# AQUIFER DECONTAMINATION BY PUMPING IN RADIAL AND ONE DIMENSIONAL UNIFORM FLOW FIELDS

INDEPENDENT STUDY OF

GREG D. WOODSIDE

SUBMITTED IN PARTIAL FULFILLMENT

OF THE DEGREE OF

MASTER OF SCIENCE

NEW MEXICO INSTITUTE OF MINING AND TECHNOLOGY

MAY 11, 1988

## ABSTRACT

Decontamination of polluted aquifers is an important environmental concern. Pumping from withdrawal wells or french drains to extract dissolved solutes is a possible remediation technique. Two mathematical models for aquifer decontamination using a single withdrawal well with radially converging flow and a french drain with one dimensional flow are developed. In the radial model, the well is taken into account as a mathematical sink located at the center of the plume which is assumed to be radially symmetric. The plume is incorporated into the model as an initial condition capable of representing a wide range of plume geometries. For the one dimensional model, two simple initial conditions of a uniform concentration given by a step function and a sloping straight line are used with a constant dispersion coefficient. Both models assume advection and longitudinal mechanical dispersion as the transport mechanisms. Solutions for the one dimensional model are obtained using the Green's function approach and LaPlace transform following the same methodology as Chen and Woodside (1988) used in deriving the solution for the radial model. The radial solution agrees well with the approximate solution of Gelhar and Collins (1971). Using the field data of Pickens and Grisak (1981), the radial model also accurately reproduces the concentration history at the withdrawal well of the single well injection–withdrawal tracer test. When the initial conditions in the two models are formulated with large concentration gradients at the plume boundary, adverse dispersion against the converging groundwater flow causes spreading of solutes beyond the original plume boundary. If the initial conditions gradually decrease to zero concentration at the plume boundary, solutes do not extend beyond the region of original contamination during the withdrawal process. Graphical relationships are developed for estimating the time required to decrease the concentration to 1% of the initial maximum concentration. In general, neglecting dispersion underestimates the total cleanup time. Comparisons between the two models for decontamination efficiency are made using the criteria of decontaminating in equivalent time or withdrawing water at equivalent rates from the single pumping well and the drain. The radial and one dimensional models yield different results over the range of Peclet numbers tested and the use of the correct model is thus important.

## ACKNOWLEDGEMENTS

The author wishes to express his gratitude to Dr. Chia-Shyun Chen for the opportunity to work on this research project and for his support. Financial support for the research project was through the National Science Foundation grant ECE-8696079 to Dr. Chen and the author is grateful for this assistance. Also, the assistance of Dr. Bill Stone and Dr. Allan Sharples from the Mathematics Department from New Mexico Tech is greatly appreciated. I also wish to thank my friends at New Mexico Tech for their encouragement, with special thanks to Li-Wei Chiang.

## TABLE OF CONTENTS

# INTRODUCTION

## STATEMENT OF THE PROBLEM

In the past few decades, groundwater pollution problems have become an important environmental concern. Approximately fifty percent of the United States drinking water needs are supplied by groundwater. A number of pollutant sources can create groundwater contamination problems. These include underground storage tanks, surface impoundments, landfills, waste piles, and a multitude of sources due to improperly stored wastes. The U. S. Environmental Protection Agency estimates that of the 3.5 million or more underground storage tanks in the United States, 10–30% may be leaking (Dowd, 1984). Comprehensive data regarding the overall extent of groundwater pollution is not available. Estimates range from 1–2% but the true extent may be much higher due to the complexity of quantifying groundwater pollution on a nationwide scale (Office of Tech. Assessment, 1984). Even if the percentage of contaminated groundwater is small, the hazard it poses is significant because much of the contamination occurs in heavily populated regions where the consumption of groundwater is high.

Much attention has been devoted to understanding and quantifying the behavior of contaminants in the subsurface. This effort includes studying the chemical, biological and physical mechanisms by which pollutants are transformed and transported in groundwater. Most of the attention to date has been devoted to how these mechanisms affect a pollutant already in the subsurface and have not considered how a contaminated aquifer may be rehabilitated. The same physical processes which govern how a contaminant is transported in the subsurface away from an pollution source also govern how the contaminant travels toward a region where it is removed. Present theories of contaminant transport can thus be modified and applied to quantify the transport of polluted groundwater to withdrawal systems.

Mechanical remediation by pumping is one of the numerous methodologies that exist for rehabilitating contaminated aquifers. Contaminated groundwater can be removed by pumping from withdrawal wells, french drains, or infiltration galleries. A hypothetical remediation scheme is depicted in Figure 1 where a single pumping well is constructed to cleanup a contamination site. The plume shown in Figure 1 is idealized because it is shown as raddially symmetric and as having a uniform decline in the concentration away from the maximum where the well is located. Note that groundwater flow would be radial to the well and the velocity profile which is important for predicting contaminant transport can be approximated with present theories of well hydraulics. A french drain is a withdrawal system in which a perforated pipe is placed in an excavated trench and backfilled with permeable material. The pipe thus acts as a groundwater collector and the water flowing into the pipe can be removed. This is depicted in Figure 2 with a hypothetical contaminant plume. Note that flow can generally be assumed to be one dimensional to the drain except near the ends of the system. An alternative to the french drain is an infiltration gallery where a trench is dug below the water table and water then flows into

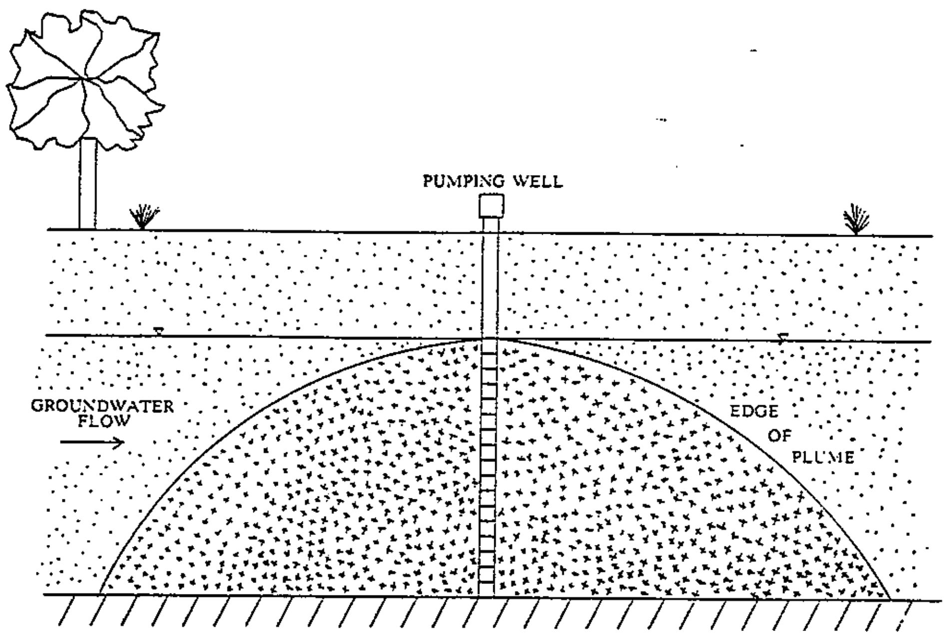



Figure 1 Generalized plume profile with withdrawal well at maximum concentration

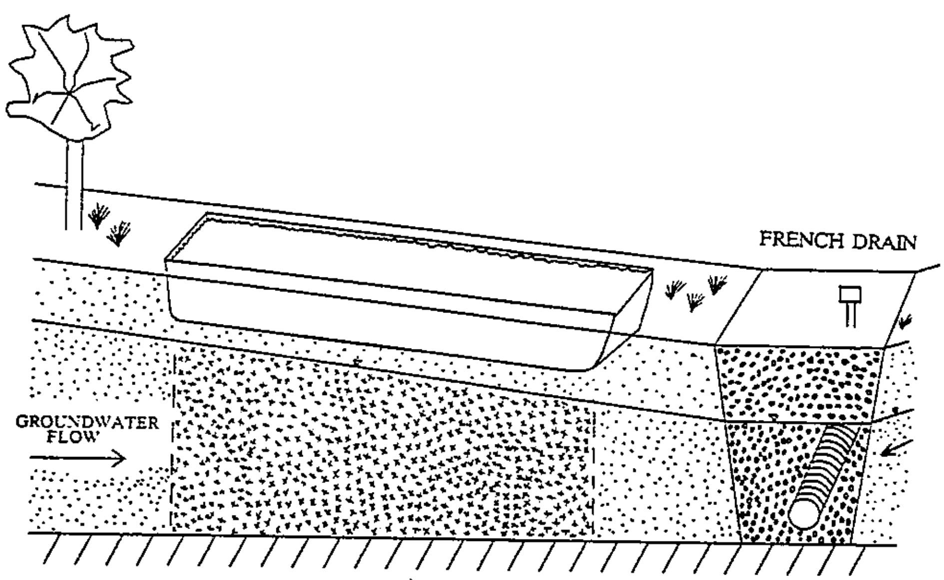



Figure 2 Decontamination of aquifer polluted by waste pond in one dimensional flow using french drain

the trench from which it is then pumped. Depending on the depth of the trench and the stability of the trench walls, it may be backfilled with permeable material or simply left open. Groundwater flow will again be approximately one dimensional to the infiltration gallery as in Figure 2 except near the ends of the trench. The usefulness of the french drain and infiltration gallery is that both are limited to shallow groundwater contamination problems since both withdrawal systems cannot be extended to large depths. In a natural groundwater flow system, one dimensional flow can also occur to constant head boundaries such as rivers or lakes. Figure 3 depicts a contaminant plume in this situation.

