(I),X2(I),X3(I)
               ELSE
                 WRITE(OUTF,350) I,C(J,OLD),CMEAN(J,IDT,L),
     3                    -SQRT(-CVAR(J,IDT,L)),
     1                   C(J,OLD) + CMEAN(J,IDT,L),X1(I),X2(I),X3(I)
               ENDIF
             ELSE
               WRITE(OUTF,350) I,X(I),0.0,0.0,X(I),X1(I),X2(I),X3(I)
             ENDIF
 350         FORMAT (I5,1X,4F16.12,3F7.2)
 360       CONTINUE
           MASS = 0.0
           TEMP1 = 0.0
           TEMP(1,1) = 0.0
C           IF (DIM .GE. 2) THEN
             TEMP2 = 0.0
             TEMP(2,1) = 0.0
             TEMP(2,2) = 0.0
C              IF (DIM .EQ. 3) THEN
C                TEMP3 = 0.0
C                TEMP(3,1) = 0.0
C                TEMP(3,2) = 0.0
C                TEMP(3,2) = 0.0
C              ENDIF
C           ENDIF
           DO 370 I = 1, NUMNOD
             IF (ICH(I) .eq. 0) THEN
               TEMP4 = C(I-LCH(I),OLD) + CMEAN(I-LCH(I),IDT,L)
             ELSE
               TEMP4 = X(I)
             ENDIF
             MASS = MASS + TEMP4
             TEMP1 = TEMP1 + X1(I) * TEMP4
             TEMP(1,1) = TEMP(1,1) + TEMP4 * X1(I) * X1(I)
C              IF (DIM .GE. 2) THEN
               TEMP2 = TEMP2 + X2(I) * TEMP4
               TEMP(2,1) = TEMP(2,1) + TEMP4 * X2(I) * X1(I)
               TEMP(2,2) = TEMP(2,2) + TEMP4 * X2(I) * X2(I)
C                IF (DIM .EQ. 3) THEN
C                  TEMP3 = TEMP3 + X3(I) * TEMP4
C                  TEMP(3,1) = TEMP(3,1) + TEMP4 * X3(I) * X1(I)
C                  TEMP(3,2) = TEMP(3,2) + TEMP4 * X3(I) * X2(I)
C                  TEMP(3,3) = TEMP(3,3) + TEMP4 * X3(I) * X3(I)
C                ENDIF
C              ENDIF
 370       CONTINUE
           WRITE (OUTF,380) MASS * ABS(X1(IN(1,3)) - X1(IN(1,1)))
     1                        * ABS(X2(IN(1,3)) - X2(IN(1,1))),
     2                    TEMP1 / MASS, TEMP2 / MASS
 380       FORMAT(/10X, 'TOTAL MASS',
     1             14X, 'X CENTER OF MASS',
     2             14X, 'Y CENTER OF MASS'
     3            /9X,F12.3,13X,F12.3,18X,F12.3)
           TEMP(1,1) = TEMP(1,1) / MASS - (TEMP1 / MASS) ** 2
           TEMP(2,1) = TEMP(2,1) / MASS - TEMP1 * TEMP2 / MASS ** 2
           TEMP(2,2) = TEMP(2,2) / MASS - (TEMP2 / MASS) ** 2
           WRITE(OUTF,390) 'RADII OF GYRATION OF AVERAGE PLUME',
     1                     SQRT(TEMP(1,1)), SQRT(ABS(TEMP(2,1))),
     2                     SQRT(TEMP(2,2))
           WRITE(OUTF,390) 'RADII OF GYRATION OF CENTER OF MASS',
     1                     SQRT(ABS(SIGMA(1,1,IDT,L) / MASS)),
     2                     SQRT(ABS(SIGMA(2,1,IDT,L) / MASS)),
     3                     SQRT(ABS(SIGMA(2,2,IDT,L) / MASS))
           WRITE(OUTF,390) '     AVERAGE RADII OF GYRATION',
     1                 SQRT(TEMP(1,1) + SIGMA(1,1,IDT,L) / MASS),
     2               SQRT(ABS(TEMP(2,1) + SIGMA(2,1,IDT,L) / MASS)),
     3                   SQRT(TEMP(2,2) + SIGMA(2,2,IDT,L) / MASS)
```

```
 390         FORMAT(/28X,A,/7X,'XX',F12.3,11X,'XY',F12.3,16X,'YY',F12.3)
           ENDIF
 400   CONTINUE
       RETURN
       END


       SUBROUTINE RHS0(C,OLD,NEW)
C*************************************************************
C
C 18.1   PURPOSE:
C          SUBROUTINE RHS0 ASSEMBLES THE RIGHT-HAND-SIDE VECTOR
C          FOR DETERMINING THE CONCENTRATIONS
C
C 18.2   FILE INPUT AND OUTPUT:
C          NONE
C
C        VARIABLE INPUT AND OUTPUT
C
C
C 18.4   DEFINITIONS OF VARIABLES:
C          DELTAT(I) = SIZE OF TIME STEP I
C            FLUX(I) = SPECIFIED VALUE OF GROUNDWATER FLOW OR
C                      SOLUTE FLUX AT NODE I
C              GT(I) = VALUE OF TIME FUNCTION AT TIME (I)
C             ICH(I) = 1 IF THE VALUE OF THE FIELD VARIABLE IS
C                      SPECIFIED AT NODE I
C                    = 0 OTHERWISE
C             B1(IJ) = MODIFIED GLOBAL MATRIX IN VECTOR STORAGE
C               NDOF = NUMBER OF NODES WHERE THE VALUE OF THE
C                      FIELD VARIABLE IS UNKNOWN
C             NUMNOD = NUMBER OF NODES
C              OMEGA = RELAXATION FACTOR
C            OMOMEGA = 1 - OMEGA
C
C 18.5   USAGE
C          FOR EACH TIME STEP, THE RIGHT-HAND-SIDE VECTOR IS
C          COMPUTED USING THE VALUES OF HEAD OR SOLUTE
C          CONCENTRATION FROM THE PREVIOUS TIME STEP, AND THE
C          MODIFIED COMBINED CONDUCTION AND CAPACITANCE MATRIX,
C          RELAXATION FACTOR, AND TIME STEP INTERVAL FOR THAT
C          TIME STEP
C
C        SUBROUTINES CALLED
C          LOCATE
C
C*************************************************************
       INCLUDE 'COMALL'
       INTEGER I, J, OLD, NEW
       DOUBLE PRECISION C(MAX1,2)
       IF (T .GT. TIME(IGT)) IGT = IGT + 1
       T = T + DELTAT(IDT)
       IF (T .GT. TIME(IGTDT)) IGTDT = IGTDT + 1
       I = 0
       DO 10 J = 1, NUMNOD
         IF (ICH(J) .EQ. 0) THEN
           I = I + 1
           C(I,NEW) = FC(I) + DELTAT(IDT) * (OMOMEGA * GT(IGT) * FLUX(J)
     1           + OMEGA * GT(IGTDT) * FLUX(J))
         ENDIF
 10    CONTINUE
       CALL DGBMV('NotTransposed',NDOF,NDOF,SBW,SBW,1D0,B2,2*MAX6+1,
     1            C(1,OLD),1,1D0,C(1,NEW),1)
       RETURN
       END

       SUBROUTINE RHS1(C,DDE,E,K,PNT,OLD,NEW)
C*************************************************************
```

```
C
C 18.1  PURPOSE:
C         SUBROUTINE RHS1 ASSEMBLES THE RIGHT-HAND-SIDE VECTOR
C         FOR DETERMINING THE FIRST DERIVATIVE OF
C         CONCENTRATION WITH RESPECT TO VELOCITY
C
C 18.2  FILE INPUT AND OUTPUT:
C         NONE
C
C       VARIABLE INPUT AND OUTPUT
C
C
C 18.4  DEFINITIONS OF VARIABLES:
C         DELTAT(I) = SIZE OF TIME STEP I
C           FLUX(I) = SPECIFIED VALUE OF GROUNDWATER FLOW OR
C                     SOLUTE FLUX AT NODE I
C             GT(I) = VALUE OF TIME FUNCTION AT TIME (I)
C            ICH(I) = 1 IF THE VALUE OF THE FIELD VARIABLE IS
C                     SPECIFIED AT NODE I
C                    = 0 OTHERWISE
C            B1(IJ) = MODIFIED GLOBAL MATRIX IN VECTOR STORAGE
C              NDOF = NUMBER OF NODES WHERE THE VALUE OF THE
C                     FIELD VARIABLE IS UNKNOWN
C            NUMNOD = NUMBER OF NODES
C             OMEGA = RELAXATION FACTOR
C           OMOMEGA = 1 - OMEGA
C
C 18.5  USAGE
C         SIMILAR TO DHDK, DCDV IS DETERMINED USING:
C
C               M * DCDV = DFDV - dt DDDV * C
C
C         NOW
C           F = (A - (1-w)dt * D) Ct-1 + dt((1-w)Ft-1 + wFt-1)
C
C         AND ONLY C AND D ARE FUNCTIONS OF V
C         THEREFORE
C
C           DFDV = (A - (1-w)dt * D) * DCDVt-1
C                      + (1-w)dt * DDDV * Ct-1
C
C         AND THE RHS IS
C                  (A - (1-w)dt * D) * DCDVt-1      # LINE 1
C                      + (1-w)dt * DDDV * Ct-1      # LINE 2
C                                -dt DDDV * C       # LINE 3
C
C       SUBROUTINES CALLED
C         LOCATE
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I, II, J, KI, KJ, K, NODETBL(13)
      INTEGER PNT, OLD, NEW
      DOUBLE PRECISION C(MAX1,2), DDE(MAX3,MAX3,3,0:3)
      DATA NODETBL/2,3,4,3,4,4,8,12,8,20,32,3,4/
C     DETERMINE (A - (1-w)dt * D) * DCDVt-1          # LINE 1
      CALL DGBMV('NotTransposed',NDOF,NDOF,SBW,SBW,1D0,B2,2*MAX6+1,
     1           DCDV(1,PNT,K,OLD),1,0D0,DCDV(1,PNT,K,NEW),1)
C     DETERMINE (1-w)dt * DDDV * Ct-1                # LINE 2
      DO 50 I = 1, NODETBL(ELEMTYP(E))
        KI = IN (E,I)
        IF (ICH(KI) .EQ. 0) THEN
          II = KI - LCH(KI)
          DO 40 J = 1, NODETBL(ELEMTYP(E))
            KJ = IN(E,J)
            IF (ICH(KJ) .NE. 0) THEN
C             RHS 'BC' TERM - X(KJ) IS A FIXED BOUNDARY CONDITION
C             WILL BE DONE BELOW!
```

```
                DCDV(II,PNT,K,NEW) = DCDV(II,PNT,K,NEW)
     1                             - DELTAT(IDT) * DDE(I,J,K,0) * X(KJ)
              ELSE
C               RHS 'DMDV*C' TERM - ONLY 'M' DIFERENTIATED NEW (MEAN) CONC.
                DCDV(II,PNT,K,NEW) = DCDV(II,PNT,K,NEW) - OMOMEGA
     1            * DELTAT(IDT)  * DDE(I,J,K,0) * C(KJ - LCH(KJ),OLD)
                DCDV(II,PNT,K,NEW) = DCDV(II,PNT,K,NEW) - OMEGA
     1               * DELTAT(IDT) * DDE(I,J,K,0) * C(KJ - LCH(KJ),NEW)
              ENDIF
 40         CONTINUE
          ENDIF
 50     CONTINUE
C       UPGRADE C FROM C(t-1) TO C(t)
C         THIS IS DONE BY CHANGING FROM 'OLD' TO 'NEW'
C       DETERMINE   - DDDV * C                              # LINE 3
        RETURN
        END

        SUBROUTINE RHS2DIFF(DDE,DDE2,E,E2,OLD,NEW)
C*************************************************************
C
C 18.1  PURPOSE:
C         SUBROUTINE RHS2 ASSEMBLES THE RIGHT-HAND-SIDE VECTOR
C         FOR DETERMINING THE SECOND DERIVATIVE OF
C         CONCENTRATION WITH RESPECT TO VELOCITY
C
C 18.2  FILE INPUT AND OUTPUT:
C         NONE
C
C       VARIABLE INPUT AND OUTPUT
C
C
C 18.4  DEFINITIONS OF VARIABLES:
C         DELTAT(I) = SIZE OF TIME STEP I
C           FLUX(I) = SPECIFIED VALUE OF GROUNDWATER FLOW OR
C                     SOLUTE FLUX AT NODE I
C             GT(I) = VALUE OF TIME FUNCTION AT TIME (I)
C            ICH(I) = 1 IF THE VALUE OF THE FIELD VARIABLE IS
C                     SPECIFIED AT NODE I
C                    = 0 OTHERWISE
C            B1(IJ) = MODIFIED GLOBAL MATRIX IN VECTOR STORAGE
C              NDOF = NUMBER OF NODES WHERE THE VALUE OF THE
C                     FIELD VARIABLE IS UNKNOWN
C            NUMNOD = NUMBER OF NODES
C             OMEGA = RELAXATION FACTOR
C           OMOMEGA = 1 - OMEGA
C
C 18.5  USAGE
C         SIMILAR TO D2HDK2, D2CDV2 IS DETERMINED USING:
C               M * D2CDV2(I,J) = D2FDV2(I,J)
C                                 - DDDV(I) * DCDV(J)
C                                 - DDDV(J) * DCDV(I)
C                                 - D2DDV2(I,J) * C
C
C         AS ABOVE
C
C           DF2DV2(I,J) = (A - (1-w)dt * D) D2CDV2(I,J)t-1
C                       + (1-w)dt * DDDV(I) * DCDV(J)t-1
C                       + (1-w)dt * DDDV(J) * DCDV(I)t-1
C                       + (1-w)dt * D2DDV2(I,J) * Ct-1
C
C         AND THE RHS IS
C           (A - (1-w)dt * D) D2CDV2(I,J)t-1      # LINE 1
C           + (1-w)dt * DDDV(I) * DCDV(J)t-1      # LINE 2
C           + (1-w)dt * DDDV(J) * DCDV(I)t-1      # LINE 3
C             + (1-w)dt * D2DDV2(I,J) * Ct-1      # LINE 4
C                       - DDDV(I) * DCDV(J)       # LINE 5
C                       - DDDV(J) * DCDV(I)       # LINE 6
```

```
C                       - D2DDV2(I,J) * C       # LINE 7
C
C       SUBROUTINES CALLED
C         LOCATE
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E, E2, I, II, J, KI, KJ
      INTEGER K, K2, OLD, NEW
      DOUBLE PRECISION DDE(MAX3,MAX3,3,0:3),R(MAX0,3,3)
      DOUBLE PRECISION DDE2(MAX3,MAX3,3,0:0)
      INTEGER NODETBL(13)
      DATA NODETBL/2,3,4,3,4,4,8,12,8,20,32,3,4/
C     DETERMINE (A - (1-w)dt * D) D2CDV2(I,J)t-1     # LINE 1
      DO 10 K = 1, DIM
        DO 10 K2 = 1, DIM
          CALL DGBMV('NotTransposed',NDOF,NDOF,SBW,SBW,1D0,B2,
     1             2*MAX6+1,D2CDV2(1,K,K2),1,0D0,R(1,K,K2),1)
 10   CONTINUE
C$DOACROSS NEST (K,K2), LOCAL (I, J, KI, II, KJ,K ,K2)
      DO 80 K = 1, DIM
        DO 80 K2 = 1, DIM
C         DETERMINE (1-w)dt * DDDV(I) * DCDV(J)t-1       # LINE 2
C         DETERMINE              - DDDV(I) * DCDV(J)     # LINE 5
          DO 50 I = 1, NODETBL(ELEMTYP(E))
            KI = IN (E,I)
            IF (ICH(KI) .EQ. 0) THEN
              II = KI - LCH(KI)
              DO 40 J = 1, NODETBL(ELEMTYP(E))
                 KJ = IN(E,J)
                IF (ICH(KJ) .EQ. 0) THEN
C                 RHS DERIVATIV OF F TERM AND
C                 RHS 'DDDV*DCDV' TERMS.
                  KJ = KJ - LCH(KJ)
                  R(II,K,K2) = R(II,K,K2) - DELTAT(IDT) * DDE (I,J,K,0)
     1                                 * ( OMOMEGA * DCDV(KJ,2,K2,OLD)
     2                                   + OMEGA * DCDV(KJ,2,K2,NEW))
                ENDIF
 40           CONTINUE
            ENDIF
 50       CONTINUE
C         DETERMINE (1-w)dt * DDDV(J) * DCDV(I)t-1       # LINE 3
C         DETERMINE              - DDDV(J) * DCDV(I)     # LINE 6
          DO 70 I = 1, NODETBL(ELEMTYP(E2))
            KI = IN (E2,I)
            IF (ICH(KI) .EQ. 0) THEN
              II = KI - LCH(KI)
              DO 60 J = 1, NODETBL(ELEMTYP(E2))
                KJ = IN(E2,J)
                IF (ICH(KJ) .EQ. 0) THEN
C                 RHS DERIVATIV OF F TERM AND
C                 RHS 'DDDV*DCDV' TERMS.
                  KJ = KJ - LCH(KJ)
                  R(II,K,K2) = R(II,K,K2) - DELTAT(IDT) * DDE2(I,J,K2,0)
     1                                 * (OMOMEGA * DCDV(KJ,1,K,OLD)
     2                                    + OMEGA * DCDV(KJ,1,K,NEW))
                ENDIF
 60           CONTINUE
            ENDIF
 70       CONTINUE
C         DETERMINE (1-w)dt * D2DDV2(I,J) * Ct-1         # LINE 4
C         DETERMINE D2DDV2(I,J) * C                      # LINE 7
C         THIS DOES NOT NEED TO BE DONE SINCE D2DDV2 = [0]
          DO 80 I = 1, NDOF
            D2CDV2(I,K,K2) = R(I,K,K2)
 80   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RHS2SAME(C,DDE,E,OLD,NEW)
C*************************************************************
C
C 18.1  PURPOSE:
C         SUBROUTINE RHS2 ASSEMBLES THE RIGHT-HAND-SIDE VECTOR
C         FOR DETERMINING THE SECOND DERIVATIVE OF
C         CONCENTRATION WITH RESPECT TO VELOCITY
C
C 18.2  FILE INPUT AND OUTPUT:
C         NONE
C
C       VARIABLE INPUT AND OUTPUT
C
C
C 18.4  DEFINITIONS OF VARIABLES:
C         DELTAT(I) = SIZE OF TIME STEP I
C           FLUX(I) = SPECIFIED VALUE OF GROUNDWATER FLOW OR
C                     SOLUTE FLUX AT NODE I
C            GT(I) = VALUE OF TIME FUNCTION AT TIME (I)
C           ICH(I) = 1 IF THE VALUE OF THE FIELD VARIABLE IS
C                     SPECIFIED AT NODE I
C                  = 0 OTHERWISE
C           B1(IJ) = MODIFIED GLOBAL MATRIX IN VECTOR STORAGE
C             NDOF = NUMBER OF NODES WHERE THE VALUE OF THE
C                     FIELD VARIABLE IS UNKNOWN
C           NUMNOD = NUMBER OF NODES
C            OMEGA = RELAXATION FACTOR
C          OMOMEGA = 1 - OMEGA
C
C 18.5  USAGE
C         SIMILAR TO D2HDK2, D2CDV2 IS DETERMINED USING:
C               M * D2CDV2(I,J) = D2FDV2(I,J)
C                                 - DDDV(I) * DCDV(J)
C                                 - DDDV(J) * DCDV(I)
C                                 - D2DDV2(I,J) * C
C
C         AS ABOVE
C
C           DF2DV2(I,J) = (A - (1-w)dt * D) D2CDV2(I,J)t-1
C                       + (1-w)dt * DDDV(I) * DCDV(J)t-1
C                       + (1-w)dt * DDDV(J) * DCDV(I)t-1
C                       + (1-w)dt * D2DDV2(I,J) * Ct-1
C
C         AND THE RHS IS
C           (A - (1-w)dt * D) D2CDV2(I,J)t-1     # LINE 1
C           + (1-w)dt * DDDV(I) * DCDV(J)t-1     # LINE 2
C           + (1-w)dt * DDDV(J) * DCDV(I)t-1     # LINE 3
C             + (1-w)dt * D2DDV2(I,J) * Ct-1     # LINE 4
C                       - DDDV(I) * DCDV(J)      # LINE 5
C                       - DDDV(J) * DCDV(I)      # LINE 6
C                       - D2DDV2(I,J) * C        # LINE 7
C
C       SUBROUTINES CALLED
C         LOCATE
C
C*************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I, II, J, KI, KJ, K, K2, OLD, NEW
      DOUBLE PRECISION C(MAX1,2), DDE(MAX3,MAX3,3,0:3)
      DOUBLE PRECISION R(MAX0,3,3)
      INTEGER NODETBL(13)
      DATA NODETBL/2,3,4,3,4,4,8,12,8,20,32,3,4/
C     DETERMINE (A - (1-w)dt * D) D2CDV2(I,J)t-1     # LINE 1
      DO 10 K = 1, DIM
        DO 10 K2 = 1, DIM
          CALL DGBMV('NotTransposed',NDOF,NDOF,SBW,SBW,1D0,B2,
     1               2*MAX6+1,D2CDV2(1,K,K2),1,0D0,R(1,K,K2),1)
```

```
 10   CONTINUE
C$DOACROSS NEST (K,K2), LOCAL (I, J, KI, II, KJ,K ,K2)
      DO 60 K = 1, DIM
        DO 60 K2 = 1, DIM
          DO 50 I = 1, NODETBL(ELEMTYP(E))
            KI = IN (E,I)
            IF (ICH(KI) .EQ. 0) THEN
              II = KI - LCH(KI)
              DO 40 J = 1, NODETBL(ELEMTYP(E))
                KJ = IN(E,J)
                IF (ICH(KJ) .NE. 0) THEN
C                 RHS 'BC' TERM - X(KJ) IS A FIXED BOUNDARY CONDITION
                  R(II,K,K2) = R(II,K,K2) - DELTAT(IDT)
     1                         * DDE(I,J,K,K2) * X(KJ)
                ELSE
C                 RHS DERIVATIV OF F TERM.
                  KJ = KJ - LCH(KJ)
C     DETERMINE (1-w)dt * DDDV(I) * DCDV(J)t-1        # LINE 2
C     DETERMINE (1-w)dt * DDDV(J) * DCDV(I)t-1        # LINE 3
C     DETERMINE (1-w)dt * D2DDV2(I,J) * Ct-1          # LINE 4
                  R(II,K,K2) = R(II,K,K2) - OMOMEGA * DELTAT(IDT)
     1                       * (  DDE(I,J,K ,0) * DCDV(KJ,2,K2,OLD)
     2                          + DDE(I,J,K2,0) * DCDV(KJ,1,K ,OLD)
     3                          + DDE(I,J,K,K2) * C(KJ,OLD))
C     DETERMINE                  - DDDV(I) * DCDV(J)      # LINE 5
C     DETERMINE                  - DDDV(J) * DCDV(I)      # LINE 6
C     DETERMINE                     D2DDV2(I,J) * C       # LINE 7
                  R(II,K,K2) = R(II,K,K2) - OMEGA * DELTAT(IDT)
     1                       * (  DDE(I,J,K, 0) * DCDV(KJ,2,K2,NEW)
     2                          + DDE(I,J,K2,0) * DCDV(KJ,1,K ,NEW)
     3                          + DDE(I,J,K,K2) * C(KJ,NEW))
                ENDIF
 40           CONTINUE
            ENDIF
 50       CONTINUE
          DO 60 I = 1, NDOF
            D2CDV2(I,K,K2) = R(I,K,K2)
 60   CONTINUE
      RETURN
      END

      SUBROUTINE CSAVE(E,E2)
C*********************************************************************
C
C       PURPOSE:
C         SUBROUTINE CSAVE SAVES THE CURRENT VALUES OF THE
C         VARIABLES INTO A FILE CALLED 'AUTOSAVE' SO THAT THE
C         PROGRAM CAN BE SHUT DOWN AND RESTARTED WHERE IT LEFT
C         OFF.
C
C*********************************************************************
      INCLUDE 'COMALL'
      INTEGER E, E2, J, K, K2, L, ISTEP
      OPEN (UNIT = PROBF, FILE = PROBFILE,
     1      FORM = 'FORMATTED', STATUS = 'UNKNOWN',
     2      ACCESS = 'SEQUENTIAL')
      WRITE (PROBF,*) 'ABOUT TO SAVE CONCENTRATION DATA'
      OPEN (UNIT = HCF, FORM = 'FORMATTED',
     1      FILE = HCFILE,
     2      STATUS = 'UNKNOWN', ACCESS = 'SEQUENTIAL')
      WRITE (HCF,*) E, E2
      DO 30 L = 1, NUMMAT
        IDT = 0
        DO 30 ISTEP = 1, MXSTEP
          IF (ISTEP .EQ. 1 .OR. ISTEP .GT. DTSTEP(IDT)) THEN
            IDT = IDT + 1
            DO 10 J = 1, NDOF
              WRITE (HCF,*) CMEAN(J,IDT,L), CVAR(J,IDT,L)
```

```
 10         CONTINUE
            DO 20 K = 1, DIM
              DO 20 K2 = 1, K
                WRITE(HCF,*) SIGMA(K,K2,IDT,L)
 20         CONTINUE
          ENDIF
 30   CONTINUE
      WRITE (HCF,*) 'CONCENTRATION FILE ENDS'
      CLOSE (UNIT = HCF, STATUS = 'KEEP')
      OPEN (UNIT = DONEF, FILE = DONEFILE, STATUS = 'UNKNOWN')
      WRITE (DONEF,*) 4,'   Concentrations in progress'
      CLOSE (UNIT = DONEF, STATUS = 'KEEP')
      PRINT *,'DATA SAVED AT E, E2 = ',E, E2
      PRINT *,NOR2,'SECOND DERIVATIVES COMPLETED'
      CLOSE (UNIT = PROBF, STATUS = 'DELETE')
      END

      SUBROUTINE CLOAD(E,E2)
C*********************************************************************
C
C       PURPOSE:
C         SUBROUTINE CLOAD LOADS THE CURRENT VALUES OF THE
C         VARIABLES FROM A FILE CALLED 'AUTOSAVE' SO THAT THE
C         PROGRAM CAN RESTART WHERE IT LEFT OFF.
C
C*********************************************************************
      INCLUDE 'COMALL'
      INTEGER E, E2, J, K, K2, L, ISTEP
      OPEN (UNIT = HCF, FORM = 'FORMATTED',
     1      FILE = HCFILE,
     2      STATUS = 'UNKNOWN', ACCESS = 'SEQUENTIAL')
      READ (HCF,*) E, E2
      DO 30 L = 1, NUMMAT
        IDT = 0
        DO 30 ISTEP = 1, MXSTEP
          IF (ISTEP .EQ. 1 .OR. ISTEP .GT. DTSTEP(IDT)) THEN
            IDT = IDT + 1
            DO 10 J = 1, NDOF
              READ (HCF,*) CMEAN(J,IDT,L), CVAR(J,IDT,L)
 10         CONTINUE
            DO 20 K = 1, DIM
              DO 20 K2 = 1, K
                READ (HCF,*) SIGMA(K,K2,IDT,L)
 20         CONTINUE
          ENDIF
 30   CONTINUE
      WRITE (HCF,*) 'CONCENTRATION FILE ENDS'
      CLOSE (UNIT = HCF, STATUS = 'KEEP')
      PRINT *,'DATA RESTORED AT E, E2 = ',E, E2
      RETURN
      END

SUBROUTINE CREATE (PRIOR)
C*********************************************************************
C
C  8.1  PURPOSE:
C         TO CREATE NODE NUMBERS AND COORDINATES
C         TO CREATE ELEMENT NUMBERS, TYPES AND NODE NUMBERS
C         TO CREATE ELEMENT MATERIAL SET NUMBERS AND MATERIAL
C                           PROPERTIES FOR EACH MATERIAL SET
C         TO CREATE DIRICHLET BOUNDARY CONDITIONS
C
C  8.2  INPUT:
C         NODE NUMBERS AND COORDINATES ARE CREATED FROM DATA
C         SUPPLIED FROM THE USER SUPPLIED FILE ASSIGNED TO UNIT
C         "INF"
C
C  8.3  OUTPUT:
```

```
C         NODE NUMBERS AND COORDINATES; ELEMENT NUMBERS, TYPES,
C         AND NODE NUMBERS; ELEMENT MATERIAL SET NUMBERS AND
C         MATERIAL PROPERTIES FOR EACH MATERIAL SET; AND
C         SPECIFIED VALUES  OF THE FIELD VARIABLE AND SPECIFIED
C         VALUES OF GROUNDWATER FLOW ARE WRITTEN TO THE USER-
C         DEFINED FILE ASSIGNED TO UNIT "OUTF"
C
C  8.4  DEFINITIONS OF VARIABLES:
C                 DIM = COORDINATE SYSTEM TYPE
C           ELEMTYP(I) = ELEMENT TYPE FOR ELEMENT I
C              FLUX(I) = SPECIFIED VALUE OF GROUNDWATER FLOW OR
C                        SOLUTE FLUX AT NODE (I)
C               ICH(I) = 1 IF THE VALUE OF THE FIELD VARIABLE IS
C                        SPECIFIED FOR NODE I
C                      = 0 OTHERWISE
C                 IDIM = DIM IF DIM = 1, 2 OR 3 OTHERWISE IDIM = 2
C              IN(I,J) = NODE NUMBER J FOR ELEMENT I
C               LABEL1 = "HYDRAULIC HEAD" OR "SOLUTE CONCENTRATION"
C               LABEL2 = "GROUNDWATER FLOW" OR "SOLUTE FLUX"
C               LCH(I) = ICH(I) + ICH(I-1) + ICH(I-2) + ...
C                        THE ARRAYSICH AND LCH ARE USED TO MODIFY
C                        GLOBAL SYSTEM OF EQUATIONS IN SUBROUTINES
C                        ASMBK, ASMBKC, AND ASMBAD
C            LENGTH(I) = LENGTH OF FLOW DOMAIN IN DIRECTION I
C            MATSET(I) = MATERIAL SET NUMBER FOR ELEMENT(I)
C                  NDN = NUMBER OF NODES WITH SPECIFIED VALUES OF THE
C                        FIELD VARIABLE (DIRICHLET NODES)
C                 NDOF = NUMBER OF DEGREES OF FREEDOM (UNSPECIFIED NODES)
C              NELM(I) = NUMBER OF ELEMENTS IN I DIRECTION
C                  NNN = NUMBER OF NODES WITH SPECIFIED VALUES OF THE
C                        GROUNDWATER FLOW OR SOLUTE FLUX (NEUMANN NODES)
C           NODETBL(I) = NUMBER OF NODES
C               NUMELM = NUMBER OF ELEMENTS IN MESH
C               NUMMAT = NUMBER OF MATERIAL SETS
C               NUMNOD = NUMBER OF NODES READ
C              NUMPROP = NUMBER OF MATERIAL PROPERTIES IN EACH
C                        MATERIAL SET
C                PLANE = NUMBER OF NODES IN A PLANE PERPENDICULAR
C                        TO X DIRECTION
C            PROP(I,J) = MATERIAL PROPERTY J FOR MATERIAL SET I
C                 X(I) = SPECIFIED VALUE OF THE FIELD VARIABLE AT NODE I
C                X1(I) = X COORDINATE FOR NODE I IF DIM = 1, 2 OR 3
C                      = R COORDINATE FOR NODE I IF DIM = 4
C                X2(I) = IS NOT USED IF DIM = 1
C                      = Y COORDINATE FOR NODE I IF DIM =  2 OR 3
C                      = Z COORDINATE FOR NODE I IF DIM = 4
C                X3(I) = IS NOT USED IF DIM = 1, 2 OR 3
C                      = Z COORDINATE FOR NODE I IF DIM = 3
C                 XELM = NUMBER OF ELEMENTS IN X DIRECTION
C                 YELM = NUMBER OF ELEMENTS IN Y DIRECTION
C                 ZELM = NUMBER OF ELEMENTS IN Z DIRECTION
C
C  8.5  USAGE:
C         THE DIMENSIONS OF THE MESH AND NUMBER OF ELEMENTS
C         ALONG EACH SIDE ARE READ IN.  FROM THIS THE
C         COORDINATES OF EACH NODE ARE CALCULATED,AND THE NODES
C         FOR EACH ELEMENT ARE DETERMINED.  ALL ELEMENTS ARE
C         ASSUMED TO BELONG TO ELEMENT MATERIAL SET (1).
C         BOTH ENDS OF THE FLOW FIELD ARE GIVEN FIXED HEADS
C         TO CREATE A UNIFORM (DETERMINISTIC) FLOW FIELD
C
C  9.6  ELEMENET TYPES:
C         TYPE  NODES  DIMENSION  DESCRIPTION
C          1      2        1        LINEAR BAR
C          2      3        1        QUADRATIC BAR
C          3      4        1        CUBIC BAR
C          4      3        2        LINEAR TRIANGLE
C          5      4        2        LINEAR RECTANGLE
```

```
C          6      4        2        LINEAR QUADRILATERAL
C          7      8        2        QUADRATIC QUADRILATERAL
C          8     12        2        CUBIC QUADRILATERAL
C          9      8        3        LINEAR PARALLELEPIPED
C         10     20        3        QUADRATIC PARALLELEPIPED
C         11     32        3        CUBIC PARALLELPIPED
C         12      3    AXI-SYMM     LINEAR TRIANGLE
C         13      4    AXI-SYMM     LINEAR RECTANGLE
c         14      8        3        LINEAR RECTANGULAR PRISM
C
C       SUBROUTINES CALLED:
C         NONE
C
C*********************************************************************
        INCLUDE 'COMALL'
        INTEGER IDIM, E, E2, I, J, K, L, PLANE, NODETBL(14), SETNUM
        INTEGER KI, II, JJ, KJ
        DATA NODETBL/2,3,4,3,4,4,8,12,8,20,32,3,4,8/
        DOUBLE PRECISION LENGTH(3), HEAD, NEWSET(MAX4)
        LOGICAL PRIOR
        CHARACTER*80 TITLE
        READ (INF,'(A)') TITLE
        READ (INF,*) DIM
        IDIM = DIM
        IF (DIM .EQ. 4) IDIM = 2
        READ (INF,*) (NELM(I), I = 1, IDIM)
        READ (INF,*) (LENGTH(I), I = 1, IDIM)
        NUMNOD = (NELM(1) + 1) * (NELM(2) + 1) * (NELM(3) + 1)
        IF (NUMNOD .GT. MAX1) THEN
          PRINT *,'TOO MANY NODES FOR THIS PROGRAM.'
          PRINT *,'AVAILABLE NODES = ',MAX1
          PRINT *,'REQUESTED NODES = ',NUMNOD
          PRINT *,'RECOMPILE WITH A LARGER MAXIMUM'
        ENDIF
        NUMELM = NELM(1) * NELM(2)
        IF (IDIM .EQ. 3) NUMELM = NUMELM * NELM(3)
        IF (NUMELM .GT. MAX2) THEN
          PRINT *,'TOO MANY ELEMENTS FOR THIS PROGRAM.'
          PRINT *,'AVAILABLE ELEMENTS = ',MAX2
          PRINT *,'REQUESTED ELEMENTS = ',NUMELM
          PRINT *,'RECOMPILE WITH A LARGER MAXIMUM'
        ENDIF
        I = 0
        DO 30, J = 0, NELM(1)
          DO 20, K = 0, NELM(2)
            IF (IDIM .EQ. 2) THEN
              I = I + 1
              X1(I) = LENGTH(1) * J / NELM(1)
              X2(I) = LENGTH(2) * K / NELM(2)
            ELSEIF (IDIM .EQ. 3) THEN
              DO 10 L = 0, NELM(3)
                I = I + 1
                X1(I) = LENGTH(1) * J / NELM(1)
                X2(I) = LENGTH(2) * K / NELM(2)
                X3(I) = LENGTH(3) * L / NELM(3)
 10           CONTINUE
            ENDIF
 20       CONTINUE
 30     CONTINUE
        IF (.NOT. PRIOR) THEN
          WRITE (OUTF,'(A)') TITLE
C         WRITE NODE NUMBERS AND NODE COORDINATES TO OUTPUT FILE
 40       IF (NUMNOD .GT. 0) THEN
            IF (DIM .EQ. 2) THEN
              WRITE (OUTF,60)
 60           FORMAT (2X, 'NODE',8X,'NODAL COORDINATES'/
     1                1X,'NUMBER',8X,'X',12X,'Y'/
     2                1X,6('-'),3X,11('-'),2X,11('-'))
```

```
            ELSEIF (DIM .EQ. 3) THEN
             WRITE (OUTF,70)
 70           FORMAT (2X,'NODE',14X,'NODAL COORDINATES'/
     1                1X,'NUMBER',8X,'X',12X,'Y',12X,'Z'/
     2                1X,6('-'),3X,11('-'),2X,11('-'),2X,11('-'))
            ELSEIF (DIM .EQ. 4) THEN
              WRITE (OUTF,80)
 80           FORMAT (2X,'NODE',8X,'NODAL COORDINATES'/
     1                1X,'NUMBER',8X,'R',12X,'Z'/
     2                1X,6('-'),3X,11('-'),2X,11('-'))
            ENDIF
            IF (IDIM .EQ. 2) THEN
              DO 100 I = 1, NUMNOD
                WRITE(OUTF,90) I, X1(I), X2(I)
 90             FORMAT (I5,2X,3(F13.4))
100           CONTINUE
            ELSE
              DO 110 I = 1, NUMNOD
                WRITE(OUTF,90) I, X1(I), X2(I), X3(I)
110           CONTINUE
            ENDIF
          ELSE
            WRITE(OUTF,120) 'NODAL POINT'
120         FORMAT(' NO ',A,' DATA CREATED.')
          ENDIF
        ENDIF
C       CREATE ELEMENT TYPE AND ELEMENT NODE NUMBERS
        IF (DIM .EQ. 2) THEN
          IF (NELM(1) / 2  .LT. (NELM(1) + 1) / 2) THEN
            PRINT *,'THIS PROGRAM REQUIRES AN EVEN NUMBER'
            PRINT *,'OF ELEMENTS IN THE X DIRECTION'
          ENDIF
          IF (NELM(2) / 2  .LT. (NELM(2) + 1) / 2) THEN
            PRINT *,'THIS PROGRAM REQUIRES AN EVEN NUMBER'
            PRINT *,'OF ELEMENTS IN THE Y DIRECTION'
          ENDIF
          E = 0
          E2 = NUMELM
          I = 0
          DO 140 J = 1, NELM(1) / 2
            DO 130 K = 1, NELM(2) / 2
              I = I + 1
              E = E + 1
              IN(E,1) = I
              IN(E,2) = I + NELM(2) + 1
              IN(E,3) = I + NELM(2) + 2
              IN(E,4) = I + 1
              ELEMTYP(E) = 6
130         CONTINUE
            E2 = E2 - NELM(2) / 2
            DO 135 K = NELM(2) / 2 + 1, NELM(2)
              I = I + 1
              E2 = E2 + 1
              IN(E2,1) = I
              IN(E2,2) = I + NELM(2) + 1
              IN(E2,3) = I + NELM(2) + 2
              IN(E2,4) = I + 1
              ELEMTYP(E2) = 6
135         CONTINUE
            E2 = E2 - NELM(2) / 2
            I = I + 1
140       CONTINUE
          E = E + NELM(2) / 2 + 1
          E2 = E2 + 1
          DO 143 J = NELM(1) / 2 + 1, NELM(1)
            DO 141 K = 1, NELM(2) / 2
              I = I + 1
              E = E - 1
```

```
              IN(E,1) = I
              IN(E,2) = I + NELM(2) + 1
              IN(E,3) = I + NELM(2) + 2
              IN(E,4) = I + 1
              ELEMTYP(E) = 6
141           CONTINUE
              E = E + NELM(2)
              DO 142 K = NELM(2) / 2 + 1, NELM(2)
                I = I + 1
                E2 = E2 - 1
                IN(E2,1) = I
                IN(E2,2) = I + NELM(2) + 1
                IN(E2,3) = I + NELM(2) + 2
                IN(E2,4) = I + 1
                ELEMTYP(E2) = 6
142             CONTINUE
              I = I + 1
143         CONTINUE
          ELSEIF (DIM .EQ. 3) THEN
            E = 0
            I = 0
            PLANE = (NELM(2) + 1) * (NELM(3) + 1)
            DO 170 J = 1, NELM(1)
              DO 160 K = 1, NELM(2)
                DO 150 L = 1, NELM(3)
                  E = E + 1
                  I = I + 1
                  IN(E,1) = I
                  IN(E,2) = I + PLANE
                  IN(E,3) = I + PLANE + NELM(3) + 1
                  IN(E,4) = I + NELM(3) + 1
                  IN(E,5) = I + 1
                  IN(E,6) = I + PLANE + 1
                  IN(E,7) = I + PLANE + NELM(3) + 2
                  IN(E,8) = I + NELM(3) + 2
                  ELEMTYP(E) = 9
150             CONTINUE
                I = I + 1
160           CONTINUE
              I = I + NELM(3)
170         CONTINUE
          ENDIF
          DO 210 E = 1, NUMELM
            MATSET(E) = 1
210       CONTINUE
          IF (.NOT. PRIOR) THEN
            IF (NUMELM .GT. 0) THEN
              IF (DIM .EQ. 2) THEN
                WRITE (OUTF,180) ('      ',I=1,2)
              ELSE
                WRITE (OUTF,180) ('              ',I=1,2)
              ENDIF
180           FORMAT (/,1X,'ELEMENT  ELEMENT  MAT''L',
     1               /,1X,'  NO.     TYPE     SET ',A,'NODE NUMBERS',
     2               /,1X,'-------  -------  -----',A,'------------')
              DO 200 I =1, NUMELM
                WRITE (OUTF,190) I,ELEMTYP(I),MATSET(I),
     1                           (IN(I,J), J=1,NODETBL(ELEMTYP(I)))
190             FORMAT(I6,I8,I9,2X,8I5:4(/25X,8I6))
200           CONTINUE
            ELSE
              WRITE (OUTF,120) 'ELEMENT'
            ENDIF
          ENDIF
C         READ MATERIAL DATA FOR EACH MATERIAL SET
C         READ FROM INPUT FILE: THE NUMBER OF PROPERTIES IN EACH
C         MATERIAL SET
          READ (INF,*) NUMPROP
```

```
      IF (NUMPROP .GT. MAX5) PRINT*,'TOO MANY PROPERTIES PER SET'
      NUMMAT = 0
C     READ FROM THE INPUT FILE: ELEMENT NUMBER AND MATERIAL SET NUMBER
250   READ (INF,*) SETNUM,(NEWSET(J),J=1,NUMPROP)
      IF (SETNUM .EQ. -1) GOTO 280
      IF (SETNUM .GT. NUMMAT) NUMMAT = SETNUM
      DO 260 J = 1, NUMPROP
        PROP(SETNUM,J) = NEWSET(J)
260   CONTINUE
      GOTO 250
280   IF (NUMMAT .GT. MAX4) THEN
        PRINT *,'TOO MANY MATERIAL SETS FOR THIS PROGRAM.'
        PRINT *,'AVAILABLE MATERIAL SETS = ',MAX4
        PRINT *,'REQUESTED MATERIAL SETS = ',NUMMAT
        PRINT *,'RECOMPILE WITH A LARGER MAXIMUM'
      ENDIF
      IF (.NOT. PRIOR) THEN
C       WRITE MATERIAL PROPERTIES INFORMATION TO OUTPUT FILE
        IF (NUMPROP .EQ. 1) THEN
          WRITE(OUTF,240) ('    ',I=1,2)
        ELSEIF (NUMPROP .EQ. 2) THEN
          WRITE(OUTF,240) ('      ',I=1,2)
        ELSEIF (NUMPROP .EQ. 3) THEN
          WRITE(OUTF,240) ('            ',I=1,2)
        ELSEIF (NUMPROP .GE. 4) THEN
          WRITE(OUTF,240) ('                  ',I=1,2)
        ENDIF
240     FORMAT(/1X,'MATERIAL'/2X,'SET NO.',A,
     1          'MATERIAL PROPERTIES'/1X,'--------',A,
     2          '------------------')
        DO 245 SETNUM = 1, NUMMAT
          WRITE (OUTF,270) SETNUM, (PROP(SETNUM,J),J=1,NUMPROP)
270       FORMAT(I6,4X,8(1P4E13.6/10X))
245     CONTINUE
        IF (NUMMAT .EQ. 0) THEN
          WRITE (OUTF,120) 'ELEMENT MATERIAL PROPERTY'
        ENDIF
      ENDIF
C     INITIALISE BOUNDARY CONDITIONS
      DO 290 I = 1, NUMNOD
        ICH(I) = 0
        FLUX(I) = 0.0
290   CONTINUE
C     CREATE DIRICHLET NODES AT ENDS OF FLOW FIELD
      READ (INF,*) HEAD
      IF (DIM .EQ. 2) THEN
        NDN = NELM(2) * 2 + 2
        DO 320 I = 1, NELM(2) + 1
          X(I) = HEAD
          ICH(I) = 1
320     CONTINUE
        DO 330 I = NUMNOD - NELM(2), NUMNOD
          X(I) = 0.0
          ICH(I) = 1
330     CONTINUE
      ELSE
        NDN = 2 * PLANE + 2
        DO 340 I = 1, PLANE + 1
          X(I) = HEAD
          ICH(I) = 1
340     CONTINUE
        DO 350 I = NUMNOD - PLANE, NUMNOD
          X(I) = 0.0
          ICH(I) = 1
350     CONTINUE
      ENDIF
      NNN=0
      LCH(1) = ICH(1)
```

```
        DO 370 I = 2, NUMNOD
          LCH(I) = LCH(I-1) + ICH(I)
370       CONTINUE
        NDOF = NUMNOD - NDN
        IF (.NOT. PRIOR) THEN
          WRITE (OUTF,300) LABEL1
300       FORMAT(//3X,'NODE',15X,'SPECIFIED'/4X,'NO.',10X,A/
     1             2X,'------',9X,'--------------------')
          DO 380 I = 1, NUMNOD
            IF (ICH(I) .EQ. 1) THEN
              WRITE (OUTF,310) I,X(I)
310           FORMAT(I6,10X,F15.4)
            ENDIF
380       CONTINUE
          WRITE (OUTF, 360) LABEL1,NDN
360       FORMAT(//' NUMBER OF NODES WITH SPECIFIED ',/2x,A,'=',I11)
          WRITE (OUTF, 360) LABEL2,NNN
        ENDIF
        READ (INF,*) R1, R2
        NOR1 = 0
        NOR2 = 0
C       COMPUTE THE SEMI-BANDWIDTH
        SBW = 1
        DO 430 E = 1, NUMELM
          DO 420 I = 1, NODETBL(ELEMTYP(E))
            KI = IN(E,I)
            IF (ICH(KI) .EQ. 0 .AND. I .LT. NODETBL(ELEMTYP(E))) THEN
              II = KI - LCH(KI)
              DO 410 J = I + 1, NODETBL(ELEMTYP(E))
                KJ = IN(E,J)
                IF (ICH(KJ) .EQ. 0) THEN
                  JJ = ABS(KJ - LCH(KJ) - II) + 1
                  IF (JJ .GT. SBW) SBW = JJ
                ENDIF
410           CONTINUE
            ENDIF
420       CONTINUE
430     CONTINUE
        IF (SBW .GT. MAX6) STOP'** EXCEEDS MAXIMUM SEMIBANDWIDTH**'
440     FORMAT(//' NUMBER OF DEGREES OF FREEDOM',/,
     1           ' IN MODIFIED K MATRIX =', I5///
     2           ' SEMI-BANDWIDTH OF MODIFIED K MATRIX =',I5)
        IF (PRIOR) THEN
390       READ (OUTF,*,END=400)
          GOTO 390
        ELSE
          WRITE (OUTF,440) NDOF,SBW
400     ENDIF
        RETURN
        END



      SUBROUTINE CREAT2 (PRIOR)
C*********************************************************************
C
C 16.1  PURPOSE:
C         FOR THE CONCENTRATION PROBLEM:
C           TO CREATE MATERIAL PROPERTIES FOR EACH MATERIAL SET
C           TO CREATE DIRICHLET BOUNDARY CONDITIONS
C           TO CREATE THE INITIAL CONCNETRATION DISTRIBUTION
C           TO READ THE RELAXATION FACTOR, TIME STEP INTERVALS
C             AND VALUES OFTHE TIME FUNCTION.
C
C 16.2  INPUT:
C         ELEMENT MATERIAL PROPERTIES FOR EACH MATERIAL SET,
C         DIRICHLET BOUNDARY CONDITIONS AND INITIAL CONCENTRATION
C         DISTRIBUTION PARAMETERS.
```

```
C
C 16.3  OUTPUT
C         MATERIAL SET PROPERTIES
C         THE RELAXATION FACTOR, TIME STEP INTERVALS, VALUES OF
C         THE TIME FUNCTION, AND INITIAL VALUES OF THE FIELD
C         VARIABLE ARE WRITTEN TO THE USER DEFINED FILE ASSIGNED
C         TO UNIT "OUTF".
C
C 16.4  DEFINITIONS OF VARIABLES:
C         DELTAT(I) = SIZE OF TIME STEP I
C         DTSTEP(I) = NUMBER OF TIME STEPS TO TAKE USING A TIME
C                     STEP OF SIZE DELTAT(I)
C             GT(I) = VALUE OF TIME FUNCTION AT TIME I
C            ICH(I) = 1 IF THE VALUE OF THE FIELD VARIABLE IS
C                     SPECIFIED FOR NODE I,
C                    = 0 OTHERWISE
C            LABEL1 = CHARACTER VARIABLE USED TO LABEL COLUMN
C                     HEADINGS FOR SPECIFIED VALUES OF THE FIELD
C                     VARIABLE ON FILE ASSIGNED TO UNIT "OUTF".
C                     LABEL1 = "HYDRAULIC HEAD", "PRESSURE HEAD"
C                     OR "SOLUTE FLUX"
C            MXSTEP = NUMBER OF DIFFERENT TIME STEP INTERVALS
C            NUMNOD = NUMBER OF NODES
C             OMEGA = RELAXATION FACTOR
C           OMOMEGA = 1 - OMEGA
C           TIME(I) = STARTING TIME FOR TIME FUNCTION VALUE GT(I)
C            TOTALT = TOTAL LENGTH OF TIME FOR WHICH CALCULATIONS
C                     ARE PERFORMED
C              X(I) = VALUE OF THE FIELD VARIABLE (HYDRAULIC HEAD,
C                     PRESSURE HEAD, OR SOLUTE CONCENTRATION)
C                     AT NODE I
C
C 16.5  USAGE
C         THE RELAXATIN FACTOR OMEGA IS READ FIRST.  THIS
C         IS FOLLOWED BY A LIST OF TIME STEPS AND TIME STEP
C         INTEVALS.  EACH LINE OF INPUT CONTAINS THE NUMBER OF
C         TIME STEPS TO TAKE FOLLOWED BY A SPECIFIED TIME STEP
C         INTERVAL.  INPUT OF TIME STEPS AND TIME STEP INTERVALS
C         IS TERMINATED BY PLACING A -1 IN BOTH FIELDS.
C         THIS IS FOLLOWED BY A LIST OF TIMES AND VALUES OF THE
C         TIME FUNCTION FOR EACH TIME.  INPUT IS TERMINATED BY
C         PLACING A -1 IN BOTH FIELDS.  FINALLY, THE INITIAL
C         VALUE OF THE FIELD VARIABLE IS READ FOR EACH NODE.
C         INPUT IS TERMINATED BY PLACING A -1 IN BOTH FIELDS.
C
C       SUBROUTINES CALLED
C         NONE
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER IT, ISTART, I, J, SETNUM, E, II, KI, KJ, JJ
      INTEGER NODETBL(14)
      DOUBLE PRECISION TOTALT, HINIT, NORM, SIGMA1
      LOGICAL PRIOR
      DATA NODETBL/2,3,4,3,4,4,8,12,8,20,32,3,4,8/
C     READ FROM INPUT FILE: THE NUMBER OF PROPERTIES IN EACH
C     MATERIAL SET
      READ (INF,*) NUMPROP
      IF (NUMPROP .GT. MAX5) PRINT*,NUMPROP,'TOO MANY PROPERTIES'
      DO 10 I = 1, NUMMAT
        READ (INF,*) SETNUM,(PROP(SETNUM,J),J=1,NUMPROP)
 10   CONTINUE
C     WRITE MATERIAL PROPERTIES INFORMATION TO OUTPUT FILE
      IF (.NOT. PRIOR) THEN
        IF (NUMPROP .EQ. 1) THEN
          WRITE(OUTF,20) ('  ',I=1,2)
        ELSEIF (NUMPROP .EQ. 2) THEN
          WRITE(OUTF,20) ('    ',I=1,2)
```

```
        ELSEIF (NUMPROP .EQ. 3) THEN
          WRITE(OUTF,20) ('          ',I=1,2)
        ELSEIF (NUMPROP .GE. 4) THEN
          WRITE(OUTF,20) ('                        ',I=1,2)
        ENDIF
 20     FORMAT(/1X,'MATERIAL'/2X,'SET NO.',A,
     1         'MATERIAL PROPERTIES'/1X,'--------',A,
     2         '-------------------')
        DO 40 I = 1, NUMMAT
          WRITE (OUTF,30) SETNUM, (PROP(SETNUM,J),J=1,NUMPROP)
 30       FORMAT(I6,2X,8(1P3E13.6/8X))
 40     CONTINUE
      ENDIF
      DO 50 I = 1, NUMELM
        MATSET(I) = SETNUM
 50   CONTINUE
C     INITIALISE BOUNDARY CONDITIONS
      DO 60 I = 1, NUMNOD
        ICH(I) = 0
        FLUX(I) = 0.0
 60   CONTINUE
C     CREATE DIRICHLET NODES AT UPSTREAM END OF FLOW FIELD
      IF (DIM .EQ. 2) THEN
        NDN = NELM(2) + 1
        DO 70 I = 1, NELM(2) + 1
          X(I) = 0.0
          ICH(I) = 1
 70     CONTINUE
      ELSE
        NDN = (NELM(2) + 1) * (NELM(3) + 1)
        DO 80 I = 1, NDN
          X(I) = 0.0
          ICH(I) = 1
 80     CONTINUE
      ENDIF
      LCH(I) = ICH(I)
      DO 85 I = 2, NUMNOD
        LCH(I) = LCH(I-1) + ICH(I)
 85   CONTINUE
      NNN=0
      IF (.NOT. PRIOR) THEN
        WRITE (OUTF,90) LABEL1
 90     FORMAT(//3X,'NODE',15X,'SPECIFIED'/4X,'NO.',10X,A/
     1           2X,'------',9X,'--------------------')
        IF (DIM .EQ. 2) THEN
          DO 110 I = 1, NELM(2) + 1
            WRITE (OUTF,100) I,X(I)
100         FORMAT(I6,10X,F15.4)
110       CONTINUE
        ELSE
          DO 120 I = 1, NDN
            WRITE (OUTF,100) I,X(I)
120       CONTINUE
        ENDIF
        WRITE (OUTF, 130) LABEL1,NDN
130     FORMAT(//' NUMBER OF NODES WITH SPECIFIED ',/2x,A,'=',I11)
        WRITE (OUTF, 130) LABEL2,NNN
      ENDIF
      LCH(1) = ICH(1)
      DO 140 I = 2, NUMNOD
        LCH(I) = LCH(I-1) + ICH(I)
140     CONTINUE
      NDOF = NUMNOD - NDN
C     INPUT OMEGA FROM INPUT FILE
      READ(INF,*) OMEGA
      OMOMEGA = 1.0 - OMEGA
C     IMPUT LIST OF TIME STEPS AND TIME STEP INTERVALS FROM INPUT FILE
      IT = 1
```

```
      MXSTEP = 0
150   READ (INF,*) DTSTEP(IT), DELTAT(IT)
      IF (DTSTEP(IT) .LE. 0) GOTO 160
      IF (DTSTEP(IT) .GT. MXSTEP) MXSTEP = DTSTEP(IT)
      IT = IT + 1
      GOTO 150
160   IT = IT - 1
      ISTART = 1
      TOTALT = 0.0
      DO 170 I = 1, IT
        TOTALT = TOTALT + (DTSTEP(I) - ISTART + 1) * DELTAT(I)
        ISTART = DTSTEP(I) + 1
170   CONTINUE
      IF (.NOT. PRIOR) THEN
        WRITE(OUTF,210) OMEGA
210     FORMAT(//2X,'OMEGA = ',F15.4)
        WRITE(OUTF,220)
220     FORMAT(//2X,'START',8X,' END ',10X,'DELTA T'/
     1          2X,5('-'),8X,5('-'),8X,11('-'))
        ISTART = 1
        DO 240 I = 1, IT
          WRITE(OUTF,230) ISTART,DTSTEP(I),DELTAT(I)
230       FORMAT (2X,I4,9X,I4,3X,F15.4)
          ISTART = DTSTEP(I) + 1
240     CONTINUE
        WRITE(OUTF,250) TOTALT
250     FORMAT (/10X,'TOTAL TIME =',F15.4)
      ENDIF
C     INPUT LIST OF TIME STEPS AND VALUES OF THE TIME FUNCTION
      IT = 1
180   READ(INF,*) TIME(IT),GT(IT)
      IF (TIME(IT) .LT. 0.0) GOTO 190
      IT = IT + 1
      GOTO 180
190   IT = IT - 1
      IF (TIME(IT) .LT. TOTALT) TIME(IT) = TOTALT
      IF (.NOT. PRIOR) THEN
        WRITE(OUTF,260)
260     FORMAT(//8X,'TIME T',11X,'G(T)'/7X,8('-'),9X,6('-'))
        DO 280 I = 1, IT
          WRITE(OUTF,270) TIME(I),GT(I)
270       FORMAT(2F15.4)
280     CONTINUE
      ENDIF
C     INPUT INITIAL VALUES OF FIELD VARIABLE FROM INPUT FILE
      READ(INF,*) IT,HINIT,SIGMA1
      DO 200 I = 1, NUMNOD
        IF (ICH(I) .NE. 1) THEN
          NORM = ((X1(I) - X1(IT)) **2
     1          + (X2(I) - X2(IT)) **2
     2          + (X3(I) - X3(IT)) **2)
          X(I) = HINIT * EXP(-NORM / 2 / SIGMA1**2)
        ENDIF
200   CONTINUE
      IF (.NOT. PRIOR) THEN
        WRITE (OUTF, 290) LABEL1, LABEL1
290     FORMAT(//2X,'INITIAL VALUES OF ',A/2X,38('-')//
     1       2X,'NODE NO.',10X,A/2X,8('-'),10X,20('-'))
        DO 310 I= 1, NUMNOD
          IF (ICH(I) .EQ. 0) THEN
            WRITE(OUTF,300) I, X(I), ' '
          ELSE
            WRITE(OUTF,300) I, X(I), '*'
          ENDIF
300       FORMAT (2X,I5,12X,F15.4,A)
310     CONTINUE
        WRITE (OUTF,320)
320     FORMAT (/23X,'* = SPECIFIED')
```

```
      ENDIF
C     COMPUTE THE SEMI-BANDWIDTH
      SBW = 1
      DO 350 E = 1, NUMELM
        DO 340 I = 1, NODETBL(ELEMTYP(E))
          KI = IN(E,I)
          IF (ICH(KI) .EQ. 0 .AND. I .LT. NODETBL(ELEMTYP(E))) THEN
            II = KI - LCH(KI)
            DO 330 J = I + 1, NODETBL(ELEMTYP(E))
              KJ = IN(E,J)
              IF (ICH(KJ) .EQ. 0) THEN
                JJ = ABS(KJ - LCH(KJ) - II) + 1
                IF (JJ .GT. SBW) SBW = JJ
              ENDIF
330         CONTINUE
          ENDIF
340     CONTINUE
350   CONTINUE
      IF (.NOT. PRIOR) THEN
        WRITE(OUTF,360) NDOF,SBW
      ENDIF
360   FORMAT(//' NUMBER OF DEGREES OF FREEDOM IN MODIFIED,'/
     1       ' GLOBAL COMBINED SORPTION AND ADVECTION-DISPERSION'/
     2       ' MATRIX =',I5///' SEMI-BANDWIDTH OF MODIFIED,'/
     3       ' GLOBAL COMBINED SORPTION AND ADVECTION-DISPERSION'/
     4       ' MATRIX =',I5)
      RETURN
      END

SUBROUTINE ASMBAD
C**********************************************************************
C
C 19.1  PURPOSE
C         TO ASSEMBLE THE COMBINE GLOBAL SORPTION AND
C         ADVECTION-DISPERSION MATRIX AND THE GLOBAL SPECIFIED
C         SOLUTE FLUX MATRIX FOR THE MESH AND TO MODIFY THE
C         SYSTEM OF EQUATIONS FOR SPECIFIED CONCENTRATION AND
C         SOLUTE FLUX BOUNDARY CONDITIONS
C
C 19.2  INPUT
C         NONE
C
C 19.3  OUTPUT
C         THE SEMI-BANDWIDTH AND NUMBER OF DEGREES OF FREEDOM
C         FOR THE MODIFIED, COMBINED GLOBAL SORPTION AND
C         ADVECTION-DISPERSION MATRIX ARE WRITTEN TO THE USER-
C         DEFINED FILE ASSIGNED TO UNIT OUTF
C
C 19.4  DEFINITIONS OF VARIABLES:
C                     AE(I,J) = SORPTION MATRIX FOR ELEMENT E IN FULL
C                               MATRIX STORAGE
C                        B(I) = GLOBAL SPECIFIED SOLUTE FLUX MATRIX
C                     DE(I,J) = ADVECTION-DISPERSION MATRIX FOR ELEMENT
C                               E IN FULL MATRIX STORAGE
C                           E = ELEMENT NUMBER
C                  ELEMTYP(E) = ELEMENT TYPE FOR ELEMENT E
C                     FLUX(I) = SPECIFIED VALUE OF SOLUTE FLUX AT NODE I
C                      ICH(I) = 1 IF THE VALUE OF THE FIELD VARIABLE IS
C                               SPECIFIED AT NODE I
C                             = 0 OTHERWISE
C                     IJSIZE = LENGTH OF ARRAY ADGLOBAL
C                     LCH(I) = ICH(I) + ICH(I-1) + ...
C                      M(IJ) = MODIFIED, COMBINED GLOBAL SORPTION AND
C                               ADVECTION-DISPERSION MATRIX IN VECTOR
C                               STORAGE
C                       NDOF = NUMBER OF NODES WHERE THE VALUE OF
C                               THE FIELD VARIABLE IS UNKNOWN
C         NODETBL(ELEMTYP(E)) = NUMBER OF NODES IN ELEMENT TYPE E
```

```
C                      NUMELM = NUMBER OF ELEMENTS IN THE MESH
C                         SBW = SEMI-BANDWIDTH
C                        X(I) = VALUE OF SOLUTE CONCENTRATION AT NODE I
C
C 19.5  USAGE:
C         THE SEMI-BANDWIDTH OF THE COMBINED GLOBAL SORPTION AND
C         ADVECTION-DISPERSION MATRIX IS COMPUTED FIRST.  THEN THE
C         ENTRIES OF THE ELEMENT SORPTION AND ADVECTION-DISPERSION
C         MATRICES ARE COMPUTED IN A SET OF SUBROUTINES, TWO
C         SUBROUTINES FOR EACH ELEMENT TYPE.  THE COMBINED GLOBAL
C         SORPTION AND ADVECTION-DISPERSION MATRIX FOR THE MESH
C         ASSEMBLED BY ADDING THE CORRESPONDING ENTRIES OF THE ELEMENT
C         SORPTION AND ADVECTION-DISPERSION MATRICES TO THE GLOBAL
C         MATRIX.  DURING THE ASSEMBLY PROCESS THE GLOBAL MATRIX IS
C         MODIFIED FOR SPECIFIED VALUES OF SOLUTE CONCENTRATION AND
C         SOLUTE FLUX ARE ADDED TO THE GLOBAL SOLUTE FLUX MATRIX.
C
C       SUBROUTINES CALLED:
C         HEAPS, INCLUDING LOCATE, A~ AND D~
C
C***********************************************************************
      INCLUDE 'COMALL'
      DOUBLE PRECISION AE(MAX3,MAX3), DE(MAX3,MAX3)
      INTEGER NODETBL(13), E, I, KI, II, J, KJ, JJ
      DATA NODETBL/2,3,4,3,4,4,8,12,8,20,32,3,4/

C     INITIALISE ENTRIES OF GLOBAL CONDUCTANCE MATRIX TO ZERO
      DO 50 I = 1, NDOF
        B(I) = 0D0
        FC(I) = 0D0
        DO 40 J = 1, 3*SBW+1
          M2(J,I) = 0D0
 40     CONTINUE
        DO 50 J = 1, 2*SBW+1
          B2(J,I) = 0D0
 50   CONTINUE
C     INITIALISE ENTRIES OF THE GLOBAL SOLUTE FLUX MATRIX TO ZERO
C     LOOP ON THE NUMBER OF ELEMENTS

      print *,'In the interest of execution speed this program'
      print *,'assumes that all elements have the same X and Y'
      print *,'groundwater velocities and equal material sets.'
      e = 1

c      DO 90 E = 1, NUMELM

C       COMPUTE THE ELEMENT SORPTION AND ADVECTION DISPERSION MATRICES
C       FOR THIS ELEMENT TYPE
        IF (ELEMTYP(E) .EQ. 1) THEN
C         ELEMENT IS A ONE DIMENSIONAL, LINEAR BAR
          CALL ADBAR2(E,AE,DE)
        ELSEIF (ELEMTYP(E) .EQ. 4) THEN
C         ELEMENT IS A TWO DIMENSIONAL, LINEAR TRIANGLE
          CALL ATRI3(E,AE)
          CALL DTRI3(E,DE)
        ELSEIF (ELEMTYP(E) .EQ. 5) THEN
C         ELEMENT IS A TWO DIMENSIONAL, LINEAR RECTANGLE
          CALL AREC4(E,AE)
          CALL DREC4(E,DE)
        ELSEIF (ELEMTYP(E) .EQ. 6) THEN
C         ELEMENT IS A TWO DIMENSIONAL, LINEAR QUADRILATERAL
          CALL ADQUA4(E,AE,DE)
        ELSEIF (ELEMTYP(E) .EQ. 9) THEN
C         ELEMENT IS A THREE DIMENSIONAL, LINEAR PARALLELEPIPED
          CALL ADPAR8(E,AE,DE)
        ENDIF

c     For explanation see print statements above.
```

```
      do 90 e = 1, numelm

C       ADD THE ELEMENT SORPTION AND ADVECTION-DISPERSION MATRICES
C       FOR THIS ELEMENT TO THE GLOBAL MATRIX
C       AE(I,J), DE(I,J) ----> M(IJ)  <=>  M(KI,KJ)
        DO 80 I = 1, NODETBL(ELEMTYP(E))
          KI = IN (E,I)
          IF (ICH(KI) .EQ. 0) THEN
            II = KI - LCH(KI)
            DO 70 J = 1, NODETBL(ELEMTYP(E))
              KJ = IN(E,J)
              IF (ICH(KJ) .NE. 0) THEN
                FC(II) = FC(II) - DELTAT(IDT) * DE(I,J) * X(KJ)
              ELSE
                JJ = KJ - LCH(KJ)
                M2(2*SBW+1+II-JJ,JJ) = M2(2*SBW+1+II-JJ,JJ)
     1                 + AE(I,J) + OMEGA * DELTAT(IDT) * DE(I,J)
                B2(SBW+1+II-JJ,JJ) = B2(SBW+1+II-JJ,JJ)
     1                  + AE(I,J) - OMOMEGA * DELTAT(IDT) * DE(I,J)
              ENDIF
 70         CONTINUE
          ENDIF
 80     CONTINUE
 90   CONTINUE
      RETURN
      END

      SUBROUTINE ATRI3(E,AE)
C*********************************************************************
C
C       PURPOSE:
C         TO COMPUTE THE CONSISTENT FORM OF THE ELEMENT SORPTION
C         MATRIX FOR A TWO-DIMENSIONAL, LINEAR TRIANGLE ELEMENT
C
C       DEFINITIONS OF VARIABLES:
C         AE(I,J) = ELEMENT SORPTION MATRIX
C               E = ELEMENT NUMBER
C             KDE = ELEMENT DISTRIBUTION COEFFICIENT
C              NE = ELEMENT POROSITY
C           RHOBE = ELEMENT BULK DENSITY
C
C*********************************************************************
      INCLUDE 'COMALL'
      INTEGER E
      DOUBLE PRECISION AE(MAX3,MAX3),KDE,NE,RHOBE,AE4,TEMP

      RHOBE = PROP(MATSET(E),4)
      KDE = PROP(MATSET(E),5)
      NE = PROP(MATSET(E),6)
      AE4 = 2.0 * (X1(IN(E,2)) * X2(IN(E,3)) + X1(IN(E,1)) *
     1      X2(IN(E,2)) + X2(IN(E,1)) * X1(IN(E,3)) -
     2      X2(IN(E,3)) * X1(IN(E,1)) - X1(IN(E,3)) *
     3      X2(IN(E,2)) - X1(IN(E,2)) * X2(IN(E,1)))
      TEMP = AE4 / 12.0 / 4.0 * (1.0 + RHOBE * KDE / NE)
      AE(1,1) = 2.0 * TEMP
      AE(1,2) = TEMP
      AE(1,3) = TEMP
      AE(2,1) = TEMP
      AE(2,2) = AE(1,1)
      AE(2,3) = TEMP
      AE(3,1) = TEMP
      AE(3,2) = TEMP
      AE(3,3) = AE(1,1)
      RETURN
      END

      SUBROUTINE AREC4(E,AE)
C*********************************************************************
```

```
C
C       PURPOSE:
C         TO COMPUTE THE CONSISTENT FORM OF THE ELEMENT SORPTION
C         MATRIX FOR A TWO-DIMENSIONAL, LINEAR RECTANGLE ELEMENT
C
C       DEFINITIONS OF VARIABLES:
C         AE(I,J) = ELEMENT SORPTION MATRIX
C               E = ELEMENT NUMBER
C             KDE = ELEMENT DISTRIBUTION COEFFICIENT
C              NE = ELEMENT POROSITY
C           RHOBE = ELEMENT BULK DENSITY
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E
      DOUBLE PRECISION AE(MAX3,MAX3),KDE,NE,RHOBE,TEMP

      RHOBE = PROP(MATSET(E),4)
      KDE = PROP(MATSET(E),5)
      NE = PROP(MATSET(E),6)
      TEMP = (1.0 + RHOBE * KDE / NE) * ABS((X2(IN(E,1)) - X2(IN(E,3)))
     1        / 2.0 * (X1(IN(E,1)) - X1(IN(E,3))) / 2.0 ) / 9.0
      AE(1,1) = 4 * TEMP
      AE(1,2) = 2 * TEMP
      AE(1,3) = TEMP
      AE(1,4) = AE(1,2)
      AE(2,1) = AE(1,2)
      AE(2,2) = AE(1,1)
      AE(2,3) = AE(1,2)
      AE(2,4) = AE(1,3)
      AE(3,1) = AE(1,3)
      AE(3,2) = AE(1,2)
      AE(3,3) = AE(1,1)
      AE(3,4) = AE(1,2)
      AE(4,1) = AE(1,2)
      AE(4,2) = AE(1,3)
      AE(4,3) = AE(1,2)
      AE(4,4) = AE(1,1)
      RETURN
      END

      SUBROUTINE ADBAR2(E,AE,DE)
C*****************************************************************
C
C       PURPOSE:
C         TO COMPUTE THE CONSISTENT FORM OF THE ELEMENT SORPTION
C         AND ADVECTION-DISPESION MATRICES FOR A ONE-DIMENSIONAL,
C         LINEAR BAR ELEMENT
C
C       DEFINITIONS OF VARIABLES:
C         AE(I,J) = ELEMENT SORPTION MATRIX
C             ALE = LONGITUDINAL DISPERSIVITY FOR ELEMENT
C         DE(I,J) = ELEMENT ADVECTION DISPERSION MATRIX
C             DXE = ELEMENT DISPESION COEFFICIENT
C               E = ELEMENT NUMBER
C             KDE = ELEMENT DISTRIBUTION COEFFICIENT
C          LAMBDA = SOLUTE DECAY COEFFICIENT
C              LE = ELEMENT LENGTH
C              NE = ELEMENT POROSITY
C           RHOBE = ELEMENT BULK DENSITY
C             VXE = APPARENT GROUNDWATER VELOCITY IN
C                   X COORDINATE DIRECTION
C            VXEP = PORE WATER VELOCITY IN X COORDINATE DIRECTION
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E
      DOUBLE PRECISION DE(MAX3,MAX3), KDE, LAMBDA, LE ,NE, ALE, RHOBE
```

```
      DOUBLE PRECISION VXE, VXEP, DXE, TEMP3, AE(MAX3,MAX3)
      ALE    = PROP(MATSET(E),1)
      LAMBDA = PROP(MATSET(E),2)
      RHOBE  = PROP(MATSET(E),3)
      KDE    = PROP(MATSET(E),4)
      NE     = PROP(MATSET(E),5)
      VXE    = V1(E)
      VXEP   = VXE / NE
      LE     = ABS(X1(IN(E,2)) - X1(IN(E,1)))
      DXE    = ALE * VXEP
      TEMP3  = LAMBDA * (1.0 + RHOBE * KDE / NE) * (LE / 6.0)
      DE(1,1) =  DXE / LE - VXEP / 2.0 + 2.0 * TEMP3
      DE(1,2) = -DXE / LE + VXEP / 2.0 +       TEMP3
      DE(2,1) = -DXE / LE - VXEP / 2.0 +       TEMP3
      DE(2,2) =  DXE / LE + VXEP / 2.0 + 2.0 * TEMP3
      AE(1,1) = (1.0 + RHOBE * KDE / NE) * (LE / 6) * 2.0
      AE(1,2) = AE(1,1) / 2.0
      AE(2,1) = AE(1,2)
      AE(2,2) = AE(1,1)
      RETURN
      END

      SUBROUTINE DTRI3(E,DE)
C*****************************************************************
C
C       PURPOSE:
C         TO COMPUTE THE CONSISTENT FORM OF THE ELEMENT
C         ADVECTION-DISPESION MATRIX FOR A TWO-DIMENSIONAL,
C         LINEAR TRIANGLE ELEMENT
C
C       DEFINITIONS OF VARIABLES:
C             AE4 = FOUR TIMES ELEMENT AREA
C             ALE = LONGITUDINAL DISPERSIVITY FOR ELEMENT
C             ATE = TRANSVERSE DISPERSIVITY FOR ELEMENT
C         DE(I,J) = ELEMENT ADVECTION DISPERSION MATRIX
C     DXXE (ETC.) = ELEMENT DISPESION COEFFICIENTS
C               E = ELEMENT NUMBER
C             KDE = ELEMENT DISTRIBUTION COEFFICIENT
C          LAMBDA = SOLUTE DECAY COEFFICIENT
C              NE = ELEMENT POROSITY
C           RHOBE = ELEMENT BULK DENSITY
C             VXE = APPARENT GROUNDWATER VELOCITY IN
C                   X COORDINATE DIRECTION
C             VYE = APPARENT GROUNDWATER VELOCITY IN
C                   Y COORDINATE DIRECTION
C            VXEP = PORE WATER VELOCITY IN X COORDINATE DIRECTION
C            VYEP = PORE WATER VELOCITY IN Y COORDINATE DIRECTION
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I, J
      DOUBLE PRECISION DE(MAX3,MAX3), KDE, LAMBDA, NE, BE(3), CE(3)
      DOUBLE PRECISION ALE, ATE, RHOBE, VXE, VYE, VXEP, VYEP, VP
      DOUBLE PRECISION DXXE, DYYE, DXYE, DYXE, TEMP, AE4
      ALE    = PROP(MATSET(E),1)
      ATE    = PROP(MATSET(E),2)
      LAMBDA = PROP(MATSET(E),3)
      RHOBE  = PROP(MATSET(E),4)
      KDE    = PROP(MATSET(E),5)
      NE     = PROP(MATSET(E),6)
      VXE    = V1(E)
      VYE    = V2(E)
      VXEP   = VXE / NE
      VYEP   = VYE / NE
      VP     = SQRT(VXEP**2 + VYEP**2)
      IF (ABS(VP) .LT. 1E-40) THEN
        DXXE = 0.0
        DYYE = 0.0
```

```
        DXYE = 0.0
      ELSE
        DXXE   = (ALE * VXEP**2 + ATE * VYEP**2) / VP
        DYYE   = (ALE * VYEP**2 + ATE * VXEP**2) / VP
        DXYE   = ((ALE  - ATE) * VXEP * VYEP) / VP
      ENDIF
      DYXE   = DXYE
      BE(1)  = X2(IN(E,2)) - X2(IN(E,3))
      BE(2)  = X2(IN(E,3)) - X2(IN(E,1))
      BE(3)  = X2(IN(E,1)) - X2(IN(E,2))
      CE(1)  = X1(IN(E,2)) - X1(IN(E,3))
      CE(2)  = X1(IN(E,3)) - X1(IN(E,1))
      CE(3)  = X1(IN(E,1)) - X1(IN(E,2))
      AE4 = 2.0 * (X1(IN(E,2)) * X2(IN(E,3)) + X1(IN(E,1)) *
     1      X2(IN(E,2)) + X2(IN(E,1)) * X1(IN(E,3)) -
     2      X2(IN(E,3)) * X1(IN(E,1)) - X1(IN(E,3)) *
     3      X2(IN(E,2)) - X1(IN(E,2)) * X2(IN(E,1)))
      TEMP = AE4 / 48.0 * LAMBDA * (1.0 + RHOBE * KDE / NE)
      DO 20 I = 1, 3
        DO 10 J = 1, 3
          DE(I,J) = (DXXE*BE(I)*BE(J) + DYYE*CE(I)*CE(J) +
     1               DXYE*BE(I)*CE(J) + DYXE*CE(I)*BE(J) ) /AE4
     2             + VXEP / 6.0 * BE(J) + VYEP / 6.0 * CE(J) + TEMP
          IF (I .EQ. J) DE(I,J) = DE(I,J) + TEMP
 10     CONTINUE
 20   CONTINUE
      RETURN
      END

      SUBROUTINE DREC4(E,DE)
C*********************************************************************
C
C       PURPOSE:
C         TO COMPUTE THE CONSISTENT FORM OF THE ELEMENT
C         ADVECTION-DISPESION MATRIX FOR A TWO-DIMENSIONAL,
C         LINEAR RECTANGLE ELEMENT
C
C       DEFINITIONS OF VARIABLES:
C             ALE = LONGITUDINAL DISPERSIVITY FOR ELEMENT
C             ATE = TRANSVERSE DISPERSIVITY FOR ELEMENT
C         DE(I,J) = ELEMENT ADVECTION DISPERSION MATRIX
C     DXXE (ETC.) = ELEMENT DISPESION COEFFICIENTS
C               E = ELEMENT NUMBER
C             KDE = ELEMENT DISTRIBUTION COEFFICIENT
C          LAMBDA = SOLUTE DECAY COEFFICIENT
C              NE = ELEMENT POROSITY
C           RHOBE = ELEMENT BULK DENSITY
C             VXE = APPARENT GROUNDWATER VELOCITY IN
C                   X COORDINATE DIRECTION
C             VYE = APPARENT GROUNDWATER VELOCITY IN
C                   Y COORDINATE DIRECTION
C            VXEP = PORE WATER VELOCITY IN X COORDINATE DIRECTION
C            VYEP = PORE WATER VELOCITY IN Y COORDINATE DIRECTION
C
C*********************************************************************
      INCLUDE 'COMALL'
      INTEGER E
      DOUBLE PRECISION DE(MAX3,MAX3), KDE, LAMBDA, NE, AE, BE
      DOUBLE PRECISION ALE, ATE, RHOBE, VXE, VYE, VXEP, VYEP, VP
      DOUBLE PRECISION DXXE, DYYE, DXYE, DYXE, TEMP1, TEMP2
      DOUBLE PRECISION TEMP3, TEMP4, TEMP5, TEMP6, TEMP7
      ALE    = PROP(MATSET(E),1)
      ATE    = PROP(MATSET(E),2)
      LAMBDA = PROP(MATSET(E),3)
      RHOBE  = PROP(MATSET(E),4)
      KDE    = PROP(MATSET(E),5)
      NE     = PROP(MATSET(E),6)
      VXE    = V1(E)
```

```
      VYE    = V2(E)
      VXEP   = VXE / NE
      VYEP   = VYE / NE
      VP     = SQRT(VXEP**2 + VYEP**2)
      IF (ABS(VP) .LT. 1E-40) THEN
        DXXE = 0.0
        DYYE = 0.0
        DXYE = 0.0
      ELSE
        DXXE   = (ALE * VXEP**2 + ATE * VYEP**2) / VP
        DYYE   = (ALE * VYEP**2 + ATE * VXEP**2) / VP
        DXYE   = ((ALE  - ATE) * VXEP * VYEP) / VP
      ENDIF
      DYXE   = DXYE
      AE     = ABS(X2(IN(E,1)) - X2(IN(E,3)))/ 2.0
      BE     = ABS(X1(IN(E,1)) - X1(IN(E,3)))/ 2.0
      TEMP1  = (DXXE * AE) / (6.0 * BE)
      TEMP2  = (DYYE * BE) / (6.0 * AE)
      TEMP3  = DXYE / 4.0
      TEMP4  = DYXE / 4.0
      TEMP5  = VXEP * AE / 6.0
      TEMP6  = VYEP * BE / 6.0
      TEMP7  = LAMBDA * (1.0 + RHOBE * KDE / NE) * (AE * BE) / 9.0
      DE(1,1)= 2.*TEMP1+2.*TEMP2+TEMP3+TEMP4-2.*TEMP5-2.*TEMP6+4.*TEMP7
      DE(1,2)=-2.*TEMP1+   TEMP2+TEMP3-TEMP4+2.*TEMP5-   TEMP6+2.*TEMP7
      DE(1,3)=-   TEMP1-   TEMP2-TEMP3-TEMP4+   TEMP5+   TEMP6+   TEMP7
      DE(1,4)=    TEMP1-2.*TEMP2-TEMP3+TEMP4-   TEMP5+2.*TEMP6+2.*TEMP7
      DE(2,1)=-2.*TEMP1+   TEMP2-TEMP3+TEMP4-2.*TEMP5-   TEMP6+2.*TEMP7
      DE(2,2)= 2.*TEMP1+2.*TEMP2-TEMP3-TEMP4+2.*TEMP5-2.*TEMP6+4.*TEMP7
      DE(2,3)=    TEMP1-2.*TEMP2+TEMP3-TEMP4+   TEMP5+2.*TEMP6+2.*TEMP7
      DE(2,4)=-   TEMP1-   TEMP2+TEMP3+TEMP4-   TEMP5+   TEMP6+   TEMP7
      DE(3,1)=-   TEMP1-   TEMP2-TEMP3-TEMP4-   TEMP5-   TEMP6+   TEMP7
      DE(3,2)=    TEMP1-2.*TEMP2-TEMP3+TEMP4+   TEMP5-2.*TEMP6+2.*TEMP7
      DE(3,3)= 2.*TEMP1+2.*TEMP2+TEMP3+TEMP4+2.*TEMP5+2.*TEMP6+4.*TEMP7
      DE(3,4)=-2.*TEMP1+   TEMP2+TEMP3-TEMP4-2.*TEMP5+   TEMP6+2.*TEMP7
      DE(4,1)=    TEMP1-2.*TEMP2+TEMP3-TEMP4-   TEMP5-2.*TEMP6+2.*TEMP7
      DE(4,2)=-   TEMP1-   TEMP2+TEMP3+TEMP4+   TEMP5-   TEMP6+   TEMP7
      DE(4,3)=-2.*TEMP1+   TEMP2-TEMP3+TEMP4+2.*TEMP5+   TEMP6+2.*TEMP7
      DE(4,4)= 2.*TEMP1+2.*TEMP2-TEMP3-TEMP4-2.*TEMP5+2.*TEMP6+4.*TEMP7
      RETURN
      END


      SUBROUTINE ADQUA4(E,AE,DE)
C*********************************************************************
C
C       PURPOSE:
C         TO COMPUTE THE CONSISTENT FORM OF THE ELEMENT SORPTION
C         AND ADVECTION-DISPESION MATRICES FOR A TWO-DIMENSIONAL,
C         LINEAR QUADRILATERAL ELEMENT
C
C       DEFINITIONS OF VARIABLES:
C         AE(I,J) = ELEMENT SORPTION MATRIX
C             ALE = LONGITUDINAL DISPERSIVITY FOR ELEMENT
C             ATE = TRANSVERSE DISPERSIVITY FOR ELEMENT
C         DE(I,J) = ELEMENT ADVECTION DISPERSION MATRIX
C          DETJAC = DETERMINANT OF THE JACOBIAN MATRIX
C        DNDXI(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                   FUNCTION WITH RESPECT TO XI AT NODE I
C         DNDX(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                   FUNCTION WITH RESPECT TO X AT NODE I
C       DNDETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                   FUNCTION WITH RESPECT TO ETA AT NODE I
C         DNDY(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                   FUNCTION WITH RESPECT TO Y AT NODE I
C     DXXE (ETC.) = ELEMENT DISPESION COEFFICIENTS
C               E = ELEMENT NUMBER
C           XI(I) = LOCATION OF GAUSS POINT IN XI COORDINATE DIRECTION
```

```
C          ETA(I) = LOCATION OF GAUSS POINT IN ETA COORDINATE DIRECTION
C        JAC(I,J) = JACOBIAN MATRIX
C           N(I) = INTERPOLATION FUNCTION FOR NODE I
C           W(I) = WEIGHT FOR GAUSS POINT I
C            KDE = ELEMENT DISTRIBUTION COEFFICIENT
C          LAMBDA = SOLUTE DECAY COEFFICIENT
C              NE = ELEMENT POROSITY
C           RHOBE = ELEMENT BULK DENSITY
C             VXE = APPARENT GROUNDWATER VELOCITY IN
C                   X COORDINATE DIRECTION
C             VYE = APPARENT GROUNDWATER VELOCITY IN
C                   Y COORDINATE DIRECTION
C            VXEP = PORE WATER VELOCITY IN X COORDINATE DIRECTION
C            VYEP = PORE WATER VELOCITY IN Y COORDINATE DIRECTION
C     X1(IN(E,I)) = X COORDINATE FOR NODE I, ELEMENT E
C     X2(IN(E,I)) = Y COORDINATE FOR NODE I, ELEMENT E
C
C*********************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I, J, K, K1
      DOUBLE PRECISION JAC(2,2), JACINV(2,2), N(4), DETJAC
      DOUBLE PRECISION DNDXI(4), DNDX(4), DNDY(4), DNDETA(4)
      DOUBLE PRECISION W(2), XI(2), ETA(2), DE(MAX3,MAX3)
      DOUBLE PRECISION SIGN1(4), SIGN2(4), KDE, NE, LAMBDA
      DOUBLE PRECISION ALE, ATE, RHOBE, VXE, VYE, VXEP, VYEP, VP
      DOUBLE PRECISION DXXE, DYYE, DXYE, DYXE, AE(MAX3,MAX3)
      DOUBLE PRECISION LENG1, LENG2, TEMP1, TEMP2, TEMPA, TEMPB
      DOUBLE PRECISION TEMP3X, TEMP3Y, TEMP4X, TEMP4Y
      DOUBLE PRECISION TEMP5, TEMP6, TEMP7, TEMP8
      DOUBLE PRECISION ALPHA, BETA, WXI(4), WETA(4), DWDXI(4)
      DOUBLE PRECISION DWDETA(4), DWDX(4), DWDY(4)
      DOUBLE PRECISION VXI, VETA, VXW(4), VYW(4)
      DATA SIGN1/-1.0, 1.0, 1.0,-1.0/
      DATA SIGN2/-1.0,-1.0, 1.0, 1.0/
      XI(1) = SQRT(DBLE(1D0/3D0))
      XI(2) = -XI(1)
      ETA(1) = XI(1)
      ETA(2) = XI(2)
      W(1) = 1.0
      W(2) = 1.0
      ALE    = PROP(MATSET(E),1)
      ATE    = PROP(MATSET(E),2)
      LAMBDA = PROP(MATSET(E),3)
      RHOBE  = PROP(MATSET(E),4)
      KDE    = PROP(MATSET(E),5)
      NE     = PROP(MATSET(E),6)
      IF (NE .EQ. 0.0) PRINT *,'NE = 0.0 CHECK NUMBER OF PROPS'
      VXE    = V1(E)
      VYE    = V2(E)
      VXEP   = VXE / NE
      VYEP   = VYE / NE
      VP     = SQRT(VXEP**2 + VYEP**2)
      IF (ABS(VP) .LT. 1E-40) THEN
        DXXE = 0.0
        DYYE = 0.0
        DXYE = 0.0
        ALPHA = 0.0
        BETA = 0.0
      ELSE
        DXXE   = (ALE * VXEP**2 + ATE * VYEP**2) / VP
        DYYE   = (ALE * VYEP**2 + ATE * VXEP**2) / VP
        DXYE   = ((ALE  - ATE) * VXEP * VYEP) / VP
        LENG1 = SQRT((X1(IN(E,2)) - X1(IN(E,1)))**2
     1              + (X2(IN(E,2)) - X2(IN(E,1)))**2)
        VXI  = (   (X1(IN(E,2)) - X1(IN(E,1))) * VXEP
     1           + (X2(IN(E,2)) - X2(IN(E,1))) * VYEP ) / LENG1
        LENG2 = SQRT((X1(IN(E,3)) - X1(IN(E,2)))**2
     1              + (X2(IN(E,3)) - X2(IN(E,2)))**2)
```

```
        VETA = (    (X1(IN(E,3)) - X1(IN(E,2))) * VXEP
     1           + (X2(IN(E,3)) - X2(IN(E,2))) * VYEP ) / LENG2
C       1E-10 IS CHOSEN TO BE SLIGHTLY MORE THAN DOUBLE PRECISION
        IF (VXI .GT.  VP * 1E-10) THEN
          ALPHA = PROP(MATSET(E),7)
        ELSEIF (VXI .LT. -VP * 1E-10) THEN
          ALPHA = -PROP(MATSET(E),7)
        ELSE
          ALPHA = 0.0
        ENDIF
        IF (VETA .GT. VP * 1E-10) THEN
          BETA = PROP(MATSET(E),7)
        ELSEIF (VETA .LT. -VP * 1E-10) THEN
          BETA = -PROP(MATSET(E),7)
        ELSE
          BETA = 0.0
        ENDIF
      ENDIF
      DYXE   = DXYE
      DO 30 I = 1, 4
        DO 20 J = 1, 4
          AE(I,J) = 0.0
          DE(I,J) = 0.0
 20     CONTINUE
 30   CONTINUE
      DO 120 I = 1, 2
        DO 110 J = 1, 2

          DO 50 K = 1, 2
            DO 40 K1 = 1, 2
              JAC(K,K1) = 0.0
 40         CONTINUE
 50       CONTINUE
          DO 60 K1 = 1, 4
            TEMPA = (1.0 + SIGN1(K1) * XI(I))
            TEMPB = (1.0 + SIGN2(K1) * ETA(J))
            TEMP1 = (1.0 + XI(I))
            TEMP2 = (1.0 + ETA(J))
            TEMP3X = (2.0 + 3.0 * ALPHA  * (1.0 - XI(I)))
     1              * SIGN1(K1)
            TEMP3Y = 2.0 * SIGN1(K1)
            TEMP4X = 2.0 * SIGN2(K1)
            TEMP4Y = (2.0 + 3.0 * BETA  * (1.0 - ETA(J)))
     1              * SIGN2(K1)
            TEMP5 = 2.0 * (1.0 - SIGN1(K1))
            TEMP6 = 2.0 * (1.0 - SIGN2(K1))
            TEMP7 = 3.0 * ALPHA * XI(I) - 1.0
            TEMP8 = 3.0 * BETA * ETA(J) - 1.0
                 N(K1) = 0.25 * TEMPA * TEMPB
             DNDXI(K1) = 0.25 * SIGN1(K1) * TEMPB
            DNDETA(K1) = 0.25 * SIGN2(K1) * TEMPA
               WXI(K1) = 0.0625 * (TEMP1 * TEMP3X + TEMP5)
     1                          * (TEMP2 * TEMP4X + TEMP6)
              WETA(K1) = 0.0625 * (TEMP1 * TEMP3Y + TEMP5)
     1                          * (TEMP2 * TEMP4Y + TEMP6)
             DWDXI(K1) = -0.125 * SIGN1(K1) * TEMP7
     1                          * (TEMP2 * TEMP4X + TEMP6)
            DWDETA(K1) = -0.125 * SIGN2(K1) * TEMP8
     1                          * (TEMP1 * TEMP3Y + TEMP5)
 60       CONTINUE
          DO 70 K1 = 1, 4
            JAC(1,1) = JAC(1,1) + DNDXI(K1) * X1(IN(E,K1))
            JAC(1,2) = JAC(1,2) + DNDXI(K1) * X2(IN(E,K1))
            JAC(2,1) = JAC(2,1) + DNDETA(K1) * X1(IN(E,K1))
            JAC(2,2) = JAC(2,2) + DNDETA(K1) * X2(IN(E,K1))
 70       CONTINUE
          DETJAC = JAC(1,1) * JAC(2,2) - JAC(1,2) * JAC(2,1)
          JACINV(1,1) =  JAC(2,2) / DETJAC
```

```
          JACINV(1,2) = -JAC(1,2) / DETJAC
          JACINV(2,1) = -JAC(2,1) / DETJAC
          JACINV(2,2) =  JAC(1,1) / DETJAC
          DO 80 K1 = 1, 4
           DNDX(K1) = JACINV(1,1) * DNDXI(K1) + JACINV(1,2) * DNDETA(K1)
           DNDY(K1) = JACINV(2,1) * DNDXI(K1) + JACINV(2,2) * DNDETA(K1)
           DWDX(K1) = JACINV(1,1) * DWDXI(K1) + JACINV(1,2) * DWDETA(K1)
           DWDY(K1) = JACINV(2,1) * DWDXI(K1) + JACINV(2,2) * DWDETA(K1)
           VXW(K1) = WXI(K1) * VXI * (X1(IN(E,2)) - X1(IN(E,1))) / LENG1
     1           + WETA(K1) * VETA * (X1(IN(E,3)) - X1(IN(E,2))) / LENG2
           VYW(K1) = WXI(K1) * VXI * (X2(IN(E,2)) - X2(IN(E,1))) / LENG1
     1           + WETA(K1) * VETA * (X2(IN(E,3)) - X2(IN(E,2))) / LENG2
 80       CONTINUE
          DO 100 K = 1, 4
            DO 90 K1 = 1, 4
              AE(K,K1) = AE(K,K1) + W(I) * W(J) * (1.0 + RHOBE*KDE/NE)
     1                  * N(K) * N(K1) * DETJAC
              DE(K,K1) = DE(K,K1) + W(I) * W(J) *
     1                    (DXXE * DWDX(K) * DNDX(K1)
     2                   + DXYE * DWDX(K) * DNDY(K1)
     3                   + DYXE * DWDY(K) * DNDX(K1)
     4                   + DYYE * DWDY(K) * DNDY(K1)
C                          SEE ABOVE FOR VXW(K) = W(K) * VXEP
     5                   + VXW(K) * DNDX(K1)
     6                   + VYW(K) * DNDY(K1)
     7                   + LAMBDA * (1.0 + RHOBE * KDE / NE)
     8                   * N(K) * N(K1)) * DETJAC
 90         CONTINUE
 100      CONTINUE
 110    CONTINUE
 120  CONTINUE
      RETURN
      END

      SUBROUTINE ADPAR8(E,AE,DE)
C*****************************************************************
C
C         PURPOSE:
C           TO COMPUTE THE CONSISTENT FORM OF THE ELEMENT
C           SORPTION MATRIX AND THE ELEMENT CONDUCTANCE
C           MATRIX FOR A THREE DIMENSIONAL, LINEAR
C           PARALLELEPIPED ELEMENT
C
C         DEFINITIONS OF VARIABLES:
C               AE(I,J) = ELEMENT CAPACITANCE MATRIX
C                   ALE = LONGITUDINAL DISPERSIVITY FOR ELEMENT
C                   ATE = TRANSVERSE DISPERSIVITY FOR ELEMENT
C               DE(I,J) = ELEMENT ADVECTION DISPERSION MATRIX
C                DETJAC = DETERMINANT OF JACOBIAN MATRIX
C              DNDXI(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                         FUNCTION WITH RESPECT TO XI AT NODE I
C               DNDX(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                         FUNCTION WITH RESPECT TO X AT NODE I
C             DNDETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                         FUNCTION WITH RESPECT TO ETA AT NODE I
C               DNDY(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                         FUNCTION WITH RESPECT TO Y AT NODE I
C            DNDZETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                         FUNCTION WITH RESPECT TO ZETA AT NODE I
C               DNDZ(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                         FUNCTION WITH RESPECT TO Z AT NODE I
C                     E = ELEMENT NUMBER
C                ETA(I) = LOCATION OF GAUSS POINT IN ETA
C                         COORDINATE DIRECTION
C               IN(I,J) = NODE NUMBER J FOR ELEMENT I
C              JAC(I,J) = JACOBIAN MATRIX
C            JACINV(I,J) = INVERSE OF JACOBIAN MATRIX
C                   KDE = ELEMENT DISTRIBUTION COEFFICIENT
```

```
C                LAMBDA = SOLUTE DECAY COEFFICIENT
C                  N(I) = INTERPOLATION FUNCTION FOR NODE I
C                    NE = ELEMENT POROSITY
C                 RHOBE = ELEMENT BULK DENSITY
C                   VXE = APPARENT GROUNDWATER VELOCITY IN
C                         X COORDINATE DIRECTION
C                   VYE = APPARENT GROUNDWATER VELOCITY IN
C                         Y COORDINATE DIRECTION
C                   VZE = APPARENT GROUNDWATER VELOCITY IN
C                         Z COORDINATE DIRECTION
C                  VXEP = PORE WATER VELOCITY IN X COORDINATE DIRECTION
C                  VYEP = PORE WATER VELOCITY IN Y COORDINATE DIRECTION
C                  VZEP = PORE WATER VELOCITY IN Z COORDINATE DIRECTION
C                  W(I) = WEIGHT FOR GAUSS POINT I
C            X1(IN(E,I) = X COORDINATE FOR NODE I, ELEMENT E
C            X2(IN(E,I) = Y COORDINATE FOR NODE I, ELEMENT E
C            X3(IN(E,I) = Z COORDINATE FOR NODE I, ELEMENT E
C                 XI(I) = LOCATION OF GAUSS POINT IN XI COORDINATE
C                         DIRECTION
C               ZETA(I) = LOCATION OF GAUSS POINT IN ZETA COORDINATE
C                         DIRECTION
C
C*********************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I, J, K, L, N1
      DOUBLE PRECISION JAC(3,3), AE(MAX3,MAX3), DE(MAX3,MAX3)
      DOUBLE PRECISION DNDXI(8), DNDX(8), DNDETA(8), DNDY(8)
      DOUBLE PRECISION DNDZETA(8), DNDZ(8), W(2), XI(2), ETA(2)
      DOUBLE PRECISION ZETA(2), SIGN1(8), SIGN2(8), SIGN3(8)
      DOUBLE PRECISION RHOBE, KDE, NE,LAMBDA, JACINV(3,3), DETJAC
      DOUBLE PRECISION VXE, VYE, VZE, VXEP, VYEP, VZEP, VP
      DOUBLE PRECISION DXXE, DXYE, DYXE, DYYE, DXZE, DZXE
      DOUBLE PRECISION DZZE, DYZE, DZYE, ALE, ATE, N(8)
      DATA SIGN1/-1.0, 1.0, 1.0,-1.0,-1.0, 1.0, 1.0,-1.0/
      DATA SIGN2/-1.0,-1.0, 1.0, 1.0,-1.0,-1.0, 1.0, 1.0/
      DATA SIGN3/-1.0,-1.0,-1.0,-1.0, 1.0, 1.0, 1.0, 1.0/

      XI(1) = SQRT(DBLE(1D0/3D0))
      XI(2) = -XI(1)
      ETA(1) = XI(1)
      ETA(2) = XI(2)
      ZETA(1) = XI(1)
      ZETA(2) = XI(2)
      W(1) = 1.0
      W(2) = 1.0
      ALE    = PROP(MATSET(E),1)
      ATE    = PROP(MATSET(E),2)
      LAMBDA = PROP(MATSET(E),3)
      RHOBE  = PROP(MATSET(E),4)
      KDE    = PROP(MATSET(E),5)
      NE     = PROP(MATSET(E),6)
      VXE    = V1(E)
      VYE    = V2(E)
      VZE    = V3(E)
      VXEP   = VXE / NE
      VYEP   = VYE / NE
      VZEP   = VZE / NE
      VP     = SQRT(VXEP**2 + VYEP**2 + VZEP**2)
      IF (ABS(VP) .LT. 1E-40) THEN
        DXXE = 0.0
        DYYE = 0.0
        DZZE = 0.0
        DXYE = 0.0
        DXZE = 0.0
        DYZE = 0.0
      ELSE
        DXXE = (ALE * VXEP**2 + ATE * (VYEP**2 + VZEP**2)) / VP
        DYYE = (ALE * VYEP**2 + ATE * (VXEP**2 + VZEP**2)) / VP
```

```
        DZZE = (ALE * VZEP**2 + ATE * (VXEP**2 + VYEP**2)) / VP
        DXYE = ((ALE  - ATE) * VXEP * VYEP) / VP
        DXZE = ((ALE  - ATE) * VXEP * VZEP) / VP
        DYZE = ((ALE  - ATE) * VYEP * VZEP) / VP
      ENDIF
      DYXE = DXYE
      DZXE = DXZE
      DZYE = DYZE
      DO 20 K = 1, 8
        DO 10 N1 = 1, 8
          AE(K,N1) = 0.0
          DE(K,N1) = 0.0
10      CONTINUE
20    CONTINUE
      DO 140 I = 1, 2
        DO 130 J = 1, 2
          DO 120 K = 1, 2
            DO 60 L = 1, 3
              DO 50 N1 = 1, 3
                JAC(L,N1) = 0.0
50            CONTINUE
60          CONTINUE
            DO 70 N1 = 1, 8
              N(N1)      = 0.125 * (1.0 + SIGN1(N1) * XI(I))
     1                           * (1.0 + SIGN2(N1) * ETA(J))
     2                           * (1.0 + SIGN3(N1) * ZETA(K))
              DNDXI(N1) = 0.125 * SIGN1(N1) * (1.0 + SIGN2(N1) * ETA(J))
     1                  * (1.0 + SIGN3(N1) * ZETA(K))
              DNDETA(N1) = 0.125 * SIGN2(N1) * (1.0 + SIGN1(N1) * XI(I))
     1                   * (1.0 + SIGN3(N1) * ZETA(K))
              DNDZETA(N1)= 0.125 * SIGN3(N1) * (1.0 + SIGN1(N1) * XI(I))
     1                   * (1.0 + SIGN2(N1) * ETA(J))
70          CONTINUE
            DO 80 N1 = 1, 8
              JAC(1,1) = JAC(1,1) + DNDXI(N1) * X1(IN(E,N1))
              JAC(1,2) = JAC(1,2) + DNDXI(N1) * X2(IN(E,N1))
              JAC(1,3) = JAC(1,3) + DNDXI(N1) * X3(IN(E,N1))
              JAC(2,1) = JAC(2,1) + DNDETA(N1) * X1(IN(E,N1))
              JAC(2,2) = JAC(2,2) + DNDETA(N1) * X2(IN(E,N1))
              JAC(2,3) = JAC(2,3) + DNDETA(N1) * X3(IN(E,N1))
              JAC(3,1) = JAC(3,1) + DNDZETA(N1) * X1(IN(E,N1))
              JAC(3,2) = JAC(3,2) + DNDZETA(N1) * X2(IN(E,N1))
              JAC(3,3) = JAC(3,3) + DNDZETA(N1) * X3(IN(E,N1))
80          CONTINUE
            DETJAC = JAC(1,1) * (JAC(2,2)*JAC(3,3) - JAC(3,2)*JAC(2,3))
     1             - JAC(1,2) * (JAC(2,1)*JAC(3,3) - JAC(3,1)*JAC(2,3))
     2             - JAC(1,3) * (JAC(2,1)*JAC(3,2) - JAC(3,1)*JAC(2,2))
           if (detjac .eq. 0) stop 'detjac = 0, check dimension'
C           INVERSE JACOBIAN MATRIX FORMULA HAS BEEN TRANSPOSED STEVE 9/7/96
            JACINV(1,1) = ( JAC(2,2) * JAC(3,3) - JAC(2,3) * JAC(3,2))
     1                    / DETJAC
            JACINV(2,1) = (-JAC(2,1) * JAC(3,3) + JAC(2,3) * JAC(3,1))
     1                    / DETJAC
            JACINV(3,1) = ( JAC(2,1) * JAC(3,2) - JAC(3,1) * JAC(2,2))
     1                    / DETJAC
            JACINV(1,2) = (-JAC(1,2) * JAC(3,3) + JAC(1,3) * JAC(3,2))
     1                    / DETJAC
            JACINV(2,2) = ( JAC(1,1) * JAC(3,3) - JAC(1,3) * JAC(3,1))
     1                    / DETJAC
            JACINV(3,2) = (-JAC(1,1) * JAC(3,2) + JAC(1,2) * JAC(3,1))
     1                    / DETJAC
            JACINV(1,3) = ( JAC(1,2) * JAC(2,3) - JAC(1,3) * JAC(2,2))
     1                    / DETJAC
            JACINV(2,3) = (-JAC(1,1) * JAC(2,3) + JAC(1,3) * JAC(2,1))
     1                    / DETJAC
            JACINV(3,3) = ( JAC(1,1) * JAC(2,2) - JAC(1,2) * JAC(2,1))
     1                    / DETJAC
            DO 90 N1 = 1, 8
```

```
              DNDX(N1) = JACINV(1,1) * DNDXI(N1) + JACINV(1,2) *
     1                   DNDETA(N1) + JACINV(1,3) * DNDZETA(N1)
              DNDY(N1) = JACINV(2,1) * DNDXI(N1) + JACINV(2,2) *
     1                   DNDETA(N1) + JACINV(2,3) * DNDZETA(N1)
              DNDZ(N1) = JACINV(3,1) * DNDXI(N1) + JACINV(3,2) *
     1                   DNDETA(N1) + JACINV(3,3) * DNDZETA(N1)
 90         CONTINUE
            DO 110 L = 1, 8
              DO 100 N1 = 1, 8
                AE(L,N1) = AE(L,N1) + W(I) * W(J) * W(K) * DETJAC *
     1                     (1.0 + RHOBE * KDE / NE) * N(L) * N(N1)
                DE(L,N1) = DE(L,N1) + W(I) * W(J) * W(K) * DETJAC * (
     1          DNDX(L)*(DXXE*DNDX(N1) + DXYE*DNDY(N1) + DXZE*DNDZ(N1))+
     2          DNDY(L)*(DYXE*DNDX(N1) + DYYE*DNDY(N1) + DYZE*DNDZ(N1))+
     3          DNDZ(L)*(DZXE*DNDX(N1) + DZYE*DNDY(N1) + DZZE*DNDZ(N1))
     4           + N(L)*(VXEP*DNDX(N1) + VYEP*DNDY(N1) + VZEP*DNDZ(N1))
     5            + LAMBDA * (1.0 + RHOBE * KDE / NE) * N(L) * N(N1) )
100           CONTINUE
110         CONTINUE
120       CONTINUE
130     CONTINUE
140   CONTINUE
      RETURN
      END