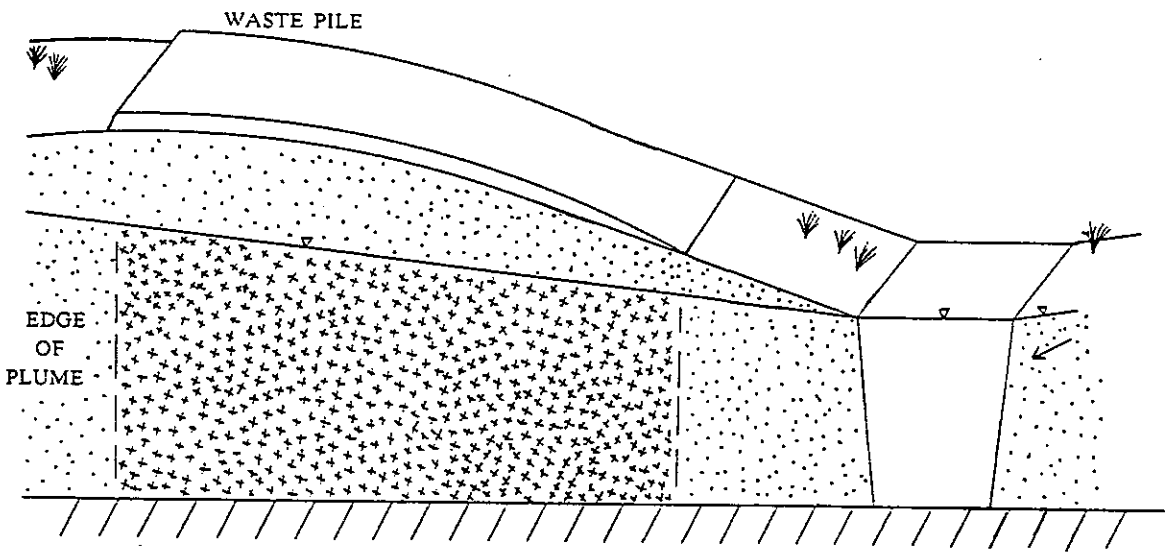



Figure 3 Transport of polluted groundwater in one dimensional flow to river

## PREVIOUS WORK

By neglecting dispersion, Kirkham and Affleck (1977) determined travel times to wells for solute transport by advection for three cases. These were a withdrawal well in a homogeneous aquifer, a withdrawal well with a region of low conductivity near the well, and a well fed by a river. Gelhar and Collins (1971) presented solutions for solute transport due to cyclic injection and withdrawal at a single well. Their solution was derived using perturbation methods and does not impose a boundary condition at the well. It is not formally valid unless the solute front has travelled on the order of fifty times the dispersivity. Using the method of inner and outer expansions, Eldor and Dagan (1972) also developed approximate solutions for dispersion in two dimensional flow. The specific cases were for diverging radial flow from a recharging well, a recharging well in a uniform flow field, and soil leaching due to uniform recharge. Phillips and Gelhar (1978) studied the transport to a partially penetrating well of a solute released from an upper aquitard. Through numerical techniques, travel time formula derived by neglecting dispersion, and the use of Gelhar and Collins (1971) solution, they studied aquifers in Long Island, New York with and without lower impervious boundaries where the solute released from the upper aquitard posed pollution hazards to water supply wells. To study the transport of a solute toward a withdrawal well due to an input at a finite distance from the well, Al-Niami and Rushton (1978) assumed a constant dispersion coefficient in developing two analytical solutions. The solutions are for a step change and an exponential increase at the boundary and can only be evaluated for certain nondimensional discharge rates. Guvanesen and Guvanesen (1987) developed approximate solutions for withdrawing tracers from a tracer ring created by an injection well. Using time-moment analysis, Valocchi (1986) analysed the validity of the local equilibrium assumption for radial dispersion problems. He presented a solution in the LaPlace domain for the concentration in a radial flow field due a uniform initial condition but did not present closed form, real time domain solutions. Ahfield et al (1988a) developed methodologies for optimizing aquifer decontamination by pumping using finite element methods for the flow and transport equations and nonlinear optimization. The two formulations were for maximizing the amount of contaminant removed or for decontaminating the aquifer to specified levels in a fixed time period at least cost. Ahfield et al (1988b) applied the two optimization formulations to a field site in Woburn, Massachusetts and discuss the competing factors of remediation effectiveness and operating cost.

Solutions for injection of solutes cannot be applied to the problem of aquifer decontamination by pumping because in the injection models the concentration of the solute input into the system is known. The injection is thus modeled as a mathematical source in which the concentration or the mass flux is specified by a known function. However, in remediation models, the initial condition is known but the concentration at the outlet is unknown. Solutions for injection of solutes in radial and one dimensional flow fields are reviewed below.

Solutions to radial dispersion problems for injection wells that were developed prior to 1985 are reviewed by Chen (1985) and those after 1985 include Chen (1986,1987) and Hsieh (1986). Lapidus and Amundson (1952) present two solutions for the one dimensional uniform dispersion problem in an infinite column with adsorption onto the solid phase. One solution assumes adsorption occurs under the local equilibrium assumption while the other assumes nonequilibrium adsorption with a first order rate law. Bastian and Lapidus (1956) derived the solution for a finite column with local equilibrium adsorption. For a semi-infinite column, Ogata and Banks (1961) developed the solution for a constant concentration boundary condition at the inlet. Gershon and Nir (1969) gave solutions for a semi-infinite column with a Cauchy boundary condition at the inlet; this boundary condition preserves mass balance at the boundary by considering dispersion across the inlet. Their transient solution accounts for adsorption under local equilibrium conditions but neglects radioactive decay while their steady state solution considers radioactive decay. Van Genuchten and Wierenga (1976) developed solution accounting for adsorption under equilibrium conditions and partitioning of the solute between mobile and immobile liquid phases. Transfer between the two liquid regions was considered by assuming diffusional transfer as proportional to the concentration difference between the mobile and immobile regions. Al-Niami and Rushton (1977) studied dispersion in the direction opposite to flow in a one dimensional finite system. They present solutions for a step change in concentration at the source of dispersion in a constant velocity flow field and discuss the conditions under which dispersion is contained by the groundwater flow. Kumar (1983) studied a similar problem to Al-Niami and Rushton (1977) but with an exponential change in concentration at the source of dispersion and an unsteady, one dimensional flow field. Solutions to the one dimensional uniform dispersion problem are systematically compiled by Van Genuchten and Alves (1982). All these works are different from the one studied here.

### PURPOSE AND SCOPE

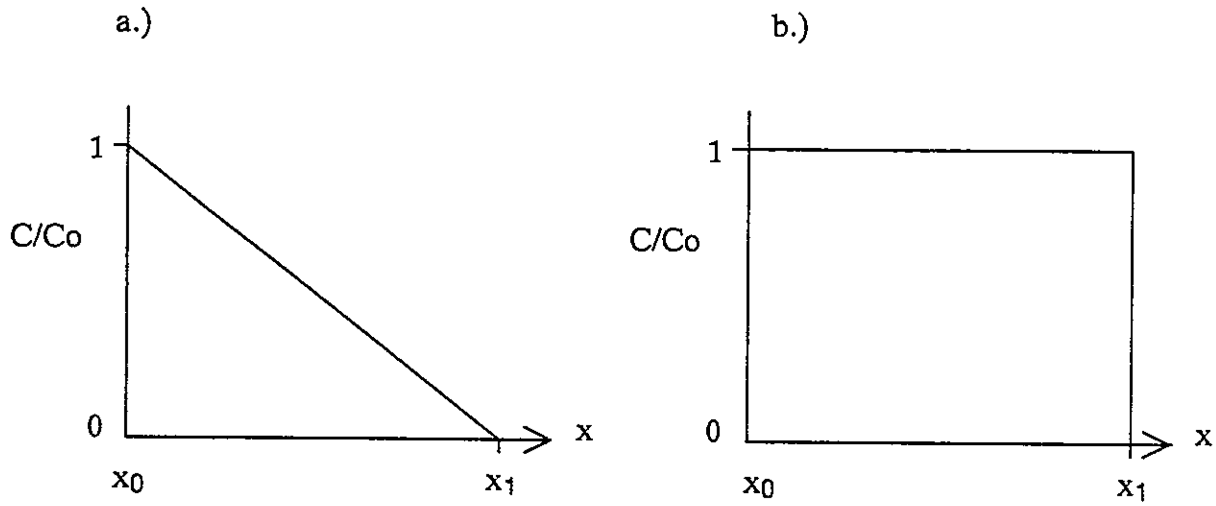
Mathematical solutions to the contamination scenarios depicted in Figures 1 through 3 are clearly relevant to address the environmental concerns of groundwater pollution. The purpose of this study is thus to develop new analytical solutions for the rehabilitation of polluted aquifers in one dimensional flow fields and to compare these solutions with the solutions for the radial flow model as developed by Chen and Woodside (1988). These solutions yield the spatial and temporal variation of the concentration in the aquifer for various withdrawal systems with which sensitivity analysis of the transport parameters and an analysis of the boundary conditions can be performed. Also, the methodology by which the solution of Chen and Woodside (1988) is numerically evaluated will be detailed.

## MODEL DEVELOPMENT AND SOLUTION DETERMINATION

In order to determine closed-form solutions to the aquifer remediation problems in Figures 1 through 3, the physical problem must be simplified for the mathematical model to be tractable. For this study, the primary simplifying assumptions are that the contaminants are not biologically or chemically transformed in the subsurface and behave as ideal tracers and that the aquifer is homogeneous and isotropic. Other assumptions will be noted as necessary. The two principal models in this study are for radial and one dimensional uniform flow. For both flow domains, the model formulation and solution technique are similar, the primary difference being the greater complexity of the radial model solution methodology.

## AQUIFER REHABILITATION MODEL IN 1D UNIFORM FLOW

Two simple initial conditions are used to model the contaminated groundwater as it exists when the remediation operation begins. These are a nonuniform initial condition which decreases from a maximum concentration to zero at some finite distance with a constant slope, and a uniform initial condition of unit concentration over some finite distance (Figure 4). All concentrations are normalized by dividing by the maximum concentration and thus vary between zero and one. The withdrawal system (french drain or infiltration gallery) should be located at the region of maximum concentration to minimize the decontamination time. Note that in Figure 4(b) the uniform initial condition must terminate at some finite distance for there to be a nontrivial solution. If $x_1$ tends to infinity, there will be no 'clean' groundwater to displace the contaminated water and the concentration in the entire aquifer for all time will be unity.




$$f(x) = 1 - m(x - x_0) \qquad x_0 \le x \le x_1$$
$$= 0 \qquad x \ge x_1$$
where $m = 1/(x_1 - x_0)$

$$f(x) = 1 \qquad x_0 \le x \le x_1$$
$$= 0 \qquad x \ge x_1$$

Figure 4 Nonuniform initial condition (a.) and uniform initial condition (b.) for one dimensional model.

When using the initial conditions in Figure 4 to approximate a plume geometry, the maximum concentration profile in the plume should be used if the plume profile varies in the direction normal to the flow induced by the withdrawal system (Figure 5). This will give the most conservative estimate for the concentration.

Using the initial conditions from Figure 4 and assuming the groundwater velocity is constant, the mathematical model is

$$D\frac{\partial^2 C}{\partial x^2} + V\frac{\partial C}{\partial x} = \frac{\partial C}{\partial t} \qquad (1)$$

Initial condition $$C(x, t = 0) = f(x) \qquad (2)$$

Boundary conditions $$\frac{\partial C}{\partial x} = 0 \qquad at\ x = x_0 \qquad (3)$$

$$C(x, t) = 0 \qquad as\ x \rightarrow \infty \qquad (4)$$

The governing equation (1) describes the movement of a conservative, nonreactive tracer in a uniform velocity flow field of a constant seepage velocity V. The dispersion coefficient, D, is assumed to be constant. In (2), the initial condition is given by one of the known functions in Figure 4. If a plume does not have a uniform initial condition in the direction normal to the groundwater flow, the maximum concentration profile should be chosen as shown in Figure 5. For a plume as in Figure 5, the one dimensional model does not incorporate the transverse dispersion that would occur but assuming the profile is uniform and using the maximum concentrations would produce a useful first approximation for the cleanup operation. For the boundary condition at the outlet, (3) indicates that the concentration just inside the withdrawal system is the same as the concentration just outside the withdrawal system. This boundary condition is applied because the concentration at the outlet is not known; it is determined by the solution and thus cannot be prescribed as a priori at the boundary. Thus, prescribed concentration or mass flux (Cauchy) boundary conditions are not applicable at the outlet. As indicated by (4) the concentration must tend to zero at large distances. As discussed earlier, this is necessary for there to be sufficient clean groundwater to displace the contaminated water and rehabilitate the aquifer. This formulation is in terms of dimensional units and any consistent set of units is acceptable.