SUBROUTINE DDBAR2(E, DDE)
C*****************************************************************
C
C       PURPOSE:
C         TO COMPUTE THE DERIVATIVE OF THE CONSISTENT FORM
C         OF THE ELEMENT ADVECTION-DISPESION MATRIX
C         FOR A ONE-DIMENSIONAL, LINEAR BAR ELEMENT.
C
C       DEFINITIONS OF VARIABLES:
C             ALE = LONGITUDINAL DISPERSIVITY FOR ELEMENT
C         DE(I,J) = DERIVATIVE OF THE ELEMENT ADVECTION-
C                   DISPERSION MATRIX
C             DXE = ELEMENT DISPESION COEFFICIENT
C               E = ELEMENT NUMBER
C             KDE = ELEMENT DISTRIBUTION COEFFICIENT
C          LAMBDA = SOLUTE DECAY COEFFICIENT
C              LE = ELEMENT LENGTH
C              NE = ELEMENT POROSITY
C           RHOBE = ELEMENT BULK DENSITY
C             VXE = APPARENT GROUNDWATER VELOCITY IN
C                   X COORDINATE DIRECTION
C            VXEP = PORE WATER VELOCITY IN X COORDINATE DIRECTION
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E
      DOUBLE PRECISION DDE(MAX3,MAX3,3,0:3), KDE, LAMBDA, LE ,NE, ALE
      DOUBLE PRECISION VXEP, DXE, RHOBE
      ALE    = PROP(MATSET(E),1)
      LAMBDA = PROP(MATSET(E),2)
      RHOBE  = PROP(MATSET(E),3)
      KDE    = PROP(MATSET(E),4)
      NE     = PROP(MATSET(E),5)
      VXEP   = 1 / NE
      LE     = ABS(X1(IN(E,2)) - X1(IN(E,1)))
      DXE    = ALE * VXEP
      DDE(1,1,1,0) =  DXE / LE - VXEP / 2.0
      DDE(1,2,1,0) = -DXE / LE + VXEP / 2.0
      DDE(2,1,1,0) = -DXE / LE - VXEP / 2.0
      DDE(2,2,1,0) =  DXE / LE + VXEP / 2.0
      RETURN
      END
```

```
      SUBROUTINE DDTRI3(E,DDE,DIM2)
C*********************************************************************
C
C       PURPOSE:
C         TO COMPUTE THE CONSISTENT FORM OF THE ELEMENT
C         ADVECTION-DISPESION MATRIX FOR A TWO-DIMENSIONAL,
C         LINEAR TRIANGLE ELEMENT
C
C       DEFINITIONS OF VARIABLES:
C             AE4 = FOUR TIMES ELEMENT AREA
C             ALE = LONGITUDINAL DISPERSIVITY FOR ELEMENT
C             ATE = TRANSVERSE DISPERSIVITY FOR ELEMENT
C         DE(I,J) = ELEMENT ADVECTION DISPERSION MATRIX
C     DXXE (ETC.) = ELEMENT DISPESION COEFFICIENTS
C               E = ELEMENT NUMBER
C             KDE = ELEMENT DISTRIBUTION COEFFICIENT
C          LAMBDA = SOLUTE DECAY COEFFICIENT
C              NE = ELEMENT POROSITY
C           RHOBE = ELEMENT BULK DENSITY
C             VXE = APPARENT GROUNDWATER VELOCITY IN
C                   X COORDINATE DIRECTION
C             VYE = APPARENT GROUNDWATER VELOCITY IN
C                   Y COORDINATE DIRECTION
C            VXEP = PORE WATER VELOCITY IN X COORDINATE DIRECTION
C            VYEP = PORE WATER VELOCITY IN Y COORDINATE DIRECTION
C
C*********************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I, J, DIMEN, DIMEN2, DIM2
      DOUBLE PRECISION DDE(MAX3,MAX3,3,0:3), KDE, LAMBDA, NE, BE(3)
      DOUBLE PRECISION ALE, ATE, RHOBE, VXE, VYE, VXEP, VYEP
      DOUBLE PRECISION DXXE, DYYE, DXYE, DYXE, AE4, CE(3)
      DOUBLE PRECISION DXZE, DYZE, DZZE, VZEP
      ALE    = PROP(MATSET(E),1)
      ATE    = PROP(MATSET(E),2)
      LAMBDA = PROP(MATSET(E),3)
      RHOBE  = PROP(MATSET(E),4)
      KDE    = PROP(MATSET(E),5)
      NE     = PROP(MATSET(E),6)
      VXE    = V1(E)
      VYE    = V2(E)
      BE(1)  = X2(IN(E,2)) - X2(IN(E,3))
      BE(2)  = X2(IN(E,3)) - X2(IN(E,1))
      BE(3)  = X2(IN(E,1)) - X2(IN(E,2))
      CE(1)  = X1(IN(E,2)) - X1(IN(E,3))
      CE(2)  = X1(IN(E,3)) - X1(IN(E,1))
      CE(3)  = X1(IN(E,1)) - X1(IN(E,2))
      AE4 = 2.0 * (X1(IN(E,2)) * X2(IN(E,3)) + X1(IN(E,1)) *
     1      X2(IN(E,2)) + X2(IN(E,1)) * X1(IN(E,3)) -
     2      X2(IN(E,3)) * X1(IN(E,1)) - X1(IN(E,3)) *
     3      X2(IN(E,2)) - X1(IN(E,2)) * X2(IN(E,1)))
       DO 40 DIMEN = 1,2
         DO 30 DIMEN2 = 0, 2 * DIM2
           VXEP   = VXE / NE
           VYEP   = VYE / NE
           VZEP   = 0.0
           CALL DERIVE(DXXE, DXYE, DYYE, DXZE, DYZE, DZZE,
     1                 VXEP, VYEP, VZEP, NE, ALE, ATE,
     2                 DIMEN, DIMEN2, 2)
           DYXE   = DXYE
           DO 20 I = 1, 3
             DO 10 J = 1, 3
               DDE(I,J,DIMEN,DIMEN2) = (DXXE * BE(I) * BE(J)
     1                                  + DYYE * CE(I) * CE(J)
     2                                  + DXYE * BE(I) * CE(J)
     3                                  + DYXE * CE(I) * BE(J) ) / AE4
     4                                  + VXEP / 6.0 * BE(J)
     5                                  + VYEP / 6.0 * CE(J)
```

```
 10         CONTINUE
 20       CONTINUE
 30     CONTINUE
 40   CONTINUE
      RETURN
      END

      SUBROUTINE DDREC4(E,DDE,DIM2)
C*****************************************************************
C
C       PURPOSE:
C         TO COMPUTE THE CONSISTENT FORM OF THE ELEMENT
C         ADVECTION-DISPESION MATRIX FOR A TWO-DIMENSIONAL,
C         LINEAR RECTANGLE ELEMENT
C
C       DEFINITIONS OF VARIABLES:
C            ALE = LONGITUDINAL DISPERSIVITY FOR ELEMENT
C            ATE = TRANSVERSE DISPERSIVITY FOR ELEMENT
C         DE(I,J) = ELEMENT ADVECTION DISPERSION MATRIX
C     DXXE (ETC.) = ELEMENT DISPESION COEFFICIENTS
C              E = ELEMENT NUMBER
C            KDE = ELEMENT DISTRIBUTION COEFFICIENT
C         LAMBDA = SOLUTE DECAY COEFFICIENT
C             NE = ELEMENT POROSITY
C          RHOBE = ELEMENT BULK DENSITY
C            VXE = APPARENT GROUNDWATER VELOCITY IN
C                  X COORDINATE DIRECTION
C            VYE = APPARENT GROUNDWATER VELOCITY IN
C                  Y COORDINATE DIRECTION
C           VXEP = PORE WATER VELOCITY IN X COORDINATE DIRECTION
C           VYEP = PORE WATER VELOCITY IN Y COORDINATE DIRECTION
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I, J, DIM2
      DOUBLE PRECISION DDE(MAX3,MAX3,3,0:3), KDE, LAMBDA, NE, AE
      DOUBLE PRECISION ALE, ATE, RHOBE, VXE, VYE, VXEP, VYEP
      DOUBLE PRECISION DXXE, DYYE, DXYE, DYXE, TEMP1, TEMP2, BE
      DOUBLE PRECISION TEMP3, TEMP4, TEMP5, TEMP6
      DOUBLE PRECISION DXZE, DYZE, DZZE, VZEP
      ALE    = PROP(MATSET(E),1)
      ATE    = PROP(MATSET(E),2)
      LAMBDA = PROP(MATSET(E),3)
      RHOBE  = PROP(MATSET(E),4)
      KDE    = PROP(MATSET(E),5)
      NE     = PROP(MATSET(E),6)
      AE     = ABS(X2(IN(E,1)) - X2(IN(E,3)))/ 2.0
      BE     = ABS(X1(IN(E,1)) - X1(IN(E,3)))/ 2.0
      VXE    = V1(E)
      VYE    = V2(E)
      DO 20 I = 1, 2
        DO 10 J = 0, 2 * DIM2
          VXEP   = VXE / NE
          VYEP   = VYE / NE
          VZEP   = 0.0
          CALL DERIVE(DXXE, DXYE, DYYE, DXZE, DYZE, DZZE,
     1                VXEP, VYEP, VZEP, NE, ALE, ATE,
     2                I, J, 2)
          DYXE   = DXYE
          TEMP1  = (DXXE * AE) / (6.0 * BE)
          TEMP2  = (DYYE * BE) / (6.0 * AE)
          TEMP3  = DXYE / 4.0
          TEMP4  = DYXE / 4.0
          TEMP5  = VXEP * AE / 6.0
          TEMP6  = VYEP * BE / 6.0
          DDE(1,1,I,J)= 2.*TEMP1+2.*TEMP2+TEMP3+TEMP4-2.*TEMP5-2.*TEMP6
          DDE(1,2,I,J)=-2.*TEMP1+   TEMP2+TEMP3-TEMP4+2.*TEMP5-   TEMP6
          DDE(1,3,I,J)=-   TEMP1-   TEMP2-TEMP3-TEMP4+   TEMP5+   TEMP6
```

```
          DDE(1,4,I,J)=    TEMP1-2.*TEMP2-TEMP3+TEMP4-   TEMP5+2.*TEMP6
          DDE(2,1,I,J)=-2.*TEMP1+   TEMP2-TEMP3+TEMP4-2.*TEMP5-   TEMP6
          DDE(2,2,I,J)= 2.*TEMP1+2.*TEMP2-TEMP3-TEMP4+2.*TEMP5-2.*TEMP6
          DDE(2,3,I,J)=    TEMP1-2.*TEMP2+TEMP3-TEMP4+   TEMP5+2.*TEMP6
          DDE(2,4,I,J)=-   TEMP1-   TEMP2+TEMP3+TEMP4-   TEMP5+   TEMP6
          DDE(3,1,I,J)=-   TEMP1-   TEMP2-TEMP3-TEMP4-   TEMP5-   TEMP6
          DDE(3,2,I,J)=    TEMP1-2.*TEMP2-TEMP3+TEMP4+   TEMP5-2.*TEMP6
          DDE(3,3,I,J)= 2.*TEMP1+2.*TEMP2+TEMP3+TEMP4+2.*TEMP5+2.*TEMP6
          DDE(3,4,I,J)=-2.*TEMP1+   TEMP2+TEMP3-TEMP4-2.*TEMP5+   TEMP6
          DDE(4,1,I,J)=    TEMP1-2.*TEMP2+TEMP3-TEMP4-   TEMP5-2.*TEMP6
          DDE(4,2,I,J)=-   TEMP1-   TEMP2+TEMP3+TEMP4+   TEMP5-   TEMP6
          DDE(4,3,I,J)=-2.*TEMP1+   TEMP2-TEMP3+TEMP4+2.*TEMP5+   TEMP6
          DDE(4,4,I,J)= 2.*TEMP1+2.*TEMP2-TEMP3-TEMP4-2.*TEMP5+2.*TEMP6
 10     CONTINUE
 20   CONTINUE
      RETURN
      END


      SUBROUTINE DDQUA4(E,DDE,DIM2)
C*****************************************************************
C
C       PURPOSE:
C         TO COMPUTE THE CONSISTENT FORM OF THE ELEMENT SORPTION
C         AND ADVECTION-DISPESION MATRICES FOR A TWO-DIMENSIONAL,
C         LINEAR QUADRILATERAL ELEMENT
C
C       DEFINITIONS OF VARIABLES:
C         AE(I,J) = ELEMENT SORPTION MATRIX
C             ALE = LONGITUDINAL DISPERSIVITY FOR ELEMENT
C             ATE = TRANSVERSE DISPERSIVITY FOR ELEMENT
C         DE(I,J) = ELEMENT ADVECTION DISPERSION MATRIX
C          DETJAC = DETERMINANT OF THE JACOBIAN MATRIX
C        DNDXI(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                   FUNCTION WITH RESPECT TO XI AT NODE I
C         DNDX(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                   FUNCTION WITH RESPECT TO X AT NODE I
C       DNDETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                   FUNCTION WITH RESPECT TO ETA AT NODE I
C         DNDY(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                   FUNCTION WITH RESPECT TO Y AT NODE I
C     DXXE (ETC.) = ELEMENT DISPESION COEFFICIENTS
C               E = ELEMENT NUMBER
C            XI(I) = LOCATION OF GAUSS POINT IN XI COORDINATE DIRECTION
C           ETA(I) = LOCATION OF GAUSS POINT IN ETA COORDINATE DIRECTION
C         JAC(I,J) = JACOBIAN MATRIX
C             N(I) = INTERPOLATION FUNCTION FOR NODE I
C             W(I) = WEIGHT FOR GAUSS POINT I
C              KDE = ELEMENT DISTRIBUTION COEFFICIENT
C           LAMBDA = SOLUTE DECAY COEFFICIENT
C               NE = ELEMENT POROSITY
C            RHOBE = ELEMENT BULK DENSITY
C              VXE = APPARENT GROUNDWATER VELOCITY IN
C                    X COORDINATE DIRECTION
C              VYE = APPARENT GROUNDWATER VELOCITY IN
C                    Y COORDINATE DIRECTION
C             VXEP = PORE WATER VELOCITY IN X COORDINATE DIRECTION
C             VYEP = PORE WATER VELOCITY IN Y COORDINATE DIRECTION
C     X1(IN(E,I)) = X COORDINATE FOR NODE I, ELEMENT E
C     X2(IN(E,I)) = Y COORDINATE FOR NODE I, ELEMENT E
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I, J, K, K1, DIMEN, DIMEN2, DIM2
      DOUBLE PRECISION JAC(2,2), JACINV(2,2), N(4), DETJAC
      DOUBLE PRECISION DNDXI(4), DNDX(4), DNDY(4), DNDETA(4)
      DOUBLE PRECISION W(2), XI(2), ETA(2), DDE(MAX3,MAX3,3,0:3)
      DOUBLE PRECISION SIGN1(4), SIGN2(4), KDE, NE, LAMBDA, VP
```

```
      DOUBLE PRECISION ALE, ATE, RHOBE, VXE, VYE, VXEP, VYEP, VZEP
      DOUBLE PRECISION DXXE, DYYE, DXYE, DYXE, DXZE, DYZE, DZZE
      DOUBLE PRECISION LENG1, LENG2, TEMP1, TEMP2, TEMPA, TEMPB
      DOUBLE PRECISION TEMP3X, TEMP3Y, TEMP4X, TEMP4Y
      DOUBLE PRECISION TEMP5, TEMP6, TEMP7, TEMP8
      DOUBLE PRECISION ALPHA, BETA, WXI(4), WETA(4), DWDXI(4)
      DOUBLE PRECISION DWDETA(4), DWDX(4), DWDY(4)
      DOUBLE PRECISION VXI, VETA, VXW(4), VYW(4)
      DATA SIGN1/-1.0, 1.0, 1.0,-1.0/
      DATA SIGN2/-1.0,-1.0, 1.0, 1.0/
      XI(1) = SQRT(DBLE(1D0/3D0))
      XI(2) = -XI(1)
      ETA(1) = XI(1)
      ETA(2) = XI(2)
      W(1) = 1.0
      W(2) = 1.0
      ALE    = PROP(MATSET(E),1)
      ATE    = PROP(MATSET(E),2)
      LAMBDA = PROP(MATSET(E),3)
      RHOBE  = PROP(MATSET(E),4)
      KDE    = PROP(MATSET(E),5)
      NE     = PROP(MATSET(E),6)
      VXE    = V1(E)
      VYE    = V2(E)
      VXEP   = VXE / NE
      VYEP   = VYE / NE
      VP     = SQRT(VXEP**2 + VYEP**2)
      IF (ABS(VP) .LT. 1E-40) THEN
        ALPHA = 0.0
        BETA = 0.0
      ELSE
        LENG1 = SQRT((X1(IN(E,2)) - X1(IN(E,1)))**2
     1              + (X2(IN(E,2)) - X2(IN(E,1)))**2)
        VXI  = (   (X1(IN(E,2)) - X1(IN(E,1))) * VXEP
     1           + (X2(IN(E,2)) - X2(IN(E,1))) * VYEP ) / LENG1
        LENG2 = SQRT((X1(IN(E,3)) - X1(IN(E,2)))**2
     1              + (X2(IN(E,3)) - X2(IN(E,2)))**2)
        VETA = (   (X1(IN(E,3)) - X1(IN(E,2))) * VXEP
     1           + (X2(IN(E,3)) - X2(IN(E,2))) * VYEP ) / LENG2
C       1E-10 IS CHOSEN TO BE SLIGHTLY MORE THAN DOUBLE PRECISION
        IF (VXI .GT.  VP * 1E-10) THEN
          ALPHA = PROP(MATSET(E),7)
        ELSEIF (VXI .LT. -VP * 1E-10) THEN
          ALPHA = -PROP(MATSET(E),7)
        ELSE
          ALPHA = 0.0
        ENDIF
        IF (VETA .GT. VP * 1E-10) THEN
          BETA = PROP(MATSET(E),7)
        ELSEIF (VETA .LT. -VP * 1E-10) THEN
          BETA = -PROP(MATSET(E),7)
        ELSE
          BETA = 0.0
        ENDIF
      ENDIF
      DO 30 I = 1, 4
        DO 20 J = 1, 4
          DO 10 DIMEN2 = 0, 2 * DIM2
            DDE(I,J,1,DIMEN2) = 0.0
            DDE(I,J,2,DIMEN2) = 0.0
 10       CONTINUE
 20     CONTINUE
 30   CONTINUE
C
      DO 140 I = 1, 2
        DO 130 J = 1, 2
          DO 50 K = 1, 2
            DO 40 K1 = 1, 2
```

```
            JAC(K,K1) = 0.0
40        CONTINUE
50      CONTINUE
        DO 60 K1 = 1, 4
          TEMPA = (1.0 + SIGN1(K1) * XI(I))
          TEMPB = (1.0 + SIGN2(K1) * ETA(J))
          TEMP1 = (1.0 + XI(I))
          TEMP2 = (1.0 + ETA(J))
          TEMP3X = (2.0 + 3.0 * ALPHA  * (1.0 - XI(I)))
     1             * SIGN1(K1)
          TEMP3Y = 2.0 * SIGN1(K1)
          TEMP4X = 2.0 * SIGN2(K1)
          TEMP4Y = (2.0 + 3.0 * BETA  * (1.0 - ETA(J)))
     1             * SIGN2(K1)
          TEMP5 = 2.0 * (1.0 - SIGN1(K1))
          TEMP6 = 2.0 * (1.0 - SIGN2(K1))
          TEMP7 = 3.0 * ALPHA * XI(I) - 1.0
          TEMP8 = 3.0 * BETA * ETA(J) - 1.0
               N(K1) = 0.25 * TEMPA * TEMPB
           DNDXI(K1) = 0.25 * SIGN1(K1) * TEMPB
          DNDETA(K1) = 0.25 * SIGN2(K1) * TEMPA
             WXI(K1) = 0.0625 * (TEMP1 * TEMP3X + TEMP5)
     1                        * (TEMP2 * TEMP4X + TEMP6)
            WETA(K1) = 0.0625 * (TEMP1 * TEMP3Y + TEMP5)
     1                        * (TEMP2 * TEMP4Y + TEMP6)
           DWDXI(K1) = -0.125 * SIGN1(K1) * TEMP7
     1                        * (TEMP2 * TEMP4X + TEMP6)
          DWDETA(K1) = -0.125 * SIGN2(K1) * TEMP8
     1                        * (TEMP1 * TEMP3Y + TEMP5)
60      CONTINUE
        DO 70 K1 = 1, 4
          JAC(1,1) = JAC(1,1) + DNDXI(K1) * X1(IN(E,K1))
          JAC(1,2) = JAC(1,2) + DNDXI(K1) * X2(IN(E,K1))
          JAC(2,1) = JAC(2,1) + DNDETA(K1) * X1(IN(E,K1))
          JAC(2,2) = JAC(2,2) + DNDETA(K1) * X2(IN(E,K1))
70      CONTINUE
        DETJAC = JAC(1,1) * JAC(2,2) - JAC(1,2) * JAC(2,1)
        JACINV(1,1) =  JAC(2,2) / DETJAC
        JACINV(1,2) = -JAC(1,2) / DETJAC
        JACINV(2,1) = -JAC(2,1) / DETJAC
        JACINV(2,2) =  JAC(1,1) / DETJAC
        DO 80 K1 = 1, 4
         DNDX(K1) = JACINV(1,1) * DNDXI(K1) + JACINV(1,2) * DNDETA(K1)
         DNDY(K1) = JACINV(2,1) * DNDXI(K1) + JACINV(2,2) * DNDETA(K1)
         DWDX(K1) = JACINV(1,1) * DWDXI(K1) + JACINV(1,2) * DWDETA(K1)
         DWDY(K1) = JACINV(2,1) * DWDXI(K1) + JACINV(2,2) * DWDETA(K1)
80      CONTINUE
        DO 120 DIMEN = 1,2
          DO 110 DIMEN2 = 0, 2 * DIM2
            VXEP   = VXE / NE
            VYEP   = VYE / NE
            VZEP   = 0.0
            CALL DERIVE(DXXE, DXYE, DYYE, DXZE, DYZE, DZZE,
     1                  VXEP, VYEP, VZEP, NE, ALE, ATE,
     2                 DIMEN, DIMEN2, 2)
            DYXE   = DXYE
            VXI  = (   (X1(IN(E,2)) - X1(IN(E,1))) * VXEP
     1               + (X2(IN(E,2)) - X2(IN(E,1))) * VYEP ) / LENG1
            VETA = (   (X1(IN(E,3)) - X1(IN(E,2))) * VXEP
     1               + (X2(IN(E,3)) - X2(IN(E,2))) * VYEP ) / LENG2
            DO 85 K1 = 1, 4
         VXW(K1) = WXI(K1) * VXI * (X1(IN(E,2)) - X1(IN(E,1))) / LENG1
     1          + WETA(K1) * VETA * (X1(IN(E,3)) - X1(IN(E,2))) / LENG2
         VYW(K1) = WXI(K1) * VXI * (X2(IN(E,2)) - X2(IN(E,1))) / LENG1
     1          + WETA(K1) * VETA * (X2(IN(E,3)) - X2(IN(E,2))) / LENG2
85          CONTINUE
            DO 100 K = 1, 4
              DO 90 K1 = 1, 4
```

```
                  DDE(K,K1,DIMEN,DIMEN2) = DDE(K,K1,DIMEN,DIMEN2)
     1             + W(I)*W(J)* (DXXE * DWDX(K) * DNDX(K1)
     2                         + DXYE * DWDX(K) * DNDY(K1)
     3                         + DYXE * DWDY(K) * DNDX(K1)
     4                         + DYYE * DWDY(K) * DNDY(K1)
C                                SEE ABOVE FOR VXW(K) = W(K) * VXEP
     5                         + VXW(K) * DNDX(K1)
     6                         + VYW(K) * DNDY(K1)
     7                         + LAMBDA * (1.0 + RHOBE * KDE / NE)
     8                         * N(K) * N(K1)) * DETJAC
 90             CONTINUE
100           CONTINUE
110         CONTINUE
120       CONTINUE
130     CONTINUE
140   CONTINUE
      RETURN
      END

      SUBROUTINE DDPAR8(E,DDE,DIM2)
C*****************************************************************
C
C          PURPOSE:
C            TO COMPUTE THE CONSISTENT FORM OF THE ELEMENT
C            SORPTION MATRIX AND THE ELEMENT CONDUCTANCE
C            MATRIX FOR A THREE DIMENSIONAL, LINEAR
C            PARALLELEPIPED ELEMENT
C
C          DEFINITIONS OF VARIABLES:
C               AE(I,J) = ELEMENT CAPACITANCE MATRIX
C                   ALE = LONGITUDINAL DISPERSIVITY FOR ELEMENT
C                   ATE = TRANSVERSE DISPERSIVITY FOR ELEMENT
C                DE(I,J) = ELEMENT ADVECTION DISPERSION MATRIX
C                 DETJAC = DETERMINANT OF JACOBIAN MATRIX
C               DNDXI(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                          FUNCTION WITH RESPECT TO XI AT NODE I
C                DNDX(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                          FUNCTION WITH RESPECT TO X AT NODE I
C              DNDETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                          FUNCTION WITH RESPECT TO ETA AT NODE I
C                DNDY(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                          FUNCTION WITH RESPECT TO Y AT NODE I
C             DNDZETA(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                          FUNCTION WITH RESPECT TO ZETA AT NODE I
C                DNDZ(I) = PARTIAL DERIVATIVE OF INTERPOLATION
C                          FUNCTION WITH RESPECT TO Z AT NODE I
C                     E = ELEMENT NUMBER
C                 ETA(I) = LOCATION OF GAUSS POINT IN ETA
C                          COORDINATE DIRECTION
C                IN(I,J) = NODE NUMBER J FOR ELEMENT I
C               JAC(I,J) = JACOBIAN MATRIX
C             JACINV(I,J) = INVERSE OF JACOBIAN MATRIX
C                    KDE = ELEMENT DISTRIBUTION COEFFICIENT
C                 LAMBDA = SOLUTE DECAY COEFFICIENT
C                   N(I) = INTERPOLATION FUNCTION FOR NODE I
C                     NE = ELEMENT POROSITY
C                  RHOBE = ELEMENT BULK DENSITY
C                    VXE = APPARENT GROUNDWATER VELOCITY IN
C                          X COORDINATE DIRECTION
C                    VYE = APPARENT GROUNDWATER VELOCITY IN
C                          Y COORDINATE DIRECTION
C                    VZE = APPARENT GROUNDWATER VELOCITY IN
C                          Z COORDINATE DIRECTION
C                   VXEP = PORE WATER VELOCITY IN X COORDINATE DIRECTION
C                   VYEP = PORE WATER VELOCITY IN Y COORDINATE DIRECTION
C                   VZEP = PORE WATER VELOCITY IN Z COORDINATE DIRECTION
C                   W(I) = WEIGHT FOR GAUSS POINT I
C             X1(IN(E,I) = X COORDINATE FOR NODE I, ELEMENT E
```

```
C            X2(IN(E,I) = Y COORDINATE FOR NODE I, ELEMENT E
C            X3(IN(E,I) = Z COORDINATE FOR NODE I, ELEMENT E
C                 XI(I) = LOCATION OF GAUSS POINT IN XI COORDINATE
C                         DIRECTION
C               ZETA(I) = LOCATION OF GAUSS POINT IN ZETA COORDINATE
C                         DIRECTION
C
C*****************************************************************
      INCLUDE 'COMALL'
      INTEGER E, I, J, K, L, N1, DIMEN, DIMEN2, DIM2
      DOUBLE PRECISION JAC(3,3), DDE(MAX3,MAX3,3,0:3)
      DOUBLE PRECISION DNDXI(8), DNDX(8), DNDETA(8), DNDY(8)
      DOUBLE PRECISION DNDZETA(8), DNDZ(8), W(2), XI(2), ETA(2)
      DOUBLE PRECISION ZETA(2), SIGN1(8), SIGN2(8), SIGN3(8)
      DOUBLE PRECISION RHOBE, KDE, NE,LAMBDA, JACINV(3,3), DETJAC
      DOUBLE PRECISION VXE, VYE, VZE, VXEP, VYEP, VZEP
      DOUBLE PRECISION DXXE, DXYE, DYXE, DYYE, DXZE, DZXE
      DOUBLE PRECISION DZZE, DYZE, DZYE, ALE, ATE, N(8)
      DATA SIGN1/-1.0, 1.0, 1.0,-1.0,-1.0, 1.0, 1.0,-1.0/
      DATA SIGN2/-1.0,-1.0, 1.0, 1.0,-1.0,-1.0, 1.0, 1.0/
      DATA SIGN3/-1.0,-1.0,-1.0,-1.0, 1.0, 1.0, 1.0, 1.0/

      XI(1) = SQRT(DBLE(1D0/3D0))
      XI(2) = -XI(1)
      ETA(1) = XI(1)
      ETA(2) = XI(2)
      ZETA(1) = XI(1)
      ZETA(2) = XI(2)
      W(1) = 1.0
      W(2) = 1.0
      ALE    = PROP(MATSET(E),1)
      ATE    = PROP(MATSET(E),2)
      LAMBDA = PROP(MATSET(E),3)
      RHOBE  = PROP(MATSET(E),4)
      KDE    = PROP(MATSET(E),5)
      NE     = PROP(MATSET(E),6)
      VXE    = V1(E)
      VYE    = V2(E)
      VZE    = V3(E)
      DO 20 K = 1, 8
        DO 10 N1 = 1, 8
          DDE(K,N1,1,0) = 0.0
          DDE(K,N1,2,0) = 0.0
          DDE(K,N1,3,0) = 0.0
          IF (DIM2 .NE. 0) THEN
            DDE(K,N1,1,1) = 0.0
            DDE(K,N1,2,1) = 0.0
            DDE(K,N1,3,1) = 0.0
            DDE(K,N1,2,2) = 0.0
            DDE(K,N1,3,2) = 0.0
            DDE(K,N1,3,3) = 0.0
          ENDIF
 10     CONTINUE
 20   CONTINUE
      DO 160 I = 1, 2
        DO 150 J = 1, 2
          DO 140 K = 1, 2
            DO 60 L = 1, 3
              DO 50 N1 = 1, 3
                JAC(L,N1) = 0.0
 50           CONTINUE
 60         CONTINUE
            DO 70 N1 = 1, 8
              N(N1)      = 0.125 * (1.0 + SIGN1(N1) * XI(I))
     1                           * (1.0 + SIGN2(N1) * ETA(J))
     2                           * (1.0 + SIGN3(N1) * ZETA(K))
              DNDXI(N1) = 0.125 * SIGN1(N1) * (1.0 + SIGN2(N1) * ETA(J))
     1                  * (1.0 + SIGN3(N1) * ZETA(K))
```

```
            DNDETA(N1) = 0.125 * SIGN2(N1) * (1.0 + SIGN1(N1) * XI(I))
     1                   * (1.0 + SIGN3(N1) * ZETA(K))
            DNDZETA(N1)= 0.125 * SIGN3(N1) * (1.0 + SIGN1(N1) * XI(I))
     1                   * (1.0 + SIGN2(N1) * ETA(J))
 70         CONTINUE
            DO 80 N1 = 1, 8
              JAC(1,1) = JAC(1,1) + DNDXI(N1) * X1(IN(E,N1))
              JAC(1,2) = JAC(1,2) + DNDXI(N1) * X2(IN(E,N1))
              JAC(1,3) = JAC(1,3) + DNDXI(N1) * X3(IN(E,N1))
              JAC(2,1) = JAC(2,1) + DNDETA(N1) * X1(IN(E,N1))
              JAC(2,2) = JAC(2,2) + DNDETA(N1) * X2(IN(E,N1))
              JAC(2,3) = JAC(2,3) + DNDETA(N1) * X3(IN(E,N1))
              JAC(3,1) = JAC(3,1) + DNDZETA(N1) * X1(IN(E,N1))
              JAC(3,2) = JAC(3,2) + DNDZETA(N1) * X2(IN(E,N1))
              JAC(3,3) = JAC(3,3) + DNDZETA(N1) * X3(IN(E,N1))
 80         CONTINUE
            DETJAC = JAC(1,1) * (JAC(2,2)*JAC(3,3) - JAC(3,2)*JAC(2,3))
     1             - JAC(1,2) * (JAC(2,1)*JAC(3,3) - JAC(3,1)*JAC(2,3))
     2             - JAC(1,3) * (JAC(2,1)*JAC(3,2) - JAC(3,1)*JAC(2,2))
           if (detjac .eq. 0) stop 'detjac = 0, check dimension'
C           INVERSE JACOBIAN MATRIX FORMULA HAS BEEN TRANSPOSED STEVE 9/7/96
            JACINV(1,1) = ( JAC(2,2) * JAC(3,3) - JAC(2,3) * JAC(3,2))
     1                    / DETJAC
            JACINV(2,1) = (-JAC(2,1) * JAC(3,3) + JAC(2,3) * JAC(3,1))
     1                    / DETJAC
            JACINV(3,1) = ( JAC(2,1) * JAC(3,2) - JAC(3,1) * JAC(2,2))
     1                    / DETJAC
            JACINV(1,2) = (-JAC(1,2) * JAC(3,3) + JAC(1,3) * JAC(3,2))
     1                    / DETJAC
            JACINV(2,2) = ( JAC(1,1) * JAC(3,3) - JAC(1,3) * JAC(3,1))
     1                    / DETJAC
            JACINV(3,2) = (-JAC(1,1) * JAC(3,2) + JAC(1,2) * JAC(3,1))
     1                    / DETJAC
            JACINV(1,3) = ( JAC(1,2) * JAC(2,3) - JAC(1,3) * JAC(2,2))
     1                    / DETJAC
            JACINV(2,3) = (-JAC(1,1) * JAC(2,3) + JAC(1,3) * JAC(2,1))
     1                    / DETJAC
            JACINV(3,3) = ( JAC(1,1) * JAC(2,2) - JAC(1,2) * JAC(2,1))
     1                    / DETJAC
            DO 90 N1 = 1, 8
              DNDX(N1) = JACINV(1,1) * DNDXI(N1) + JACINV(1,2) *
     1                   DNDETA(N1) + JACINV(1,3) * DNDZETA(N1)
              DNDY(N1) = JACINV(2,1) * DNDXI(N1) + JACINV(2,2) *
     1                   DNDETA(N1) + JACINV(2,3) * DNDZETA(N1)
              DNDZ(N1) = JACINV(3,1) * DNDXI(N1) + JACINV(3,2) *
     1                   DNDETA(N1) + JACINV(3,3) * DNDZETA(N1)
 90         CONTINUE
            DO 130 DIMEN = 1, 3
              DO 120 DIMEN2 = 0, 3 * DIM2
                VXEP   = VXE / NE
                VYEP   = VYE / NE
                VZEP   = VZE / NE
                CALL DERIVE(DXXE, DXYE, DYYE, DXZE, DYZE, DZZE,
     1                      VXEP, VYEP, VZEP, NE, ALE, ATE,
     2                      DIMEN, DIMEN2, 3)
                DYXE = DXYE
                DZXE = DXZE
                DZYE = DYZE
                DO 110 L = 1, 8
                  DO 100 N1 = 1, 8
                    DDE(L,N1,DIMEN,DIMEN2) = DDE(L,N1,DIMEN,DIMEN2) + (
     1            DNDX(L)*(DXXE*DNDX(N1) + DXYE*DNDY(N1) + DXZE*DNDZ(N1))+
     2            DNDY(L)*(DYXE*DNDX(N1) + DYYE*DNDY(N1) + DYZE*DNDZ(N1))+
     3            DNDZ(L)*(DZXE*DNDX(N1) + DZYE*DNDY(N1) + DZZE*DNDZ(N1))
     4             + N(L)*(VXEP*DNDX(N1) + VYEP*DNDY(N1) + VZEP*DNDZ(N1)))
     5               * W(I) * W(J) * W(K) * DETJAC
100               CONTINUE
110             CONTINUE
```

```
120          CONTINUE
130         CONTINUE
140       CONTINUE
150     CONTINUE
160   CONTINUE
      RETURN
      END

      SUBROUTINE DERIVE(DXXE, DXYE, DYYE, DXZE, DYZE, DZZE,
     1                  VXEP, VYEP, VZEP, NE, ALE, ATE,
     2                  DIMEN, DIMEN2, DIM)
C*********************************************************************
C
C     THIS SUBROUTINE DETERMINES THE DERIVATIVES OF THE DISPERSION
C     COEFFICIENT DXXE ETC FOR ALL TWO AND THREE DIMENSIONAL ELEMENTS
C
C*********************************************************************
      IMPLICIT NONE
      DOUBLE PRECISION DXXE, DXYE, DYYE, DXZE, DYZE, DZZE
      DOUBLE PRECISION VXEP, VYEP, VZEP, NE, ALE, ATE, VP
      INTEGER DIMEN, DIMEN2, DIM
      VP     = SQRT(VXEP**2 + VYEP**2 + VZEP**2)
      IF (ABS(VP) .LT. 1E-40) THEN
        DXXE = 0.0
        DYYE = 0.0
        DZZE = 0.0
        DXYE = 0.0
        DXZE = 0.0
        DYZE = 0.0
      ELSEIF (DIMEN2 .EQ. 0) THEN
        IF (DIMEN .EQ. 1) THEN
C         DERIVATIVE WITH RESPECT TO VXE
          DXXE   = (2 * ALE * VXEP / VP
     1              - (ALE * VXEP**2 + ATE * (VYEP**2 + VZEP**2))
     2                * VXEP / VP ** 3)                      / NE
          DYYE   = (2 * ATE * VXEP / VP
     1              - (ALE * VYEP**2 + ATE * (VXEP**2 + VZEP**2))
     2                * VXEP / VP ** 3)                      / NE
          DXYE   = ((ALE  - ATE) * VYEP / VP
     1              - (ALE - ATE) * VXEP ** 2
     2                * VYEP / VP ** 3)                      / NE
          IF (DIM .EQ. 3) THEN
            DXZE   = ((ALE  - ATE) * VZEP / VP
     1                - (ALE - ATE) * VXEP ** 2
     2                * VZEP / VP ** 3)                      / NE
            DYZE   =  - (ALE - ATE) * VXEP * VZEP
     1                * VYEP / VP ** 3                       / NE
            DZZE   = (2 * ATE * VXEP / VP
     1                - (ALE * VZEP**2 + ATE * (VXEP**2 + VYEP**2))
     2                * VXEP / VP ** 3)                      / NE
            VZEP = 0.0
          ENDIF
          VXEP = 1 / NE
          VYEP = 0.0
        ELSEIF (DIMEN .EQ. 2) THEN
C         DERIVATIVE WITH RESPECT TO VYE
          DXXE   = (2 * ATE * VYEP / VP
     1              - (ALE * VXEP**2 + ATE * (VYEP**2 + VZEP**2))
     2                * VYEP / VP ** 3)                      / NE
          DYYE   = (2 * ALE * VYEP / VP
     1              - (ALE * VYEP**2 + ATE * (VXEP**2 + VZEP**2))
     2                * VYEP / VP ** 3)                      / NE
          DXYE   = ((ALE  - ATE) * VXEP / VP
     1              - (ALE - ATE) * VYEP ** 2
     2                * VXEP / VP ** 3)                      / NE
          IF (DIM .EQ. 3) THEN
            DXZE   =  - (ALE - ATE) * VXEP * VZEP
     1                * VYEP / VP ** 3                       / NE
```

```
            DYZE   = ((ALE  - ATE) * VZEP / VP
     1                 - (ALE - ATE) * VYEP ** 2
     2                 * VZEP / VP ** 3)                        / NE
            DZZE   = (2 * ATE * VYEP / VP
     1                 - (ALE * VZEP**2 + ATE * (VXEP**2 + VYEP**2))
     2                 * VYEP / VP ** 3)                        / NE
            VZEP = 0.0
          ENDIF
          VXEP = 0.0
          VYEP = 1 / NE
        ELSEIF (DIMEN .EQ. 3) THEN
C         DERIVATIVE WITH RESPECT TO VZE
          DXXE   = (2 * ATE * VZEP / VP
     1              - (ALE * VXEP**2 + ATE * (VYEP**2 + VZEP**2))
     2                * VZEP / VP ** 3)                        / NE
          DYYE   = (2 * ATE * VZEP / VP
     1              - (ALE * VYEP**2 + ATE * (VXEP**2 + VZEP**2))
     2                * VZEP / VP ** 3)                        / NE
          DZZE   = (2 * ALE * VZEP / VP
     1              - (ALE * VZEP**2 + ATE * (VXEP**2 + VYEP**2))
     2                * VZEP / VP ** 3)                        / NE
          DXYE   =  - (ALE - ATE) * VXEP * VYEP
     1                * VZEP / VP ** 3                         / NE
          DXZE   = ((ALE  - ATE) * VXEP / VP
     1                - (ALE - ATE) * VZEP ** 2
     2                * VXEP / VP ** 3)                        / NE
          DYZE   = ((ALE  - ATE) * VYEP / VP
     1                - (ALE - ATE) * VZEP ** 2
     2                * VYEP / VP ** 3)                        / NE
          VZEP = 1 / NE
          VXEP = 0.0
          VYEP = 0.0
        ENDIF
       ELSEIF (DIMEN2 .EQ. 1 .AND. DIMEN .EQ. 1) THEN
C         SECOND DERIVATIVE WITH RESPECT TO VXE AND VXE
          DXXE = (2 * ALE / VP
     1          - (5 * ALE * VXEP**2 + ATE * (VYEP**2 + VZEP**2)) /VP**3
     2          + 3 * (ALE * VXEP**2 + ATE * (VYEP**2 + VZEP**2))
     3              * VXEP**2 / VP**5                            ) /NE**2
          DXYE = 3 * (ALE - ATE) * VXEP * VYEP
     1              * ((VXEP / VP)**2 - 1) / VP**3                 /NE**2
          DYYE = (2 * ATE / VP
     1          - (ALE * VYEP**2 + ATE * (5 * VXEP**2 + VZEP**2)) /VP**3
     2          + 3 * (ALE * VYEP**2 + ATE * (VXEP**2 + VZEP**2))
     3              * VXEP**2 / VP**5                            ) /NE**2
          IF (DIM .EQ. 3) THEN
            DXZE = 3 * (ALE - ATE) * VXEP * VZEP
     1                * ((VXEP / VP)**2 - 1) / VP**3               /NE**2
            DYZE = (3 * VXEP**2 / VP**2 - 1)
     1                * (ALE - ATE) * VYEP * VZEP / VP**3          /NE**2
            DZZE = (2 * ATE / VP
     1            - (ALE * VZEP**2 + ATE * (5 * VXEP**2 + VYEP**2)) /VP**3
     2            + 3 * (ALE * VZEP**2 + ATE * (VXEP**2 + VYEP**2))
     3                * VXEP**2 / VP**5                          ) /NE**2
            VZEP = 0.0
          ENDIF
          VXEP = 0.0
          VYEP = 0.0
       ELSEIF(DIMEN .EQ. 2 .AND. DIMEN2 .EQ. 1 .OR.
     1        DIMEN .EQ. 1 .AND. DIMEN2 .EQ. 2) THEN
C         SECOND DERIVATIVE WITH RESPECT TO VXE AND VYE
          DXXE = (- 2 * (ALE + ATE) * VXEP * VYEP / VP **3
     1            + 3 * (ALE * VXEP**2 + ATE * (VYEP**2 + VZEP**2))
     2                * VXEP * VYEP / VP**5                      ) /NE**2
          DXYE = ((ALE-ATE) / VP
     1          - (ALE-ATE) * (VXEP**2 + VYEP**2) / VP**3
     2          + 3 * (ALE-ATE) * VXEP**2 * VYEP**2 / VP**5      ) /NE**2
          DYYE = (- 2 * (ALE + ATE) * VXEP * VYEP / VP**3
```

```
     1           + 3 * (ALE * VYEP**2 + ATE * (VXEP**2 + VZEP**2))
     2               * VXEP * VYEP / VP**5                      ) /NE**2
          IF (DIM .EQ. 3) THEN
            DXZE = (3 * VXEP**2 / VP**2 - 1) * (ALE-ATE)
     1              * VYEP * VZEP / VP**3                        /NE**2
            DYZE = (3 * VYEP**2 / VP**2 - 1) * (ALE-ATE)
     1              * VXEP * VZEP / VP**3                        /NE**2
            DZZE = (- 4 * ATE * VXEP * VYEP/ VP**3
     1              + 3 * (ALE * VZEP**2 + ATE * (VXEP**2 + VYEP**2))
     2                  * VXEP * VYEP / VP**5                  ) /NE**2
            VZEP = 0.0
          ENDIF
          VXEP = 0.0
          VYEP = 0.0
      ELSEIF (DIMEN .EQ. 3 .AND. DIMEN2 .EQ. 1 .OR.
     1        DIMEN .EQ. 1 .AND. DIMEN2 .EQ. 3) THEN
C         SECOND DERIVATIVE WITH RESPECT TO VXE AND VZE
          DXXE = (- 2 * (ALE + ATE) * VXEP * VZEP / VP **3
     1            + 3 * (ALE * VXEP**2 + ATE * (VYEP**2 + VZEP**2))
     2                * VXEP * VZEP / VP**5                    ) /NE**2
          DXYE = (3 * VXEP**2 / VP**2 - 1) * (ALE-ATE)
     1              * VYEP * VZEP / VP**3                        /NE**2
          DYYE = (- 4 * ATE * VXEP * VZEP/ VP**3
     1            + 3 * (ALE * VYEP**2 + ATE * (VXEP**2 + VZEP**2))
     2                * VXEP * VZEP / VP**5                    ) /NE**2
          DXZE = ((ALE-ATE) / VP
     1            - (ALE-ATE) * (VXEP**2 + VZEP**2) / VP**3
     2            + 3 * (ALE-ATE) * VXEP**2 * VZEP**2 / VP**5   ) /NE**2
          DYZE = (3 * VZEP**2 / VP**2 - 1) * (ALE-ATE)
     1              * VXEP * VYEP / VP**3                        /NE**2
          DZZE = (- 2 * (ALE + ATE) * VXEP * VZEP / VP**3
     1            + 3 * (ALE * VZEP**2 + ATE * (VXEP**2 + VYEP**2))
     2                * VXEP * VZEP / VP**5                    ) /NE**2
          VXEP = 0.0
          VYEP = 0.0
          VZEP = 0.0
      ELSEIF(DIMEN2 .EQ. 2 .AND. DIMEN .EQ. 2) THEN
C         SECOND DERIVATIVE WITH RESPECT TO VYE AND VYE
          DXXE = (2 * ATE / VP
     1          - (ALE * VXEP**2 + ATE * (5 * VYEP**2 + VZEP**2)) /VP**3
     2          + 3 * (ALE * VXEP**2 + ATE * (VYEP**2 + VZEP**2))
     3              * VYEP**2 / VP**5                          ) /NE**2
          DXYE = 3 * (ALE - ATE) * VXEP * VYEP
     1              * ((VYEP / VP)**2 - 1) / VP**3               /NE**2
          DYYE = (2 * ALE / VP
     1          - (5 * ALE * VYEP**2 + ATE * (VXEP**2 + VZEP**2)) /VP**3
     2          + 3 * (ALE * VYEP**2 + ATE * (VXEP**2 + VZEP**2))
     3              * VYEP**2 / VP**5                          ) /NE**2
          IF (DIM .EQ. 3) THEN
            DXZE = (3 * VYEP**2 / VP**2 - 1)
     1                * (ALE - ATE) * VXEP * VZEP / VP**3        /NE**2
            DYZE = 3 * (ALE - ATE) * VYEP * VZEP
     1                * ((VYEP / VP)**2 - 1) / VP**3             /NE**2
            DZZE = (2 * ATE / VP
     1           - (ALE * VZEP**2 + ATE * (VXEP**2 + 5 * VYEP**2)) /VP**3
     2           + 3 * (ALE * VZEP**2 + ATE * (VXEP**2 + VYEP**2))
     3               * VYEP**2 / VP**5                         ) /NE**2
            VZEP = 0.0
          ENDIF
          VXEP = 0.0
          VYEP = 0.0
      ELSEIF (DIMEN .EQ. 3 .AND. DIMEN2 .EQ. 2 .OR.
     1        DIMEN .EQ. 2 .AND. DIMEN2 .EQ. 3) THEN
C         SECOND DERIVATIVE WITH RESPECT TO VYE AND VZE
          DXXE = (- 4 * ATE * VYEP * VZEP/ VP**3
     1            + 3 * (ALE * VXEP**2 + ATE * (VYEP**2 + VZEP**2))
     2                * VYEP * VZEP / VP**5                    ) /NE**2
          DXYE = (3 * VYEP**2 / VP**2 - 1) * (ALE-ATE)
```

```
     1             * VXEP * VZEP / VP**3                          /NE**2
          DYYE = (- 2 * (ALE + ATE) * VYEP * VZEP / VP**3
     1           + 3 * (ALE * VYEP**2 + ATE * (VXEP**2 + VZEP**2))
     2               * VYEP * VZEP / VP**5                     ) /NE**2
          DXZE = (3 * VZEP**2 / VP**2 - 1) * (ALE-ATE)
     1           * VXEP * VYEP / VP**3                          /NE**2
          DYZE = ((ALE-ATE) / VP
     1         - (ALE-ATE) * (VYEP**2 + VZEP**2) / VP**3
     2         + 3 * (ALE-ATE) * VYEP**2 * VZEP**2 / VP**5     ) /NE**2
          DZZE = (- 2 * (ALE + ATE) * VYEP * VZEP / VP **3
     1           + 3 * (ALE * VZEP**2 + ATE * (VXEP**2 + VYEP**2))
     2                * VYEP * VZEP / VP**5                    ) /NE**2
          VXEP = 0.0
          VYEP = 0.0
          VZEP = 0.0
        ELSEIF (DIMEN2 .EQ. 3 .AND. DIMEN .EQ. 3) THEN
C         SECOND DERIVATIVE WITH RESPECT TO VZE AND VZE
          DXXE = (2 * ATE / VP
     1         - (ALE * VXEP**2 + ATE * (VYEP**2 + 5 * VZEP**2)) /VP**3
     2         + 3 * (ALE * VXEP**2 + ATE * (VYEP**2 + VZEP**2))
     3             * VZEP**2 / VP**5                            ) /NE**2
          DXYE = (3 * VZEP**2 / VP**2 - 1)
     1             * (ALE - ATE) * VXEP * VYEP / VP**3          /NE**2
          DYYE = (2 * ATE / VP
     1         - (ALE * VYEP**2 + ATE * (VXEP**2 + 5 * VZEP**2)) /VP**3
     2         + 3 * (ALE * VYEP**2 + ATE * (VXEP**2 + VZEP**2))
     3             * VZEP**2 / VP**5                            ) /NE**2
          DXZE = 3 * (ALE - ATE) * VXEP * VZEP
     1             * ((VZEP / VP)**2 - 1) / VP**3               /NE**2
          DYZE = 3 * (ALE - ATE) * VYEP * VZEP
     1             * ((VZEP / VP)**2 - 1) / VP**3               /NE**2
          DZZE = (2 * ALE / VP
     1         - (5 * ALE * VZEP**2 + ATE * (VXEP**2 + VYEP**2)) /VP**3
     2         + 3 * (ALE * VZEP**2 + ATE * (VXEP**2 + VYEP**2))
     3             * VZEP**2 / VP**5                            ) /NE**2
          VXEP = 0.0
          VYEP = 0.0
          VZEP = 0.0
        ENDIF
        RETURN
        END
```

Appendix B Verification of Original Deterministic Flow Program

A number of test data files have been created to verify the program. The output files, which include the input data, are presented here.

B.1 Examples 1, 2, 3, 4, 5 and 6

The aim of these examples is to test all subroutines using simple cases for which analytical solutions are available. The domain of these problems is an orthogonal region that has fixed heads at either end, for example see figure 1. Therefore the analytical solution is a linear variation of head with distance from one fixed head boundary to the other.

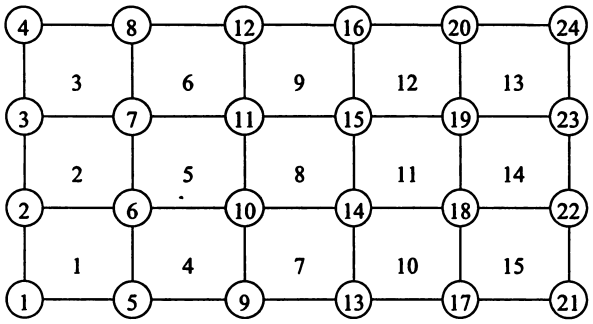


Figure 1. Element mesh used in examples 3 and 4. Example 3 considers the elements to be rectangles, example 4 considers them to be quadrilaterals.

Example 1 tests linear bar elements, example 2 tests triangle elements, example 3 tests rectangular elements, example 4 tests quadrilateral elements, example 5 tests parallelepiped elements and example 6 tests a combination of triangular, rectangular and quadrilateral elements. These examples together use every line in the program except for the parts of LOC,

DECOMP and SOLVE that are used for non symmetric matrices (used in the transport problem) and the lines referring to Neuman boundary conditions.

As can be seen below the exact result determined by the analytical solution is obtained in each example.

B.2 Examples 7, 8, 9, 10 and 11.

The aim of these examples is to test a pumping case. This has two distinguishing features from the examples above that it involves a Neuman boundary condition, thus using the few unused lines in the program referring to the groundwater flow problem, and the solution is not a planar surface, thus the result from the finite element program will only be an approximation and thus can be tested for convergence.

The analytical solution for the pumping case in a confined aquifer is given by the Thiem equation, which when rearranged gives:

$$h_1 - h_2 = \frac{Q \ln\left(r_2 / r_1\right)}{2\pi KD}$$

where

h_1, h_2 = hydraulic heads at radii r_1, r_2

Q = Pumping flow from the well

K = Hydraulic conductivity of the aquifer

D = Depth of the aquifer

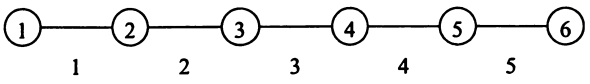
Using the data in the examples h_1-h_2 should be 1.74849576283 when r_1/r_2 is 3. Example 7 has 6 elements around the circumference and gives an answer of 1.44337567, giving a 17 percent error. Example 8 has 200 elements around the circumference and gives an answer of 1.59141853, giving a 9 percent error. Example 9 has 200 elements around the circumference and also has ratios of root three instead of three for the radial lengths of elements in

successive rings of elements. Comparing the same radial ratio it gives an answer of 1.70567724, giving a 2.4 percent error. Clearly the finite element formulation does converge.

Example 10 is used to check quadrilateral elements. Appendix A shows that the nodes on half of the radii have different heads to the other half even though the problem is symetric and the heads should be identical. This is a result of some radii having four elements meeting at their inner ring of nodes and some having three. It is noticed that overall the accuracy is similar to the triangle case, with the two sets of radial results straddling the triangle results.

Example 11 is a three dimensional version of example 10. It used to check that the subroutines for parallelepiped elements and should give the same results as quadrilateral elements. Appendix A shows that this is true.

Example 1: Bar elements using fixed head boundaries.



NODE NUMBER	NODAL COORDINATE X
1	.0000
2	1.0000
3	2.0000
4	3.0000
5	4.0000
6	5.0000

ELEMENT NO.	TYPE	NODE NUMBERS	
1	1	1	2
2	1	2	3
3	1	3	4
4	1	4	5
5	1	5	6

ELEMENT NO.	MATERIAL SET NUMBER
1	1
2	1
3	1
4	1
5	1

MATERIAL SET NO.	MATERIAL PROPERTIES
1	1.000000E-02

NODE NO.	SPECIFIED HYDRAULIC HEAD
1	.0000
6	10.0000

NUMBER OF NODES WITH SPECIFIED HYDRAULIC HEAD = 2

NUMBER OF NODES WITH SPECIFIED GROUNDWATER FLOW = 0

NUMBER OF DEGREES OF FREEDOM IN MODIFIED K MATRIX = 4

SEMI-BANDWIDTH OF MODIFIED K MATRIX = 2

****

COMPUTED VALUES OF HYDRAULIC HEAD

NODE NO.	HYDRAULIC HEAD
1	.00000000*
2	2.00000000
3	4.00000000
4	6.00000000
5	8.00000000
6	10.00000000*

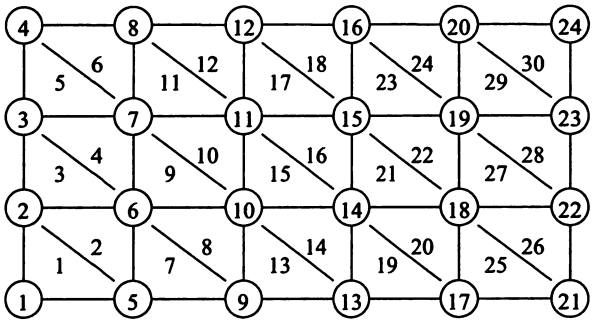
* = SPECIFIED VALUE

****

COMPUTED VALUES OF APPARENT GROUNDWATER VELOCITY

ELEMENT	VX
1	-2.000000E-02
2	-2.000000E-02
3	-2.000000E-02
4	-2.000000E-02
5	-2.000000E-02

Example 2: Orthogonal data with triangles and fixed boundaries.



NODE NUMBER	NODAL COORDINATES X	Y
1	.0000	.0000
2	.0000	2.0000
3	.0000	4.0000
4	.0000	6.0000
5	2.0000	.0000
6	2.0000	2.0000
7	2.0000	4.0000
8	2.0000	6.0000
9	4.0000	.0000
10	4.0000	2.0000
11	4.0000	4.0000
12	4.0000	6.0000
13	6.0000	.0000
14	6.0000	2.0000
15	6.0000	4.0000
16	6.0000	6.0000
17	8.0000	.0000
18	8.0000	2.0000
19	8.0000	4.0000
20	8.0000	6.0000
21	10.0000	.0000
22	10.0000	2.0000
23	10.0000	4.0000
24	10.0000	6.0000

ELEMENT NO.	TYPE	NODE NUMBERS		
1	4	1	5	2
2	4	5	6	2
3	4	2	6	3
4	4	6	7	3
5	4	3	7	4
6	4	7	8	4
7	4	5	9	6
8	4	9	10	6
9	4	6	10	7
10	4	10	11	7
11	4	7	11	8
12	4	11	12	8
13	4	9	13	10
14	4	13	14	10
15	4	10	14	11
16	4	14	15	11
17	4	11	15	12
18	4	15	16	12
19	4	13	17	14
20	4	17	18	14
21	4	14	18	15
22	4	18	19	15
23	4	15	19	16
24	4	19	20	16
25	4	17	21	18
26	4	21	22	18
27	4	18	22	19
28	4	22	23	19
29	4	19	23	20
30	4	23	24	20

ELEMENT NO.	MATERIAL SET NUMBER
1	1

{Each element uses material set 1}

MATERIAL SET NO.	MATERIAL PROPERTIES	
1	1.000000E-02	1.000000E-02

NODE NO.	SPECIFIED HYDRAULIC HEAD
1	.0000
2	.0000
3	.0000
4	.0000
21	10.0000
22	10.0000
23	10.0000
24	10.0000

NUMBER OF NODES WITH SPECIFIED HYDRAULIC HEAD = 8

NUMBER OF NODES WITH SPECIFIED GROUNDWATER FLOW = 0

NUMBER OF DEGREES OF FREEDOM IN MODIFIED K MATRIX = 16

SEMI-BANDWIDTH OF MODIFIED K MATRIX = 5

****

COMPUTED VALUES OF HYDRAULIC HEAD

NODE NO.	HYDRAULIC HEAD
1	.00000000*
2	.00000000*
3	.00000000*
4	.00000000*
5	2.00000000
6	2.00000000
7	2.00000000
8	2.00000000
9	4.00000000
10	4.00000000
11	4.00000000
12	4.00000000
13	6.00000000
14	6.00000000
15	6.00000000
16	6.00000000
17	8.00000000
18	8.00000000
19	8.00000000