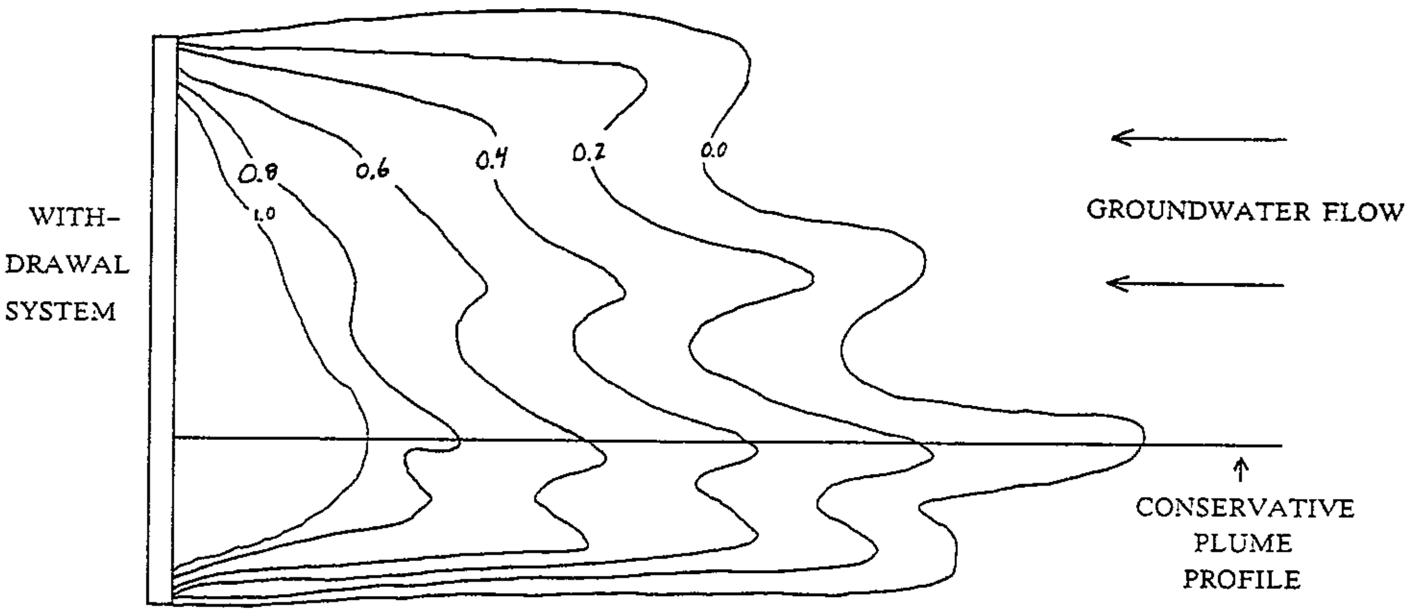



Figure 5 Plan view of isoconcentration contours of C/Co and conservative plume profile in one dimensional flow

## SOLUTION TECHNIQUE FOR 1D MODEL

By applying a change of variable as shown below, the governing equation (1) is transformed to the diffusion equation. The change of variable is

$$G(x,t) = C(x,t) exp(Vx/2D + V^2 t/4D) \tag{5}$$

Application of (5) to (1) through (4) yields

$$D\frac{\partial^2 G}{\partial x^2} = \frac{\partial G}{\partial t} \tag{6}$$

Initial condition $$G(x, t = 0) = f(x)exp(Vx/2D) \tag{7}$$

Boundary conditions $$\frac{\partial G}{\partial x} - (V/2D)G = 0 \qquad at\ x = x_0 \tag{8}$$

$$G(x,t) = 0 \qquad as\ x \rightarrow \infty \tag{9}$$

To remove the time derivative in (6), the LaPlace transform is applied to (6) as

$$\overline{G} = \int_0^\infty e^{-pt}\, G(x,t)dt \tag{10}$$

Application of (10) to (6) yields

$$D\,\frac{d^2\overline{G}}{dx^2} = p\overline{G} - \overline{G}(x, t = 0) \tag{11a}$$

$$D\,\frac{d^2\overline{G}}{dx^2} - p\overline{G} = -f(x)exp(Vx/2D) \tag{11b}$$

$$\frac{d\overline{G}}{dx} - (V/2D)\overline{G} = 0 \qquad at\ x = x_0 \qquad (12)$$

$$\overline{G}(x,t) = 0 \qquad as\ x \rightarrow \infty \qquad (13)$$

Note that the relationship (7) has been used in (11b). Equation (11b) is a nonhomogeneous ordinary differential equation and the Green's function technique will be used to solve it. The theory and application of Green's functions in determining solutions for nonhomogeneous differential equations can be found in Wylie and Barrett (1982). By letting the variable s be the Green's function parameter, the two Green's functions for this problem are

$$g_1(x,p,s) = A_1\, e^{ax} + A_2\, e^{-ax} \qquad s \leq x < \infty \qquad (14)$$

$$g_2(x,p,s) = A_3\, e^{ax} + A_4\, e^{-ax} \qquad x_0 \leq x \leq s \qquad (15)$$

where $a = \sqrt{p/D}$

where $A_1$, $A_2$, $A_3$, and $A_4$ are four coefficients which can be uniquely determined by the four properties of the Green's function: the two boundary conditions given in (12), and (13) and the two properties

$$g_1(x,p,s) = g_2(x,p,s) \qquad at\ x = s \qquad (16)$$

$$\frac{dg_1}{dx} - \frac{dg_2}{dx} = -1/D \qquad at\ x = s \qquad (17)$$

The condition in (16) indicates that the Green's function are continuous at x = s and (17) that at this point the first derivatives of the Green's function have a step discontinuity of magnitude 1/D. The use of these four conditions to determine the unknown coefficients $A_1$ through $A_4$ is shown in Appendix A and yields (14) and (15) as

$$g_1(x,p,s) = (1/2\sqrt{pD})\{\exp[a(s-x)] + \exp[a(2x_0-s-x)]/X\} \qquad s \leq x \leq \infty \tag{18}$$

$$g_2(x,p,s) = (1/2\sqrt{pD})\{\exp[a(x-s)] + \exp[a(2x_0-s-x)]/X\} \qquad x_0 \leq x \leq s \tag{19}$$

where $$X = \frac{\sqrt{p} + V/2\sqrt{D}}{\sqrt{p} - V/2\sqrt{D}}$$

With the Green's functions defined, the solution of (11b) is thus

$$\overline{G} = \int_{x_0}^{x_1} g(x,p,s)F(s)ds \tag{20}$$

where $F(s) = f(s)\exp(Vs/2D)$

where g represents the Green's function in (18) and (19). Substituting (18) and (19) into (20) yields the solution in the LaPlace domain as

$$\overline{G} = \int_{x_0}^{x} g_1\, F(s)ds + \int_{x}^{x_1} g_2\, F(s)ds \tag{21}$$

The useful feature of the Green's function technique is that the Green's functions given by (18) and (19) are independent of the nonhomogeneous term F(s) in (20). Thus, the solution (21) is valid for a wide range of initial conditions, provided that the initial condition tends to zero as x approaches infinity for the reasons discussed earlier.

If the initial condition f(x) is specified by a simple function, the integration in (21) can be performed analytically. Then, the solution in the LaPlace domain can be inverted to give the real time domain solution to the problem. The integration in (21) is performed for two forms of the initial condition. The first is using the uniform initial condition as given by Figure (4b) and results in

$$\bar{G} = \frac{\exp(Vx/2D)}{P - V^2/4D} + \frac{\exp(Vx_1/2D)}{2\sqrt{p}}\left\{\frac{\exp(a(x-x_1))}{\beta - \sqrt{p}} - \frac{\exp(a(2x_0 - x - x_1))}{\beta + \sqrt{p}}\right\} \quad (22)$$

where $\beta = V/2\sqrt{D}$

The LaPlace inversion of the terms in (22) is straightforward and can be found in most tables of LaPlace inversions (see for example Abramowitz and Stegun (1972), pp. 1022–1027). Evaluation of the LaPlace inversion and application of (5) yields the solution to (1) for the uniform initial condition as

$$C(x,t) = 1 - 0.5\{\mathrm{erfc}(j) + \exp(V(x_1-x_0)/D)*\mathrm{erfc}(k)\} \quad (23)$$

where $j = -\beta\sqrt{t} + (x_1 - x)/2\sqrt{Dt}$

$k = \beta\sqrt{t} + (x + x_1 - 2x_0)/2\sqrt{Dt}$

For the nonuniform initial condition in Figure (4a) where $f(x) = 1 - m(x-x_0)$, the integration and inversion of (11) is given in detail in Appendix B. The solution is

$$C(x,t) = 1 - m(x - x_0) - mvt + m\sqrt{Dt/\pi}\,\{\exp(-j^2) - \exp(-l^2)\} +$$
$$0.5m(vt + x - x_0 + D/V)\mathrm{erfc}(l) + 0.5m(vt - x_1 + x)\mathrm{erfc}(j) +$$
$$0.5mD/V\{-\exp(V(x_1 - x_0)/D)\mathrm{erfc}(k) - \exp(V(x_0 - x)/D)\mathrm{erfc}(m) +$$
$$\exp(V(x_0 - x)/D)\mathrm{erfc}(h)\} \quad (24)$$

where $l = \beta\sqrt{t} + (x - x_0)/2\sqrt{Dt}$

$$m = \beta\sqrt{t} - (x + x_1 - 2x_0)/2\sqrt{Dt}$$

$$h = \beta\sqrt{t} - (x - x_0)/2\sqrt{Dt}$$

## AQUIFER REHABILITATION MODEL IN RADIAL FLOW

For radial flow to a withdrawal well, the model for the transport of a conservative tracer by advection and longitudinal mechanical dispersion is (Chen and Woodside, 1988)

$$\frac{\partial^2 C}{\partial^2 \rho} + \frac{\partial C}{\partial \rho} = \rho \frac{\partial C}{\partial \tau} \qquad (25)$$

$$\frac{\partial C}{\partial \rho} = 0 \qquad \text{at } \rho = \rho_0 \qquad (26)$$

$$C(\rho, \tau) = 0 \qquad \rho \rightarrow \infty \qquad (27)$$

$$C(\rho, \tau = 0) = f(\rho) \qquad (28)$$

where $\alpha$ = dispersivity

$\rho = r/\alpha$

$\tau = At/\alpha^2$

In the derivation of (25) the velocity is assumed to be described by $V = A/r$, where A is equal to $Q/2\pi bn$ with Q as the pumping rate, b as the uniform aquifer thickness, and n is the porosity. The primary physical difference between the two models is that the radial model accounts for a velocity dependent dispersion coefficient while the one dimensional model assumes a constant dispersion coefficient. This causes (25) to be a partial differential equation with variable coefficients while (1) has constant coefficients. The boundary condition at the outlet for the radial model as given by (26) is the same as for the one dimensional model in (3) and indicates there is no concentration gradient across the well bore. The initial condition in (28) is for an arbitrary function f(ρ) which can be a wide range of functions provided that it tends to zero as ρ approaches infinity. Chen and Woodside (1988) determine the solution for the model given by (25), (26), (27), and (28) as

$$C(\rho,\tau) = 0.5\exp(-\rho/2)\int_{\rho_0}^{\infty} F(s)\int_{0}^{\infty} x^{1/3}\exp(-x^2\tau)\left[f_5 - \frac{(4f_1f_2f_3 + f_4(3f_1f_1 - f_2f_2))}{f_1f_1 + f_2f_2}\right]dx\,ds \qquad (29)$$

where
$$f_1 = x^{2/3}Ai'(\phi_0) + 0.5Ai(\phi_0)$$

$$f_2 = x^{2/3}Bi'(\phi_0) + 0.5Bi(\phi_0)$$

$$f_3 = Ai(\phi)Bi(\psi) + Ai(\psi)Bi(\phi)$$

$$f_4 = Ai(\phi)Ai(\psi) - Bi(\phi)Bi(\psi)$$

$$f_5 = 3Ai(\phi)Ai(\psi) + Bi(\phi)Bi(\psi)$$

$$\phi = (1 - 4\rho x^2)/4x^{4/3}$$

$$\phi_0 = (1 - 4\rho_0 x^2)/4x^{4/3}$$

$$\psi = (1 - 4sx^2)/4x^{4/3}$$

$$F(s) = f(s)*s*\exp(s/2)$$

Equation (29) involves a double integral which must be evaluated numerically. Chen and Woodside (1988) briefly discuss its evaluation. A detailed discussion of the numerical integration and a computer program for evaluating (29) are given in Appendix C.