20	8.00000000
21	10.00000000*
22	10.00000000*
23	10.00000000*
24	10.00000000*

* = SPECIFIED VALUE

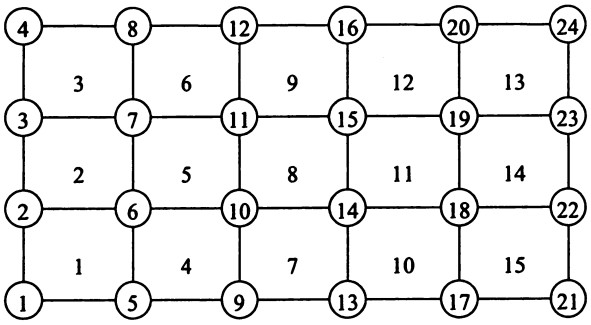
****

COMPUTED VALUES OF APPARENT GROUNDWATER VELOCITY

ELEMENT	VX	VY
1	-1.000000E-02	0.000000E+00

{All elements have the same velocity}

Example 3: Orthogonal data using rectangles and fixed head boundaries.



NODE NUMBER	NODAL COORDINATES X	Y

{Same as for example 2}

ELEMENT NO.	TYPE	NODE NUMBERS			
1	5	1	5	6	2
2	5	2	6	7	3
3	5	3	7	8	4
4	5	5	9	10	6
5	5	6	10	11	7
6	5	7	11	12	8
7	5	9	13	14	10
8	5	10	14	15	11
9	5	11	15	16	12
10	5	13	17	18	14
11	5	14	18	19	15
12	5	15	19	20	16
13	5	17	21	22	18
14	5	18	22	23	19
15	5	19	23	24	20

ELEMENT NO.	MATERIAL SET NUMBER
1	1

{Each element uses material set 1}

MATERIAL SET NO.	MATERIAL PROPERTIES	
1	1.000000E-02	1.000000E-02

NODE NO.	SPECIFIED HYDRAULIC HEAD

{Same as for example 2}

NUMBER OF NODES WITH SPECIFIED HYDRAULIC HEAD = 8

NUMBER OF NODES WITH SPECIFIED GROUNDWATER FLOW = 0

NUMBER OF DEGREES OF FREEDOM IN MODIFIED K MATRIX = 16

SEMI-BANDWIDTH OF MODIFIED K MATRIX = 6

****

COMPUTED VALUES OF HYDRAULIC HEAD

NODE NO.	HYDRAULIC HEAD
1	.00000000*
2	.00000000*
3	.00000000*
4	.00000000*
5	2.00000000
6	2.00000000
7	2.00000000
8	2.00000000
9	4.00000000
10	4.00000000
11	4.00000000
12	4.00000000
13	6.00000000
14	6.00000000
15	6.00000000
16	6.00000000
17	8.00000000
18	8.00000000
19	8.00000000
20	8.00000000
21	10.00000000*
22	10.00000000*
23	10.00000000*
24	10.00000000*

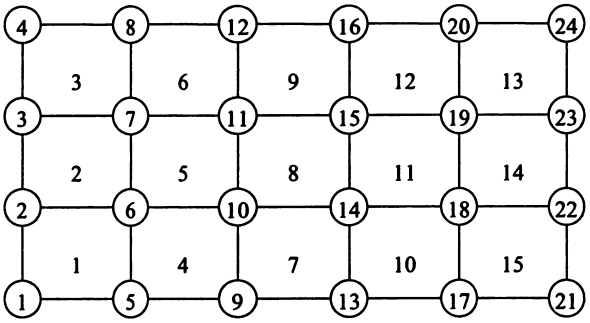
* = SPECIFIED VALUE

****

COMPUTED VALUES OF APPARENT GROUNDWATER VELOCITY

ELEMENT	VX	VY
1	-1.000000E-02	0.000000E+00
2	-1.000000E-02	0.000000E+00
3	-1.000000E-02	0.000000E+00
4	-1.000000E-02	-1.110223E-18
5	-1.000000E-02	-1.110223E-18
6	-1.000000E-02	1.110223E-18
7	-1.000000E-02	-2.220446E-18
8	-1.000000E-02	0.000000E+00
9	-1.000000E-02	0.000000E+00
10	-1.000000E-02	0.000000E+00
11	-1.000000E-02	-4.440892E-18
12	-1.000000E-02	0.000000E+00
13	-1.000000E-02	-4.440892E-18
14	-1.000000E-02	0.000000E+00
15	-1.000000E-02	-8.881784E-18

Example 4: Orthogonal data using quadrilaterals and fixed head boundaries.



NODE NUMBER	NODAL COORDINATES X	Y

{Same as for example 2}

ELEMENT NO.	TYPE	NODE NUMBERS
1	6	1 5 6 2

{Same as for example 3 except that element type is 6 instead of 4}

15	6	19 23 24 20

ELEMENT NO.	MATERIAL SET NUMBER
1	1

{Each element uses material set 1}

MATERIAL SET NO.	MATERIAL PROPERTIES	
1	1.000000E-02	1.000000E-02

NODE NO.	SPECIFIED HYDRAULIC HEAD

{Same as for example 2}

NUMBER OF NODES WITH SPECIFIED HYDRAULIC HEAD = 8

NUMBER OF NODES WITH SPECIFIED GROUNDWATER FLOW = 0

NUMBER OF DEGREES OF FREEDOM IN MODIFIED K MATRIX = 16

SEMI-BANDWIDTH OF MODIFIED K MATRIX = 6

****

COMPUTED VALUES OF HYDRAULIC HEAD

NODE NO.	HYDRAULIC HEAD
1	.00000000*
2	.00000000*
3	.00000000*
4	.00000000*
5	2.00000000
6	2.00000000
7	2.00000000
8	2.00000000
9	4.00000000
10	4.00000000
11	4.00000000
12	4.00000000
13	6.00000000
14	6.00000000
15	6.00000000
16	6.00000000
17	8.00000000
18	8.00000000
19	8.00000000
20	8.00000000
21	10.00000000*
22	10.00000000*
23	10.00000000*
24	10.00000000*

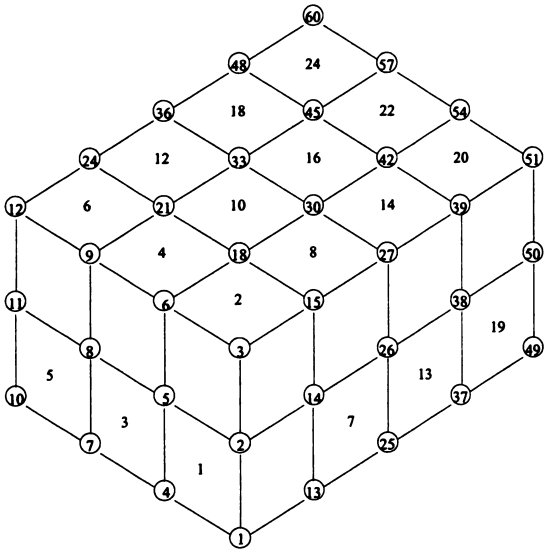
* = SPECIFIED VALUE

****

COMPUTED VALUES OF APPARENT GROUNDWATER VELOCITY

ELEMENT	VX	VY
1	-1.000000E-02	-5.551115E-19
2	-1.000000E-02	0.000000E+00
3	-1.000000E-02	1.665335E-18
4	-1.000000E-02	1.665335E-18
5	-1.000000E-02	-5.551115E-19
6	-1.000000E-02	2.220446E-18
7	-1.000000E-02	1.110223E-18
8	-1.000000E-02	7.771561E-18
9	-1.000000E-02	2.220446E-18
10	-1.000000E-02	0.000000E+00
11	-1.000000E-02	1.110223E-17
12	-1.000000E-02	4.440892E-18
13	-1.000000E-02	0.000000E+00
14	-1.000000E-02	4.440892E-18
15	-1.000000E-02	4.440892E-18

Example 5: Orthogonal data using parallelepipeds and fixed head boundaries.



NODE NUMBER	NODAL COORDINATES X	Y	Z
1	.0000	.0000	.0000
2	.0000	.0000	2.0000
3	.0000	.0000	4.0000
4	.0000	2.0000	.0000
5	.0000	2.0000	2.0000
6	.0000	2.0000	4.0000
7	.0000	4.0000	.0000
8	.0000	4.0000	2.0000
9	.0000	4.0000	4.0000
10	.0000	6.0000	.0000
11	.0000	6.0000	2.0000
12	.0000	6.0000	4.0000
13	2.0000	.0000	.0000
14	2.0000	.0000	2.0000
15	2.0000	.0000	4.0000
16	2.0000	2.0000	.0000
17	2.0000	2.0000	2.0000
18	2.0000	2.0000	4.0000
19	2.0000	4.0000	.0000
20	2.0000	4.0000	2.0000
21	2.0000	4.0000	4.0000
22	2.0000	6.0000	.0000
23	2.0000	6.0000	2.0000
24	2.0000	6.0000	4.0000
25	4.0000	.0000	.0000
26	4.0000	.0000	2.0000
27	4.0000	.0000	4.0000
28	4.0000	2.0000	.0000
29	4.0000	2.0000	2.0000
30	4.0000	2.0000	4.0000
31	4.0000	4.0000	.0000
32	4.0000	4.0000	2.0000
33	4.0000	4.0000	4.0000
34	4.0000	6.0000	.0000
35	4.0000	6.0000	2.0000
36	4.0000	6.0000	4.0000
37	6.0000	.0000	.0000
38	6.0000	.0000	2.0000
39	6.0000	.0000	4.0000
40	6.0000	2.0000	.0000
41	6.0000	2.0000	2.0000
42	6.0000	2.0000	4.0000
43	6.0000	4.0000	.0000
44	6.0000	4.0000	2.0000
45	6.0000	4.0000	4.0000
46	6.0000	6.0000	.0000
47	6.0000	6.0000	2.0000
48	6.0000	6.0000	4.0000
49	8.0000	.0000	.0000
50	8.0000	.0000	2.0000
51	8.0000	.0000	4.0000
52	8.0000	2.0000	.0000
53	8.0000	2.0000	2.0000
54	8.0000	2.0000	4.0000
55	8.0000	4.0000	.0000
56	8.0000	4.0000	2.0000
57	8.0000	4.0000	4.0000
58	8.0000	6.0000	.0000
59	8.0000	6.0000	2.0000
60	8.0000	6.0000	4.0000

ELEMENT NO.	TYPE	NODE NUMBERS							
1	9	1	13	16	4	2	14	17	5
2	9	2	14	17	5	3	15	18	6
3	9	4	16	19	7	5	17	20	8
4	9	5	17	20	8	6	18	21	9
5	9	7	19	22	10	8	20	23	11
6	9	8	20	23	11	9	21	24	12
7	9	13	25	28	16	14	26	29	17
8	9	14	26	29	17	15	27	30	18
9	9	16	28	31	19	17	29	32	20
10	9	17	29	32	20	18	30	33	21
11	9	19	31	34	22	20	32	35	23
12	9	20	32	35	23	21	33	36	24
13	9	25	37	40	28	26	38	41	29
14	9	26	38	41	29	27	39	42	30
15	9	28	40	43	31	29	41	44	32
16	9	29	41	44	32	30	42	45	33
17	9	31	43	46	34	32	44	47	35
18	9	32	44	47	35	33	45	48	36
19	9	37	49	52	40	38	50	53	41
20	9	38	50	53	41	39	51	54	42
21	9	40	52	55	43	41	53	56	44
22	9	41	53	56	44	42	54	57	45
23	9	43	55	58	46	44	56	59	47
24	9	44	56	59	47	45	57	60	48

ELEMENT NO.	MATERIAL SET NUMBER
1	1

{Each element uses material set 1}

MATERIAL SET NO.	MATERIAL PROPERTIES
1	1.000000E-02 1.000000E-02 1.000000E-02

NODE NO.	SPECIFIED HYDRAULIC HEAD
1	.0000
2	.0000
3	.0000
4	.0000
5	.0000
6	.0000
7	.0000
8	.0000
9	.0000
10	.0000
11	.0000
12	.0000
49	8.0000
50	8.0000
51	8.0000
52	8.0000
53	8.0000

54	8.0000
55	8.0000
56	8.0000
57	8.0000
58	8.0000
59	8.0000
60	8.0000

NUMBER OF NODES WITH SPECIFIED
HYDRAULIC HEAD = 24

NUMBER OF NODES WITH SPECIFIED
GROUNDWATER FLOW = 0

NUMBER OF DEGREES OF FREEDOM
IN MODIFIED K MATRIX = 36

SEMI-BANDWIDTH OF MODIFIED K MATRIX = 17

****
****

COMPUTED VALUES OF HYDRAULIC HEAD

NODE NO.	HYDRAULIC HEAD
1	.00000000*
2	.00000000*
3	.00000000*
4	.00000000*
5	.00000000*
6	.00000000*
7	.00000000*
8	.00000000*
9	.00000000*
10	.00000000*
11	.00000000*
12	.00000000*
13	2.00000000
14	2.00000000
15	2.00000000
16	2.00000000
17	2.00000000
18	2.00000000
19	2.00000000
20	2.00000000
21	2.00000000
22	2.00000000
23	2.00000000
24	2.00000000
25	4.00000000
26	4.00000000
27	4.00000000
28	4.00000000
29	4.00000000
30	4.00000000
31	4.00000000
32	4.00000000
33	4.00000000
34	4.00000000
35	4.00000000
36	4.00000000
37	6.00000000
38	6.00000000
39	6.00000000
40	6.00000000
41	6.00000000
42	6.00000000
43	6.00000000
44	6.00000000
45	6.00000000
46	6.00000000
47	6.00000000
48	6.00000000
49	8.00000000*
50	8.00000000*
51	8.00000000*
52	8.00000000*
53	8.00000000*
54	8.00000000*
55	8.00000000*
56	8.00000000*
57	8.00000000*
58	8.00000000*
59	8.00000000*
60	8.00000000*

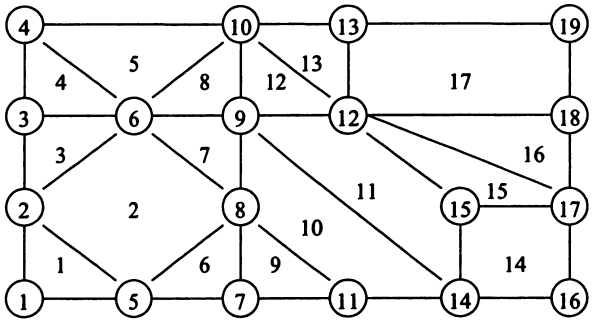
* = SPECIFIED VALUE

****
****

COMPUTED VALUES OF APPARENT GROUNDWATER VELOCITY

ELEMENT	VX	VY	VZ
1	-1.000000E-02	0.000000E+00	0.000000E+00
2	-1.000000E-02	8.326673E-19	4.163336E-19
3	-1.000000E-02	5.551115E-19	-3.238376E-19
4	-1.000000E-02	-8.326673E-19	4.163336E-19
5	-1.000000E-02	1.110223E-18	1.248865E-19
6	-1.000000E-02	2.775558E-18	4.624800E-20
7	-1.000000E-02	1.942890E-18	5.946171E-20
8	-1.000000E-02	-1.665335E-18	2.081668E-19
9	-1.000000E-02	1.110223E-18	-1.888883E-19
10	-1.000000E-02	2.220446E-18	2.774880E-20
11	-1.000000E-02	2.498002E-18	-1.040834E-19
12	-1.000000E-02	2.220446E-18	3.353742E-19
13	-1.000000E-02	-5.551115E-19	7.474219E-20
14	-1.000000E-02	1.665335E-18	-1.982057E-20
15	-1.000000E-02	6.661338E-18	-1.248865E-19
16	-1.000000E-02	2.775558E-18	1.734723E-19