## ANALYSIS OF RESULTS

Using the radial and one dimensional solutions for the uniform and nonuniform initial conditions, the effect of varying the dispersivity can be analyzed. First, the variation in concentration over distance is analyzed as the dispersivity is changed. The results for the one dimensional model will be analyzed using a dimensionless time $T_r = Vt/x_1$ which represents the number of pore volumes withdrawn. As can be seen in Figure 6a for the nonuniform initial condition, the areas beneath the two curves are not equal. This is because for a withdrawal problem, the mass balance is between the amount of mass initially in the aquifer and the mass remaining in the aquifer plus the mass already re-

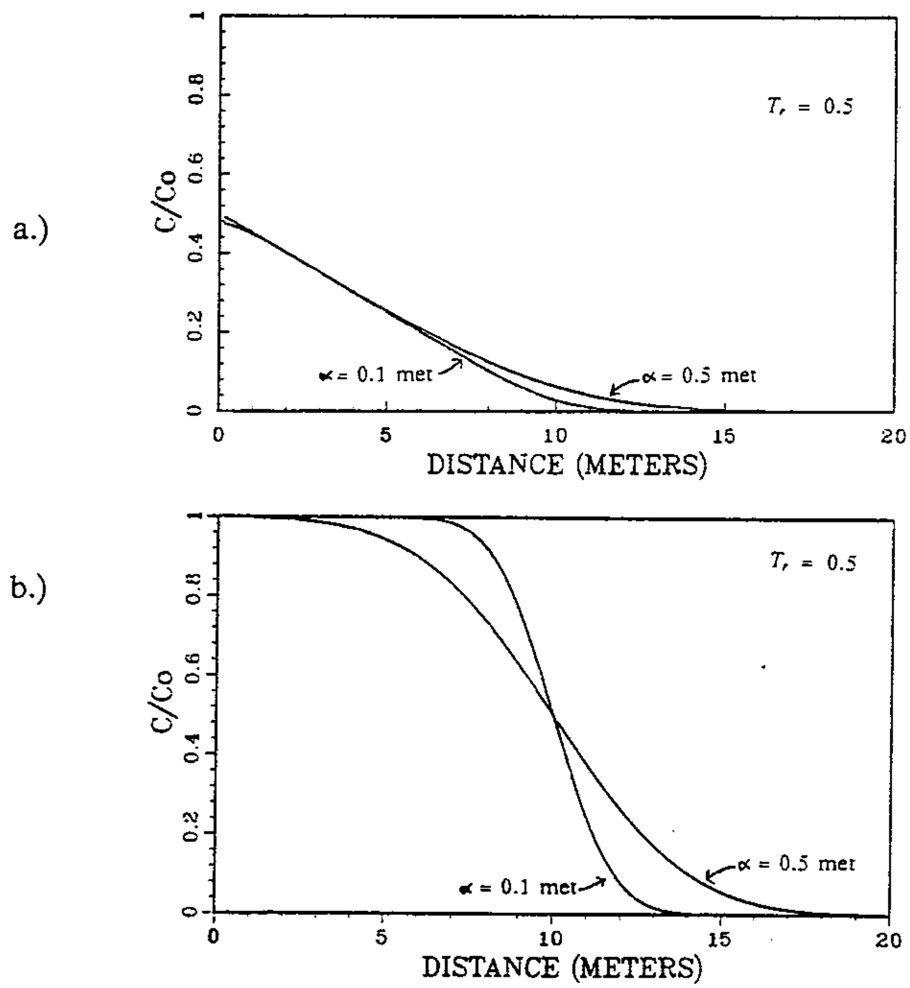



Figure 6 Concentration for one dimensional model for different dispersivities; nonuniform initial condition (a) and uniform initial condition (b)

moved. For two different dispersivities, the mass flux at the outlet is different when the concentration decreases below one and mass will be removed at different rates. Thus, the area under the two curves in Figure 6a are not equal. In Figure 6b, the concentration at the outlet is still one for the time used so mass is removed at the same rate. At a later time as is shown in Figure 7, the concentration at the outlet is less than one and mass is removed at different rates, causing the area under the two curves to not be equal. An–

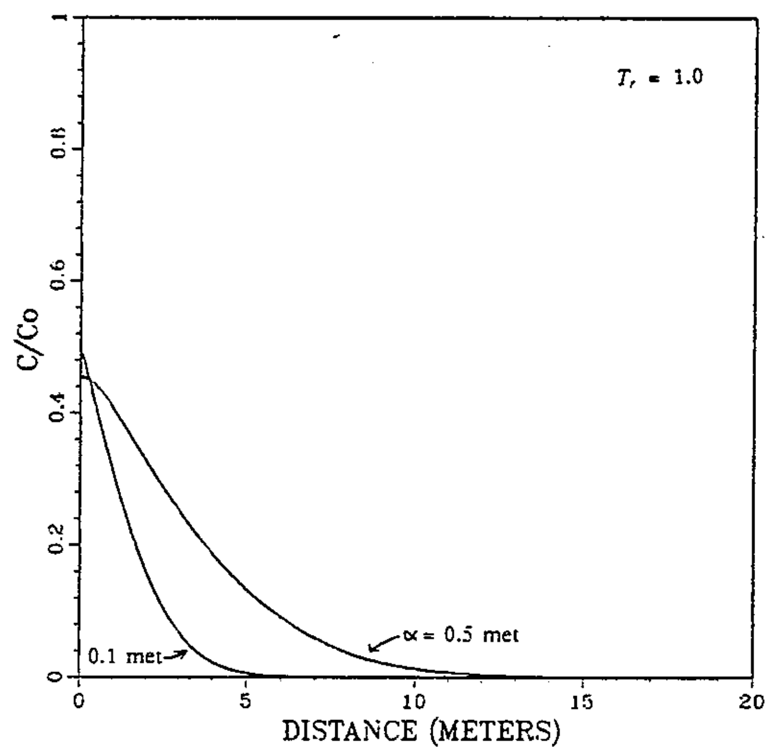



Figure 7 Concentration for uniform initial condition for different dispersivities

other method of analyzing the effect of dispersivity is to consider how the concentration at the outlet varies in time as the dispersivity is changed. As shown in Figure 8, the area under the two curves are approximately equal. The initial condition in Figure 8 is of the form given in (30)

$$\begin{aligned} f(\rho) &= 1 & \rho_0 \le \rho \le \rho_1 \\ f(\rho) &= \exp(-\omega(\rho - \rho_1)^2) & \rho \ge \rho_1 \end{aligned} \tag{30}$$

where $\omega$ is an empirical coefficient determined by trial–and–error procedure when (30) is used to approximate a known curve. By varying $\omega$ and $\rho_1$, (30) can describe a wide range of initial conditions. It produces a smooth and continuous curve of which the gradient tends to zero as $\rho$ is large. The area under the curve refers to the zeroeth time moment defined as

$$m_0 = \int_0^\infty C(x_0, t)dt \tag{31}$$

In (31) the time moment represents the total amount of mass passing the outlet (i.e. at $x_0$). According to the mass balance principle, it should be equal for different dispersivities. Figure 8 supports this fact as the areas under the two curves are approximately equal.

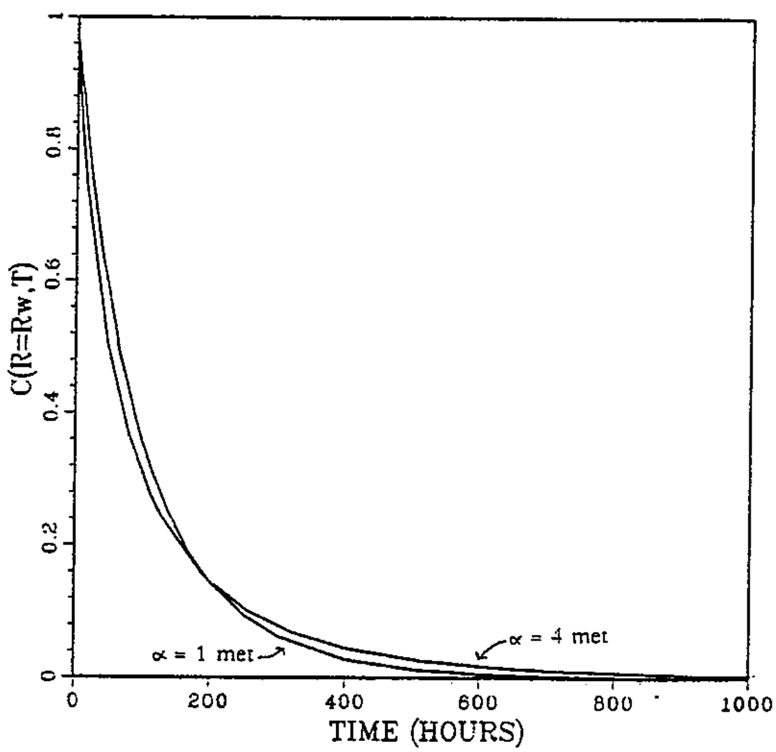



Figure 8 Mass balance in radial model for different dispersivities; initial condition given by (30) with $\omega = 0.005$ and $\rho_1 = \rho_0$

In the radial model, the nonuniform initial condition as in Figure 4a is $f(\rho) = 1-m(\rho - \rho_0)$ and the uniform initial condition as in Figure 4b is simply $f(\rho) = 1$ between the well and the edge of the plume. The radial solution for these two initial conditions is shown in Figure 9a and 9b respectively.

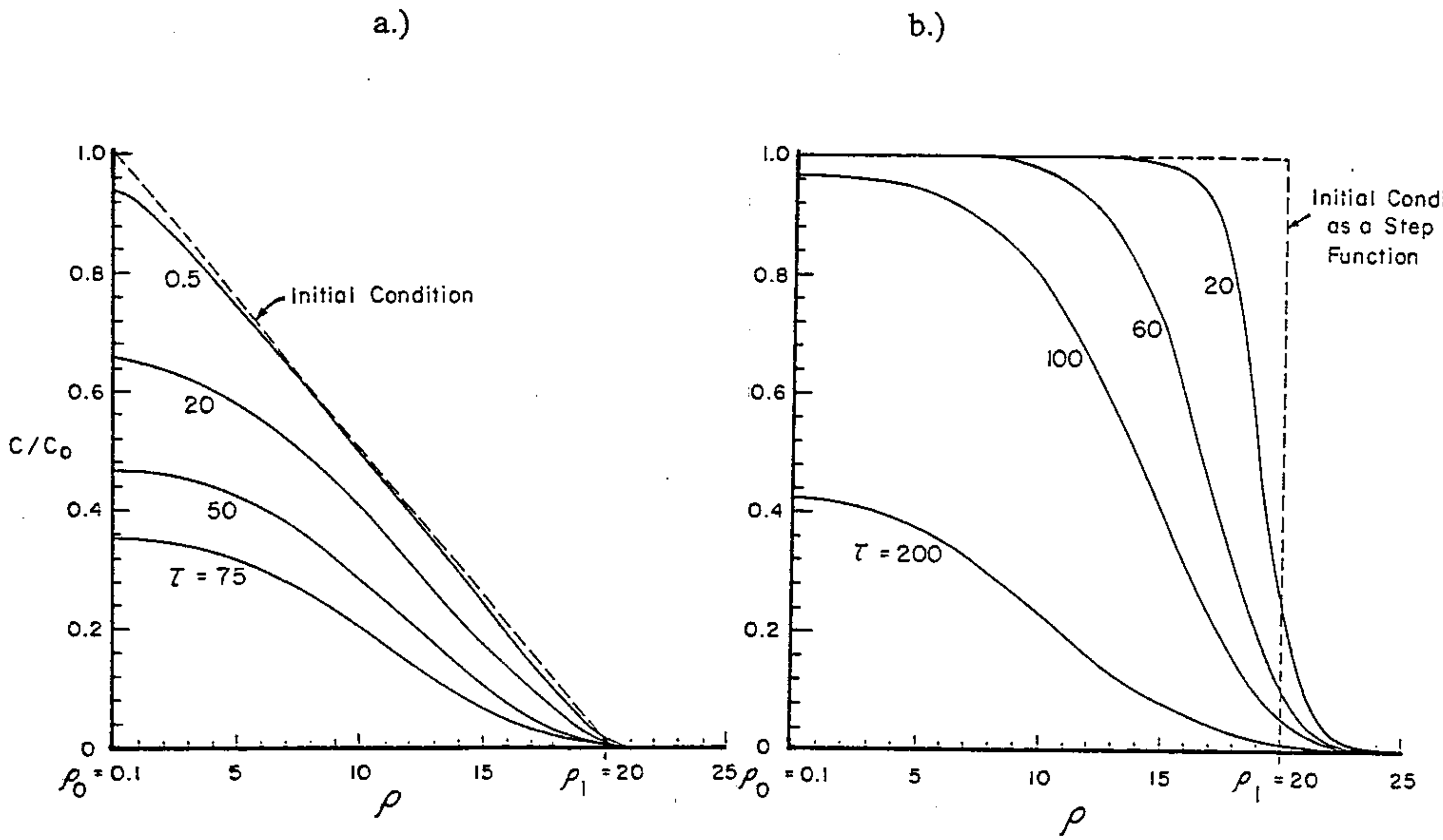



Figure 9 Radial solution for nonuniform initial condition (a) and uniform initial condition (b)

For these two simple initial conditions, the solute spreads beyond the original region of contamination as can be seen in Figures 9a and 9b. When the withdrawal process first begins the concentration gradient at the plume boundary is infinite for the uniform initial condition, due to the discontinuity in concentration at the edge of the plume. Thus, backwards or adverse dispersion is expected due to the large concentration gradient against the converging groundwater flow direction and this causes the plume to spread beyond the original contaminated region. If the nonuniform initial condition is used, the concentration gradient is finite at the plume boundary and less backward dispersion occurs as can be seen by comparing Figure 9a to Figure 9b. If the mechanism causing the backward dispersion is the large concentration gradient at the boundary, then the use of the smooth initial condition given by (30) should result in little or no adverse dispersion. Figure 10 indicates this to be the case where it is noted that during the withdrawal process the contamination never extends beyond the original plume boundary. Consequently, the use of a more realistic initial condition as in (30) leads to smaller gradients at the plume boundary and backwards dispersion does not occur. If the initial condition is formulated as a step

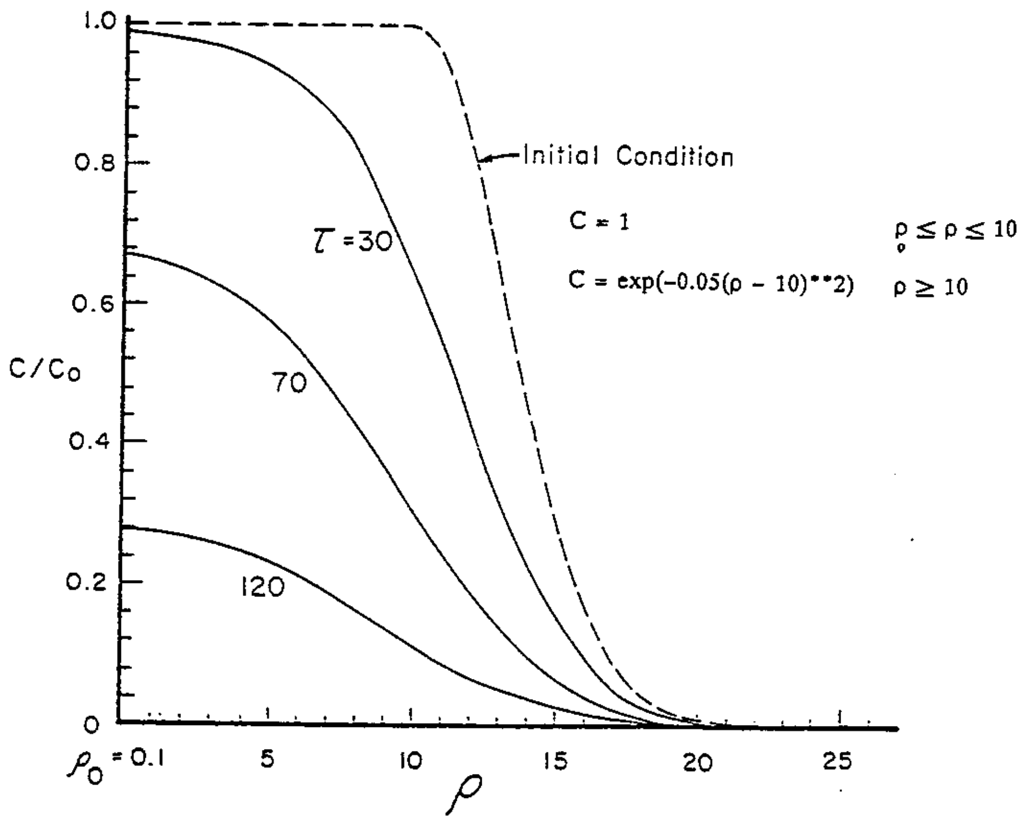



Figure 10 Radial solution with smooth initial condition

function as in Figure 4b for the one dimensional model, backwards dispersion will occur even if the dispersivity is small. This will be analyzed using the one dimensional model. Figure 11 represents the same initial condition as in Figure 9b except that Figure 11 is for the one dimensional model. It is noted that backwards dispersion occurs in Figure 11 as expected. With the same initial condition as used in Figure 11, when the dispersivity is decreased by a factor of ten, Figure 12 shows that less backwards dispersion has occurred. For example, by letting the dispersivity be equal to one meter in Figure 11, it can be seen that the plume has extended beyond the original plume boundary by about four meters. Then in Figure 12 the dispersivity is one–tenth of a meter and backwards dispersion has spread the plume about one–half of a meter beyond the initial edge of the plume. Thus, less backwards dispersion has occurred in Figure 12 where the dispersivity is one–two hundredth of the size of the plume than in Figure 11 where the dispersivity is one–twentieth of the plume length. However, even with this much smaller dispersivity, backwards dispersion still occurs as expected. As being hardly conceivable under field conditions, this backwards dispersion appears to be just a mathematical artifact due to the initial conditions selected.

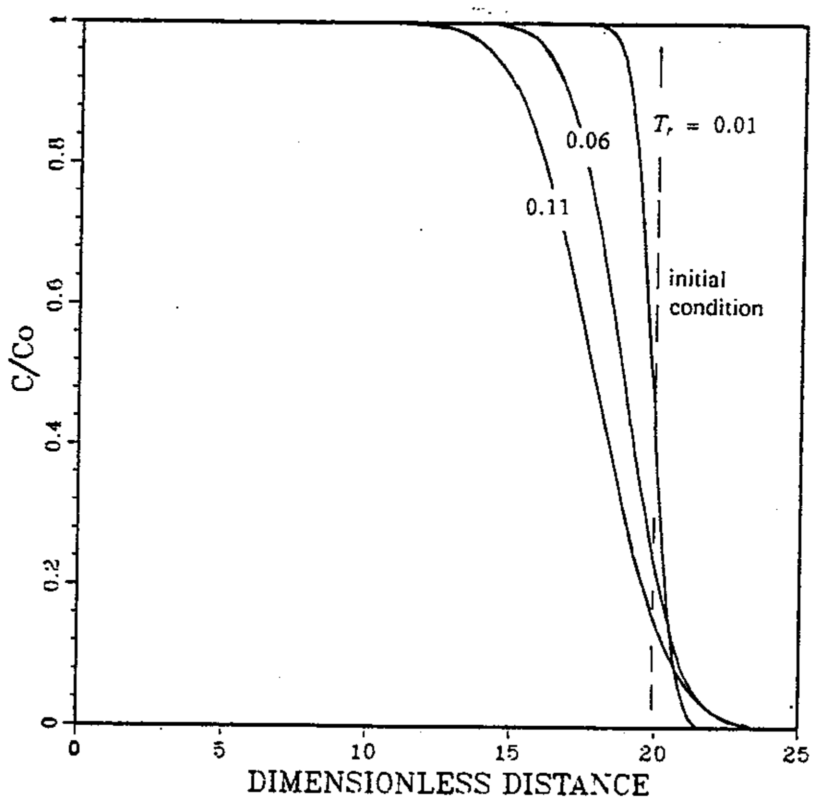



Figure 11 One dimensional solution for uniform initial condition with $x_1$ = 20

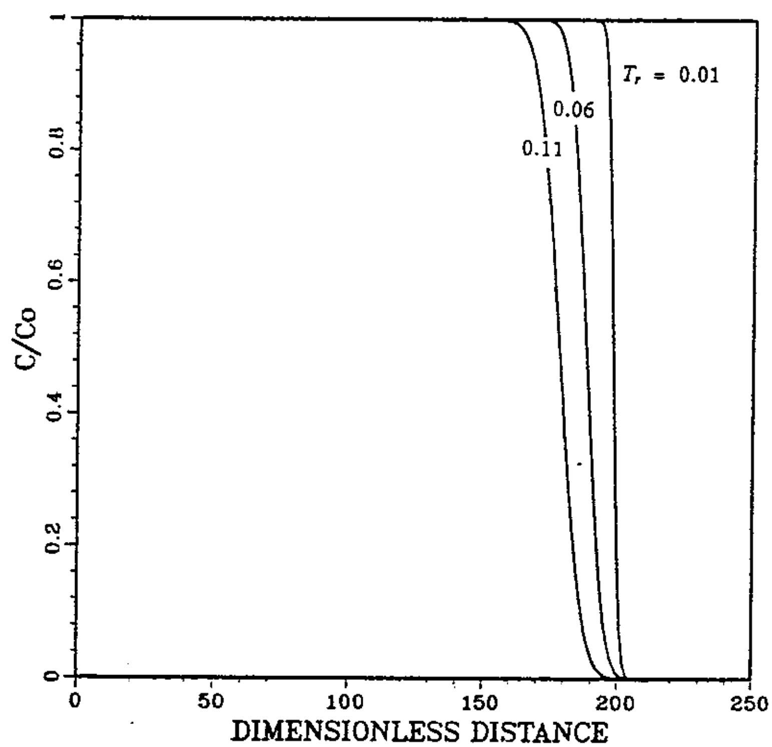



Figure 12 One dimensional solution for uniform initial condition with $x_1$ = 200

While the graphs that have been presented all behave as the physics of solute transport would indicate, other methods of validating the solutions are desired. For the one dimensional solutions as given by (23) and (24), the solutions can be put back into the mathematical model to determine if the governing equation, boundary conditions, and initial condition are satisfied. By setting t equal to zero it can be readily seen that (24) and (25) satisfy the initial condition and through the use of the Liebnitz rule it can be shown that they satisfy the boundary condition (3) and the governing equation (1). However, for the radial solution as given by (29), the solution cannot be directly put back into the mathematical model due to the difficulty of taking the derivatives of (29). Other methods of verifying the radial solution are thus important.

One method of verifying (29) is by comparison to Gelhar and Collins (1971) approximate solution for cyclic injection and withdrawal of a conservative tracer in a radial flow field. This is not a rigorous verification but a simple comparison using a special case of (29) when the initial condition is generated by injecting a given amount of solute as in a single well tracer test. Thus, the hypothetical injection of constant concentration solute for a specific time interval is modeled under given hydrogeological conditions using available analytical solutions. The analytical solution is that of Chen (1987) for resident concentrations and will be used to generate the initial condition for use in (29). For a hypothetical injection, a dimensionless time of $\tau = 200$ and $\rho_0 = 0.1$ were chosen. When plotted in dimensionless variables C/Co and $\rho$ the use of Chen's (1987) solution yields the solid line in Figure 13. Since this initial condition cannot be directly incorporated into (29), it is approximated with the dashed curve in Figure 13 which is developed by (30) as

$$
\begin{aligned}
f(\rho) &= 1 & \rho_0 \le \rho \le 12.2 \\
f(\rho) &= \exp(-0.015(\rho - 12.2)^2) & \rho \ge 12.2
\end{aligned}
\tag{32}
$$

Using the initial condition (32), the results of the radial solution (29) evaluated at the well bore are compared with those of Gelhar and Collins (1971) in Figure 14. The comparison is good for the degree of approximation in the initial condition in Figure 13. This supports the validity of (29).