17	-1.000000E-02	2.775558E-18	-2.183651E-19
18	-1.000000E-02	3.885781E-18	4.857226E-19
19	-1.000000E-02	5.551115E-18	-2.410317E-19
20	-1.000000E-02	7.771561E-18	-1.595810E-19
21	-1.000000E-02	0.000000E+00	-1.189573E-19
22	-1.000000E-02	2.220446E-18	-4.099639E-20
23	-1.000000E-02	-1.110223E-18	1.237007E-19
24	-1.000000E-02	6.661338E-18	1.676956E-19

Example 6: Orthogonal data using a mixture of two dimensional elements.



NODE NUMBER	NODAL COORDINATES X	Y
1	.0000	.0000
2	.0000	2.0000
3	.0000	4.0000
4	.0000	6.0000
5	2.0000	.0000
6	2.0000	4.0000
7	4.0000	.0000
8	4.0000	2.0000
9	4.0000	4.0000
10	4.0000	6.0000
11	6.0000	.0000
12	6.0000	4.0000
13	6.0000	6.0000
14	8.0000	.0000
15	8.0000	2.0000
16	10.0000	.0000
17	10.0000	2.0000
18	10.0000	4.0000
19	10.0000	6.0000

ELEMENT NO.	TYPE	NODE NUMBERS			
1	4	1	5	2	
2	6	5	8	6	2
3	4	2	6	3	
4	4	6	4	3	
5	4	6	10	4	
6	4	7	8	5	
7	4	8	9	6	
8	4	9	10	6	
9	4	7	11	8	
10	6	11	14	9	8
11	6	9	14	15	12
12	4	9	12	10	
13	4	12	13	10	
14	5	14	16	17	15
15	4	15	17	12	
16	4	17	18	12	
17	5	12	18	19	13

ELEMENT NO.	MATERIAL SET NUMBER
1	1

{Each element uses material set 1}

MATERIAL SET NO.	MATERIAL PROPERTIES	
1	1.000000E-02	1.000000E-02

{Each element uses material set 1}

NODE NO.	SPECIFIED HYDRAULIC HEAD
1	.0000
2	.0000
3	.0000
4	.0000
16	10.0000
17	10.0000
18	10.0000
19	10.0000

NUMBER OF NODES WITH SPECIFIED HYDRAULIC HEAD = 8

NUMBER OF NODES WITH SPECIFIED GROUNDWATER FLOW = 0

NUMBER OF DEGREES OF FREEDOM IN MODIFIED K MATRIX = 11

SEMI-BANDWIDTH OF MODIFIED K MATRIX = 7

****

COMPUTED VALUES OF HYDRAULIC HEAD

NODE NO.	HYDRAULIC HEAD
1	.00000000*
2	.00000000*
3	.00000000*
4	.00000000*
5	2.00000000
6	2.00000000
7	4.00000000
8	4.00000000
9	4.00000000
10	4.00000000
11	6.00000000
12	6.00000000
13	6.00000000
14	8.00000000
15	8.00000000
16	10.00000000*
17	10.00000000*
18	10.00000000*
19	10.00000000*

* = SPECIFIED VALUE

****

COMPUTED VALUES OF APPARENT GROUNDWATER VELOCITY

ELEMENT	VX	VY
1	-1.000000E-02	0.000000E+00

{All velocities the same as element 1}

Example 7: Pumping data using triangular elements.

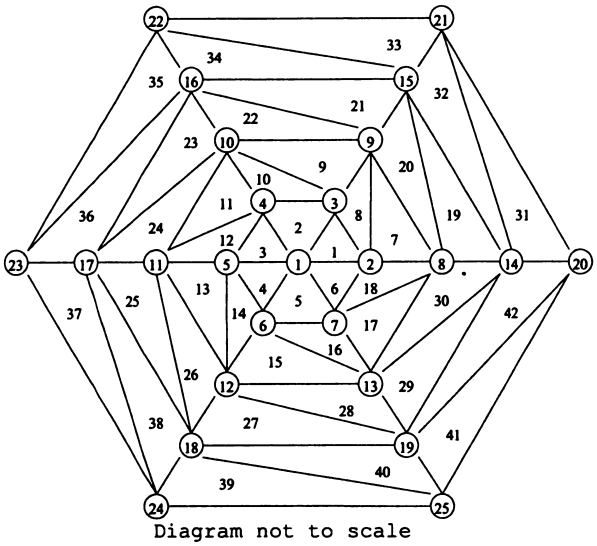


Diagram not to scale

NODE NUMBER	NODAL COORDINATES X	Y
1	.0000	.0000
2	1.0000	.0000
3	.5000	.8660
4	-.5000	.8660
5	-1.0000	.0000
6	-.5000	-.8660
7	.5000	-.8660
8	3.0000	.0000
9	1.5000	2.5981
10	-1.5000	2.5981
11	-3.0000	.0000
12	-1.5000	-2.5981
13	1.5000	-2.5981
14	9.0000	.0000
15	4.5000	7.7942
16	-4.5000	7.7942
17	-9.0000	.0000
18	-4.5000	-7.7942
19	4.5000	-7.7942
20	27.0000	.0000
21	13.5000	23.3827
22	-13.5000	23.3827
23	-27.0000	.0000
24	-13.5000	-23.3827
25	13.5000	-23.3827

ELEMENT NO.	TYPE	NODE NUMBERS		
1	4	1	2	3
2	4	1	3	4
3	4	1	4	5
4	4	1	5	6
5	4	1	6	7
6	4	1	7	2
7	4	2	8	9
8	4	2	9	3
9	4	3	9	10
10	4	3	10	4
11	4	4	10	11
12	4	4	11	5
13	4	5	11	12
14	4	5	12	6
15	4	6	12	13
16	4	6	13	7
17	4	7	13	8
18	4	7	8	2
19	4	8	14	15
20	4	8	15	9
21	4	9	15	16
22	4	9	16	10
23	4	10	16	17
24	4	10	17	11
25	4	11	17	18
26	4	11	18	12
27	4	12	18	19
28	4	12	19	13
29	4	13	19	14
30	4	13	14	8
31	4	14	20	21
32	4	14	21	15
33	4	15	21	22
34	4	15	22	16
35	4	16	22	23
36	4	16	23	17
37	4	17	23	24
38	4	17	24	18
39	4	18	24	25
40	4	18	25	19
41	4	19	25	20
42	4	19	20	14

ELEMENT NO.	MATERIAL SET NUMBER

(Each element uses material set 1)

MATERIAL SET NO.	MATERIAL PROPERTIES	
1	1.000000E-02	1.000000E-02

NODE NO.	SPECIFIED HYDRAULIC HEAD
20	.0000
21	.0000
22	.0000
23	.0000
24	.0000
25	.0000

NUMBER OF NODES WITH SPECIFIED HYDRAULIC HEAD = 6

NODE NO.	SPECIFIED GROUNDWATER FLOW
1	-.1000

NUMBER OF NODES WITH SPECIFIED GROUNDWATER FLOW = 1

NUMBER OF DEGREES OF FREEDOM IN MODIFIED K MATRIX = 19

SEMI-BANDWIDTH OF MODIFIED K MATRIX = 12

****

COMPUTED VALUES OF HYDRAULIC HEAD

NODE NO.	HYDRAULIC HEAD
1	-7.21687836
2	-4.33012702
3	-4.33012702

4	-4.33012702
5	-4.33012702
6	-4.33012702
7	-4.33012702
8	-2.88675135
9	-2.88675135
10	-2.88675135
11	-2.88675135
12	-2.88675135
13	-2.88675135
14	-1.44337567
15	-1.44337567
16	-1.44337567
17	-1.44337567
18	-1.44337567
19	-1.44337567
20	.00000000*
21	.00000000*
22	.00000000*
23	.00000000*
24	.00000000*
25	.00000000*

* = SPECIFIED VALUE

****

COMPUTED VALUES OF APPARENT GROUNDWATER VELOCITY

ELEMENT	VX	VY
1	-2.886751E-02	-1.666667E-02
2	8.881784E-18	-3.333333E-02
3	2.886751E-02	-1.666667E-02
4	2.886751E-02	1.666667E-02
5	2.664535E-17	3.333333E-02
6	-2.886751E-02	1.666667E-02
7	-7.216878E-03	-4.166667E-03
8	-7.216878E-03	-4.166667E-03
9	8.881784E-18	-8.333333E-03
10	0.000000E+00	-8.333333E-03
11	7.216878E-03	-4.166667E-03
12	7.216878E-03	-4.166667E-03
13	7.216878E-03	4.166667E-03
14	7.216878E-03	4.166667E-03
15	2.664535E-17	8.333333E-03
16	2.220446E-18	8.333333E-03
17	-7.216878E-03	4.166667E-03
18	-7.216878E-03	4.166667E-03
19	-2.405626E-03	-1.388889E-03
20	-2.405626E-03	-1.388889E-03
21	0.000000E+00	-2.777778E-03
22	8.258140E-19	-2.777778E-03
23	2.405626E-03	-1.388889E-03
24	2.405626E-03	-1.388889E-03
25	2.405626E-03	1.388889E-03
26	2.405626E-03	1.388889E-03
27	1.110223E-18	2.777778E-03
28	-5.345518E-19	2.777778E-03
29	-2.405626E-03	1.388889E-03
30	-2.405626E-03	1.388889E-03
31	-8.018754E-04	-4.629630E-04
32	-8.018754E-04	-4.629630E-04
33	5.551115E-19	-9.259259E-04
34	0.000000E+00	-9.259259E-04
35	8.018754E-04	-4.629630E-04
36	8.018754E-04	-4.629630E-04
37	8.018754E-04	4.629630E-04
38	8.018754E-04	4.629630E-04
39	2.775558E-19	9.259259E-04
40	-3.655055E-19	9.259259E-04
41	-8.018754E-04	4.629630E-04
42	-8.018754E-04	4.629630E-04

Example 8: Pumping data using triangular elements with 200 nodes around the circumference, radius ratio = 3.

This example is similar to example 7 except that the flow field is a polygon with 200 sides instead of a hexagon.

NODE NUMBER	NODAL COORDINATES X	Y
1	.0000	.0000
2	1.0000	.0000
202	3.0000	.0000
402	9.0000	.0000
602	27.0000	.0000

****

COMPUTED VALUES OF HYDRAULIC HEAD

NODE NO.	HYDRAULIC HEAD
1	-7.95709265
2	-4.77425559
202	-3.18283706
402	-1.59141853
602	.00000000*

* = SPECIFIED VALUE

Example 9: Pumping data using triangular elements with 200 nodes around the circumference, radius ratio = root 3.

This example is similar to example 8 except that 7 rings of elements instead of 4 model the flow field.

NODE NUMBER	NODAL COORDINATES X	Y
1	.0000	.0000
2	1.0000	.0000
202	1.7321	.0000
402	3.0000	.0000
602	5.1962	.0000
802	9.0000	.0000
1002	15.5885	.0000
1202	27.0000	.0000

****

COMPUTED VALUES OF HYDRAULIC HEAD

NODE NO.	HYDRAULIC HEAD
1	-8.29986877
2	-5.11703172
202	-4.26419310
402	-3.41135448
602	-2.55851586
802	-1.70567724
1002	-.85283862
1202	.00000000*

* = SPECIFIED VALUE

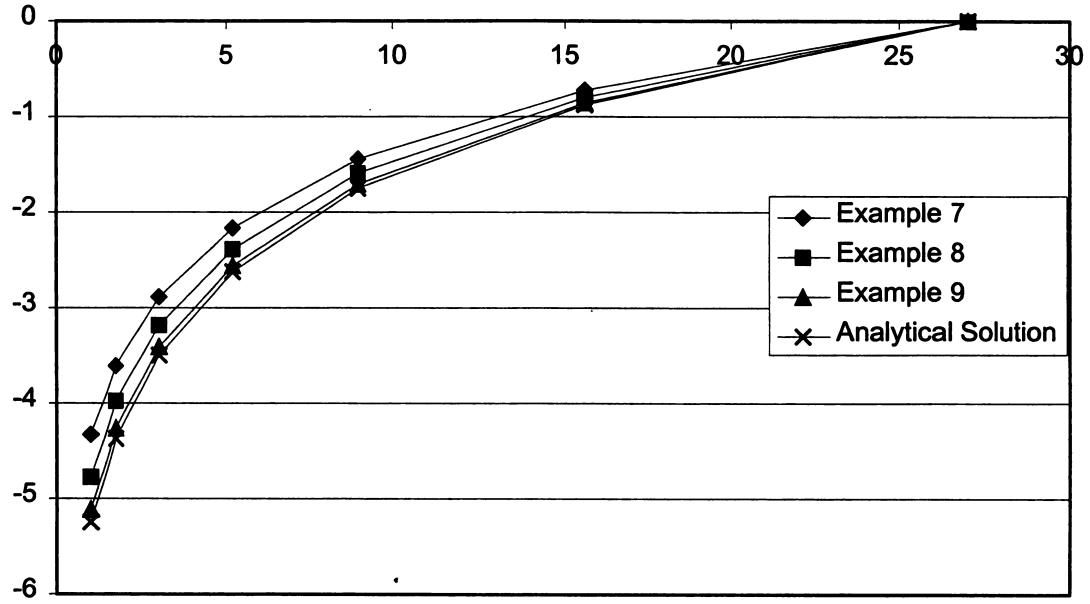
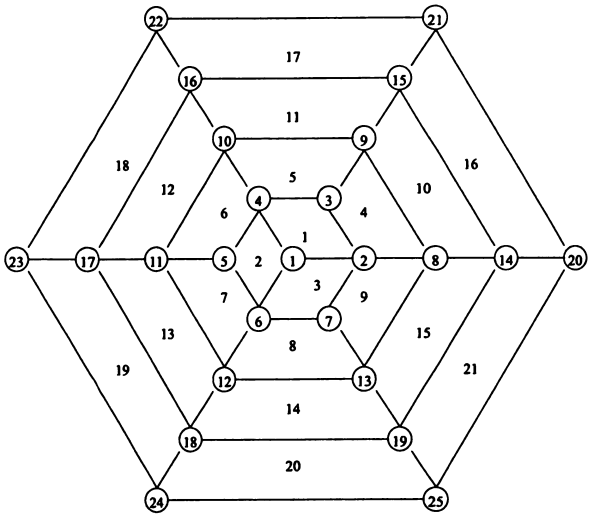


Chart showing relationship between draw down and mesh resolution.

Example 9 gives the highest mesh resolution and is much closer to the analytical solution given by the Theim equation.

Example 10: Pumping data using quadrilateral elements.



NODE NUMBER	NODAL COORDINATES X	Y

{Same as for example 7}

ELEMENT NO.	TYPE	NODE NUMBERS			
1	6	1	2	3	4
2	6	1	4	5	6
3	6	1	6	7	2
4	6	2	8	9	3
5	6	3	9	10	4
6	6	4	10	11	5
7	6	5	11	12	6
8	6	6	12	13	7
9	6	7	13	8	2
10	6	8	14	15	9
11	6	9	15	16	10
12	6	10	16	17	11
13	6	11	17	18	12
14	6	12	18	19	13
15	6	13	19	14	8
16	6	14	20	21	15
17	6	15	21	22	16
18	6	16	22	23	17
19	6	17	23	24	18
20	6	18	24	25	19
21	6	19	25	20	14

ELEMENT NO.	MATERIAL SET NUMBER

{Each element uses material set 1}

MATERIAL SET NO.	MATERIAL PROPERTIES	
1	1.000000E-02	1.000000E-02

NODE NO.	SPECIFIED HYDRAULIC HEAD
20	10.0000
21	10.0000
22	10.0000
23	10.0000
24	10.0000
25	10.0000

NUMBER OF NODES WITH SPECIFIED HYDRAULIC HEAD = 6

NODE NO.	SPECIFIED GROUNDWATER FLOW
1	-.1000

NUMBER OF NODES WITH SPECIFIED GROUNDWATER FLOW = 1

NUMBER OF DEGREES OF FREEDOM IN MODIFIED K MATRIX = 19

SEMI-BANDWIDTH OF MODIFIED K MATRIX = 12

****

COMPUTED VALUES OF HYDRAULIC HEAD

NODE NO.	HYDRAULIC HEAD
1	-7.51235921
2	-4.20894770
3	-4.45130634
4	-4.20894770
5	-4.45130634
6	-4.20894770
7	-4.45130634
8	-2.89752754
9	-2.87597516
10	-2.89752754
11	-2.87597516
12	-2.89752754
13	-2.87597516
14	-1.44242483
15	-1.44432651
16	-1.44242483
17	-1.44432651
18	-1.44242483
19	-1.44432651
20	.00000000*
21	.00000000*
22	.00000000*
23	.00000000*
24	.00000000*
25	.00000000*

* = SPECIFIED VALUE

****

COMPUTED VALUES OF APPARENT GROUNDWATER VELOCITY

ELEMENT	VX	VY
1	-1.530526E-02	-2.650950E-02
2	3.061053E-02	1.908196E-19