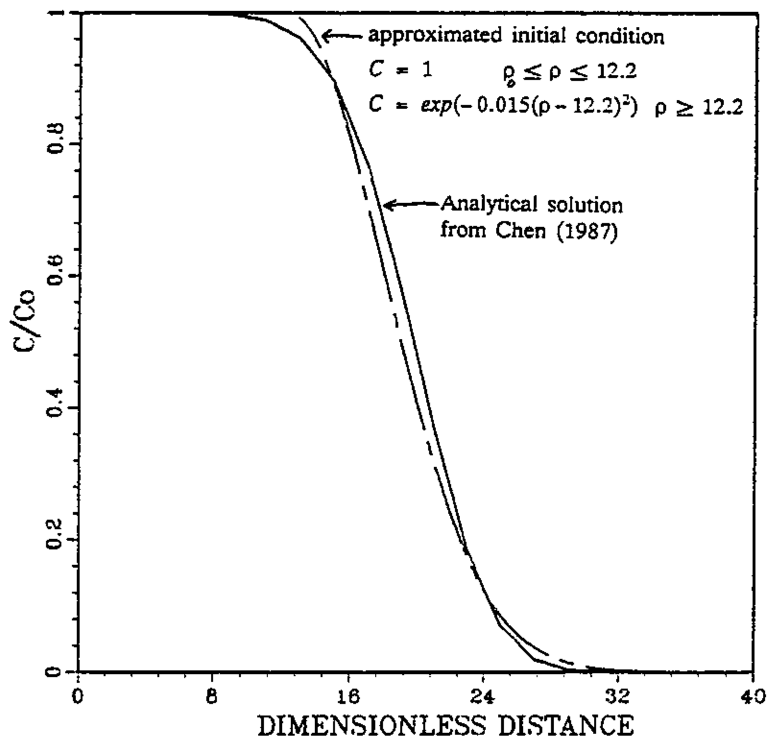



Figure 13 Chen (1987) analytical solution (solid line) and approximated initial condition given by (32) (dashed line)

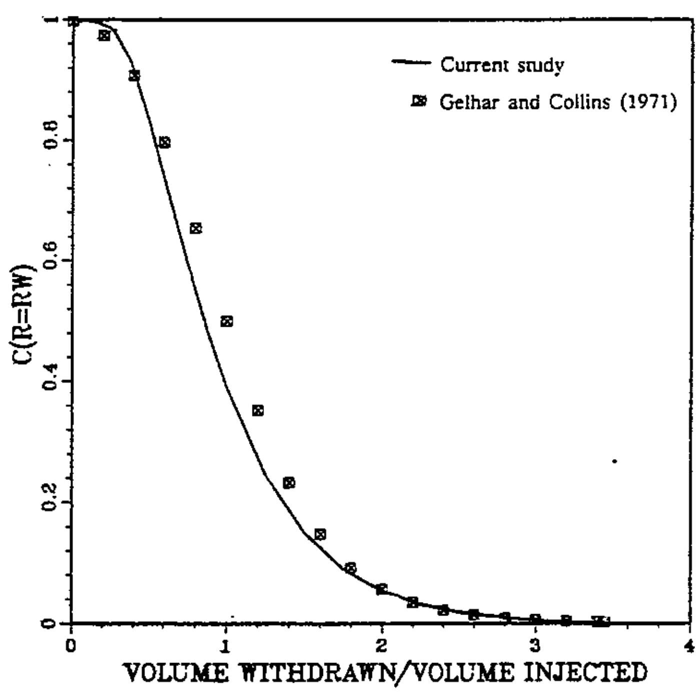



Figure 14 Comparison between radial solution (solid line) and Gelhar and Collins (1971) (boxes)

Another method of verifying (29) is by comparison to field data. Pickens and Grisak (1981) published results from their test SW2 for a single well tracer test using radioactive iodide injected for 3.93 days. The injection rate was 0.719 liters per second and the withdrawal rate was 0.606 liters per second and lasted for 16.9 days from the end of the injection period. The average aquifer thickness was estimated to be 8.2 meters with a porosity of 0.38. A dispersivity of 0.09 meters was determined from the withdrawal data using the Mercado method (1966). Pickens and Grisak (1981) did not publish enough observation well data to describe the concentration at the end of the injection period such that the initial condition in (29) can be formulated. Thus, the initial condition is approximated as in the comparison to Gelhar and Collins approximate solution by using (30), where the analytical solution of Chen (1987) is used to generate the initial condition. As shown in Figure 15 the initial condition is approximated by

$$
\begin{aligned}
f(\rho) &= 1 && \rho_0 \le \rho \le 42 \\
f(\rho) &= \exp(-0.005(\rho - 42)^2) && \rho \ge 42
\end{aligned}
\tag{33}
$$

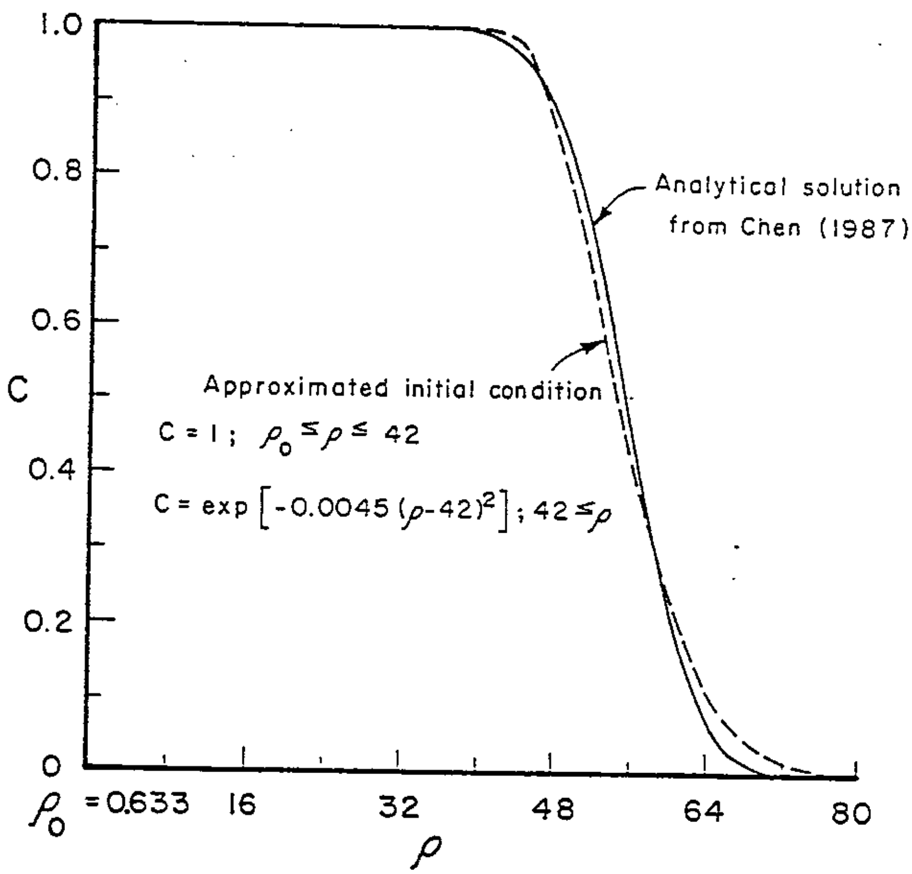



Figure 15 Chen (1987) analytical solution (solid line) and approximated initial condition given by (33) (dashed line)

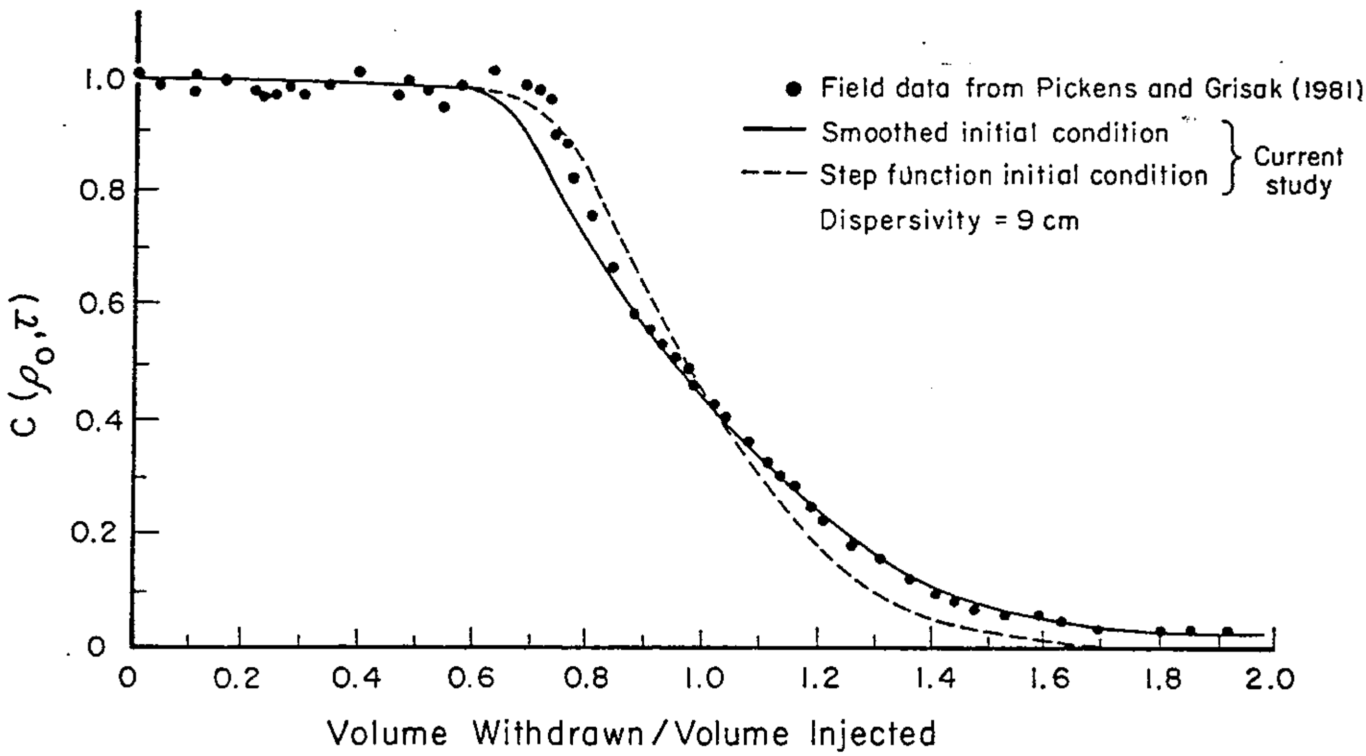



Figure 16 Radial solution for smooth initial condition (dashed line) and step function initial condition (dashed line) for Pickens and Grisak (1981) test SW2

Using the initial condition (33), the results of (29) evaluated at the well are shown as the solid line in Figure 16 and compared with the reported withdrawal phase concentration history of Pickens and Grisak (1981). Also shown in Figure 16 is the results of (29) if the initial condition is simply a step function of normalized concentration equal to one from the well to a dimensionless distance of 55.4, which refers to an average solute frontal distance of 4.99 m as reported by Pickens and Grisak (1981). Using the step function initial condition the analytical solution underestimates the concentration after one pore volume has been removed and does not have a long enough tail as expected. When the smoothed initial condition given by (33) is used, the concentration history at the well is reproduced very accurately, with the only error between when 0.6 and 0.8 pore volumes have been removed. This also supports the validity of (29).

For the decontamination of a polluted aquifer by a withdrawal well in which the initial condition may be represented by simple initial conditions, a decontamination rate curve may be developed. The two initial conditions used here are the zero slope initial condition given by a step function and the nonzero slope initial condition represented by a sloping straight line. An arbitrary level of decontamination of one percent of the initial maximum concentration is chosen here. This level of decontamination may not be appropriate for some polluted aquifers and is simply proposed as a representative decontamination index of aquifer restoration. Figure 17 shows the dimensionless time required to reach a one percent decontamination level using a single pumping well for a specified distance to the edge of the plume given by $\rho_1$ . The concentrations plotted are accurate to four decimal places and approximately straight line relationships are present. For the initial condition given be the step function (zero slope), the time required to reach one percent decontamination is always longer than for the nonzero slope straight line because there is more mass initially in the system for the step function initial condition for the same $\rho_1$ . In Figure 17 the time required to reach one percent decontamination if advection is the only transport mechanism is also shown. This time is given by

$$t = \frac{\pi r_1^2 bn}{Q} \tag{34}$$

In the dimensionless units of the radial model, this line is given by $\tau = (\rho_1 * \rho_1)/2$ where the well radius is neglected by considering that the well radius is usually small compared to the initial plume size. The one dimensional model can also be used to develop a one percent decontamination rate curve. Dimensionless variables of $X_1 = x_1/\alpha$ and $T = Vt/\alpha$ are used in Figure 18 which shows the time required to reach a one percent decontamination level for a specified distance to the edge of the plume given by $X_1$. The straight line for the time to reach one percent decontamination if dispersion is neglected is given by $T = X_1$ . As in Figure 17, approximate straight line relationships are present except for small $X_1$ . In Figure 18, the effect of changing $x_0$ is negligible as in Figure 17, indicating that the divergence from straight line relationships for small $x_1$ is not due to the value of $x_0$ that is chosen. The divergence from approximate straight line relationships in Figure 18 may be due to the assumption of a constant velocity in the one dimensional model. Both Figures 17 and 18 indicate that neglecting dispersion underestimates the total cleanup time as is also inferred from the prior analysis of dispersivity as in Figure 9 where a smaller dispersivity is associated with a more rapid removal of the plume.

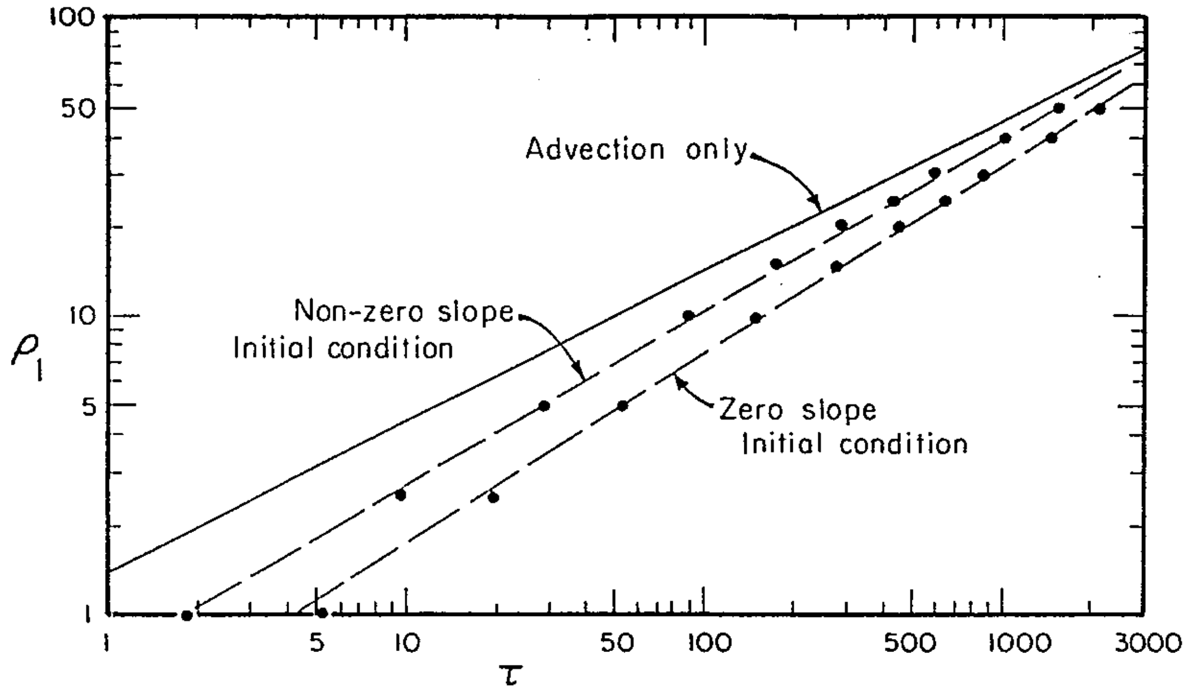



Figure 17 One percent decontamination rate curve for uniform and non-uniform initial conditions in radial model

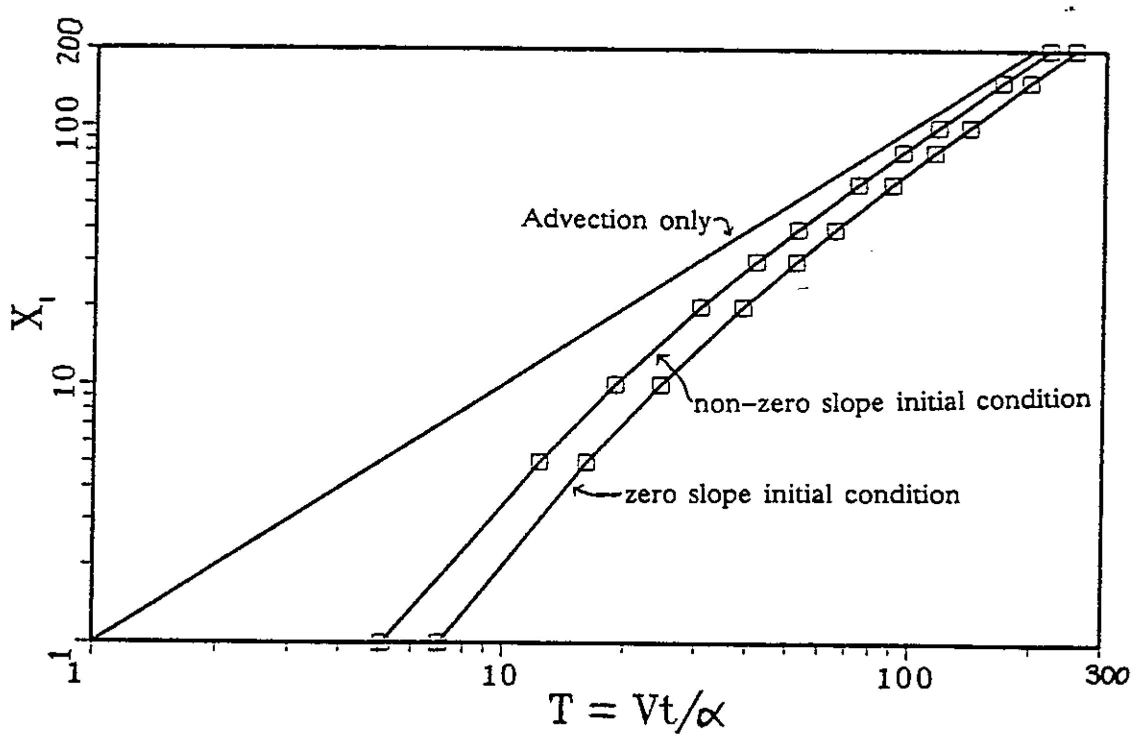



Figure 18 One percent decontamination rate curve for uniform and non-uniform initial conditions in one dimensional model

As an example of the use of Figure 17, consider a circular plume with a uniform concentration to a radius of 50 m in an aquifer 5 m thick with a porosity of 0.2 and a dispersivity of 1 m. If a single well pumping at 10 m3/hr is used to decontaminate the plume, the time required to reach one percent decontamination can be determined as follows. The dimensionless distance ρ for this plume is 50/1 = 50 and the dimensionless time corresponding to this plume size from Figure 17 is $\tau = 2100$. Noting the definition of dimensionless time as $\tau = At/\alpha^2$, the time to reach one percent decontamination can be determined as

$$2100 = \frac{10t}{2\pi(0.2)(5)1^2}$$

or

$$t = 1319.5 \text{ hrs} \simeq 55 \text{ days}$$

Thus, approximately 55 days would be required to decontaminate the aquifer for the specified conditions. If advection was assumed to be the only transport mechanism, the cleanup time would be estimated as approximately 33 days, indicating the underestimation that occurs by neglecting dispersion.

As an illustration of the use of Figure 18, consider the same 50 m radius plume that was studied with the radial model. If a 100 m drain is located as shown in Figure 19, the radial and one dimensional models can be compared to determine which most efficiently decontaminates the aquifer. The flow into the drain can be modeled using the Dupuit–Forchheimer discharge formula as

$$Q_x = \frac{K(h_c^2 - h_d^2)}{2L} \tag{35}$$

where $Q_x$ = *flow rate per unit length of drain*

$h_c$ = *head in contaminated region*

$h_d$ = *head in drain*

K = hydraulic conductivity

L = distance from the drain to the edge of the plume

$= x_1 - x_0$

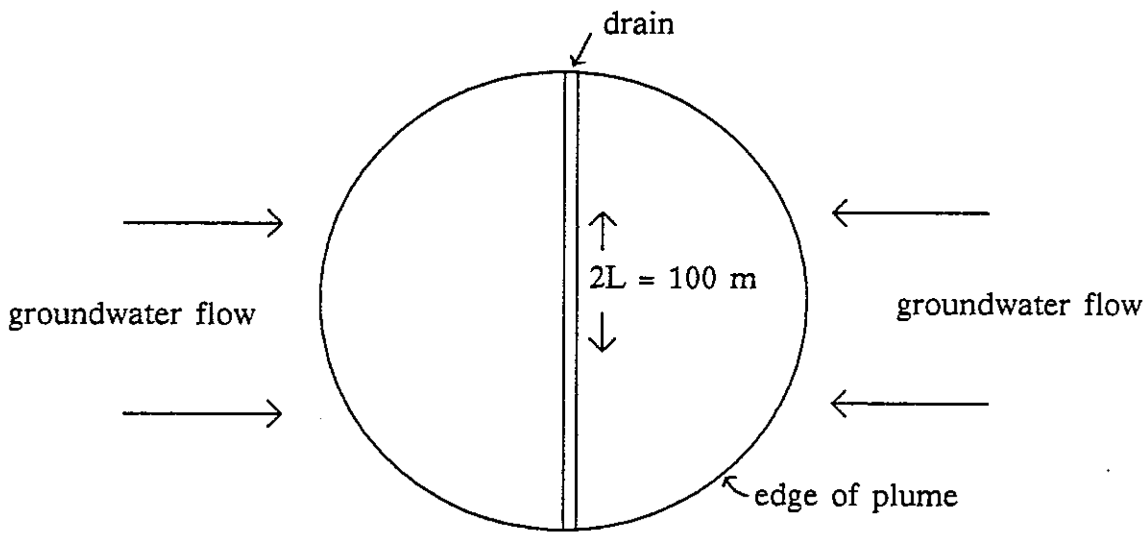



Figure 19 Restoration of polluted aquifer with plume of radius 50 m using one dimensional flow to a drain