3	-1.530526E-02	2.650950E-02
4	-7.492886E-03	-3.688607E-03
5	5.520157E-04	-8.333333E-03
6	6.940871E-03	-4.644726E-03
7	6.940871E-03	4.644726E-03
8	5.520157E-04	8.333333E-03
9	-7.492886E-03	3.688607E-03
10	-2.397438E-03	-1.403071E-03
11	-1.637558E-05	-2.777778E-03
12	2.413814E-03	-1.374707E-03
13	2.413814E-03	1.374707E-03
14	-1.637558E-05	2.777778E-03
15	-2.397438E-03	1.403071E-03
16	-8.021395E-04	-4.625055E-04
17	5.282446E-07	-9.259259E-04
18	8.016113E-04	-4.634204E-04
19	8.016113E-04	4.634204E-04
20	5.282446E-07	9.259259E-04
21	-8.021395E-04	4.625055E-04

Example 11: Pumping data using parallelepiped elements.

This example is a three dimensional version of example 10. It has two layers of elements and three layers of nodes.

NODE NUMBER	NODAL COORDINATES X	Y	Z
1	.0000	.0000	.0000
2	.0000	.0000	2.0000
3	.0000	.0000	4.0000
4	1.0000	.0000	.0000
5	1.0000	.0000	2.0000
6	1.0000	.0000	4.0000
7	.5000	.8660	.0000
8	.5000	.8660	2.0000
9	.5000	.8660	4.0000
10	-.5000	.8660	.0000
11	-.5000	.8660	2.0000
12	-.5000	.8660	4.0000
13	-1.0000	.0000	.0000
14	-1.0000	.0000	2.0000
15	-1.0000	.0000	4.0000
16	-.5000	-.8660	.0000
17	-.5000	-.8660	2.0000
18	-.5000	-.8660	4.0000
19	.5000	-.8660	.0000
20	.5000	-.8660	2.0000
21	.5000	-.8660	4.0000
22	3.0000	.0000	.0000
23	3.0000	.0000	2.0000
24	3.0000	.0000	4.0000
25	1.5000	2.5981	.0000
26	1.5000	2.5981	2.0000
27	1.5000	2.5981	4.0000
28	-1.5000	2.5981	.0000
29	-1.5000	2.5981	2.0000
30	-1.5000	2.5981	4.0000
31	-3.0000	.0000	.0000
32	-3.0000	.0000	2.0000
33	-3.0000	.0000	4.0000
34	-1.5000	-2.5981	.0000
35	-1.5000	-2.5981	2.0000
36	-1.5000	-2.5981	4.0000
37	1.5000	-2.5981	.0000
38	1.5000	-2.5981	2.0000
39	1.5000	-2.5981	4.0000
40	9.0000	.0000	.0000
41	9.0000	.0000	2.0000
42	9.0000	.0000	4.0000
43	4.5000	7.7942	.0000
44	4.5000	7.7942	2.0000
45	4.5000	7.7942	4.0000
46	-4.5000	7.7942	.0000
47	-4.5000	7.7942	2.0000
48	-4.5000	7.7942	4.0000
49	-9.0000	.0000	.0000
50	-9.0000	.0000	2.0000
51	-9.0000	.0000	4.0000
52	-4.5000	-7.7942	.0000
53	-4.5000	-7.7942	2.0000
54	-4.5000	-7.7942	4.0000
55	4.5000	-7.7942	.0000
56	4.5000	-7.7942	2.0000
57	4.5000	-7.7942	4.0000
58	27.0000	.0000	.0000
59	27.0000	.0000	2.0000
60	27.0000	.0000	4.0000
61	13.5000	23.3827	.0000
62	13.5000	23.3827	2.0000
63	13.5000	23.3827	4.0000
64	-13.5000	23.3827	.0000
65	-13.5000	23.3827	2.0000
66	-13.5000	23.3827	4.0000
67	-27.0000	.0000	.0000
68	-27.0000	.0000	2.0000
69	-27.0000	.0000	4.0000
70	-13.5000	-23.3827	.0000
71	-13.5000	-23.3827	2.0000
72	-13.5000	-23.3827	4.0000
73	13.5000	-23.3827	.0000
74	13.5000	-23.3827	2.0000
75	13.5000	-23.3827	4.0000

ELEMENT NO.	TYPE	NODE NUMBERS							
1	9	1	4	7	10	2	5	8	11
2	9	2	5	8	11	3	6	9	12
3	9	1	10	13	16	2	11	14	17
4	9	2	11	14	17	3	12	15	18
5	9	1	16	19	4	2	17	20	5
6	9	2	17	20	5	3	18	21	6
7	9	4	22	25	7	5	23	26	8
8	9	5	23	26	8	6	24	27	9
9	9	7	25	28	10	8	26	29	11
10	9	8	26	29	11	9	27	30	12
11	9	10	28	31	13	11	29	32	14
12	9	11	29	32	14	12	30	33	15
13	9	13	31	34	16	14	32	35	17
14	9	14	32	35	17	15	33	36	18
15	9	16	34	37	19	17	35	38	20
16	9	17	35	38	20	18	36	39	21
17	9	19	37	22	4	20	38	23	5
18	9	20	38	23	5	21	39	24	6
19	9	22	40	43	25	23	41	44	26
20	9	23	41	44	26	24	42	45	27
21	9	25	43	46	28	26	44	47	29
22	9	26	44	47	29	27	45	48	30
23	9	28	46	49	31	29	47	50	32
24	9	29	47	50	32	30	48	51	33
25	9	31	49	52	34	32	50	53	35
26	9	32	50	53	35	33	51	54	36
27	9	34	52	55	37	35	53	56	38
28	9	35	53	56	38	36	54	57	39
29	9	37	55	40	22	38	56	41	23
30	9	38	56	41	23	39	57	42	24
31	9	40	58	61	43	41	59	62	44
32	9	41	59	62	44	42	60	63	45
33	9	43	61	64	46	44	62	65	47
34	9	44	62	65	47	45	63	66	48
35	9	46	64	67	49	47	65	68	50
36	9	47	65	68	50	48	66	69	51
37	9	49	67	70	52	50	68	71	53
38	9	50	68	71	53	51	69	72	54
39	9	52	70	73	55	53	71	74	56
40	9	53	71	74	56	54	72	75	57
41	9	55	73	58	40	56	74	59	41
42	9	56	74	59	41	57	75	60	42

ELEMENT NO.	MATERIAL SET NUMBER
1	1

{Each element uses material set 1}

MATERIAL SET NO.	MATERIAL PROPERTIES
1	1.000000E-02 1.000000E-02 1.000000E-02

NODE NO.	SPECIFIED HYDRAULIC HEAD
58	0.0000
59	0.0000
60	0.0000
61	0.0000
62	0.0000
63	0.0000
64	0.0000
65	0.0000
66	0.0000
67	0.0000
68	0.0000
69	0.0000
70	0.0000
71	0.0000
72	0.0000
73	0.0000
74	0.0000
75	0.0000

NUMBER OF NODES WITH SPECIFIED HYDRAULIC HEAD = 18

NODE NO.	SPECIFIED GROUNDWATER FLOW
1	-.1000
2	-.2000
3	-.1000

NUMBER OF NODES WITH SPECIFIED GROUNDWATER FLOW = 3

NUMBER OF DEGREES OF FREEDOM
IN MODIFIED K MATRIX = 57

SEMI-BANDWIDTH OF MODIFIED K MATRIX = 35

****

COMPUTED VALUES OF HYDRAULIC HEAD

NODE NO.	HYDRAULIC HEAD
1	-7.51235921
2	-7.51235921
3	-7.51235921
4	-4.20894770
5	-4.20894770
6	-4.20894770
7	-4.45130634
8	-4.45130634
9	-4.45130634
10	-4.20894770
11	-4.20894770
12	-4.20894770
13	-4.45130634
14	-4.45130634
15	-4.45130634
16	-4.20894770
17	-4.20894770
18	-4.20894770
19	-4.45130634
20	-4.45130634
21	-4.45130634
22	-2.89752754
23	-2.89752754
24	-2.89752754
25	-2.87597516
26	-2.87597516
27	-2.87597516
28	-2.89752754
29	-2.89752754
30	-2.89752754
31	-2.87597516
32	-2.87597516
33	-2.87597516
34	-2.89752754
35	-2.89752754
36	-2.89752754
37	-2.87597516
38	-2.87597516
39	-2.87597516
40	-1.44242483
41	-1.44242483
42	-1.44242483
43	-1.44432651
44	-1.44432651
45	-1.44432651
46	-1.44242483
47	-1.44242483
48	-1.44242483
49	-1.44432651
50	-1.44432651
51	-1.44432651
52	-1.44242483
53	-1.44242483
54	-1.44242483
55	-1.44432651
56	-1.44432651
57	-1.44432651
58	.00000000*
59	.00000000*
60	.00000000*
61	.00000000*
62	.00000000*
63	.00000000*
64	.00000000*
65	.00000000*
66	.00000000*
67	.00000000*
68	.00000000*
69	.00000000*
70	.00000000*
71	.00000000*
72	.00000000*
73	.00000000*
74	.00000000*
75	.00000000*

* = SPECIFIED VALUE

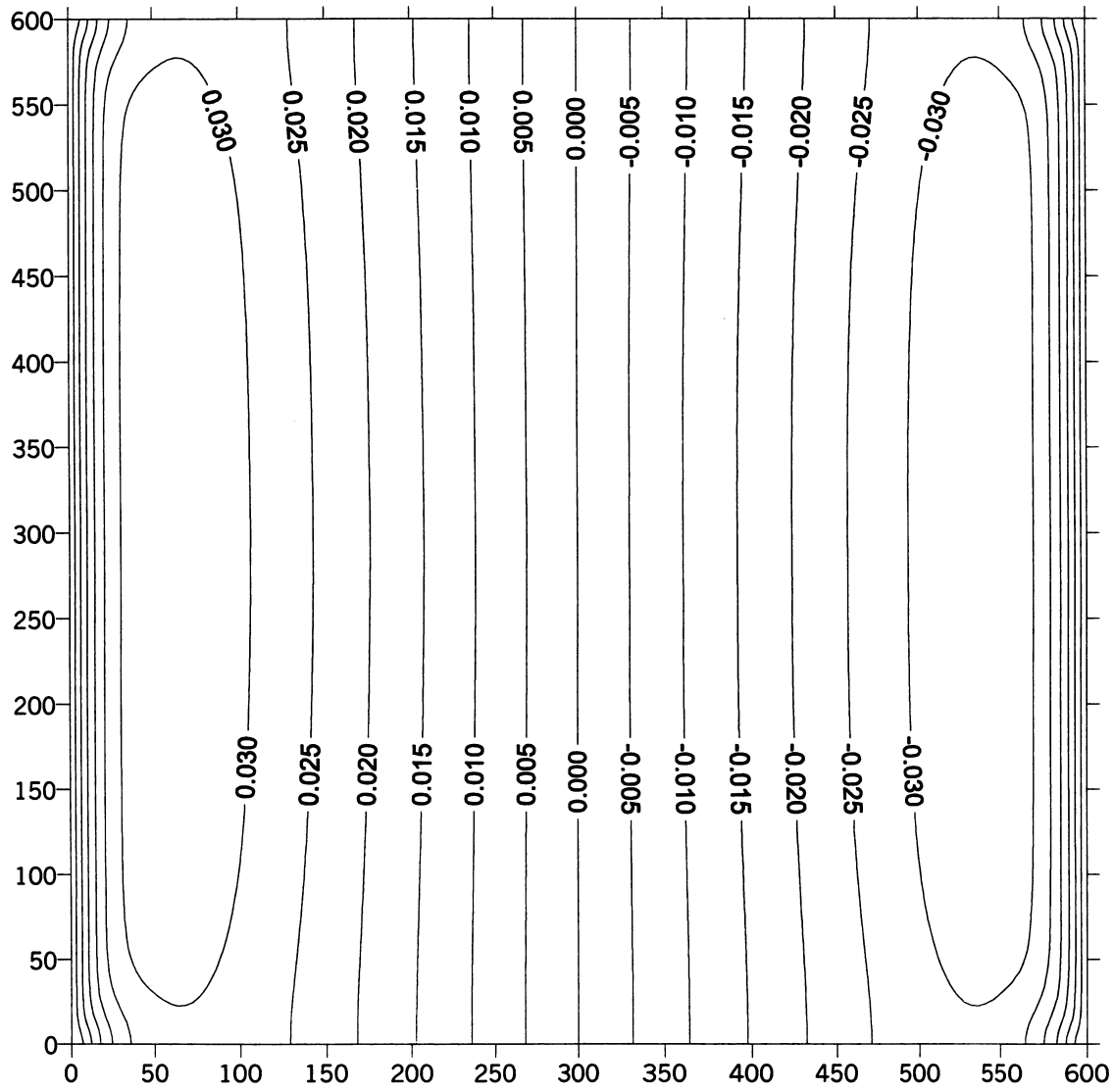
****
******

COMPUTED VALUES OF APPARENT GROUNDWATER VELOCITY

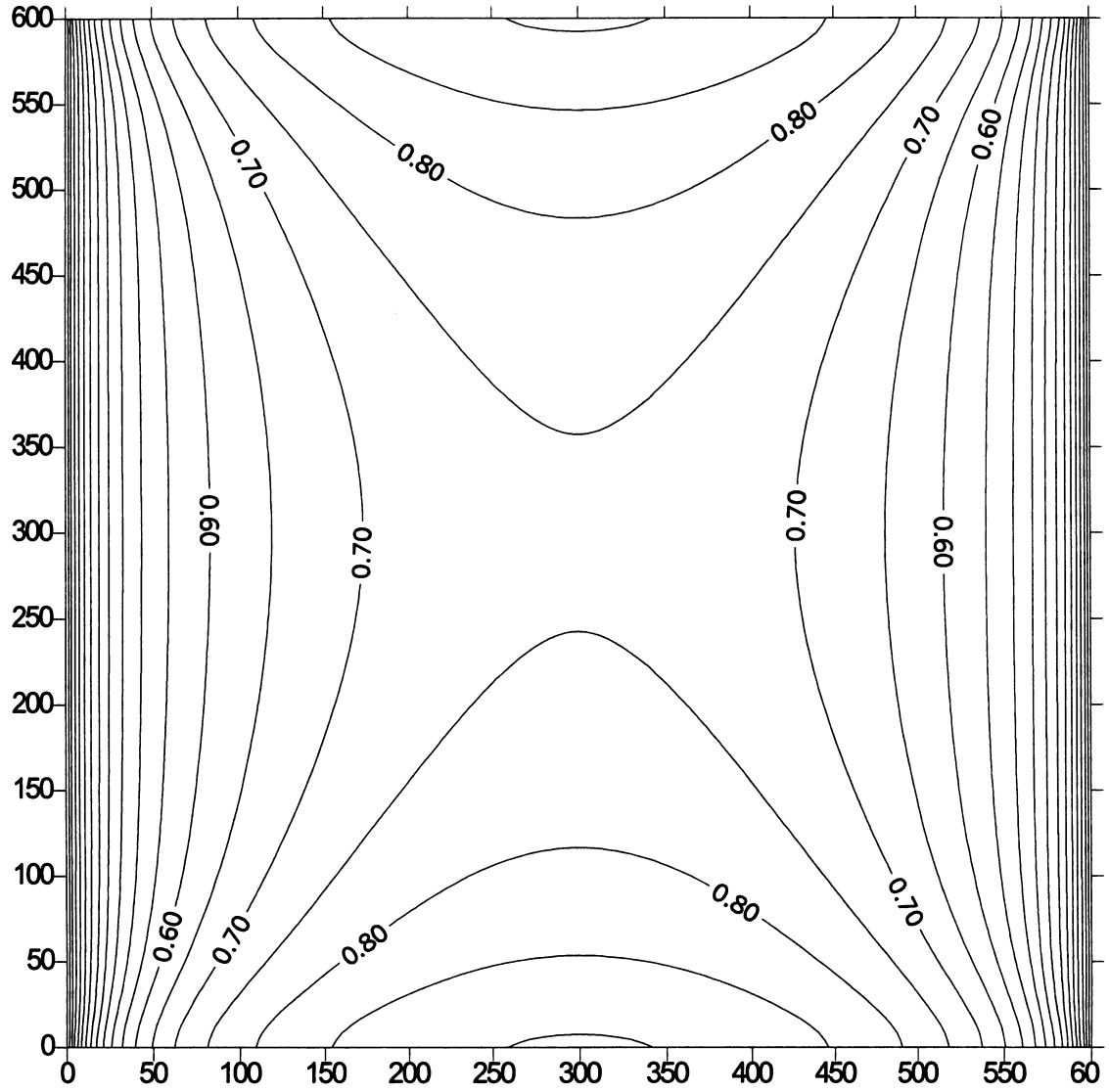
ELEMENT	VX	VY	VZ
1	-1.530526E-02	-2.650950E-02	0.000000E+00
2	-1.530526E-02	-2.650950E-02	3.885781E-18
3	3.061053E-02	7.542795E-18	3.987967E-18
4	3.061053E-02	4.978656E-18	5.551115E-19
5	-1.530526E-02	2.650950E-02	6.757290E-19
6	-1.530526E-02	2.650950E-02	3.286488E-19
7	-7.492886E-03	-3.688607E-03	7.092715E-19
8	-7.492886E-03	-3.688607E-03	9.714451E-19
9	5.520157E-04	-8.333333E-03	-1.404042E-19
10	5.520157E-04	-8.333333E-03	1.236600E-18
11	6.940871E-03	-4.644726E-03	-3.834010E-19
12	6.940871E-03	-4.644726E-03	1.281087E-18
13	6.940871E-03	4.644726E-03	1.057978E-18
14	6.940871E-03	4.644726E-03	-7.043926E-20
15	5.520157E-04	8.333333E-03	4.875183E-19
16	5.520157E-04	8.333333E-03	-5.763890E-19
17	-7.492886E-03	3.688607E-03	-1.328825E-19
18	-7.492886E-03	3.688607E-03	-5.478609E-20
19	-2.397438E-03	-1.403071E-03	4.021712E-19
20	-2.397438E-03	-1.403071E-03	3.361027E-19
21	-1.637558E-05	-2.777778E-03	-5.262107E-19
22	-1.637558E-05	-2.777778E-03	-2.208384E-19
23	2.413814E-03	-1.374707E-03	-3.754050E-19
24	2.413814E-03	-1.374707E-03	2.750147E-19
25	2.413814E-03	1.374707E-03	-4.679010E-20
26	2.413814E-03	1.374707E-03	-5.922454E-20
27	-1.637558E-05	2.777778E-03	-3.647324E-19
28	-1.637558E-05	2.777778E-03	-8.378850E-20
29	-2.397438E-03	1.403071E-03	-1.285457E-19
30	-2.397438E-03	1.403071E-03	-1.971215E-19
31	-8.021395E-04	-4.625055E-04	4.763713E-20
32	-8.021395E-04	-4.625055E-04	0.000000E+00
33	5.282446E-07	-9.259259E-04	-4.316480E-19
34	5.282446E-07	-9.259259E-04	-1.303414E-19
35	8.016113E-04	-4.634204E-04	1.407769E-20
36	8.016113E-04	-4.634204E-04	-1.555322E-19
37	8.016113E-04	4.634204E-04	1.761829E-19
38	8.016113E-04	4.634204E-04	-1.054895E-19
39	5.282446E-07	9.259259E-04	-3.896521E-19
40	5.282446E-07	9.259259E-04	-1.002887E-20
41	-8.021395E-04	4.625055E-04	-8.895540E-20
42	-8.021395E-04	4.625055E-04	1.428775E-19

Appendix C Hydraulic Head Results

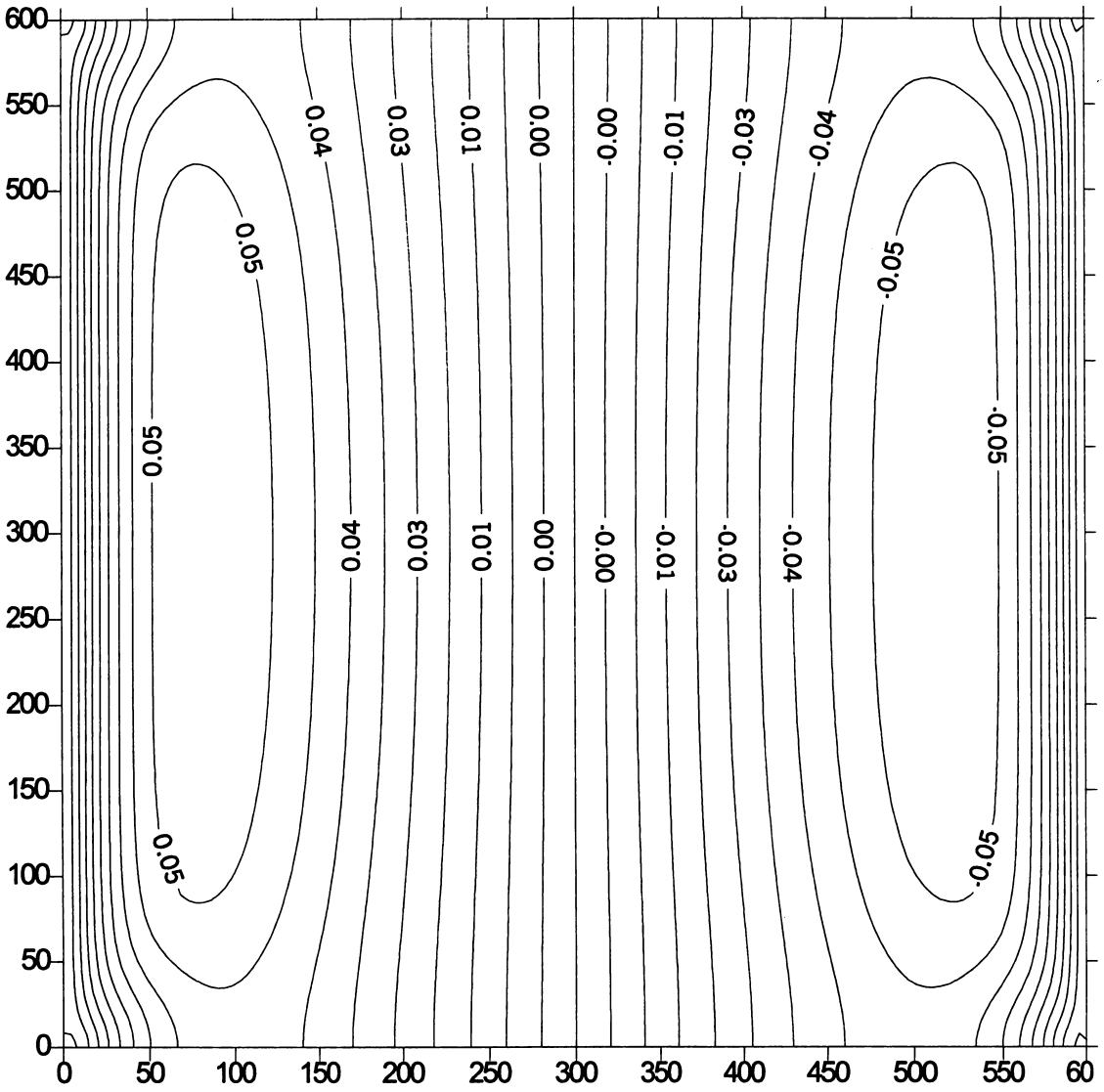
All data sets have 140 by 140 element grid.



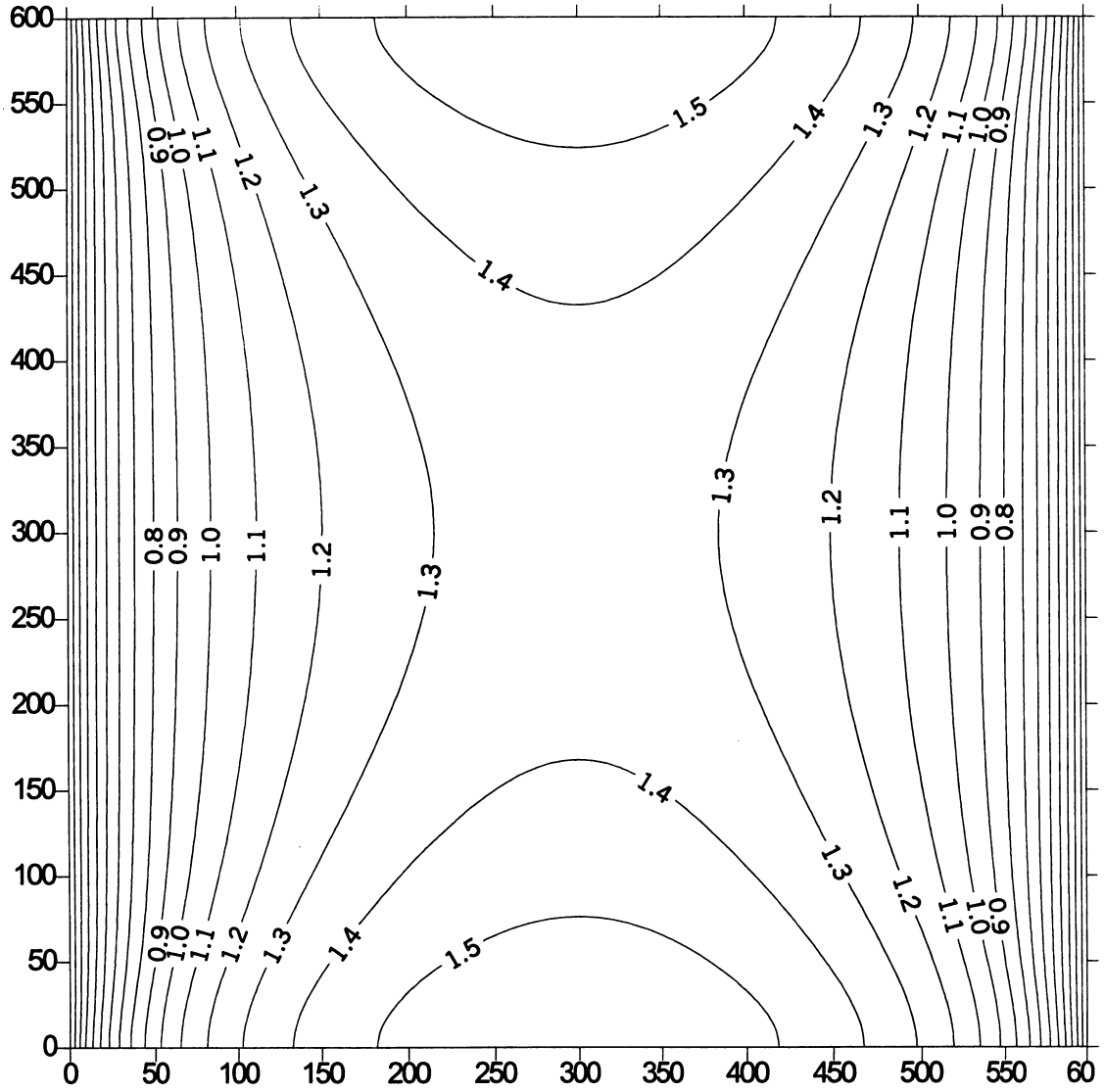
Difference between stochatistic mean hydraulic head values and deterministic result for isotropic 10m integral scale



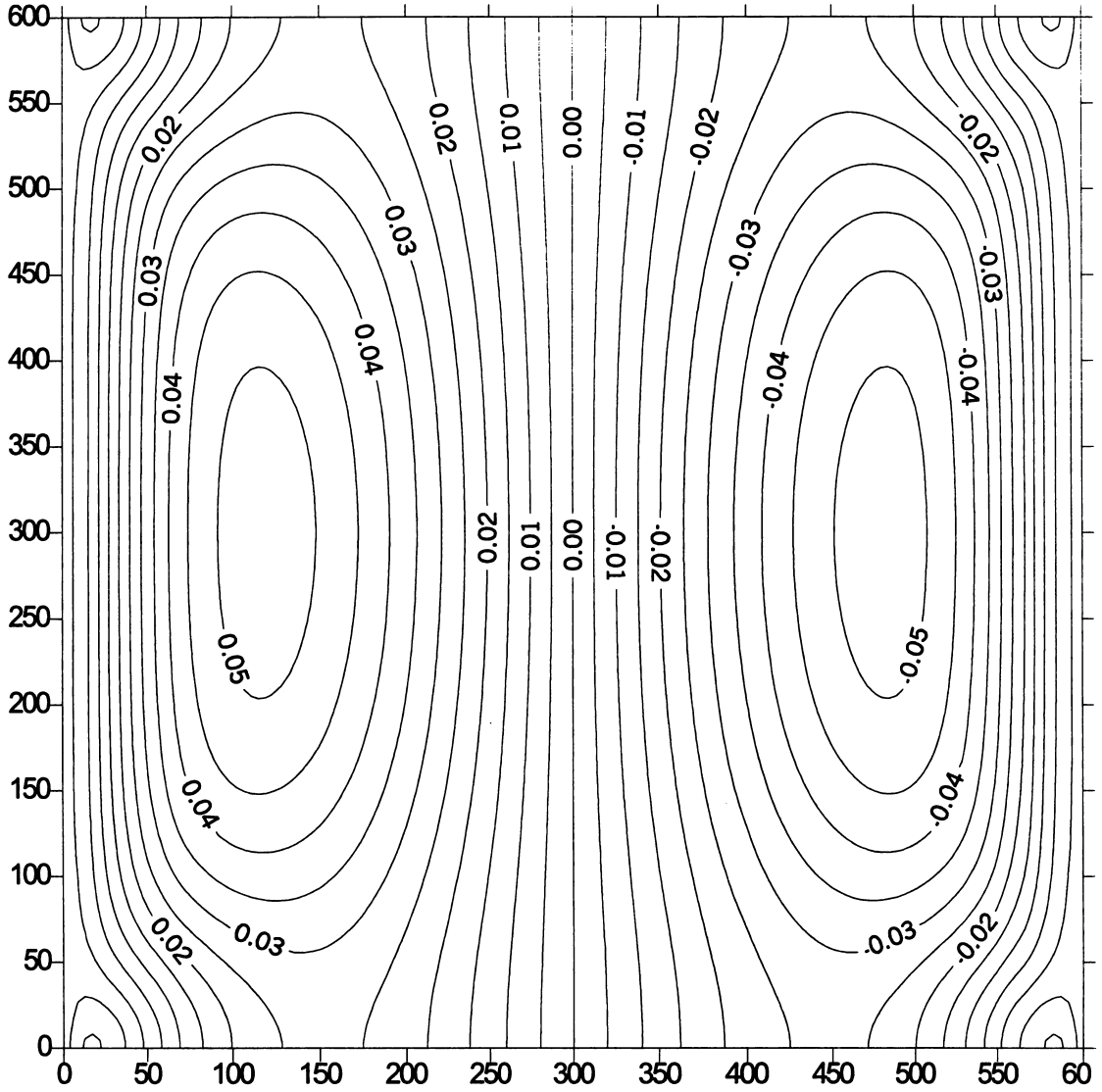
Standard deviation of hyraulic head values for isotropic 10m integral scale



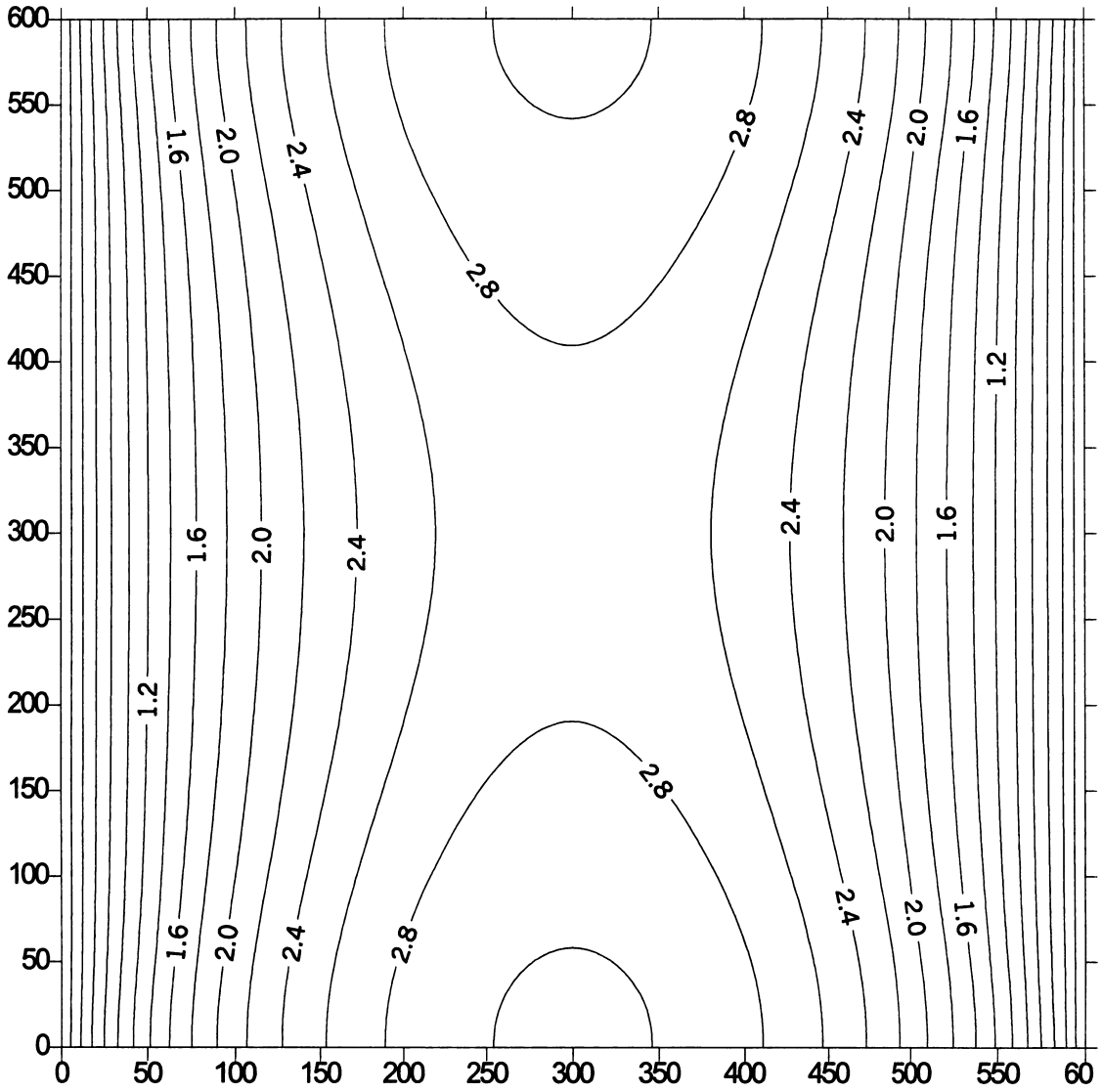
Difference between stochatistic mean hydraulic head values and deterministic result for isotropic 20m integral scale



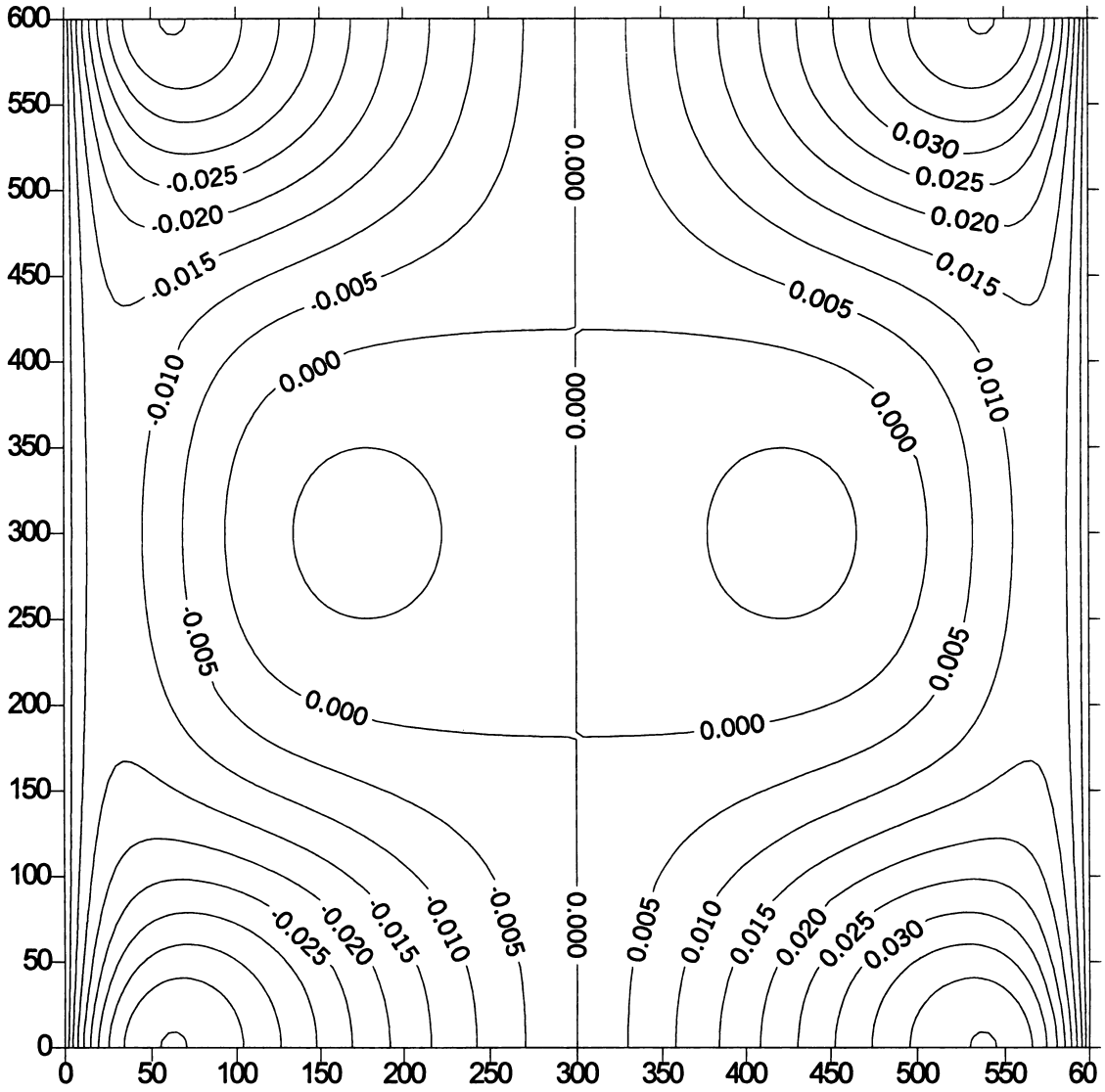
Standard deviation of hyraulic head values for isotropic 20m integral scale



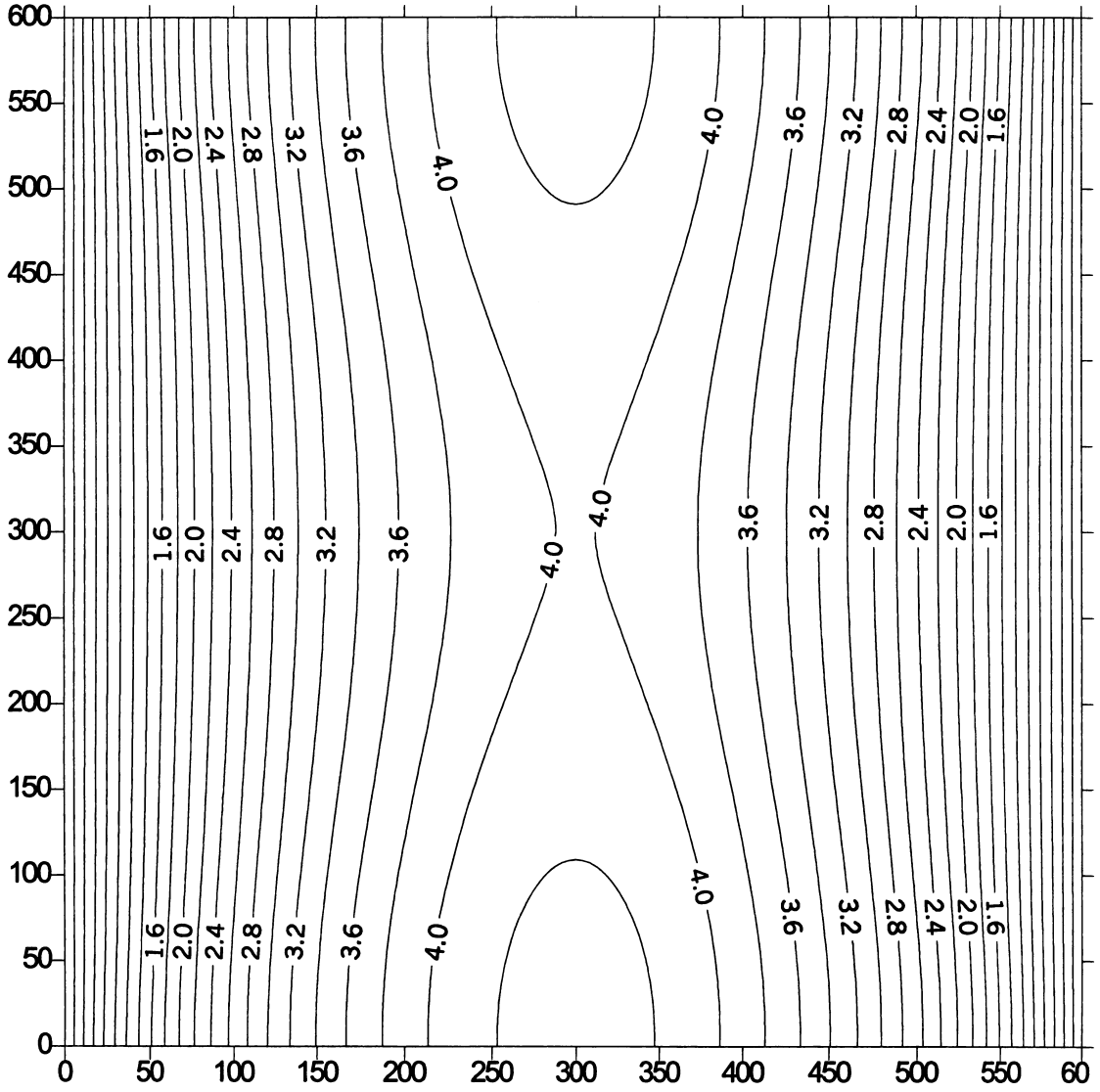
Difference between stochatistic mean hydraulic head values and deterministic result for isotropic 50m integral scale



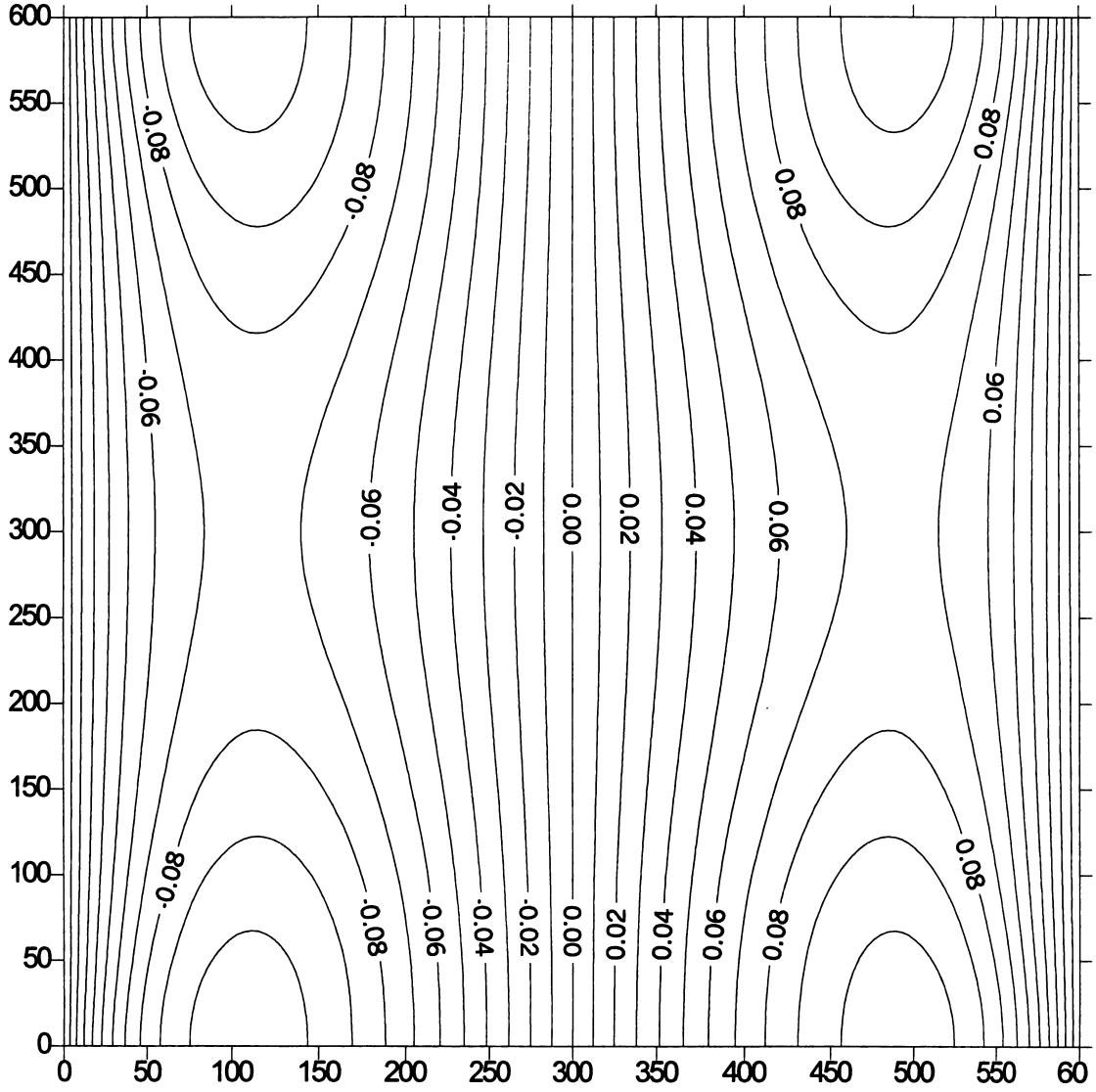
Standard deviation of hyraulic head values for isotropic 50m integral scale



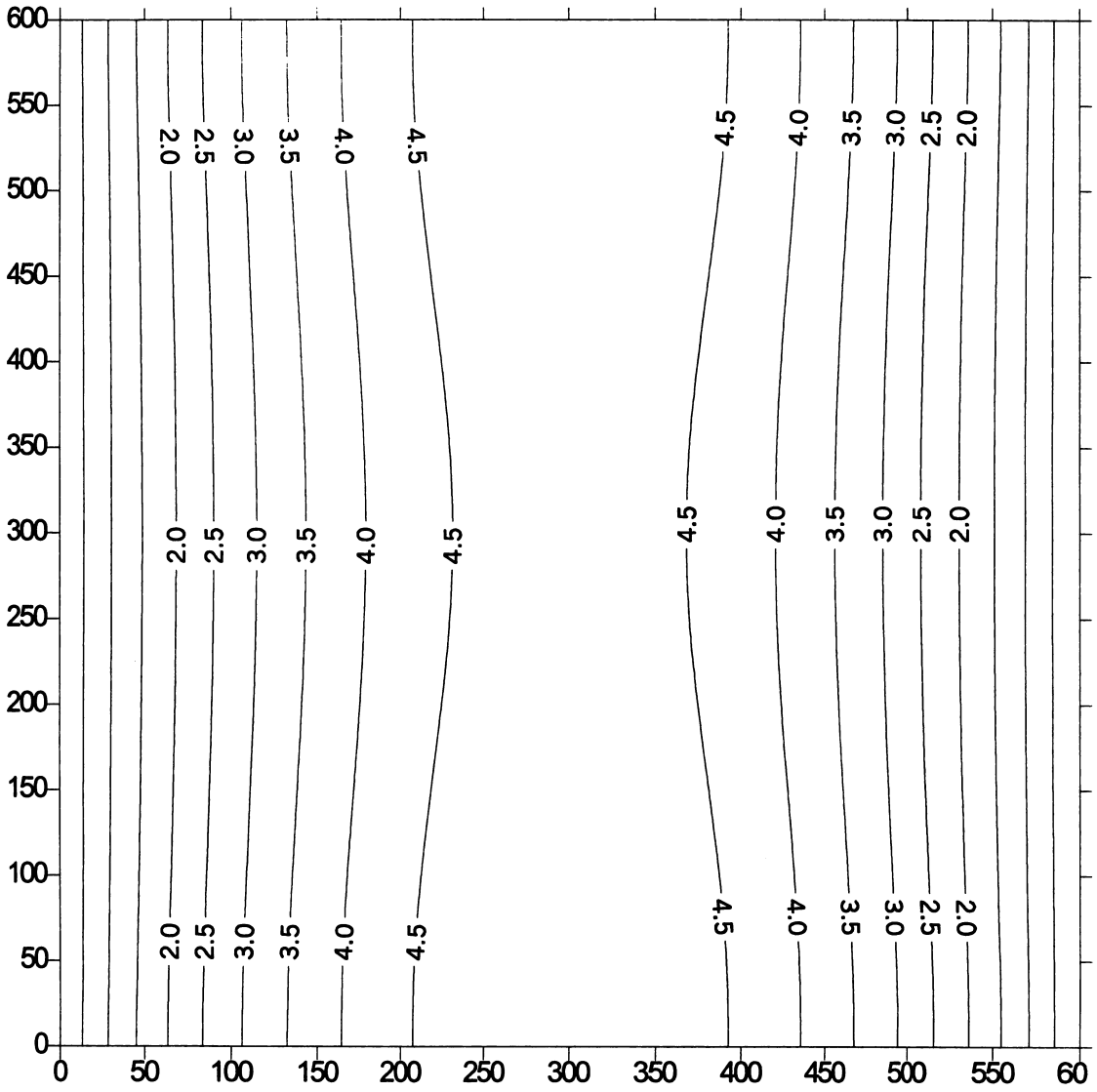
Difference between stochatistic mean hydraulic head values and deterministic result for isotropic 100m integral scale



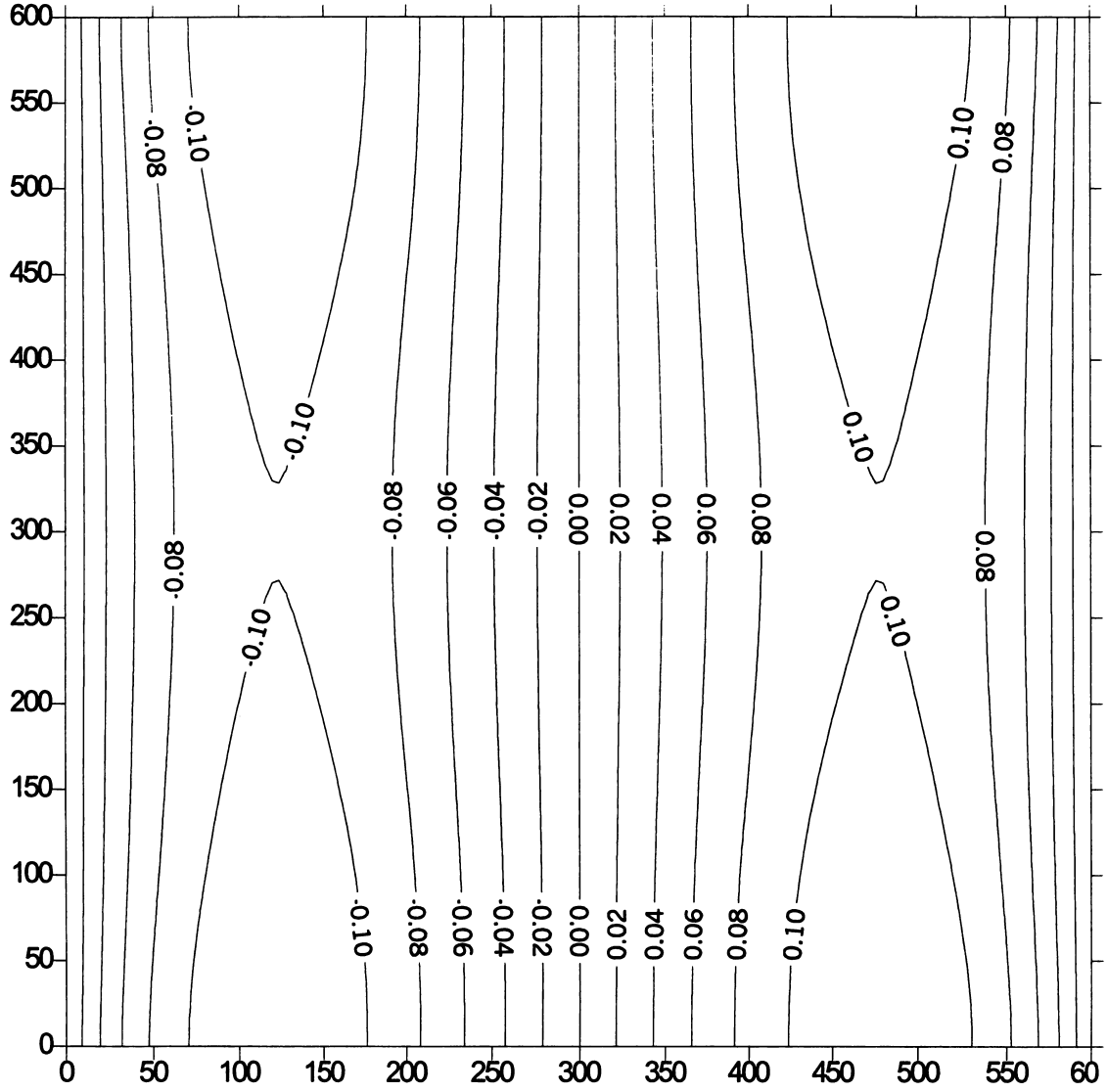
Standard deviation of hyraulic head values for isotropic 100m integral scale



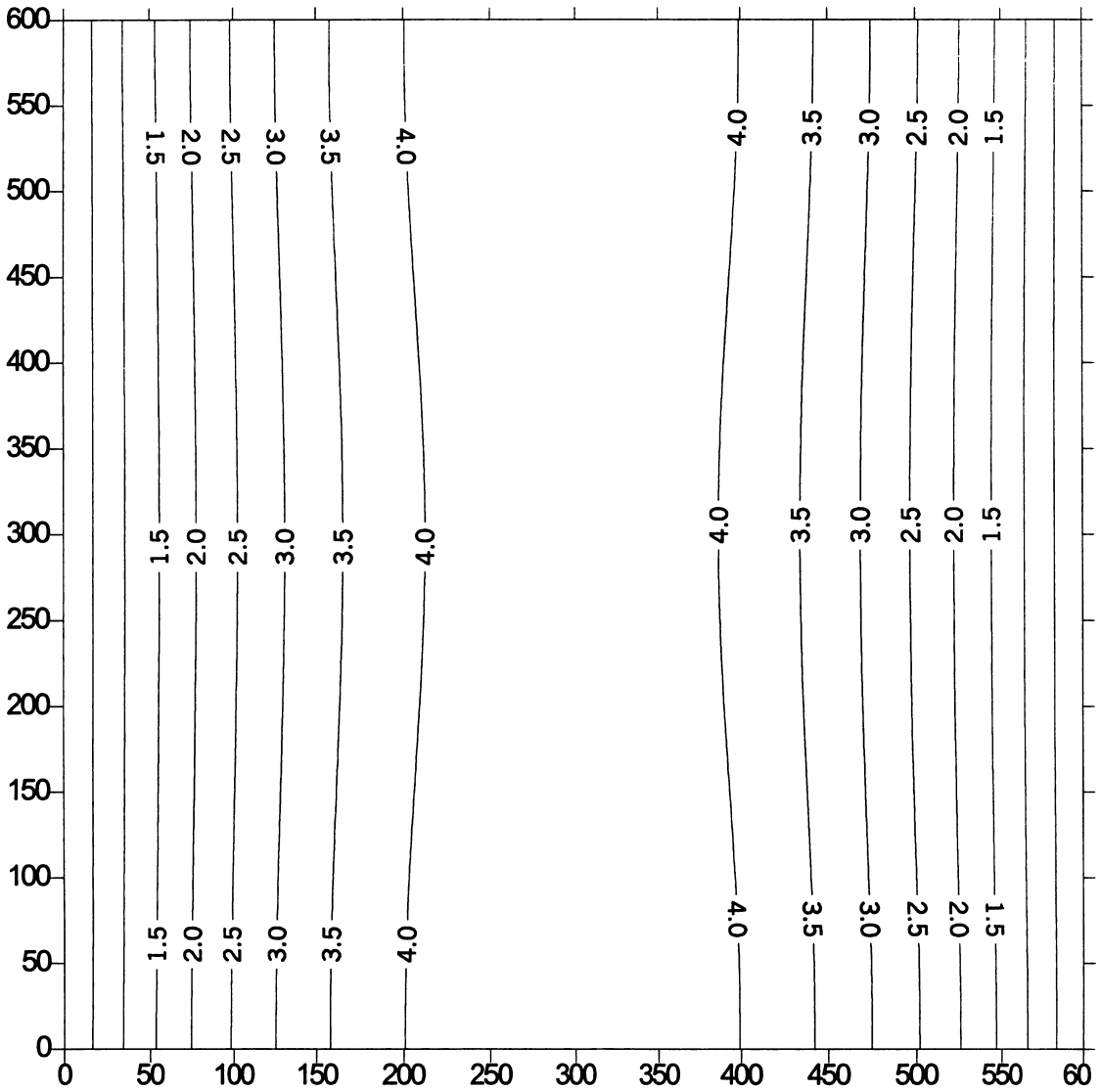
Difference between stochatistic mean hydraulic head values and deterministic result for isotropic 200m integral scale



Standard deviation of hyraulic head values for isotropic 200m integral scale



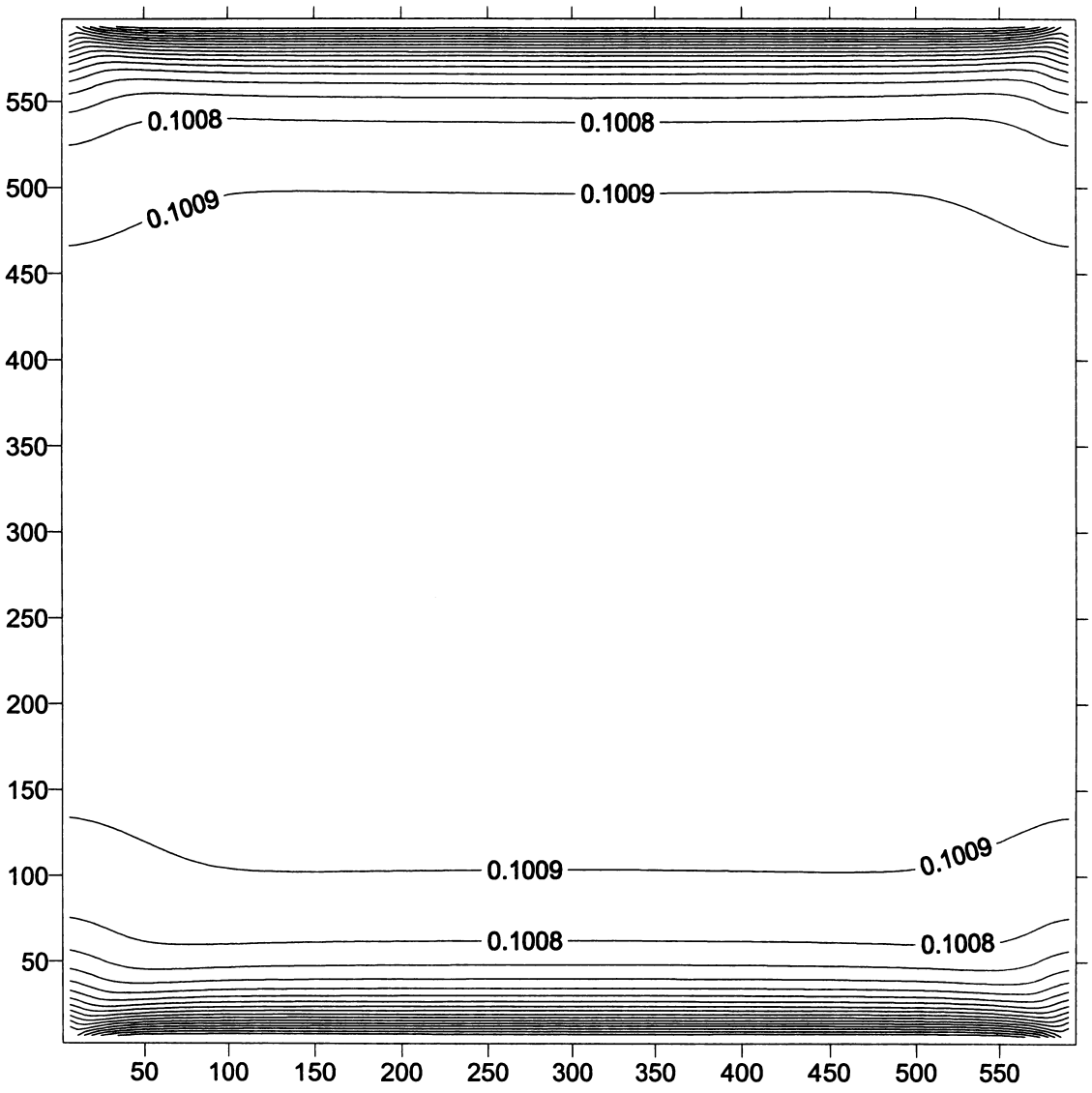
Difference between stochatistic mean hydraulic head values and deterministic result for isotropic 500m integral scale



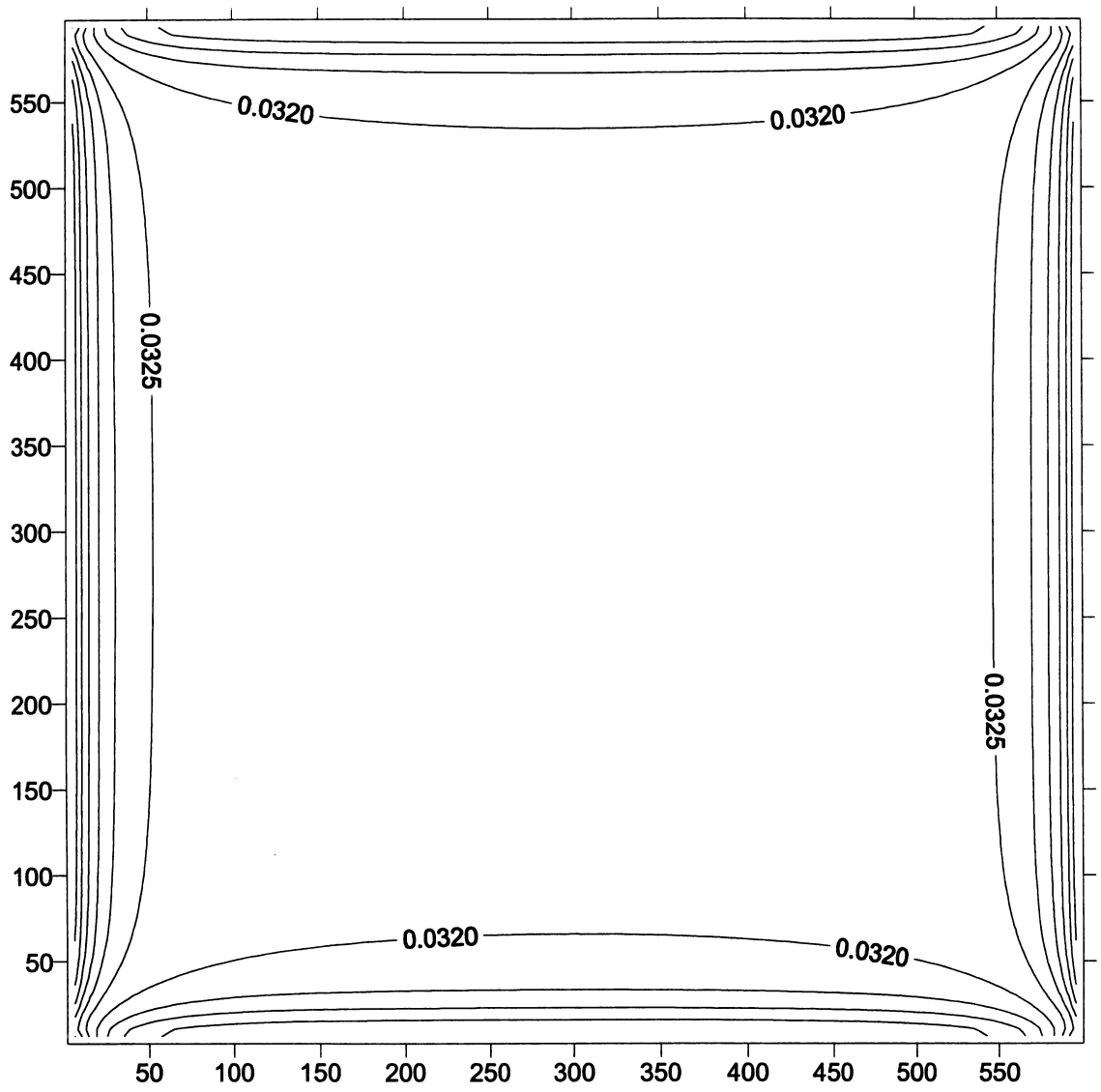
Standard deviation of hyraulic head values for isotropic 500m integral scale

Appendix D Velocity Results

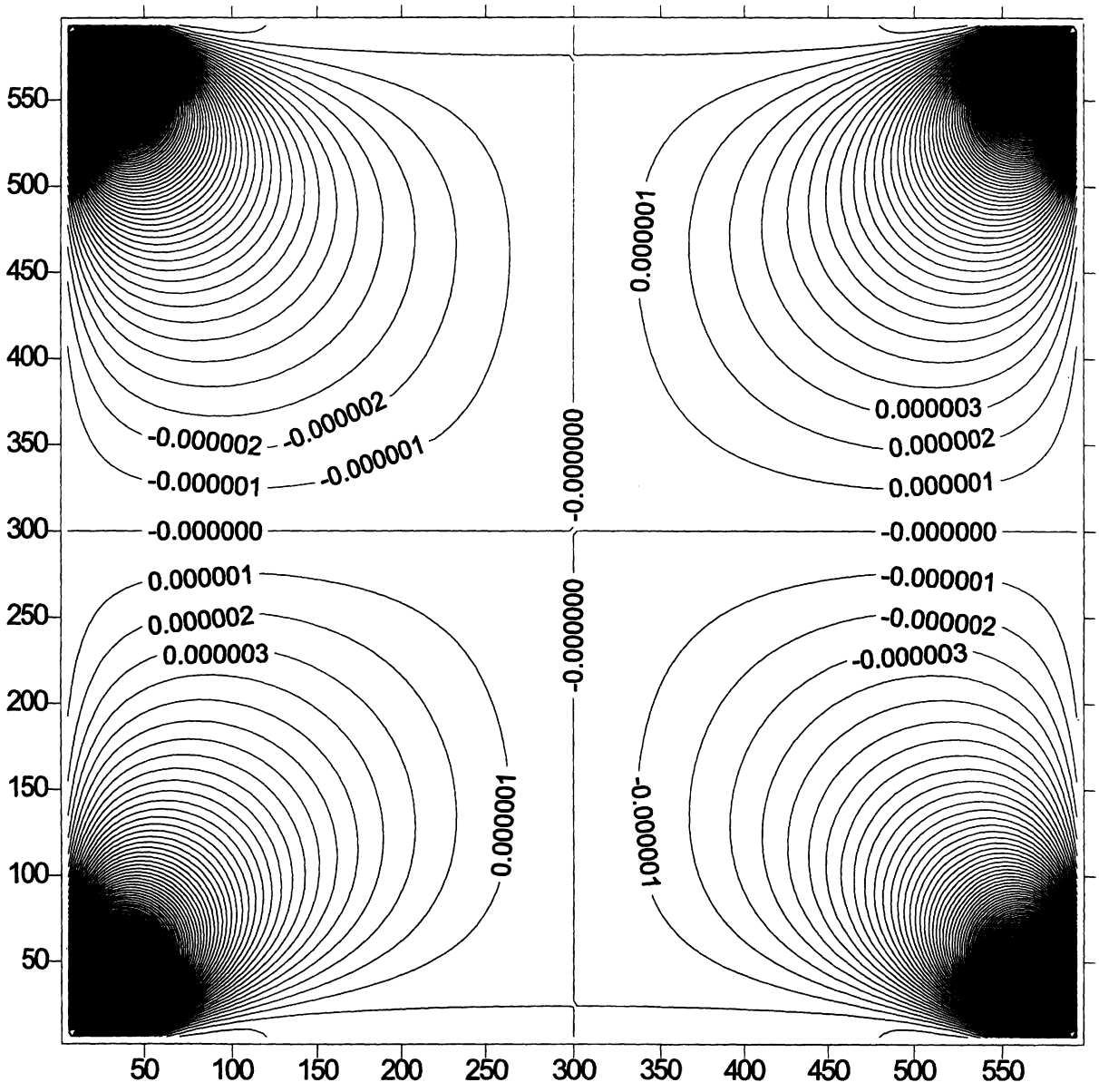
All data sets have 140 by 140 grid.



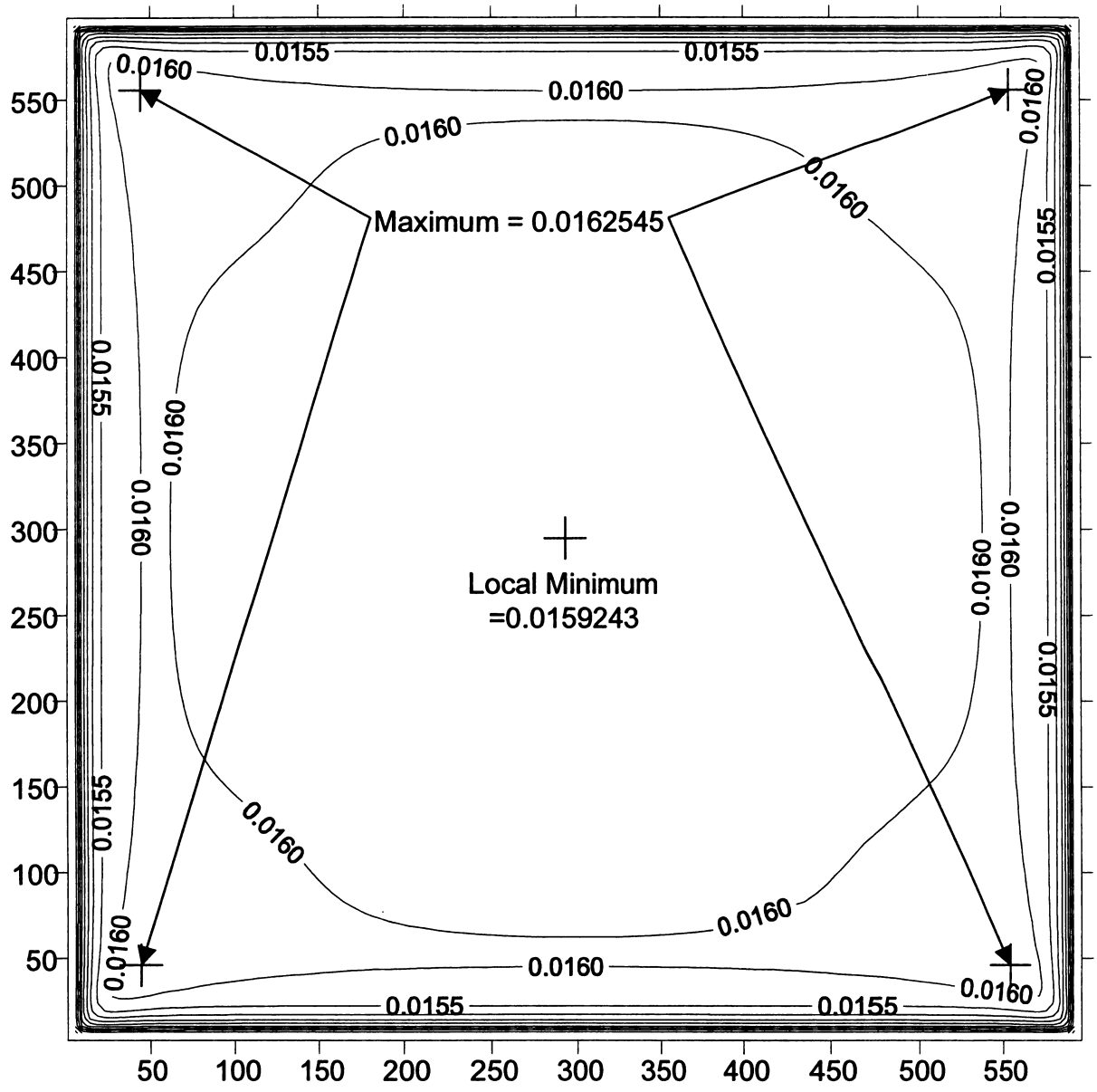
Mean longitudinal velocity for isotropic 10m integral scale



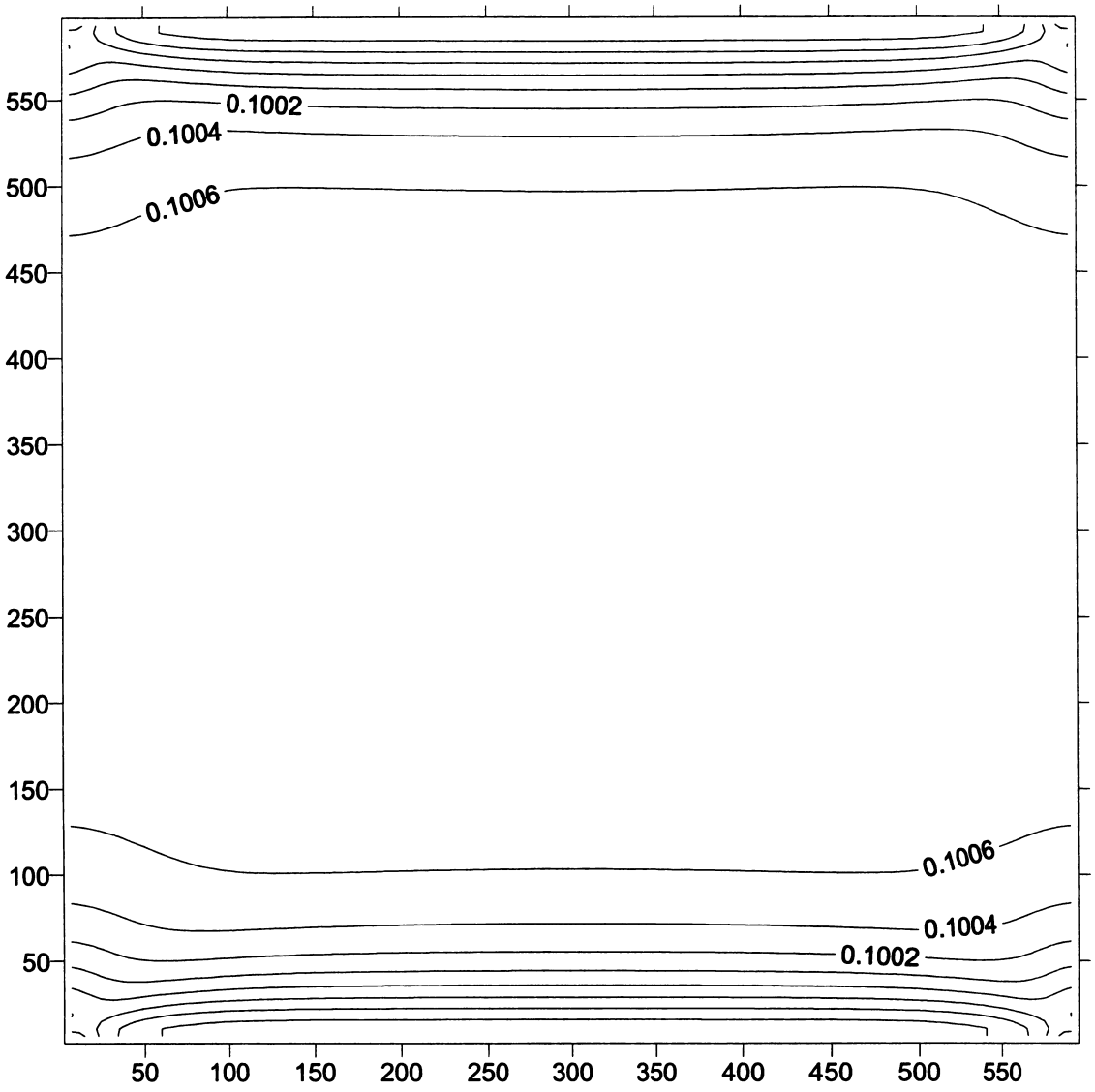
Standard deviations of longitudinal velocity for isotropic 10m integral scale



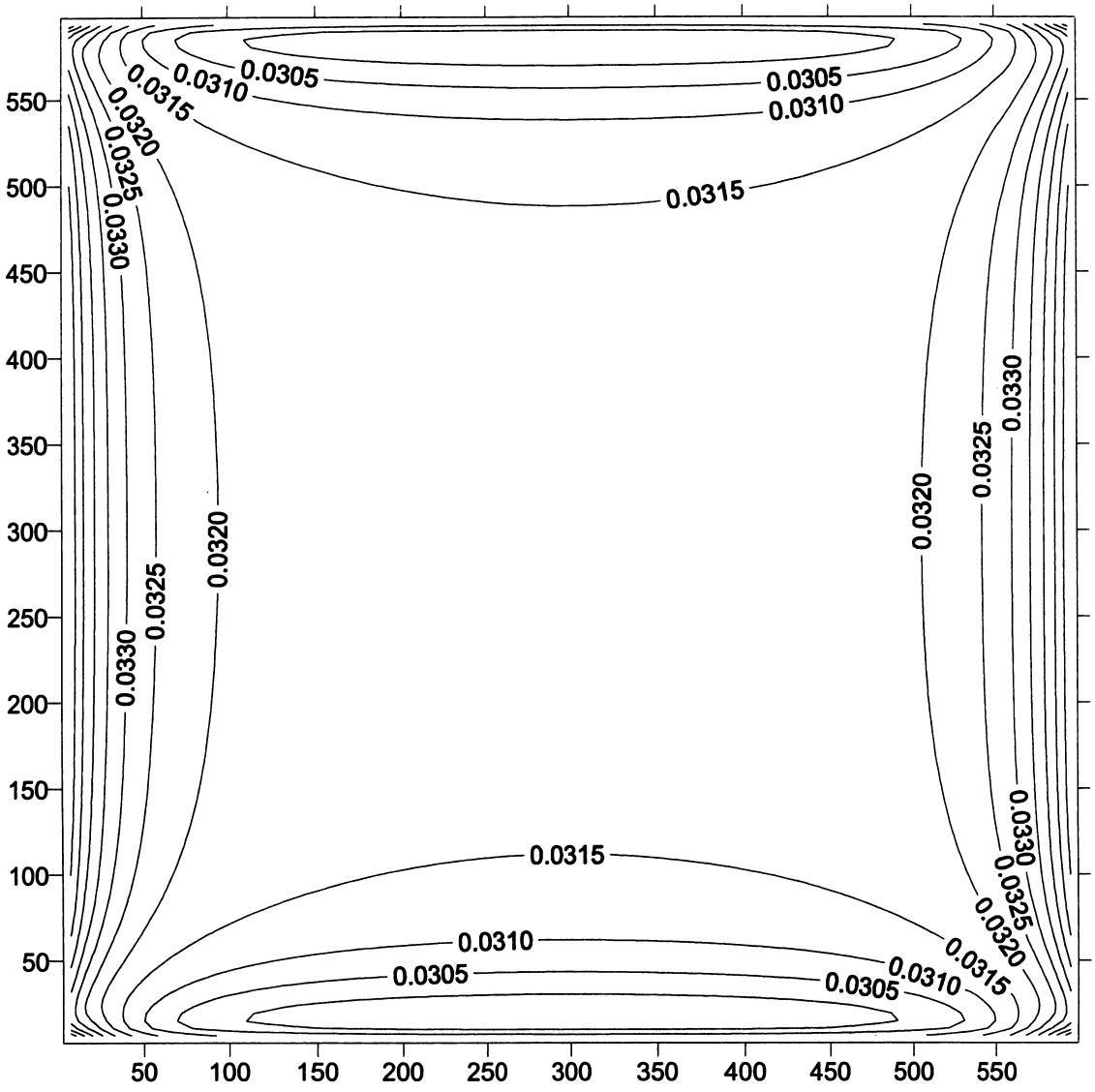
Mean transverse velocity for isotropic 10m integral scale



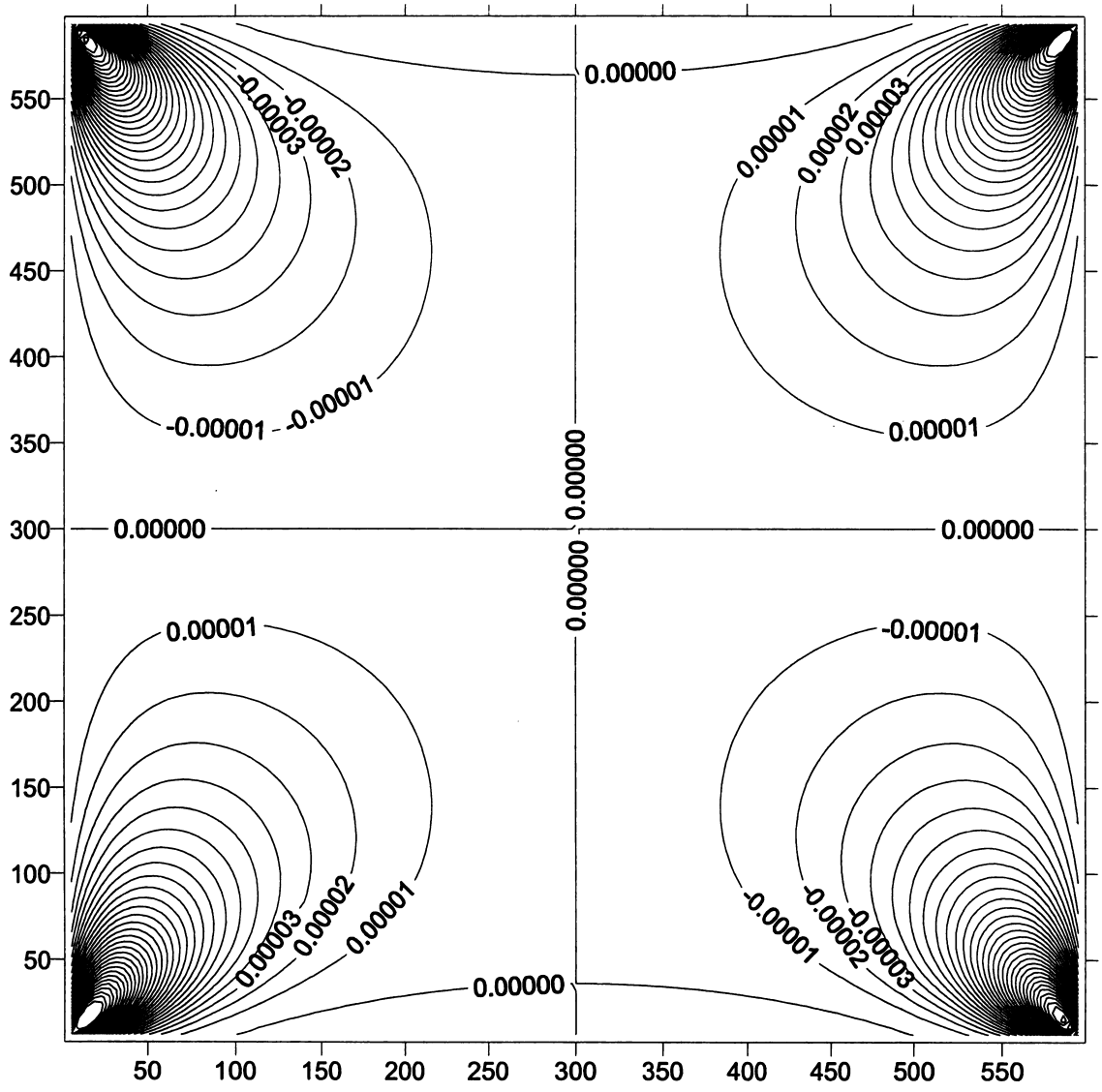
Standard deviation of transverse velocity for isotropic 10m integral scale



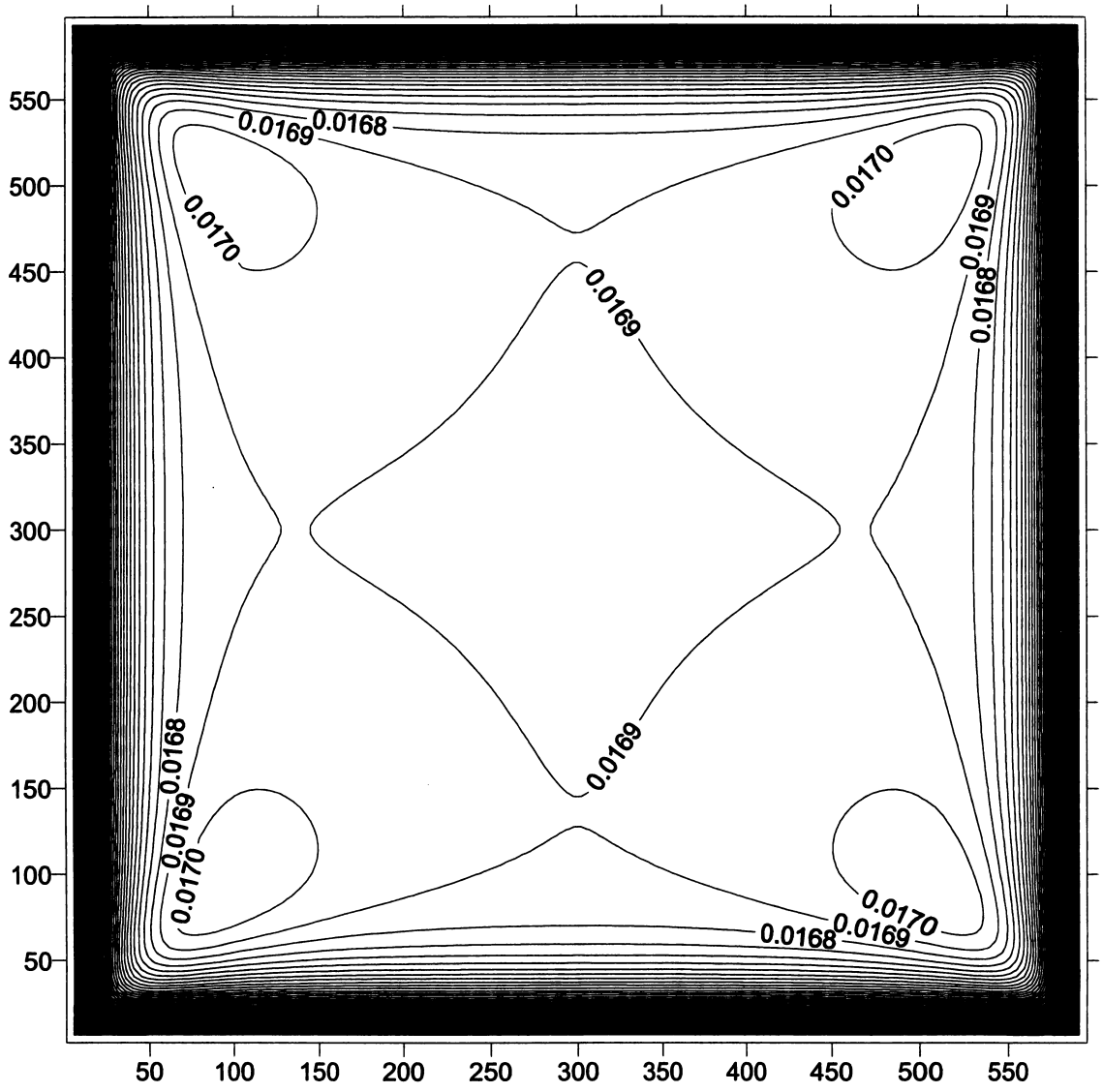
Mean longitudinal velocity for isotropic 20m integral scale



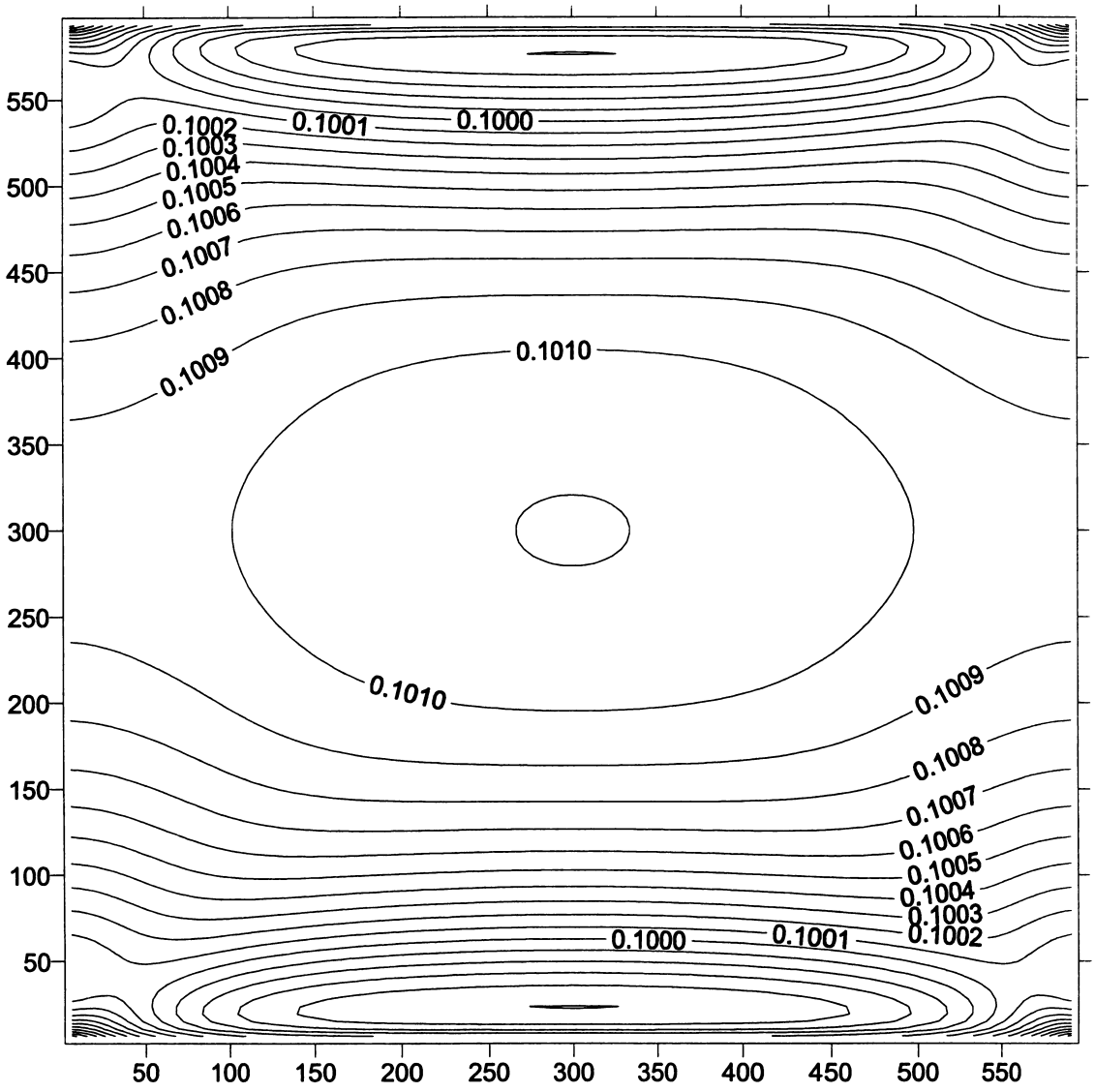
Standard deviations of longitudinal velocity for isotropic 20m integral scale



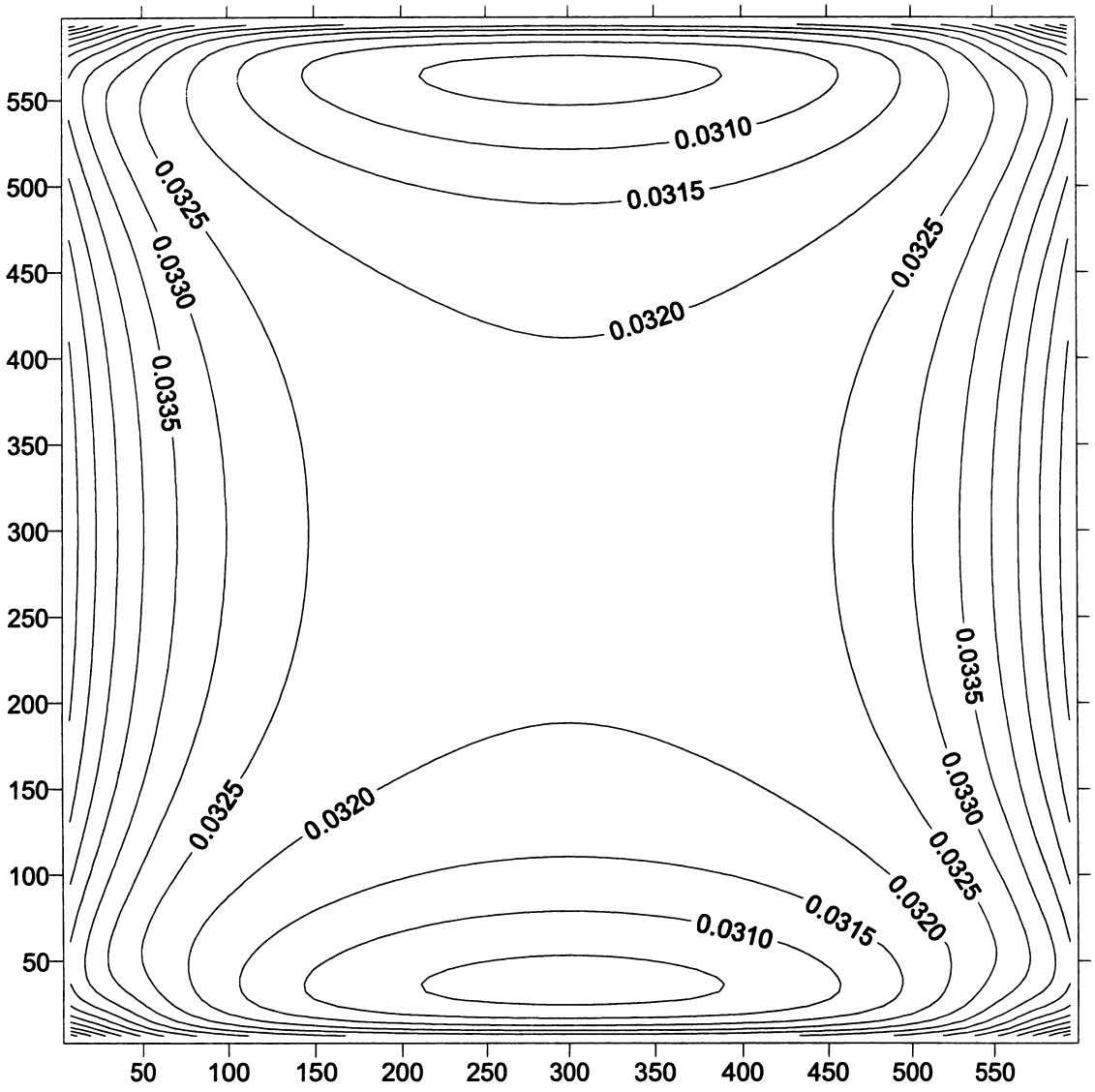
Mean transverse velocity for isotropic 20m integral scale



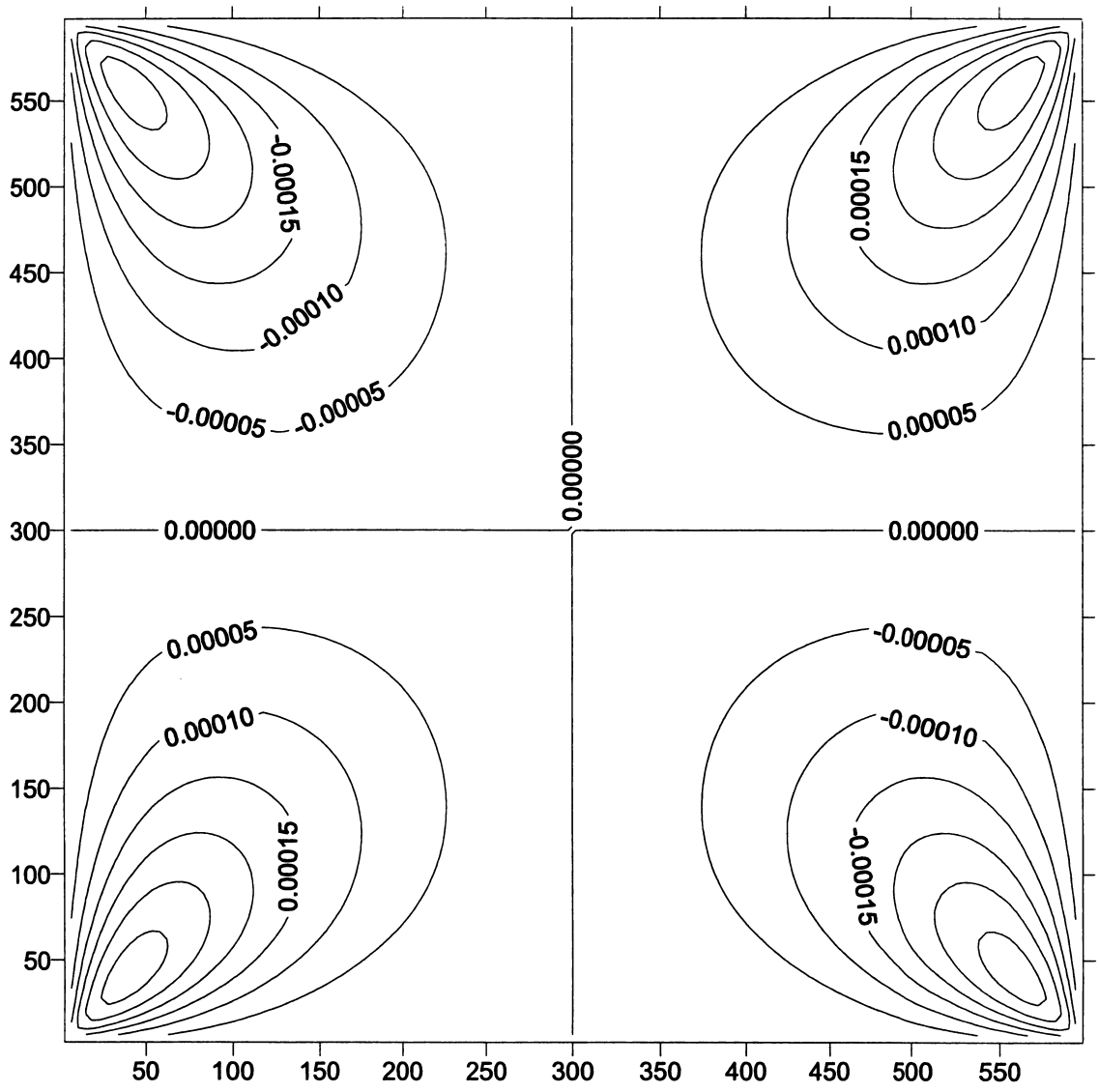
Standard deviation of transverse velocity for isotropic 20m integral scale



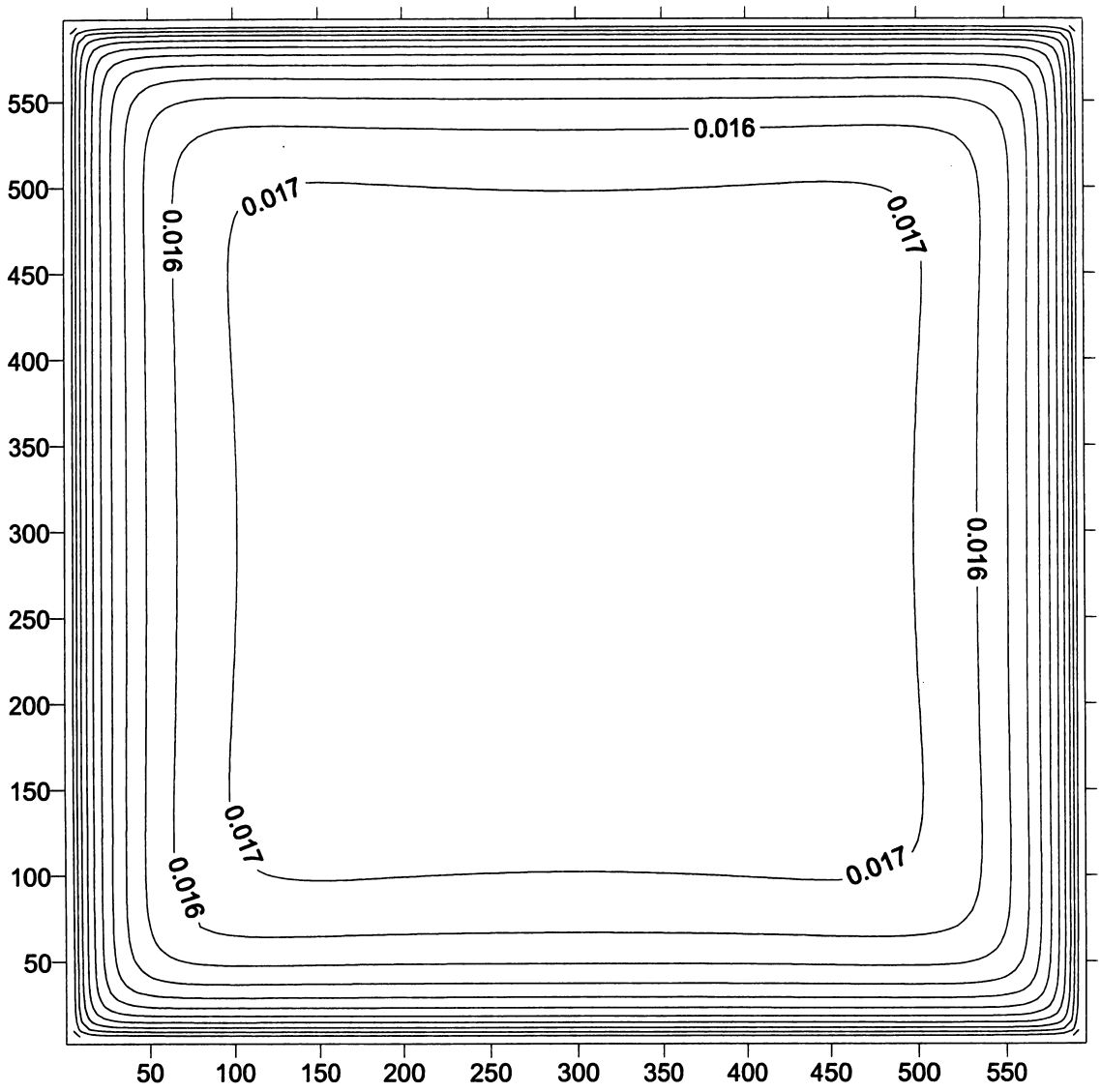
Mean longitudinal velocity for isotropic 50m integral scale



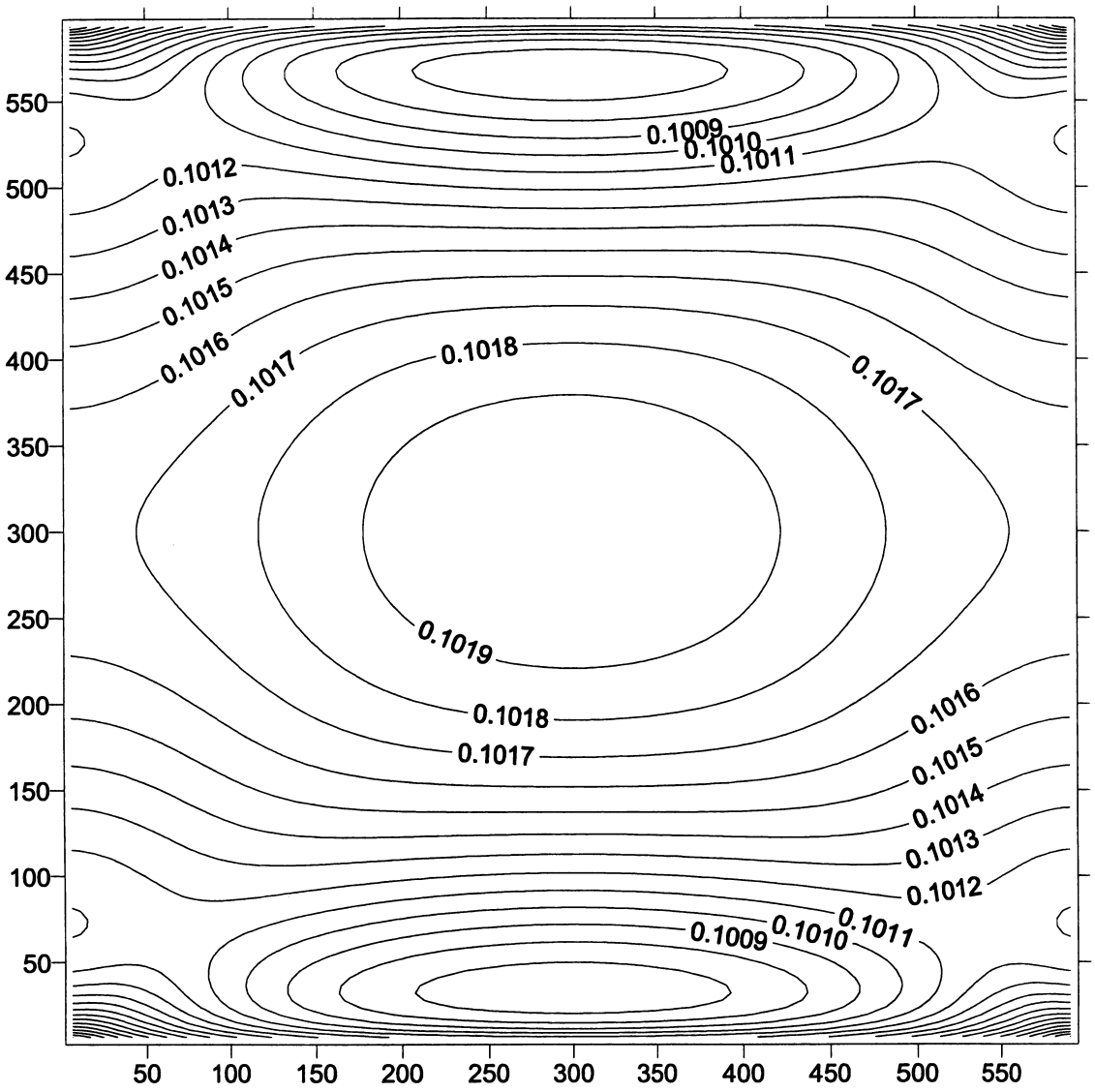
Standard deviations of longitudinal velocity for isotropic 50m integral scale



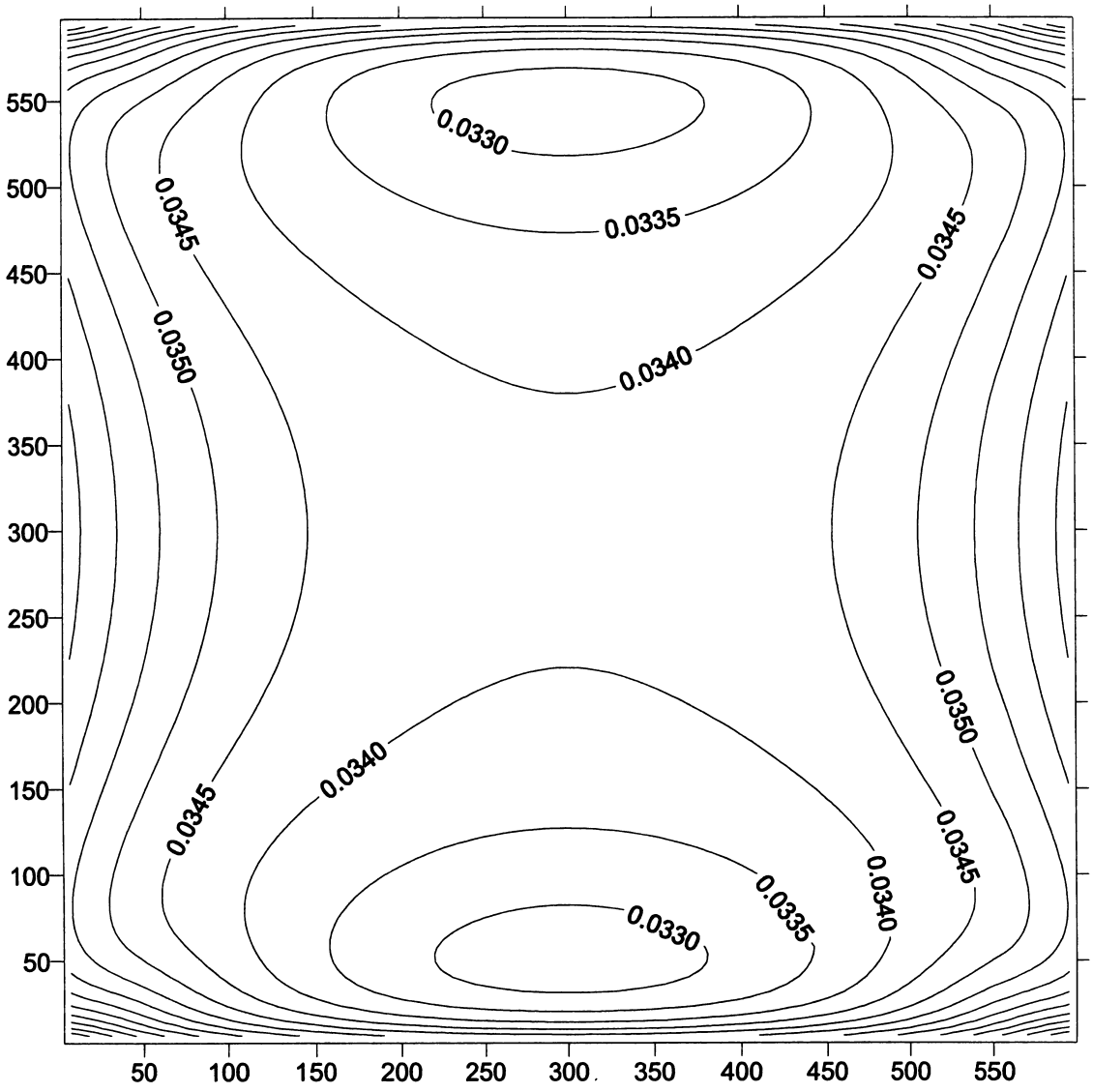
Mean transverse velocity for isotropic 50m integral scale



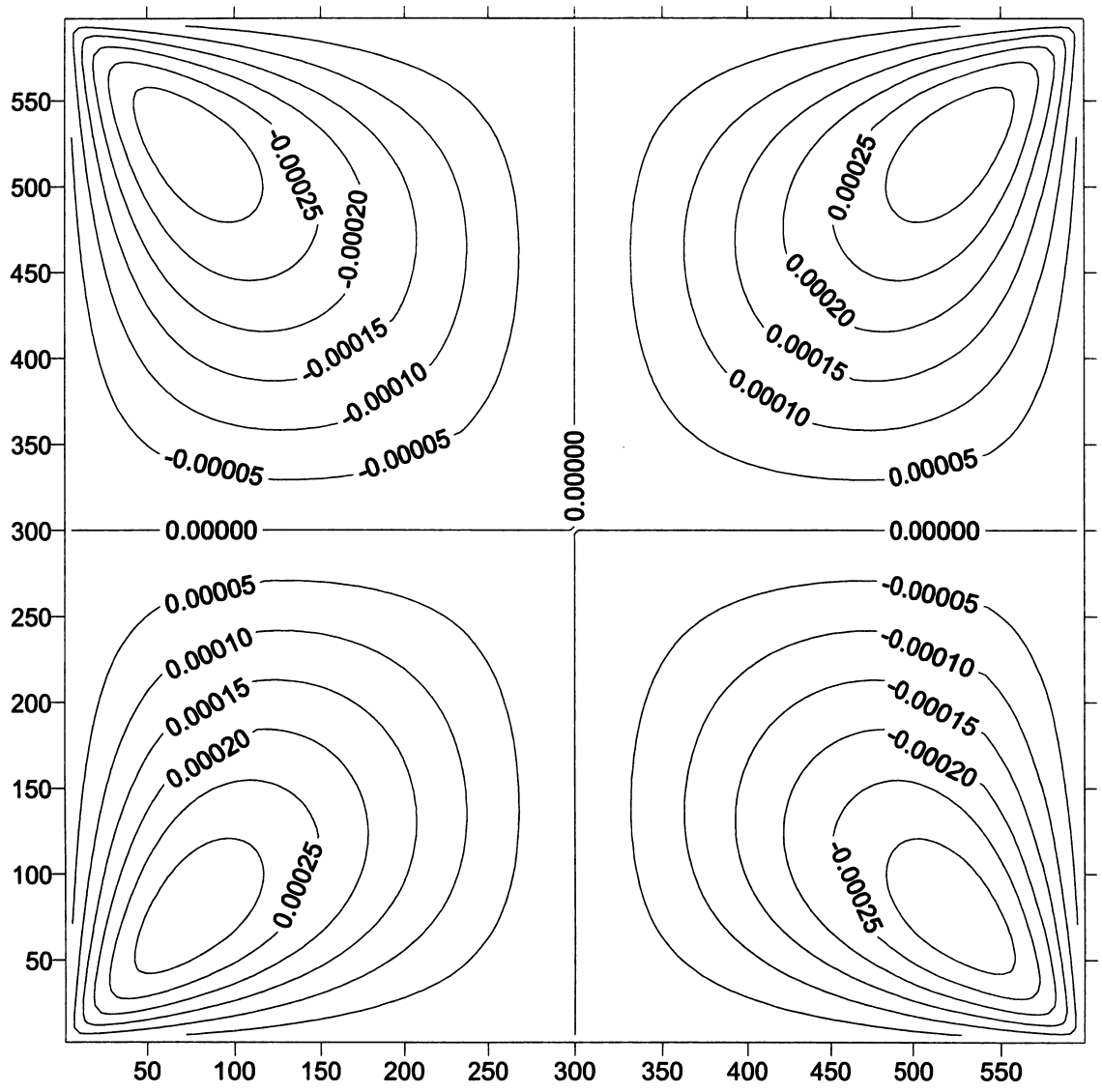
Standard deviation of transverse velocity for isotropic 50m integral scale



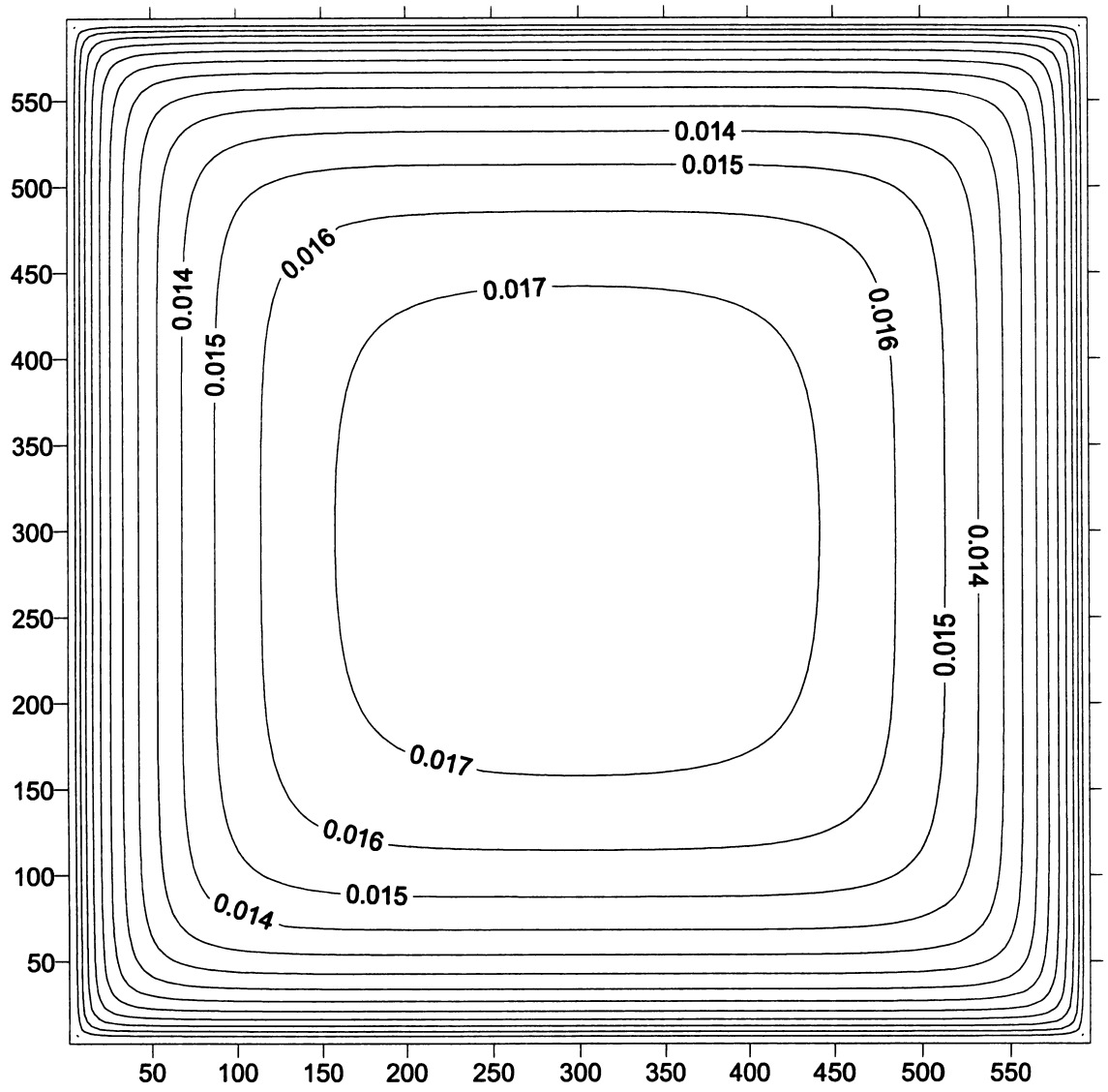
Mean longitudinal velocity for isotropic 100m integral scale



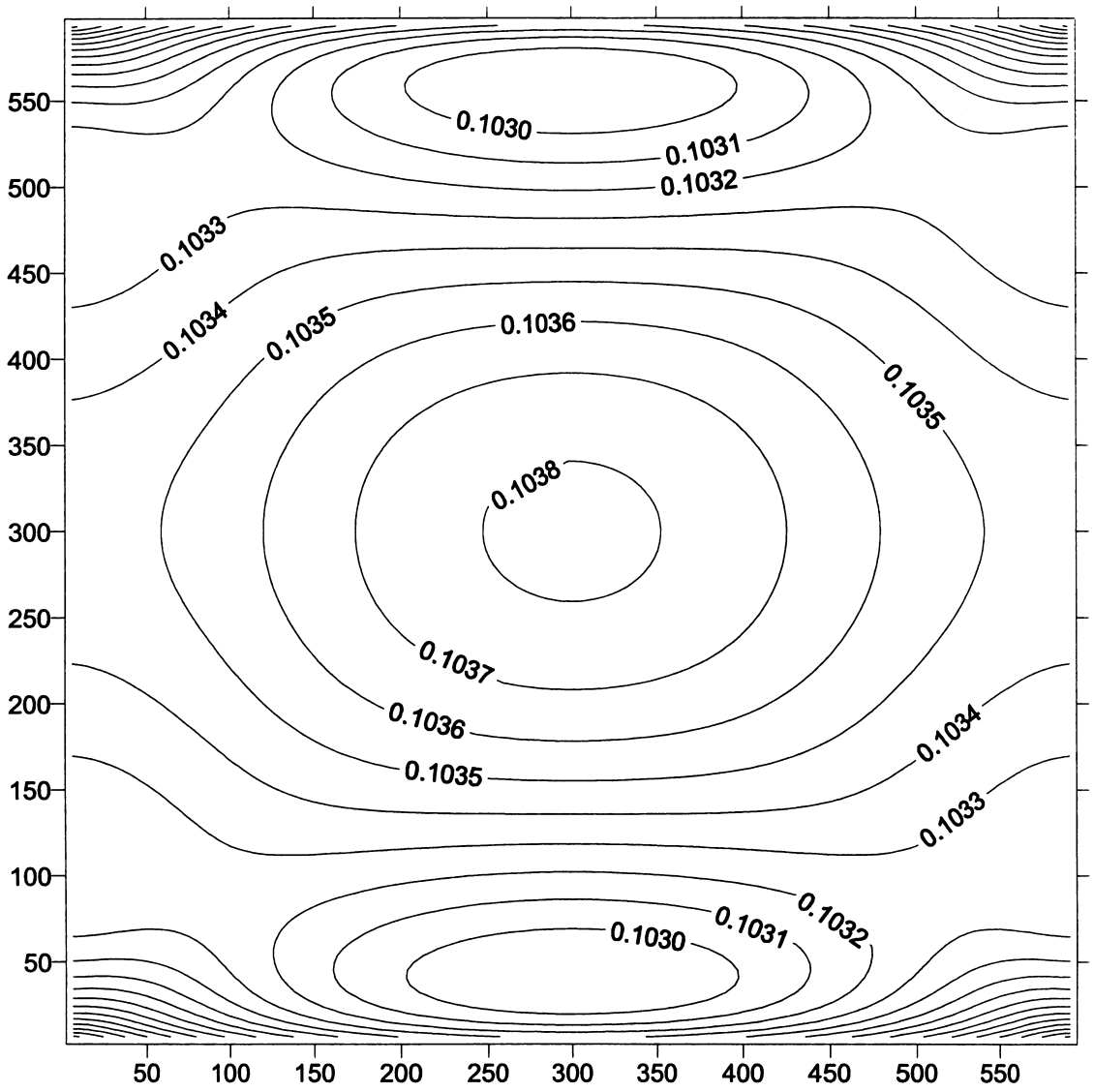
Standard deviations of longitudinal velocity for isotropic 100m integral scale



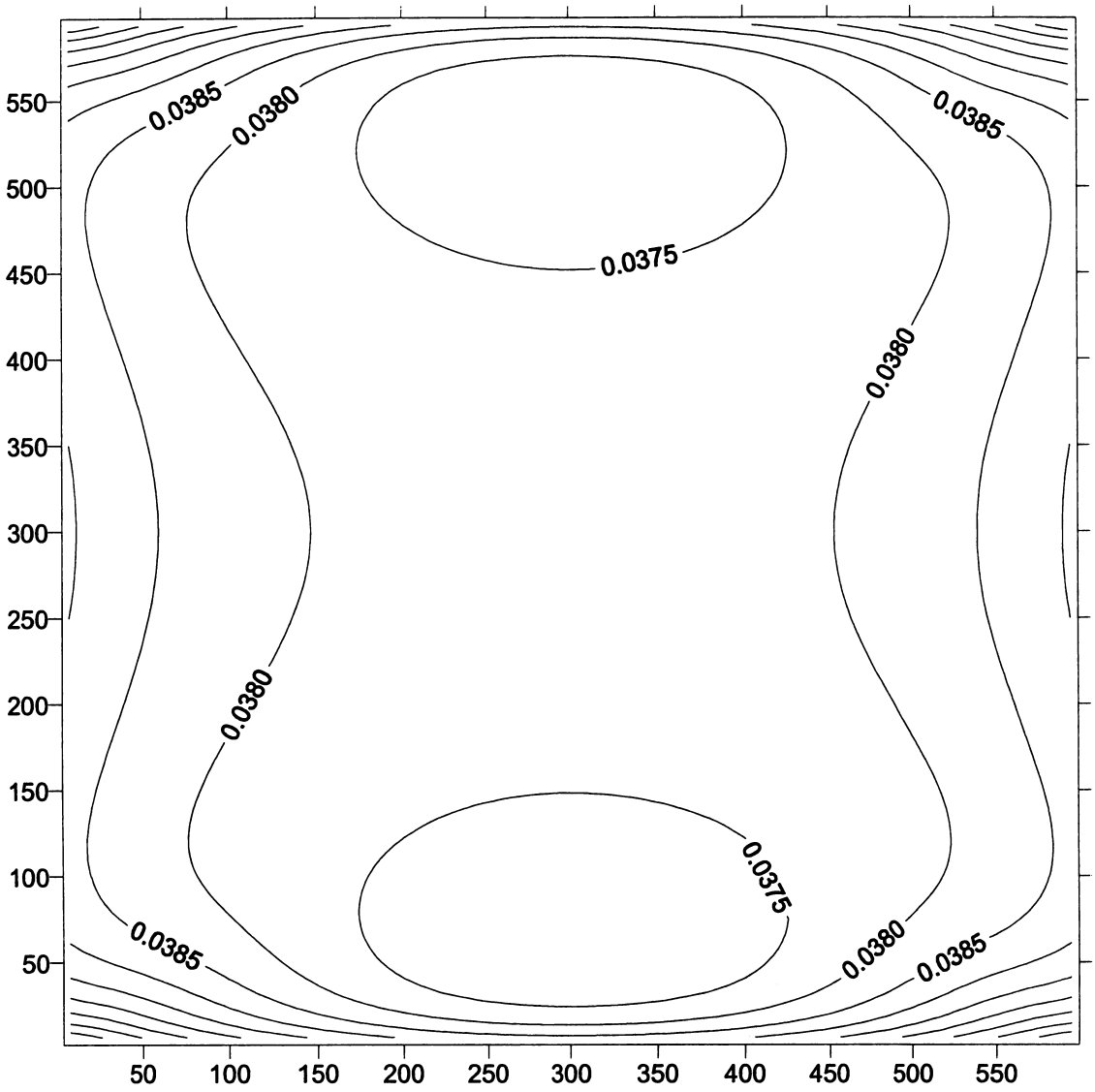
Mean transverse velocity for isotropic 100m integral scale



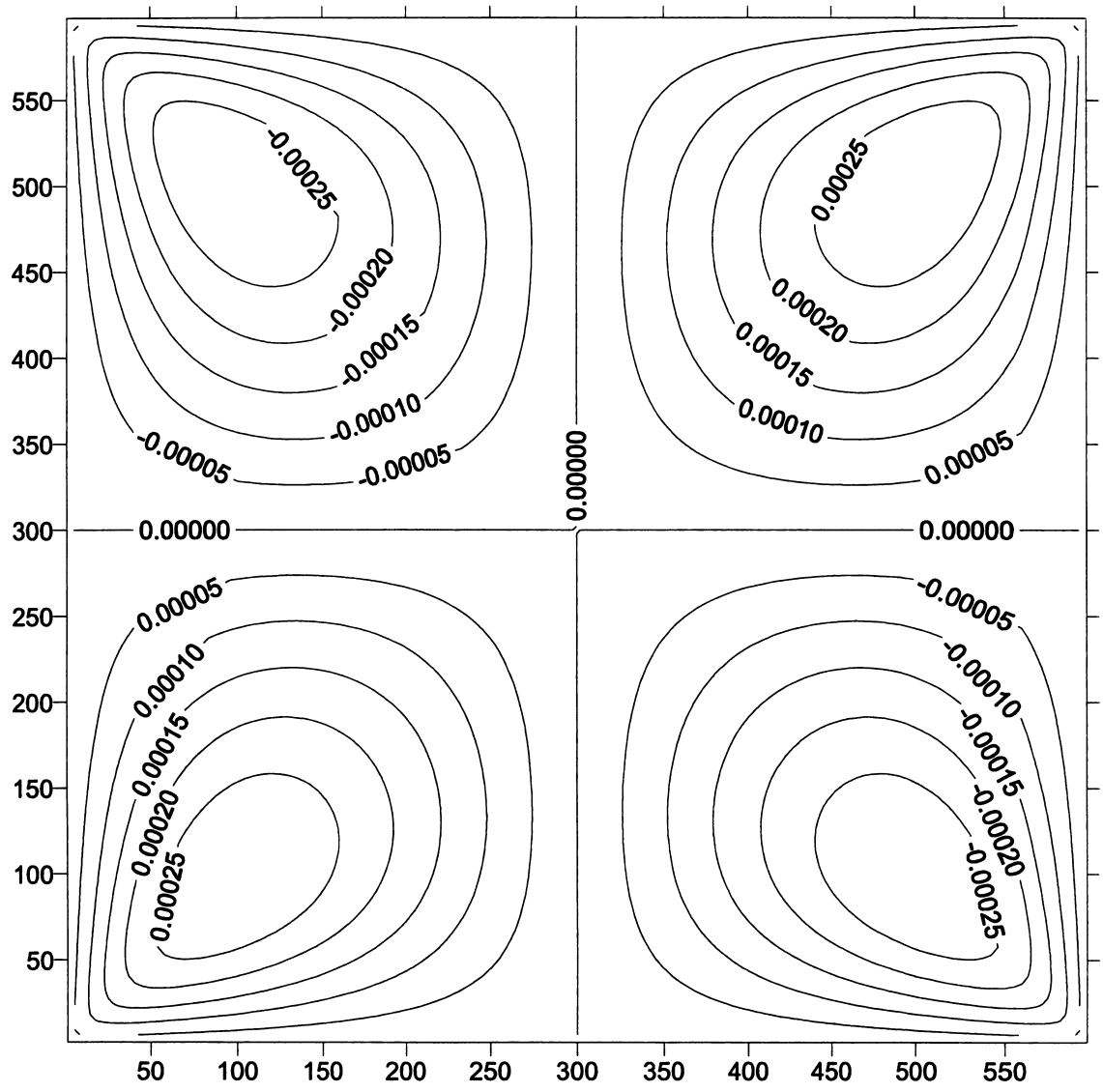
Standard deviation of transverse velocity for isotropic 100m integral scale



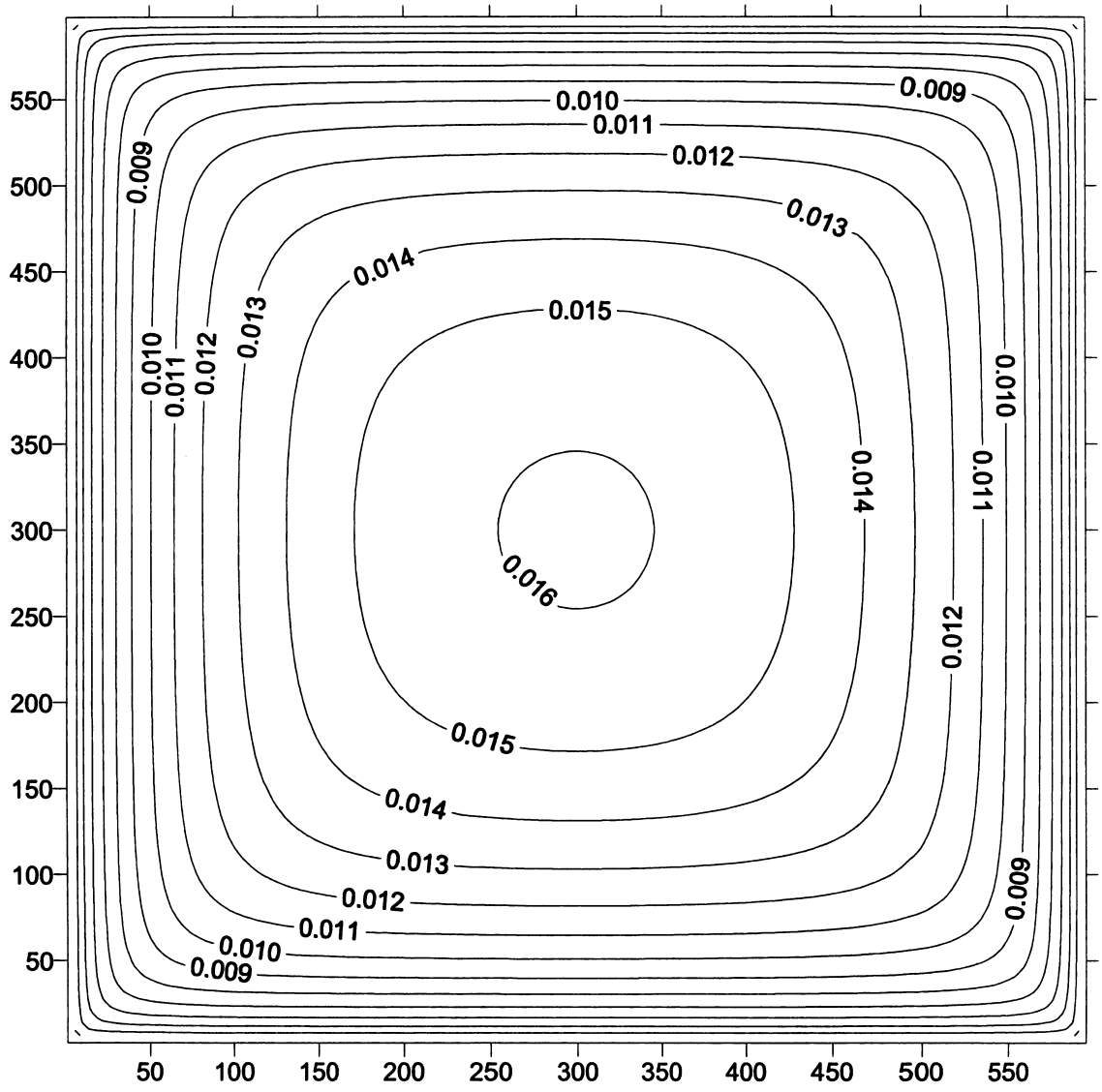
Mean longitudinal velocity for isotropic 200m integral scale



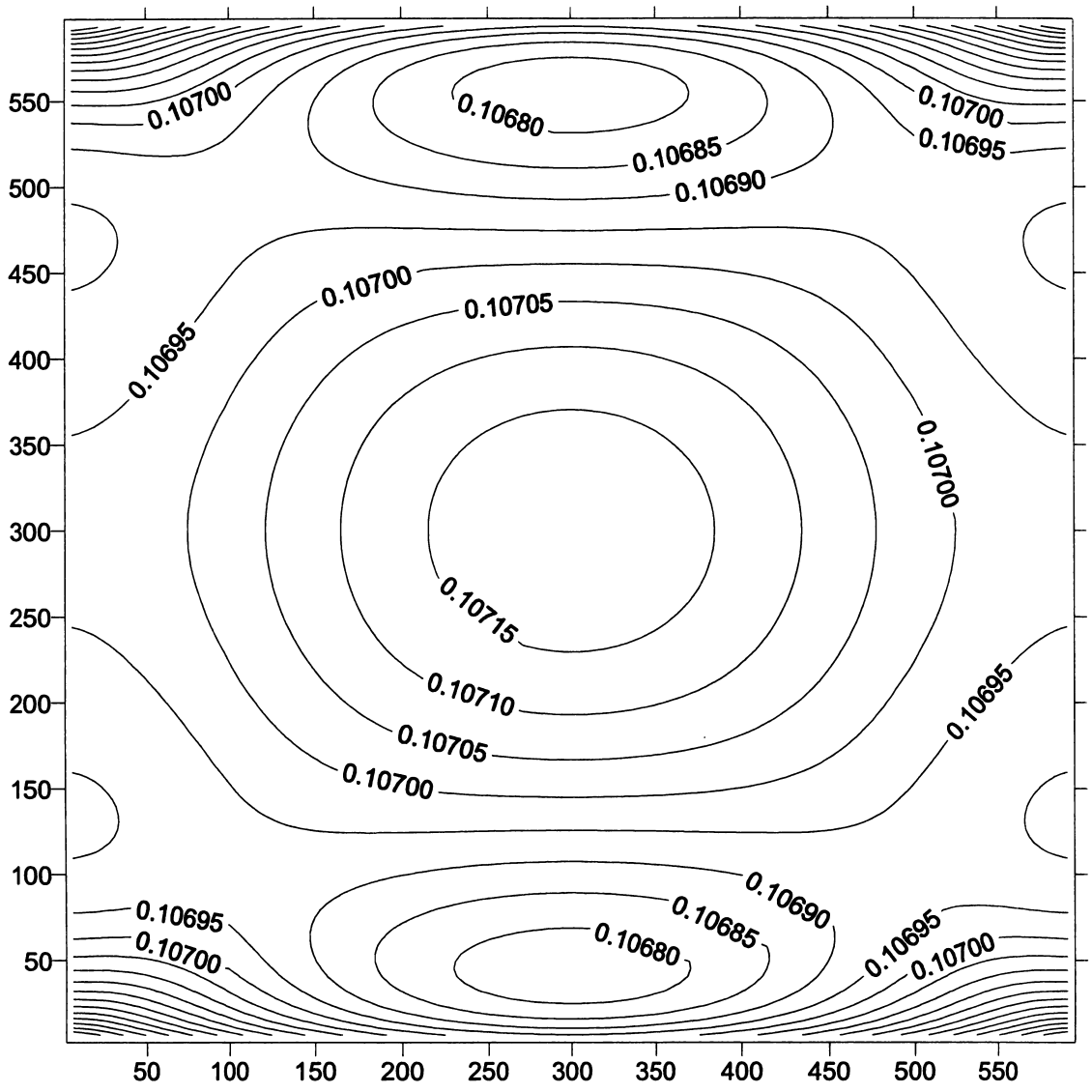
Standard deviations of longitudinal velocity for isotropic 200m integral scale



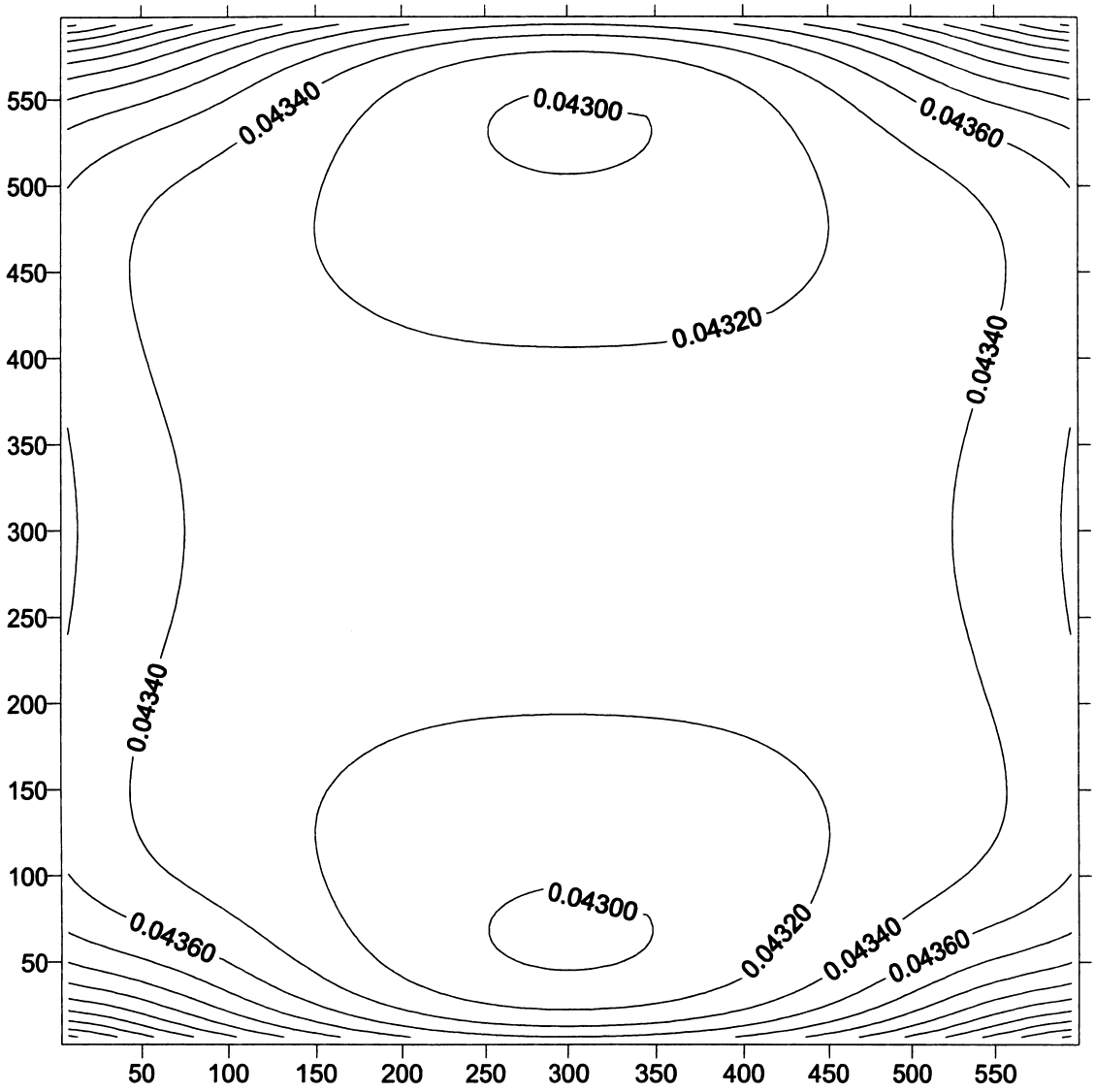
Mean transverse velocity for isotropic 200m integral scale



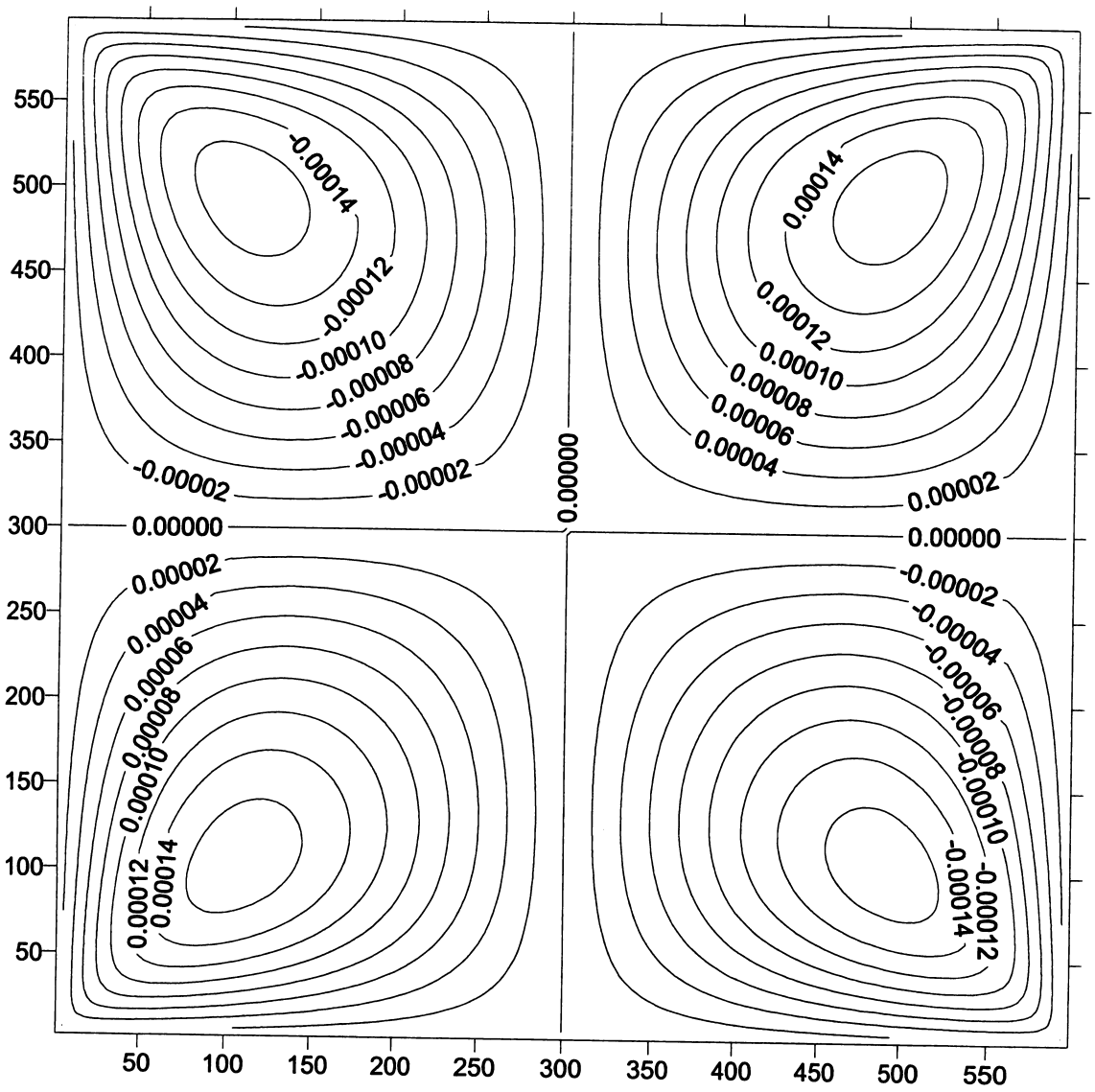
Standard deviation of transverse velocity for isotropic 200m integral scale



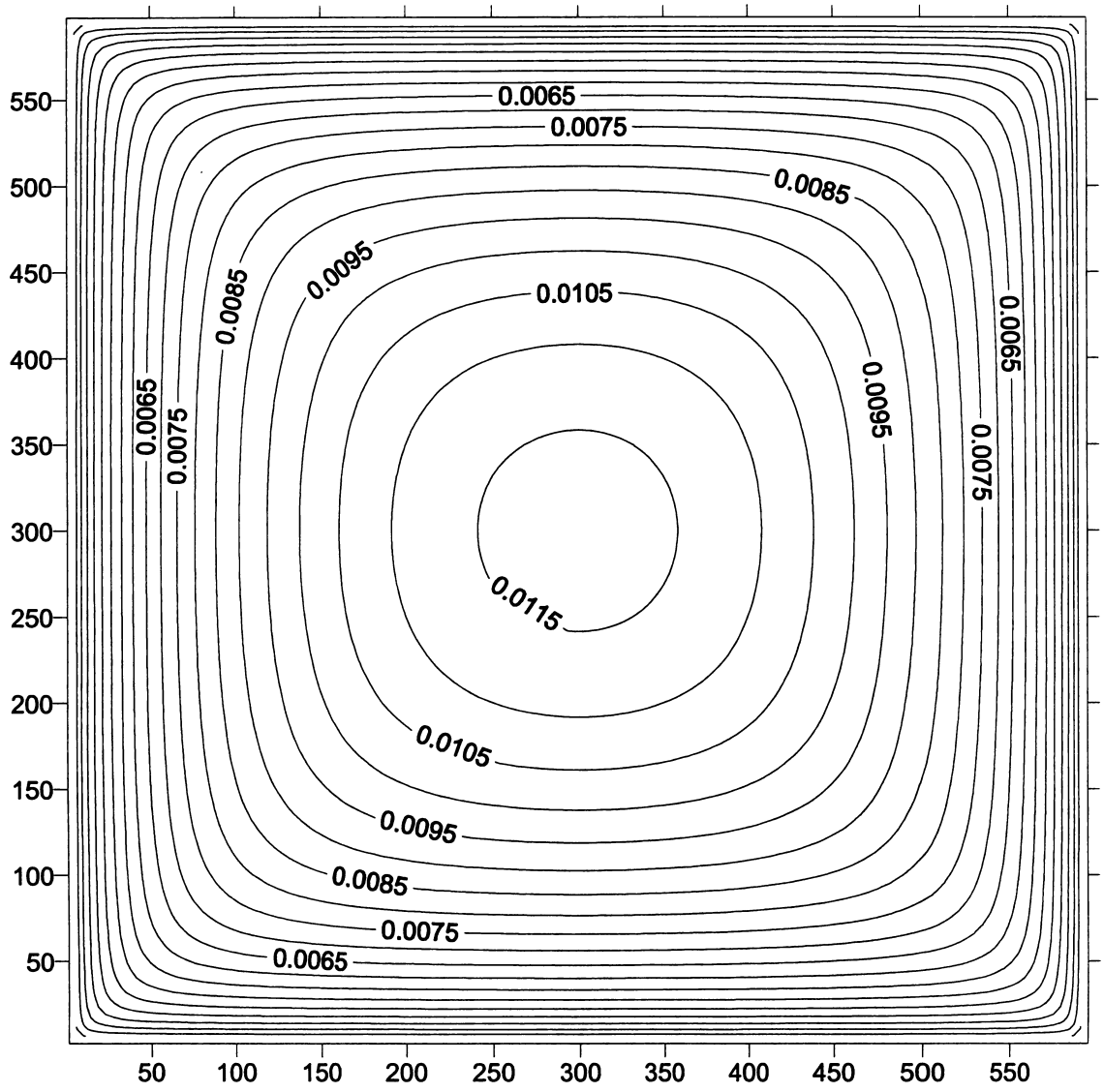
Mean longitudinal velocity for isotropic 500m integral scale



Standard deviations of longitudinal velocity for isotropic 500m integral scale



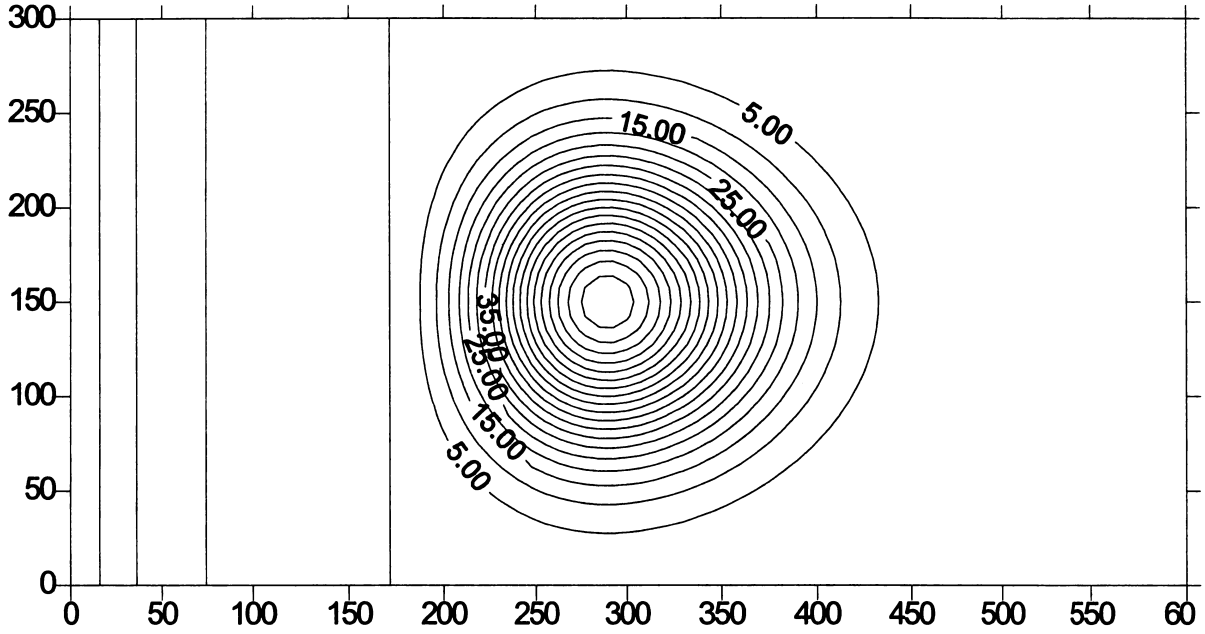
Mean transverse velocity for isotropic 500m integral scale



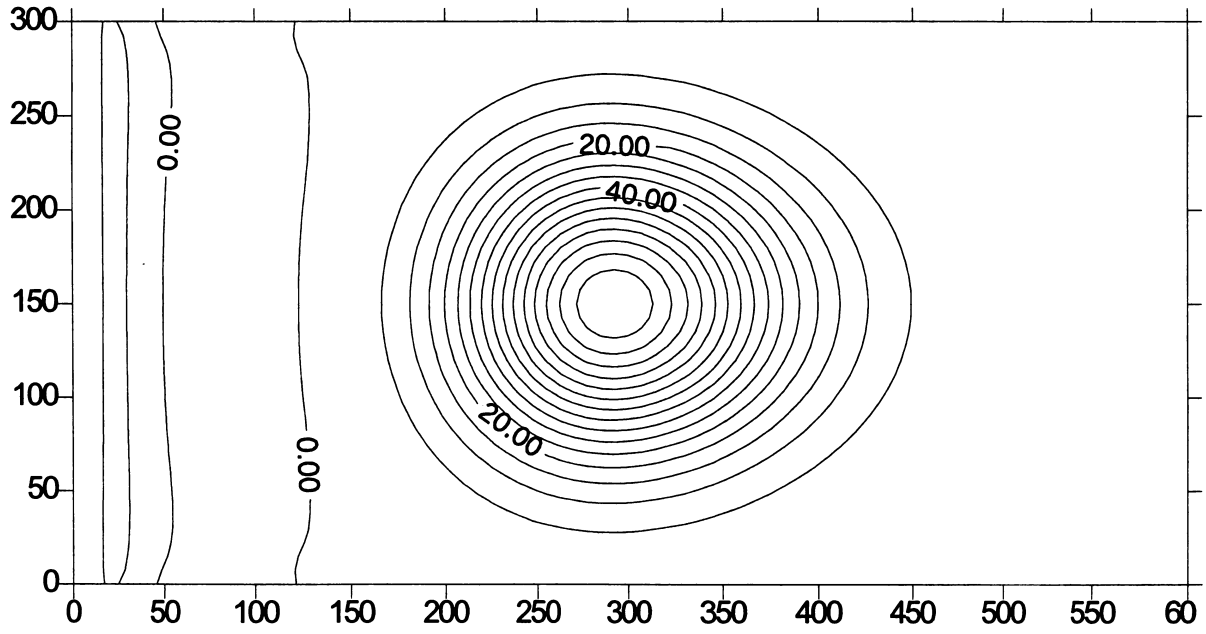
Standard deviation of transverse velocity for isotropic 500m integral scale

Appendix E Concentration Results

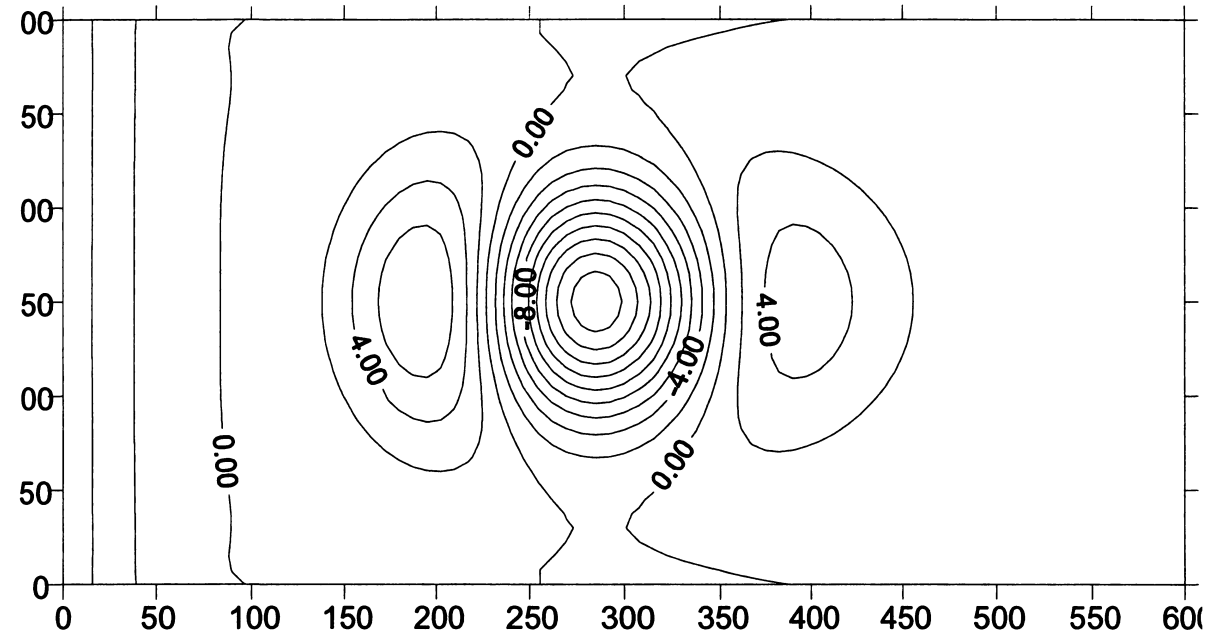
All data sets have 80 by 40 grid.



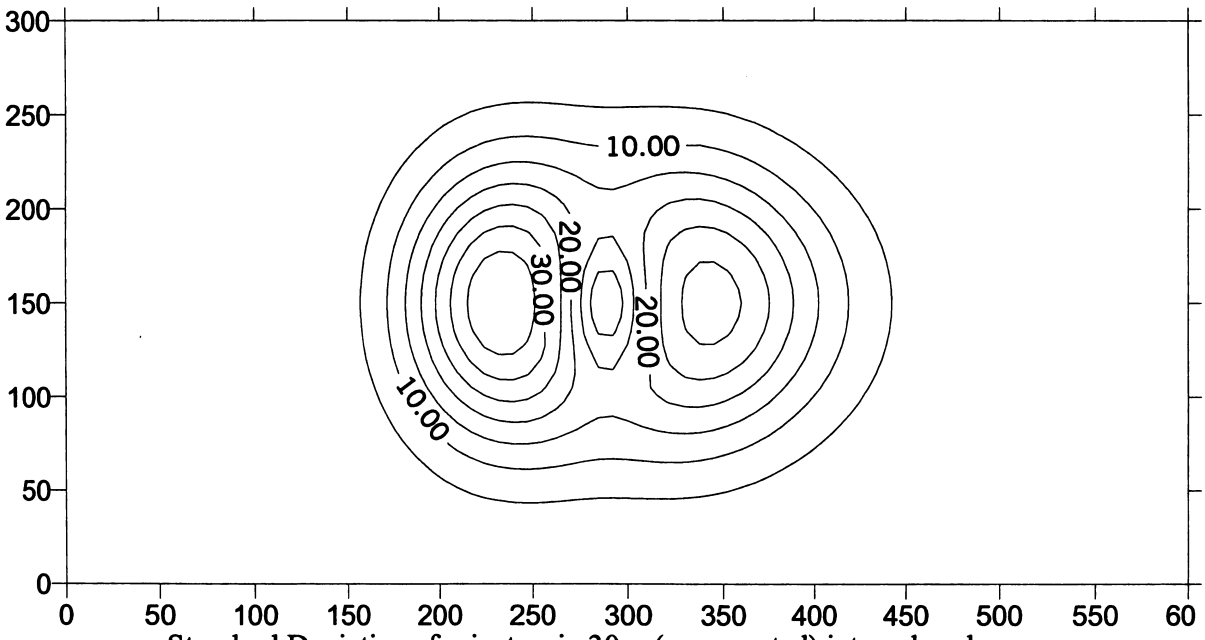
Deterministic Result



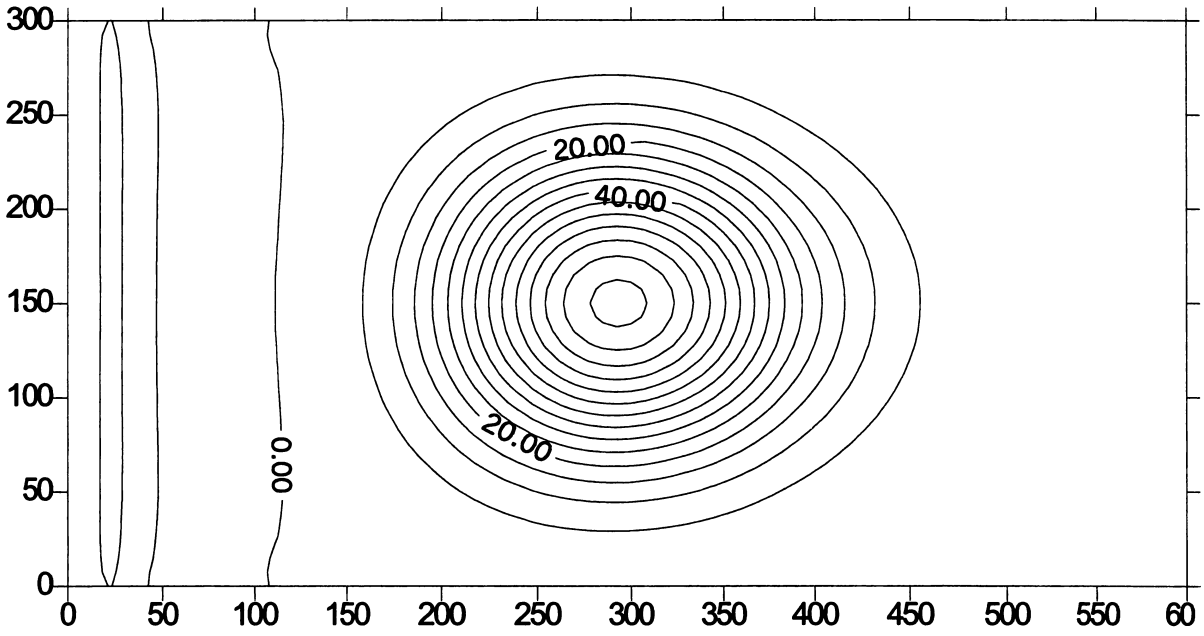
Mean concentration for isotropic 30m (uncorrected) integral scale



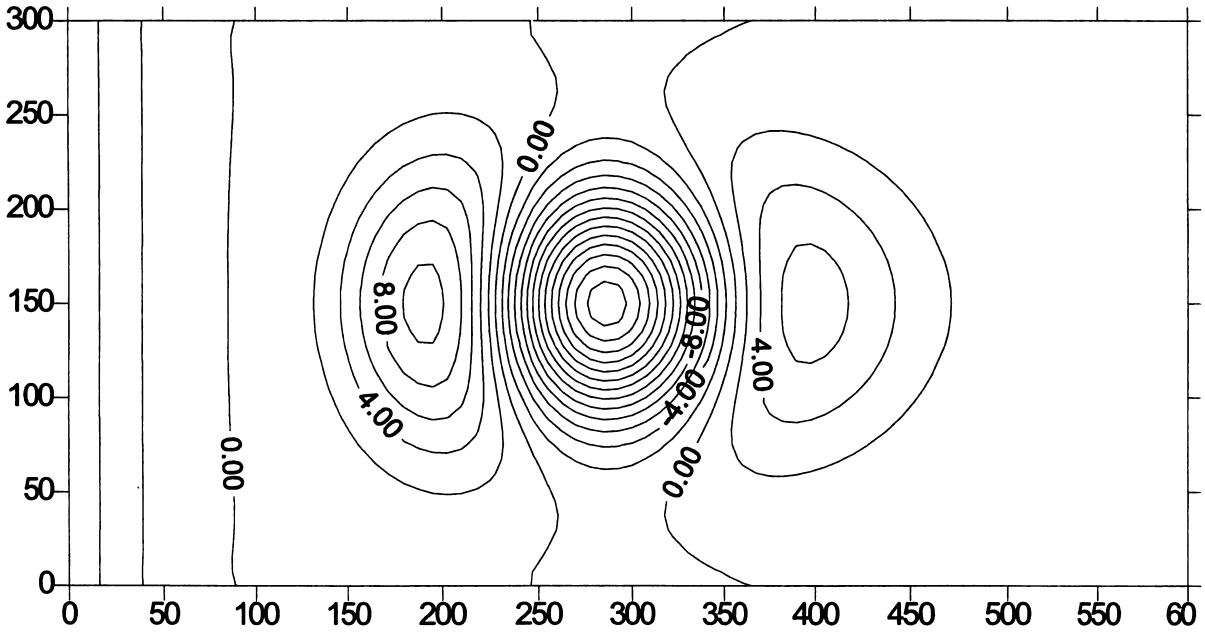
Difference between stochatistic mean concentration and deterministic result for isotropic 30m (uncorrected) integral scale



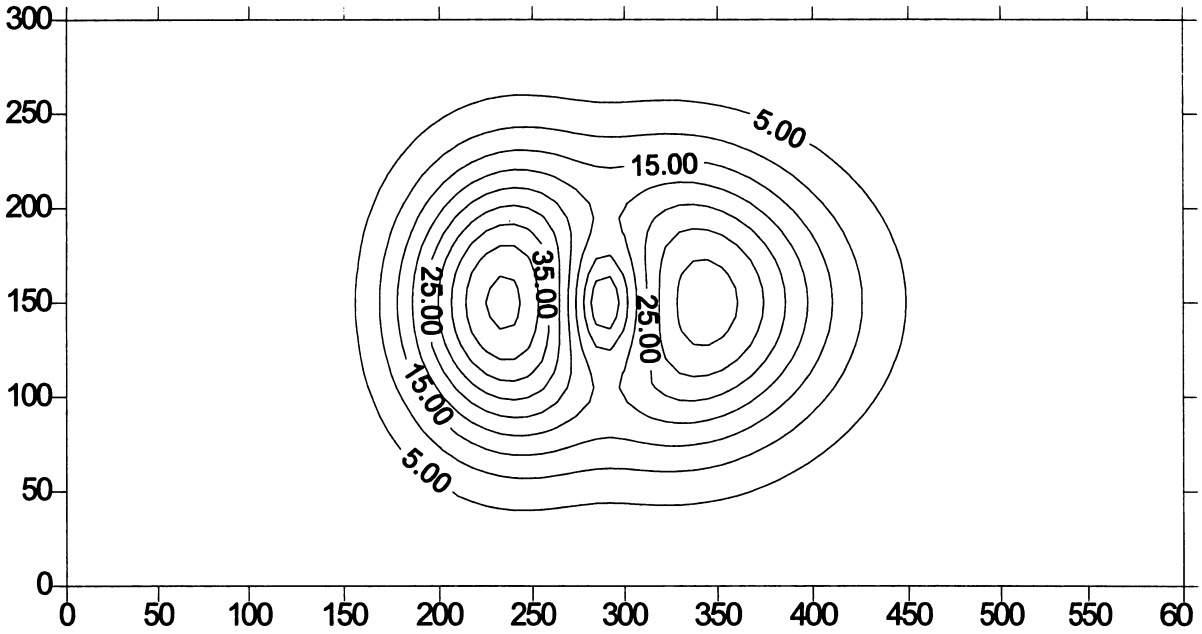
Standard Deviations for isotropic 30m (uncorrected) integral scale



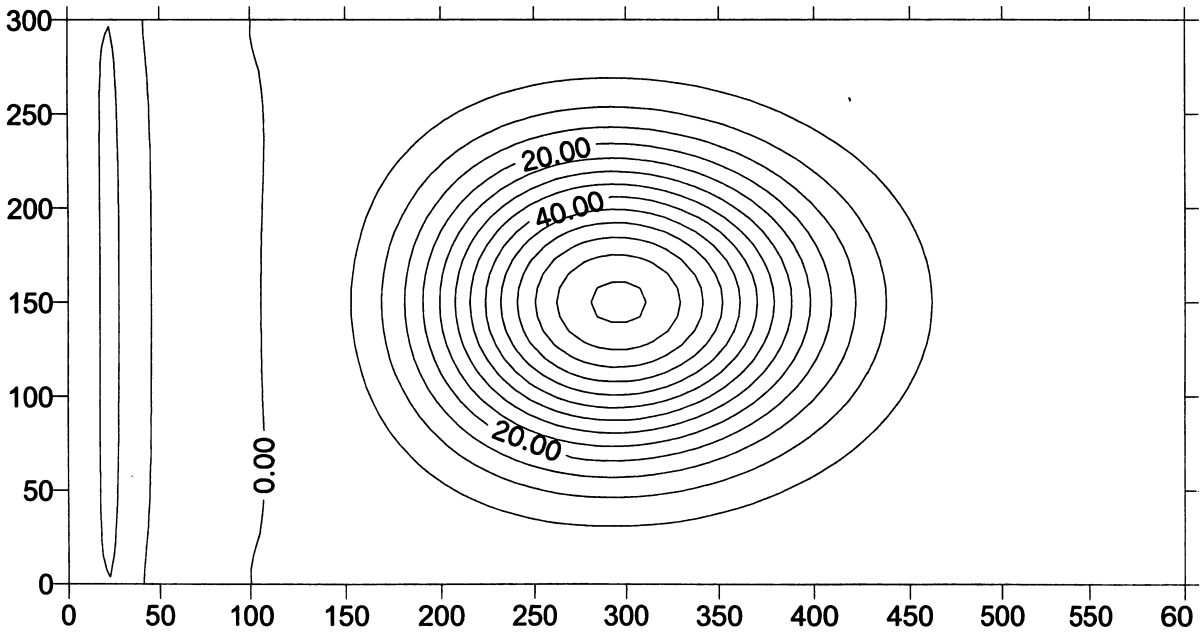
Mean concentration for isotropic 60m (uncorrected) integral scale



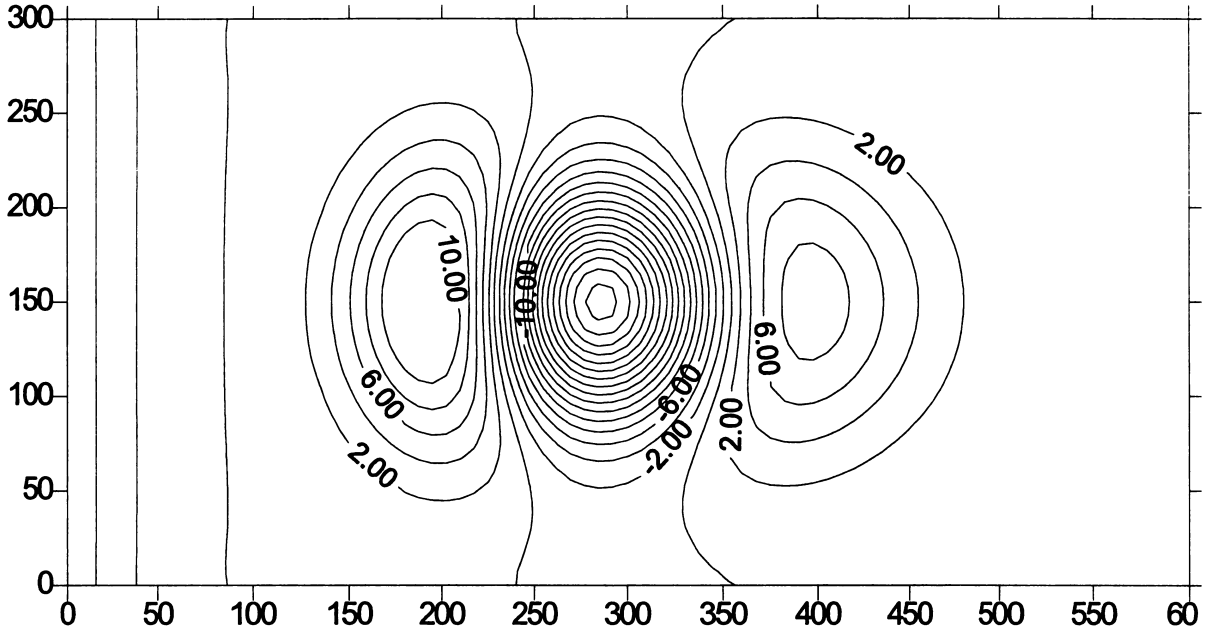
Difference between stochatistic mean concentration and deterministic result for isotropic 60m (uncorrected) integral scale



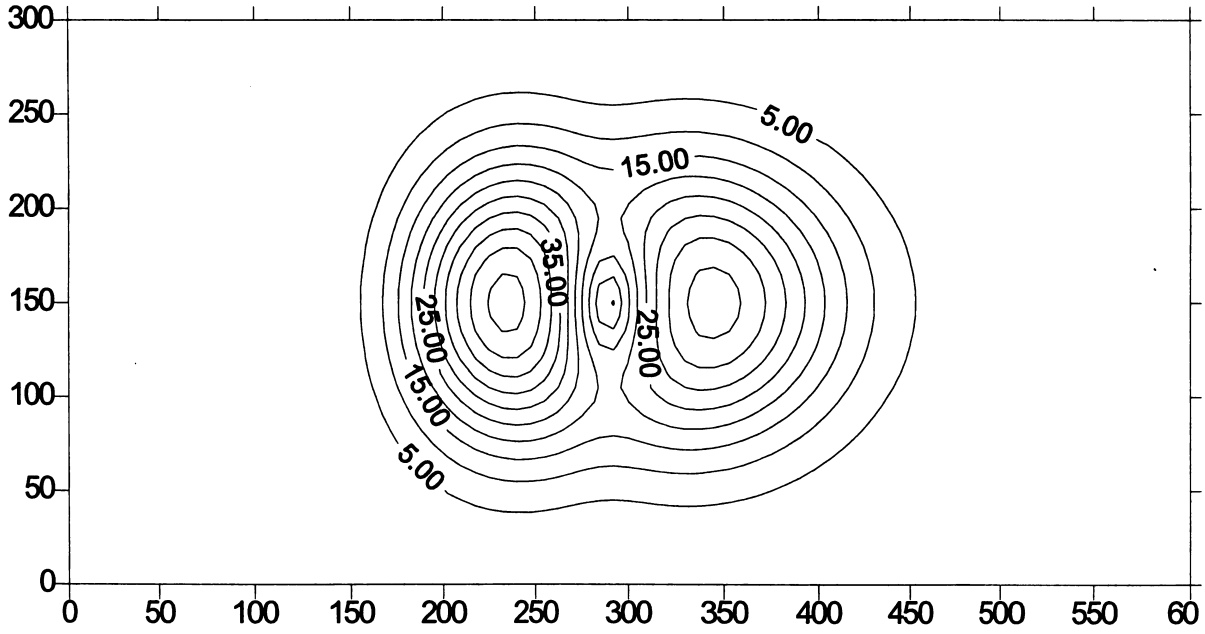
Standard Deviations for isotropic 60m (uncorrected) integral scale



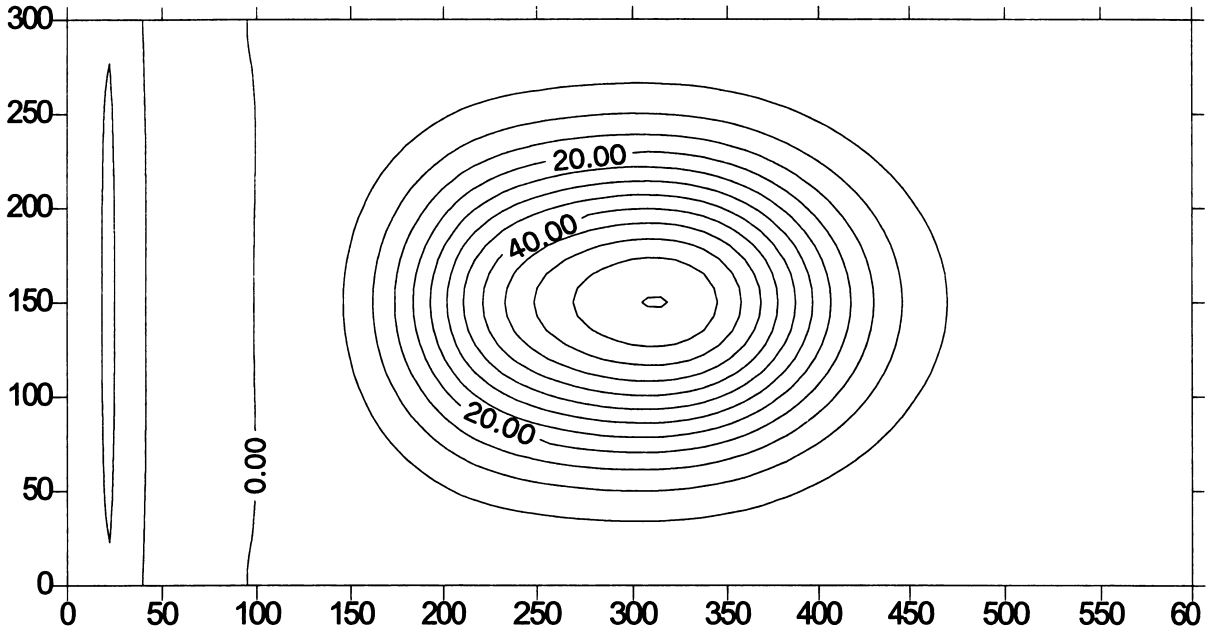
Mean concentration for isotropic 100m (uncorrected) integral scale



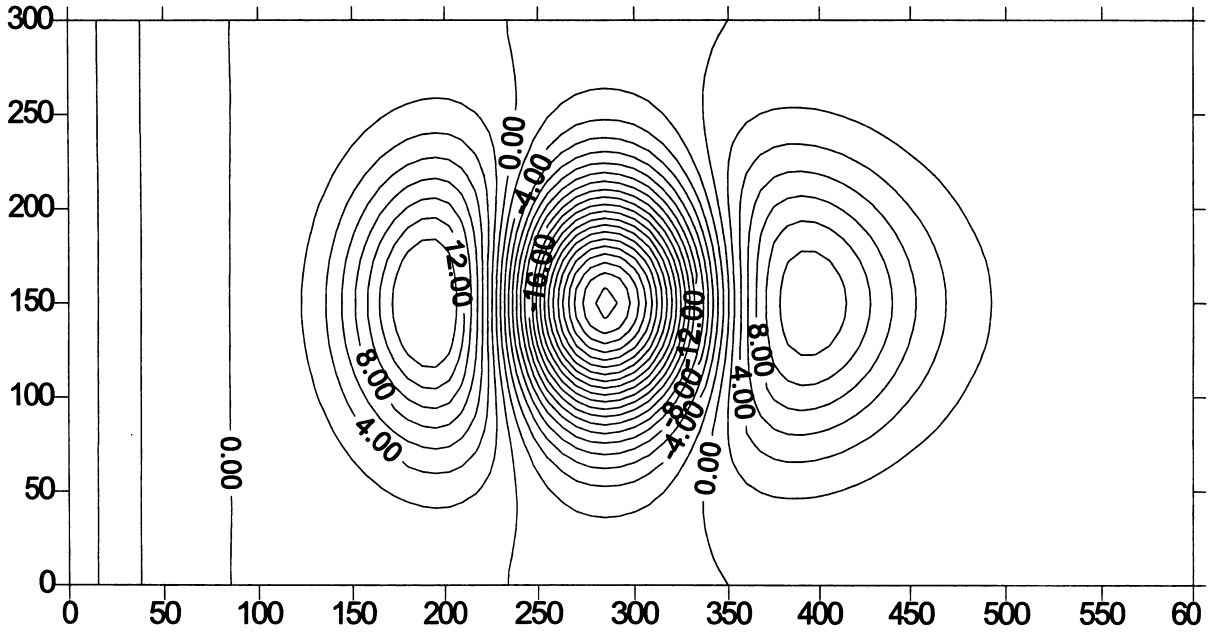
Difference between stochatistic mean concentration and deterministic result for isotropic 100m (uncorrected) integral scale



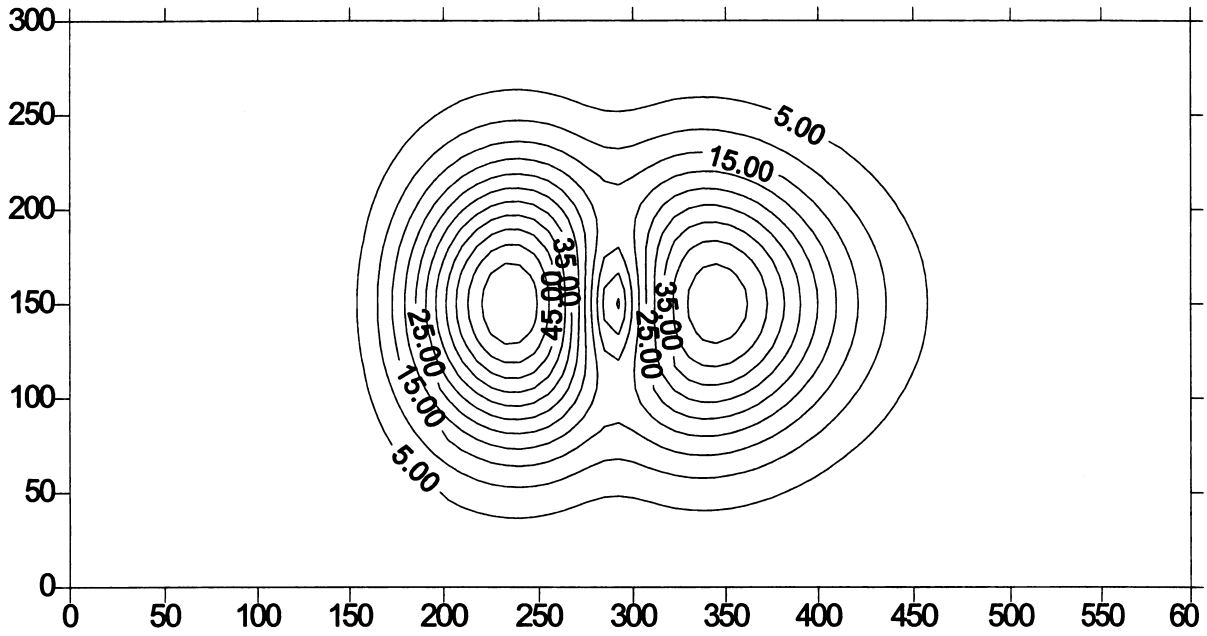
Standard Deviations for isotropic 100m (uncorrected) integral scale



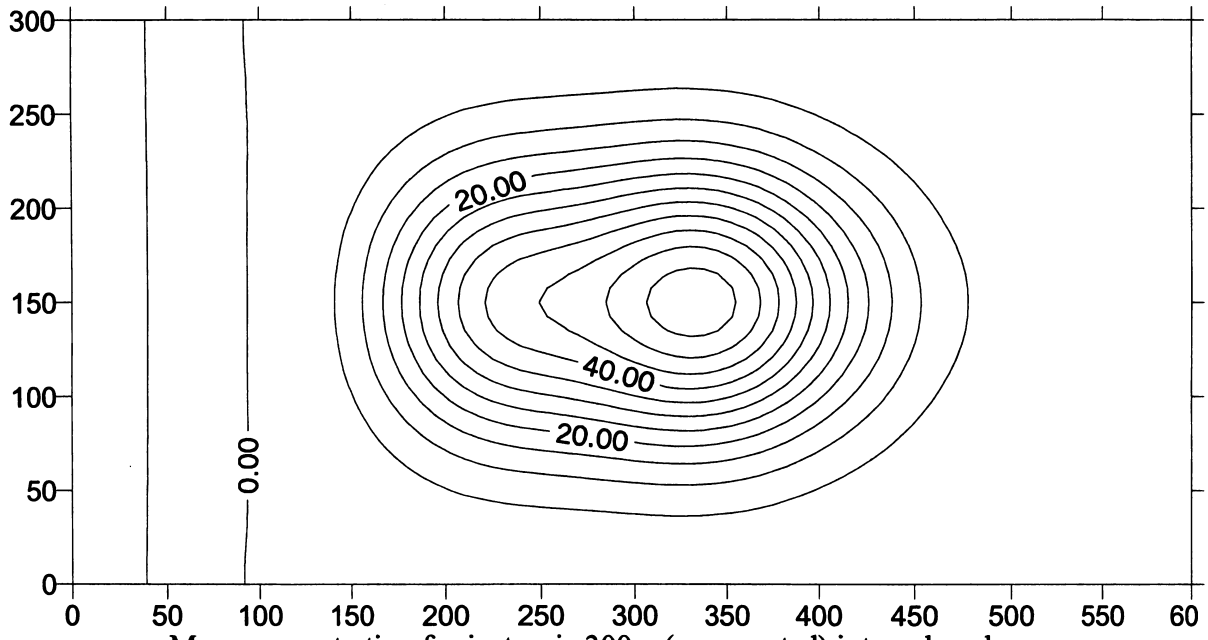
Mean concentration for isotropic 180m (uncorrected) integral scale



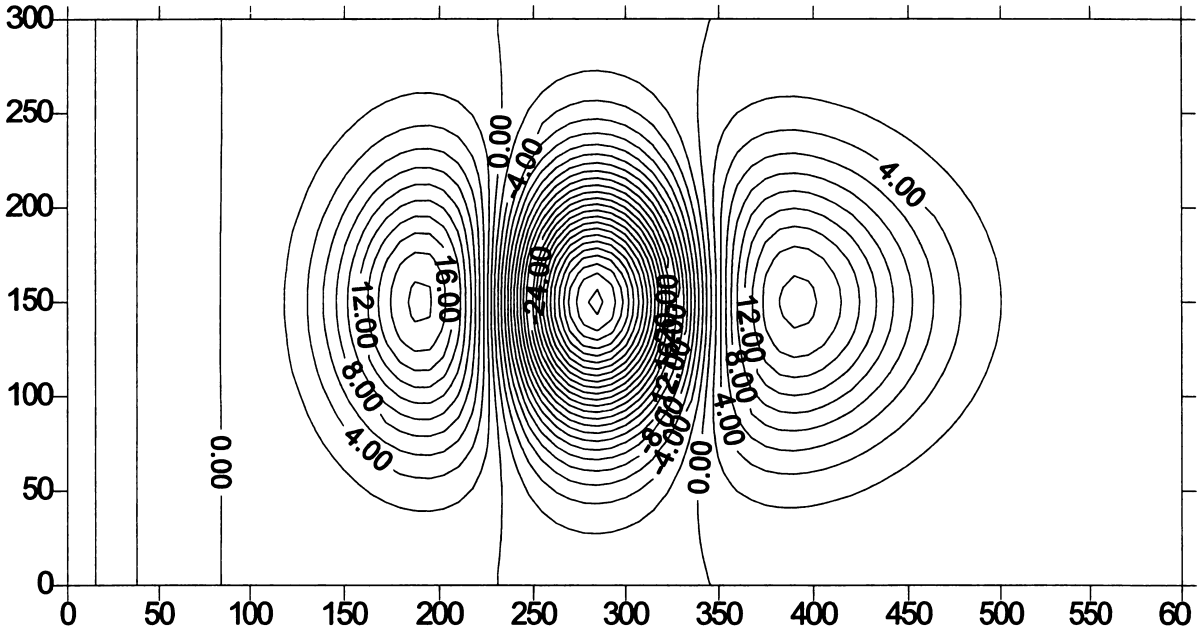
Difference between stochastic mean concentration and deterministic result for isotropic 180m (uncorrected) integral scale



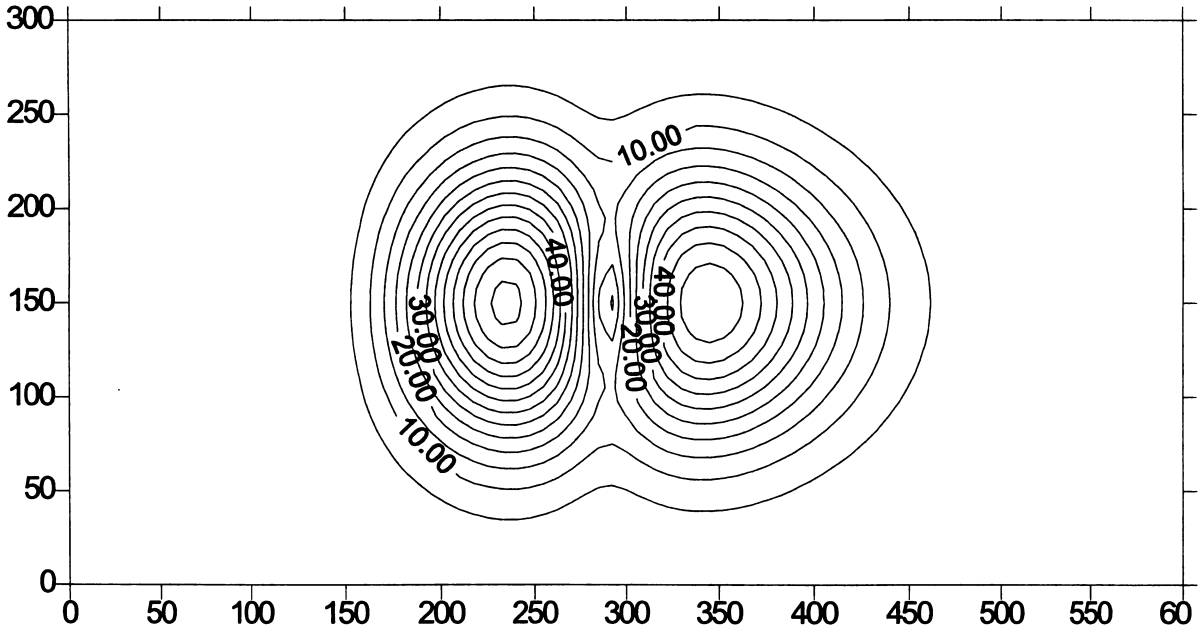
Standard Deviations for isotropic 180m (uncorrected) integral scale



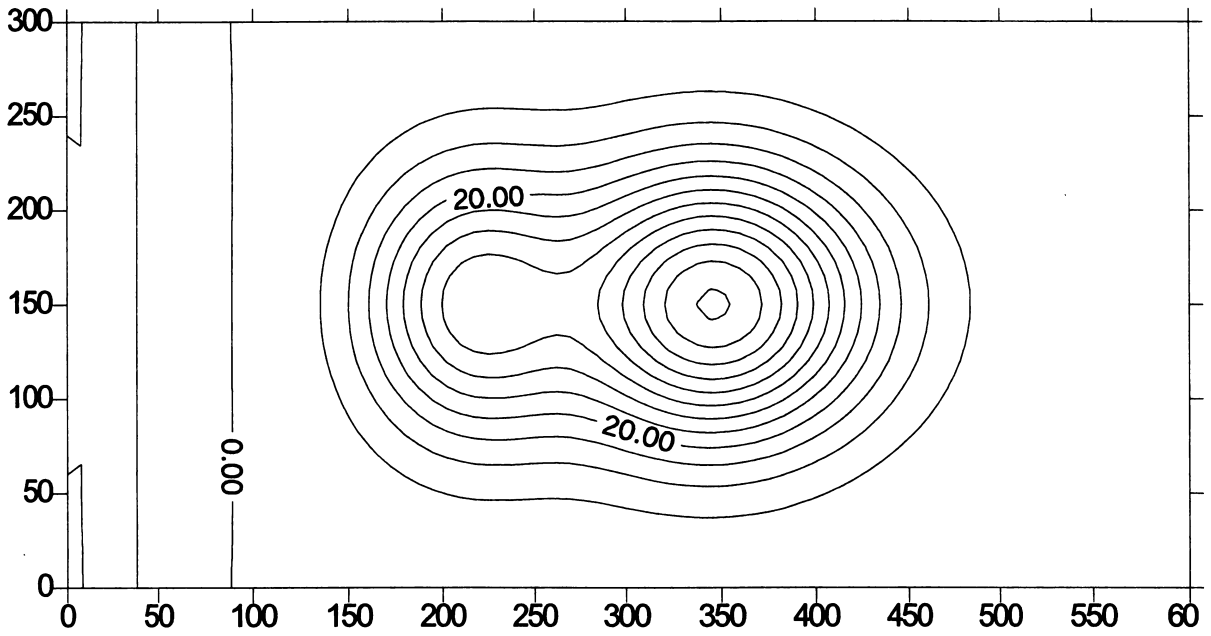
Mean concentration for isotropic 300m (uncorrected) integral scale



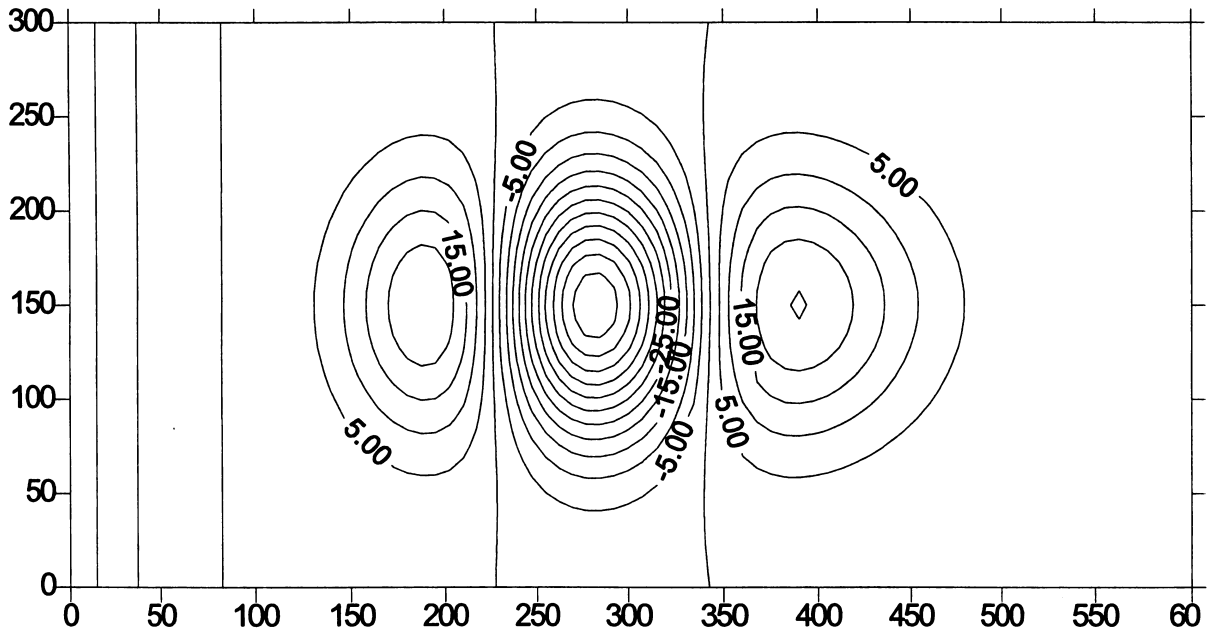
Difference between stochatistic mean concentration and deterministic result for isotropic 300m (uncorrected) integral scale



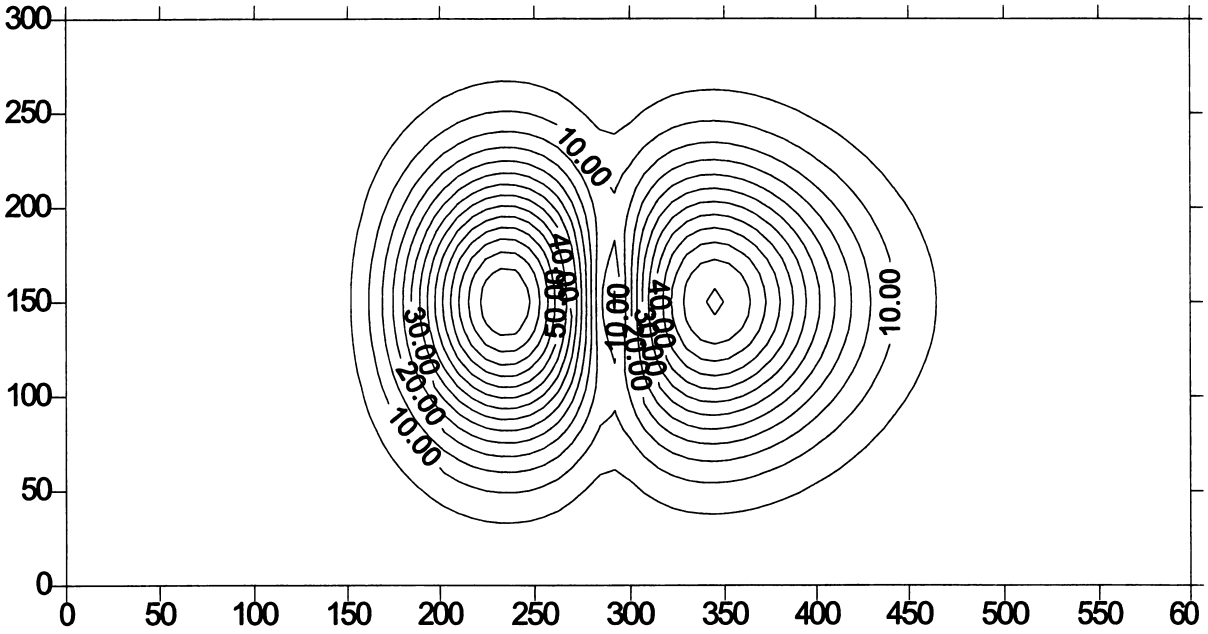
Standard Deviations for isotropic 300m (uncorrected) integral scale



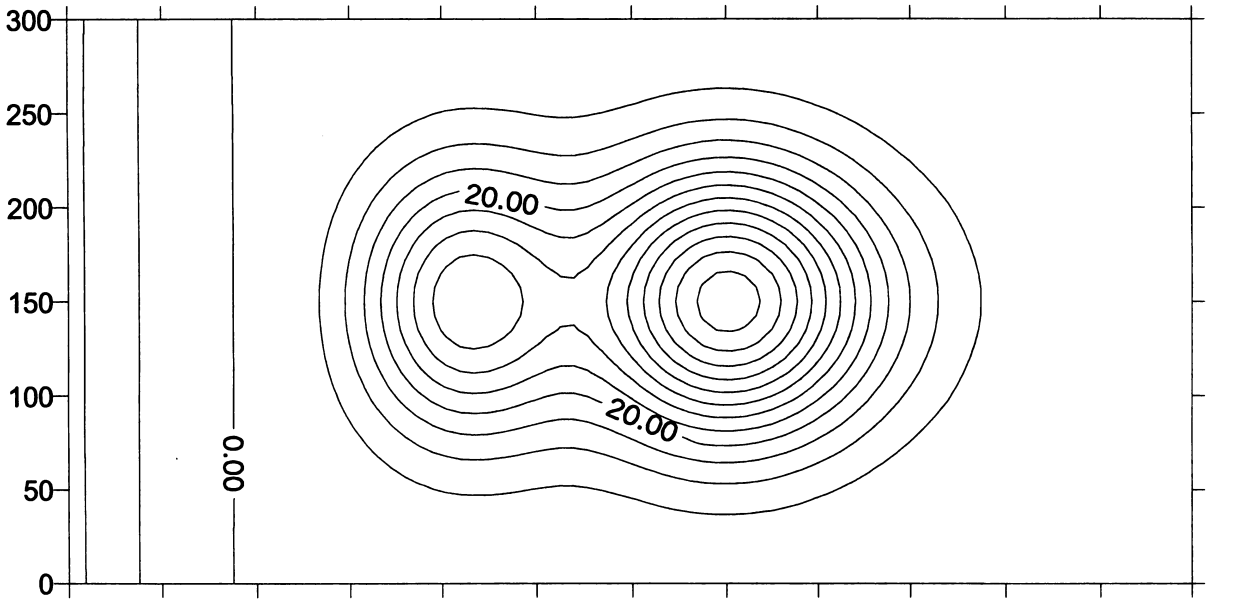
Mean concentration for isotropic 600m (uncorrected) integral scale



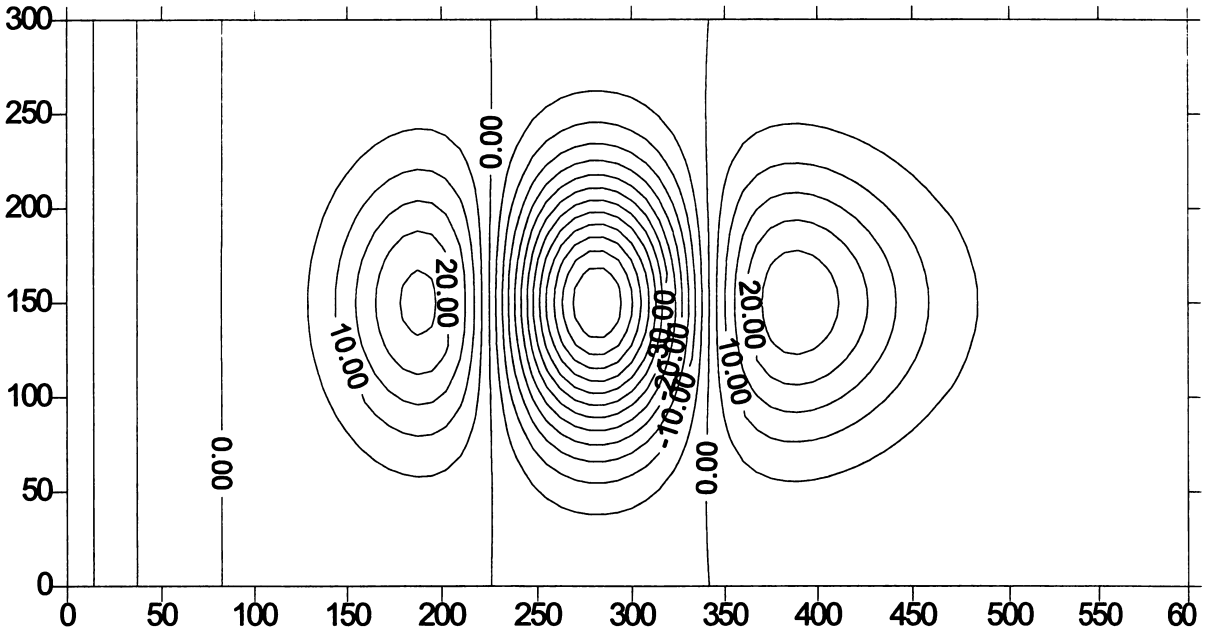
Difference between stochatistic mean concentration and deterministic result for isotropic 600m (uncorrected) integral scale



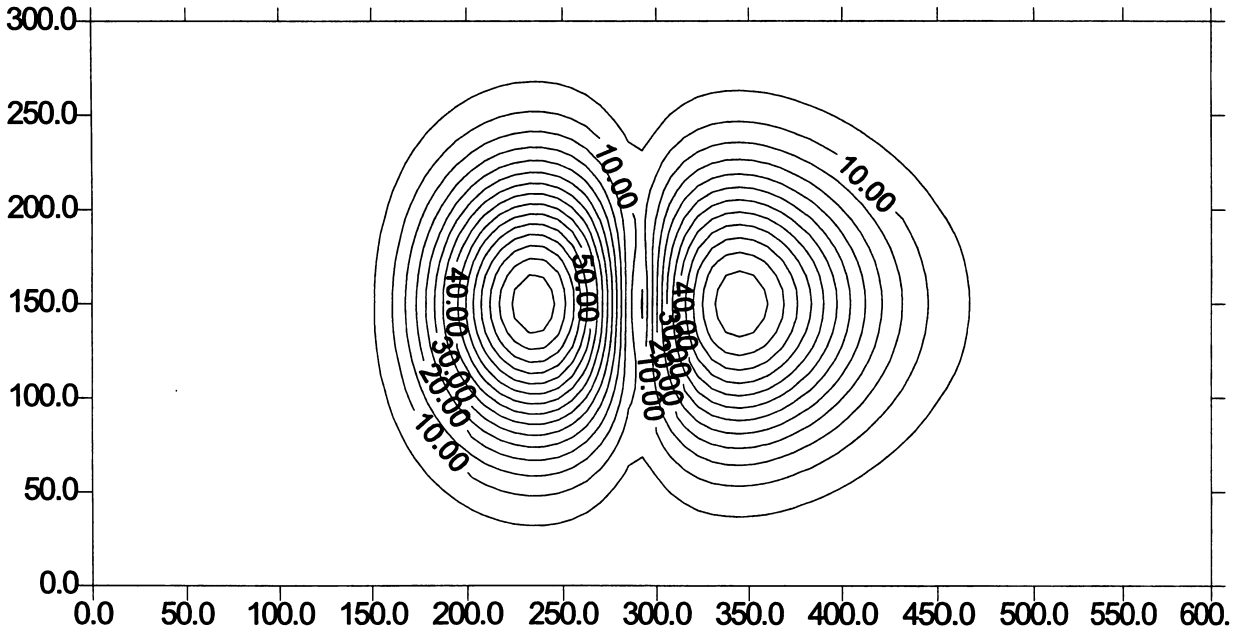
Standard Deviations for isotropic 600m (uncorrected) integral scale



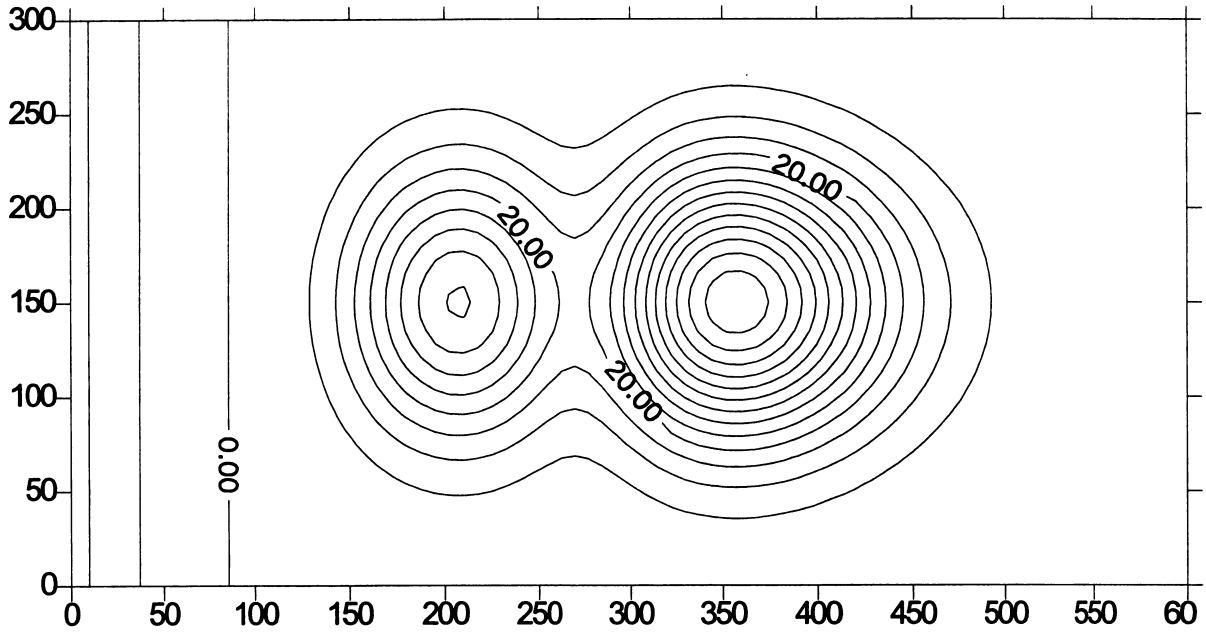
Mean concentration for isotropic 1000m (uncorrected) integral scale



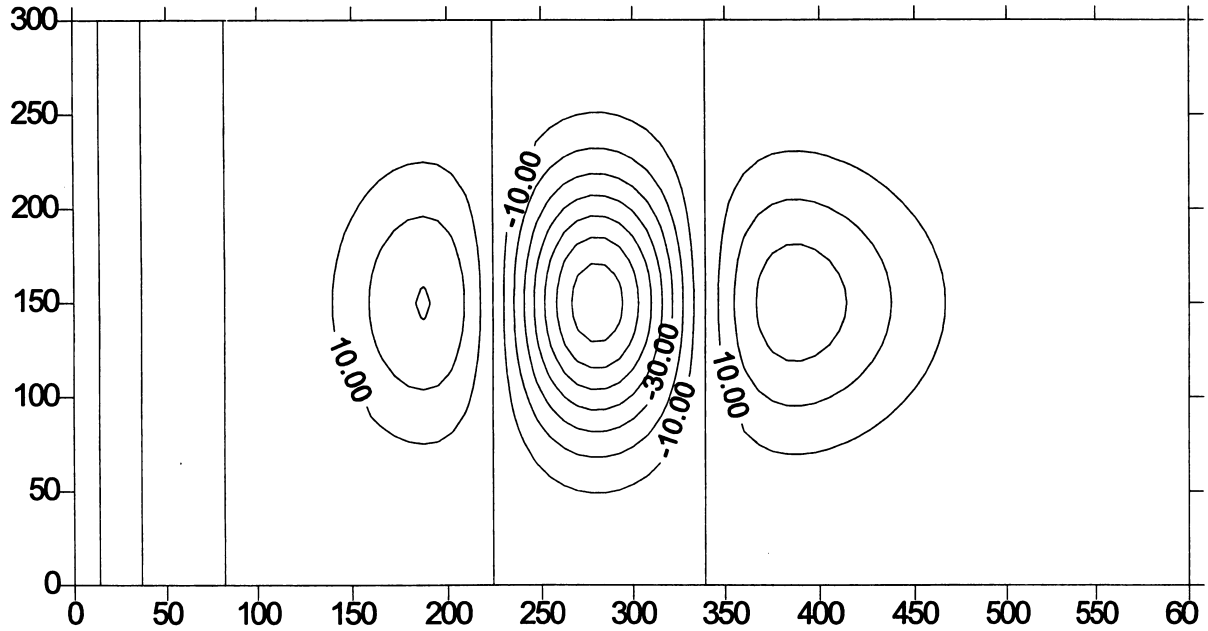
Difference between stochatistic mean concentration and deterministic result for isotropic 1000m (uncorrected) integral scale



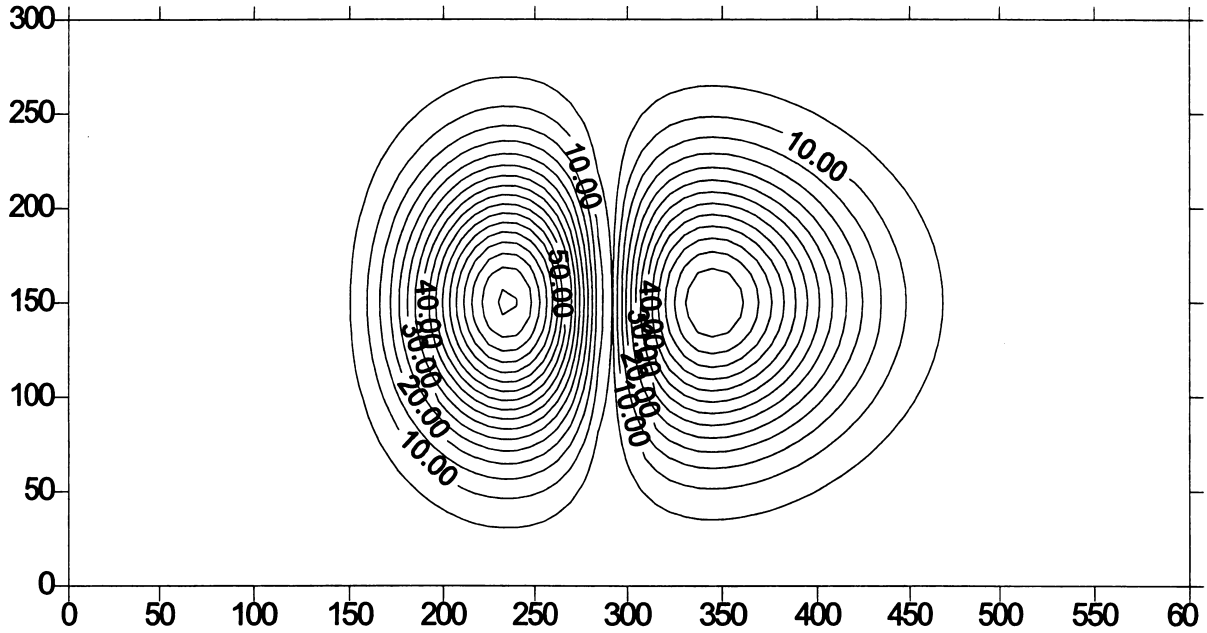
Standard Deviations for isotropic 1000m (uncorrected) integral scale



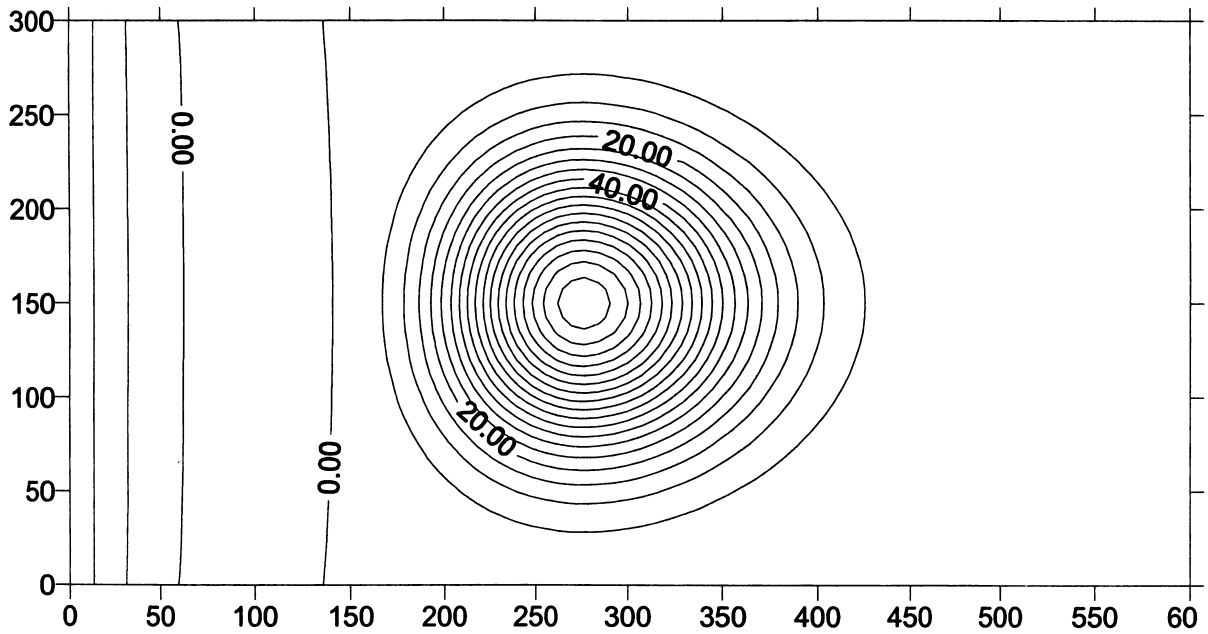
Mean concentration for isotropic 1,000,000,000m (uncorrected) integral scale



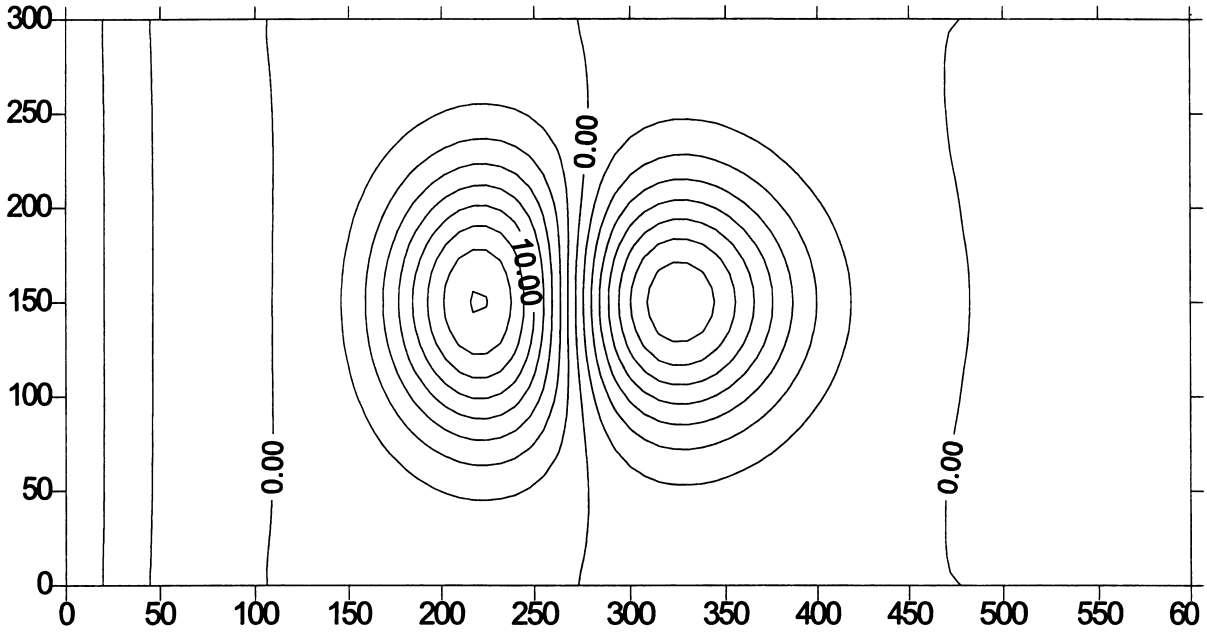
Difference between stochatistic mean concentration and deterministic result for isotropic 1,000,000,000m (uncorrected) integral scale



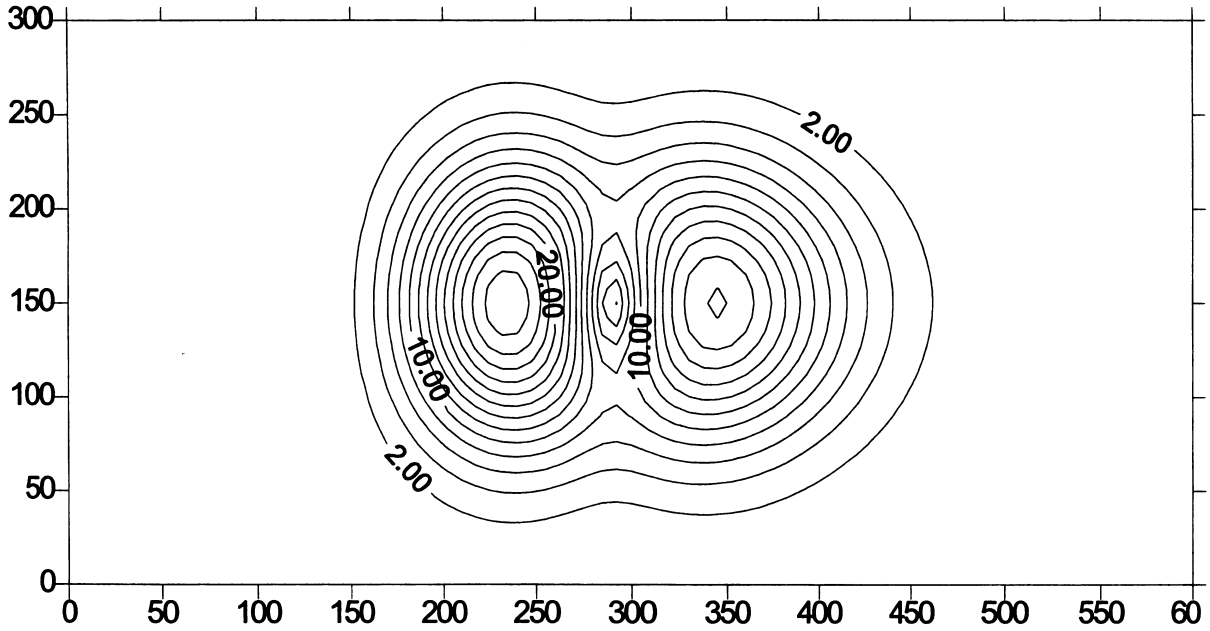
Standard Deviations for isotropic 1,000,000,000m (uncorrected) integral scale



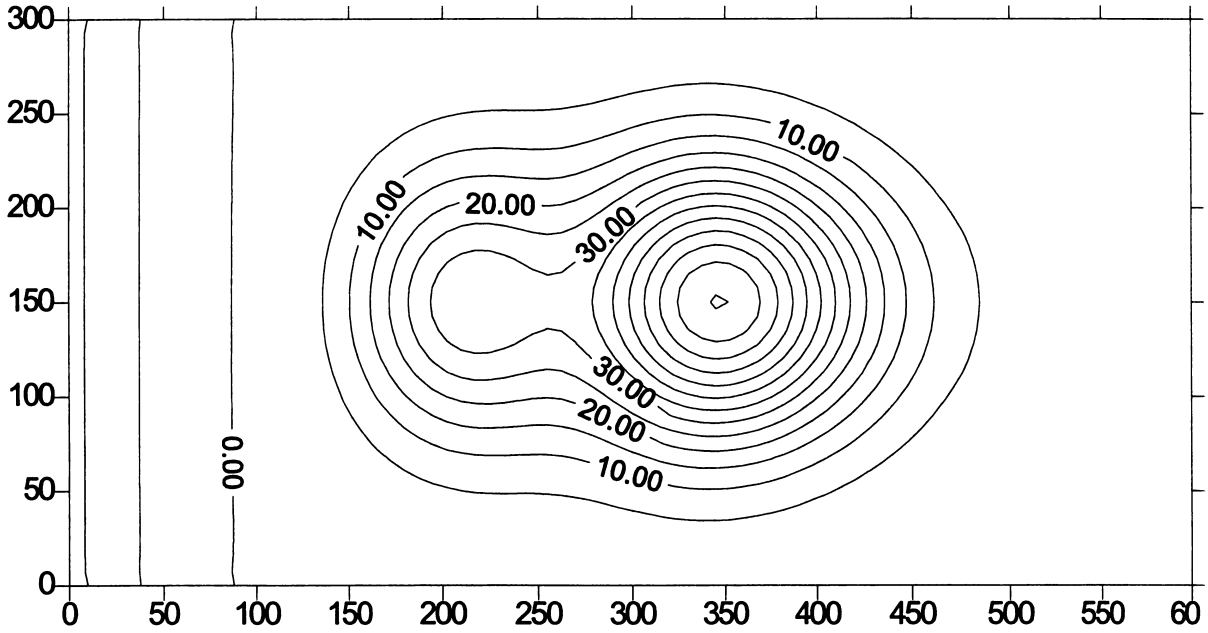
Mean concentration for anisotropic (uncorrected) integral scale.
Longitudinal (x) integral scale = 30m, Transverse (y) integral scale = 300m



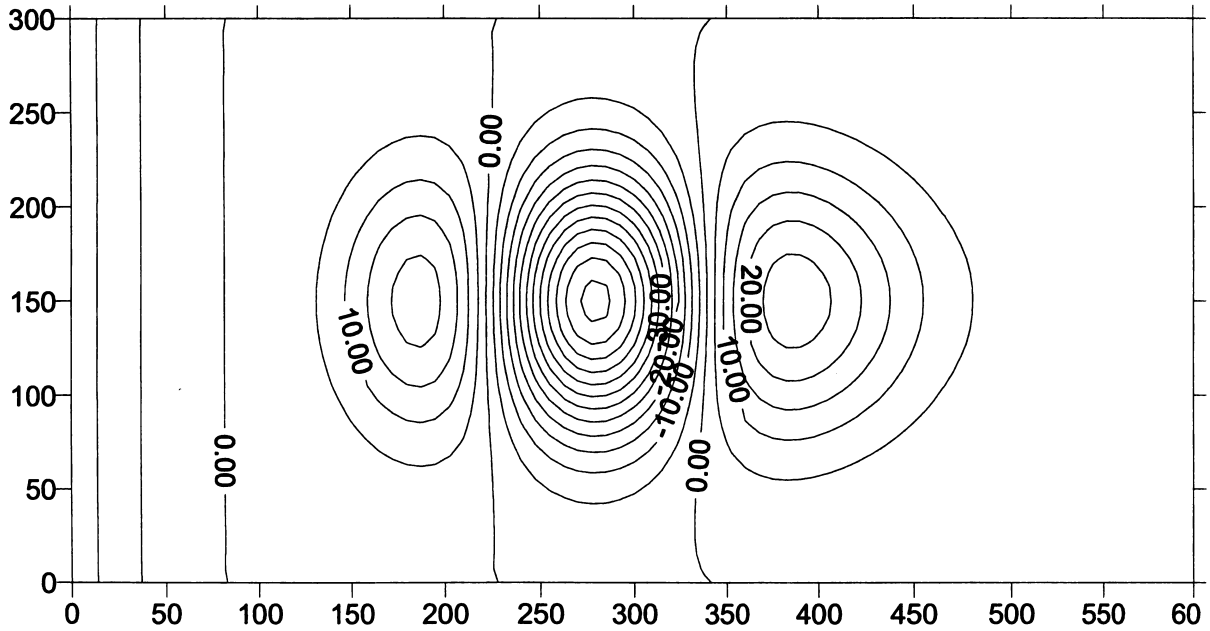
Difference between stochatistic mean concentration and deterministic result for anisotropic (uncorrected) integral scale
Longitudinal (x) integral scale = 30m, Transverse (y) integral scale = 300m



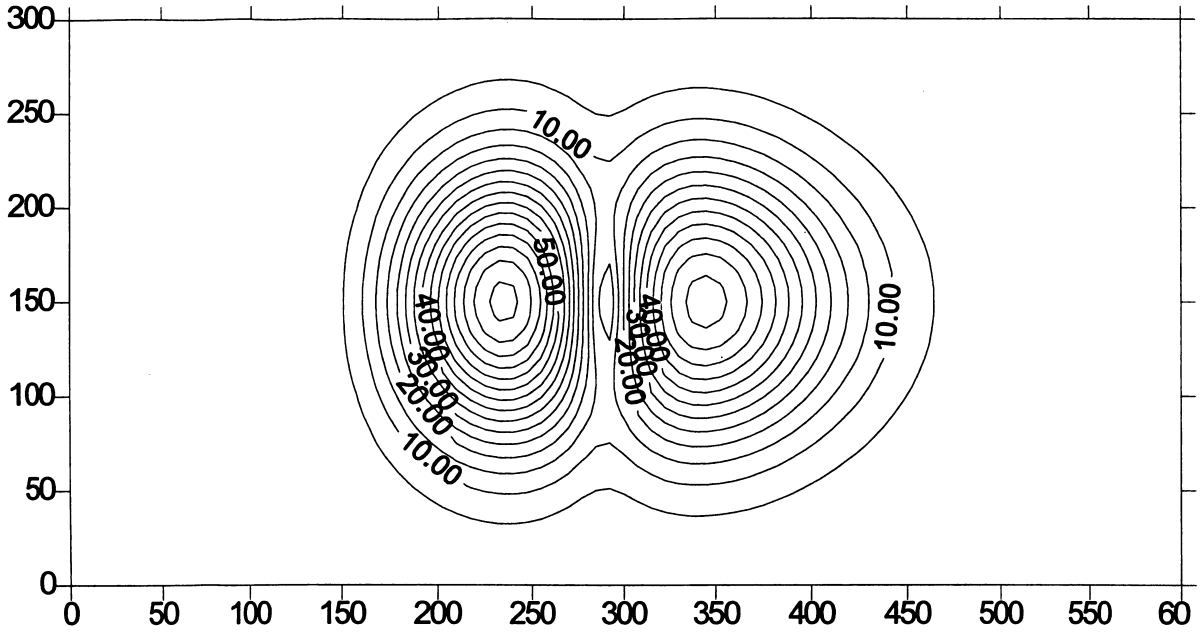
Standard Deviations for anisotropic (uncorrected) integral scale
Longitudinal (x) integral scale = 30m, Transverse (y) integral scale = 300m



Mean concentration for anisotropic (uncorrected) integral scale.
Longitudinal (x) integral scale = 300m, Transverse (y) integral scale = 30m



Difference between stochatistic mean concentration and deterministic result for anisotropic (uncorrected) integral scale
Longitudinal (x) integral scale = 300m, Transverse (y) integral scale = 30m



Standard Deviations for anisotropic (uncorrected) integral scale
Longitudinal (x) integral scale = 300m, Transverse (y) integral scale = 30m