Equation (35) gives the flow into one side of the drain in an unconfined aquifer and neglects the seepage face in the drain. For this model of flow into the drain, it is assumed that the head in the contaminated region $h_c$ is the head at the edge of the plume (at $x = x_1$) and it remains constant. Vertical or lateral recharge into the contaminated region could maintain $h_c$ as constant. The seepage velocity is determined as

$$V = q/n \qquad (36)$$

where $q$ = *specific discharge*

= *flow rate per unit area*

To determine q, it is noted that

$$q = \frac{Q_x}{(h_c + h_d)/2} \tag{37}$$

Using (35) and (37), V is determined as

$$V = \frac{K(h_c - h_d)}{nL} \tag{38}$$

For the plume under consideration, assume that K = 0.0001 m/s and $h_c$ = 5 m. The seepage velocity for the groundwater flowing to the drain is then

$$V = 0.864(5 - h_d) \qquad (met/day) \tag{39}$$

Equation (39) gives the seepage velocity as a function of the head in the drain. From Figure 18, the dimensionless time to reach 1% decontamination using the drain is T = 78. Noting the definition of the dimensionless time as T = $Vt/\alpha$, (39) can be used to determine the velocity and then the time to reach 1% decontamination can be determined. For comparing the efficiency of the radial and one dimensional models, two different design criteria can be used. The two criteria are maintaining the head in the drain such that the radial and one dimensional models reach 1% decontamination at the same time or such that water is removed at the same rate from the drain as is pumped by the well. If the criteria is to decontaminate with both models in the same amount of time, the velocity required for the one dimensional model is determined as follows by recalling that the time for the radial model was 55 days and the dispersivity was 1 m:

$$T = 78 = Vt/\alpha$$

or

$$V = 1.42\, m/d$$

Using this velocity of 1.42 m/d, (39) can be used to determine the head in the drain as

3.36 m. Thus, for the drain to decontaminate the aquifer to 1% of the original concentration the head in the drain should be maintained at 3.36 m. Given this value of $h_d$, the amount of water flowing into the drain can be calculated using (35) as

$$
\begin{aligned}
Q' &= 2(100)Q_x \\
&= 236 \ m^3/day
\end{aligned}
$$

For the criteria to remove the water from the drain at the same rate as the well pumps in the radial model, it is recalled that the pumping rate for the single well was 10 m3/hr which for the drain corresponds to a flow rate per unit length $Q_x$ of 1.2 m2/d. Using this value of $Q_x$, (35) can be used to determine the required $h_d$ as 3.33 m. The head in the drain should thus be maintained at 3.33 m for 10 m3/hr of water to flow into the drain. The seepage velocity with this value of $h_d$ can be determined and then the time to reach 1% decontamination is found to be 54.2 days. In this example, both models yield approximately equal decontamination times and pumping rates. It is noted that the velocity and flow rate to the drain are both a function of the hydraulic conductivity in the one dimensional model, while the calculations in the radial model are independent of K. Thus, if a different value of K was assumed, the decontamination times and pumping rates for the two models would have been different.

Due to the simplicity of the one dimensional uniform dispersion solutions, any relationship that could be developed between the one dimensional and radial solutions would be of value. If some relationship could be derived or the two solutions could be shown to converge to each other under certain conditions, then the difficulties encountered in evaluating the radial solution under certain conditions as discussed in Appendix C may be avoidable. Thus, the radial and one dimensional solutions are compared for different Peclet numbers for the nonuniform and uniform initial conditions as in Figure 4a and Figure 4b. The Peclet number represents a ratio of advective to dispersive forces and is equal to the dimensionless distance $\rho$ for the radial model and the dimensionless distance $\chi = x/\alpha$ in the one dimensional model. Figure 20a shows the comparison for the nonuniform initial condition and Figure 20b for uniform initial condition. It is clearly seen that the two models yield very different results for the range of Peclet numbers shown in these figures. Also there is no apparent trend that would indicate the solutions are converging as the Peclet number becomes larger. When the Peclet number is 45, large differences are still present between the two solutions. This has been tested for Peclet numbers up to 75 and no trend is still detected. Using numerical methods, Sauty (1981) reported that the one dimensional and radial solutions converge for large Peclet numbers for the injection and withdrawal problems in his study. These included pulse and continuous injection problems and the withdrawal of a slug injected in an adjacent well. For the radial and one dimensional uniform dispersion models in this study, Figures 20a and 20b indicate that for the Peclet numbers analyzed here, convergence does not occur and no other trends are apparent. The use of the correct model which corresponds to the flow

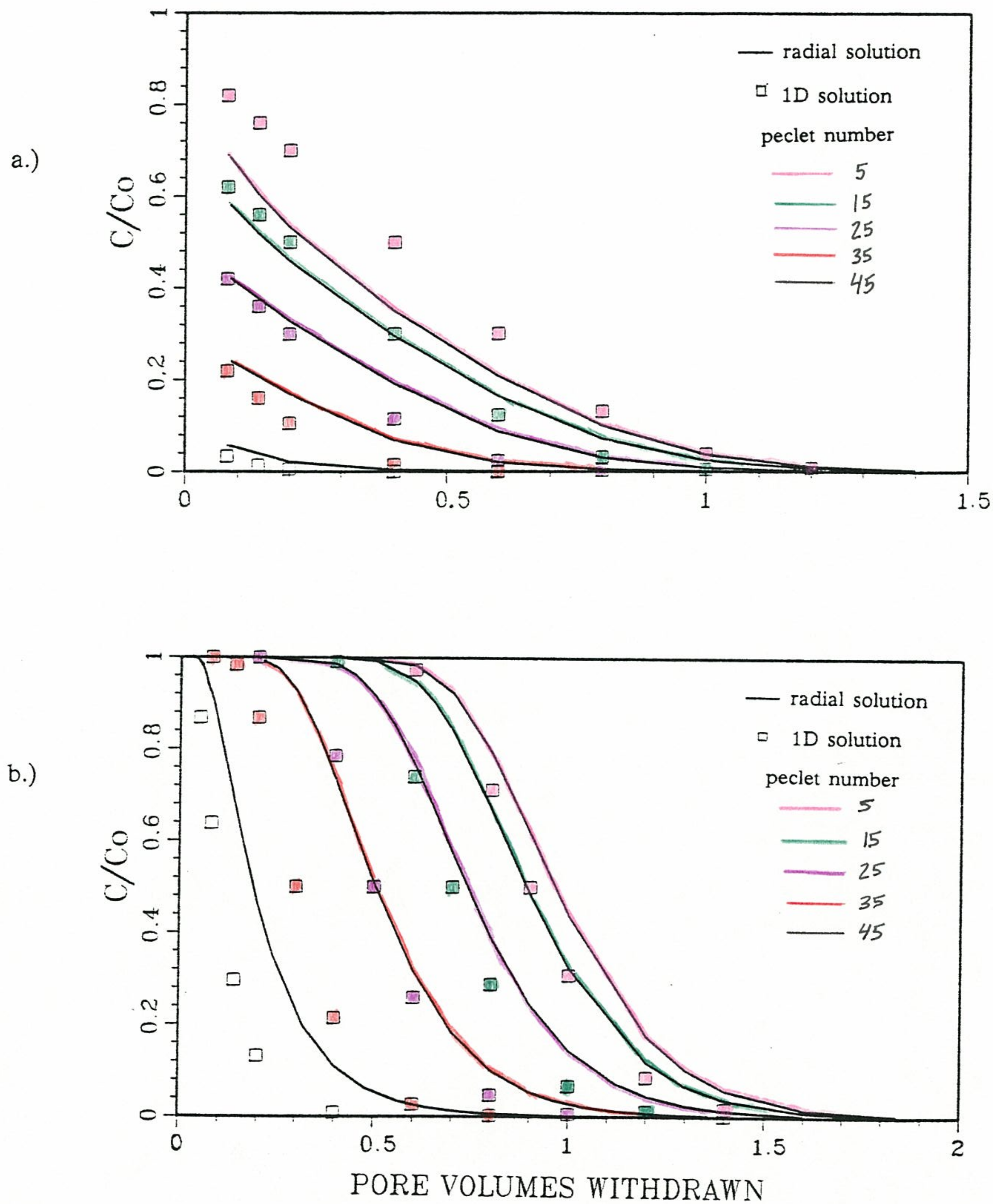



Figure 20 Radial solution (solid line) compared to one dimensional solution (boxes) for nonuniform (a) and uniform (b) initial conditions

field in the aquifer remediation operation is thus important. If the decontamination is performed with a pumping well, the one dimensional uniform dispersion solution cannot be substituted for the radial solution.

## SUMMARY AND CONCLUSIONS

Two mathematical models for aquifer remediation by pumping are proposed. Both assume that advection and longitudinal mechanical dispersion are the transport mechanisms. The model for decontamination by a withdrawal well accounts for a velocity dependent dispersion coefficient in the radially converging flow field. For simple one dimensional flow, the one dimensional model assumes a constant dispersion coefficient in the constant velocity flow field which may occur due to withdrawal using french drains or infiltration galleries. The radial dispersion model is solved for an arbitrary initial condition which can be formulated in a wide range of flexible, useful forms. The uniform dispersion model is solved for two simple initial conditions, resulting in solutions which are easily evaluated. Solutions for both models are derived through the use of Green's functions and the LaPlace transform.

When the initial conditions input into the models have large concentration gradients at the plume boundary, adverse backward dispersion occurs causing the solute to spread beyond the initial plume boundary. Adverse dispersion does not occur when the initial conditions are formulated such that gradients near the outer plume boundary gradually decrease to zero. The occurrence of adverse dispersion thus may be a mathematical artifact that occurs when the initial condition is formulated with large gradients near the plume boundary. As a special case of the radial model, the solution is compared to the approximate solution of Gelhar and Collins (1971) for the injection and withdrawal of solute from a single well and good agreement is found between the two solutions. Using the field data of Pickens and Grisak (1981), the radial solution also accurately models the concentration history of the single well tracer test. Graphical relationships giving the time required to reach one percent of the maximum initial concentration are developed for two simple initial conditions. It is determined that neglecting dispersion underestimates the total cleanup time for aquifer decontamination. A comparison of the two models at equivalent Peclet numbers shows that the two models do not converge over the range of Peclet numbers tested and that the use of the incorrect model would cause large error.

## NOMENCLATURE

$a = \sqrt{p/D}$ (s /m)
$A = Q/2\pi bn$ (m2/s)
b = aquifer thickness (m)
D = dispersion coefficient = $\alpha V$ (m2/s)
K = hydraulic conductivity (m/s)
L = distance from drain to edge of plume in 1D model (m)
n = porosity (dimensionless)
q = specific discharge = flow rate per unit area (m/s)
Q = pumping rate of well (m3/s)
Q' = flow rate into drain in radial model (m3/s)
r = radial distance (m)
t = time (s)
$Tr = Vt/x_1$ (m/m)
T = dimensionless time in 1D model = $Vt/\alpha$
V = seepage velocity (m/s)
x = distance (m)
X = dimensionless distance in 1D model = $x/\alpha$
$x_0$ = outlet distance (m)
$x_1$ = distance to edge of plume (m)
$\alpha$ = dispersivity (m)
$\beta = V/2\sqrt{D}$ (s )
$\rho$ = dimensionless distance in radial model = $r/\alpha$
$\rho_0$ = dimensionless well radius
$\tau$ = dimensionless time in radial model = $At/\alpha^2$

## REFERENCES


Abramowitz, M. and I. A. Stegun (Eds.), Handbook of Mathematical Functions, Applied Math Series, 55, National Bureau of Standards, Washington, D. C., 1970.

Ahlfeld, D. P., J. M. Mulvey, G. F. Pinder, and E. F. Wood, Contaminated groundwater remediation design using simulation, optimization and sensitivity theory, 1, Model Development, Water Resources Research, 24(3), 431–441, 1988.

Ahlfeld, D. P., J. M. Mulvey, and G. F. Pinder, Contaminated groundwater remediation design using simulation, optimization and sensitivity theory, 2, Analysis of a field site, Water Resources Research, 24(3), 443–452, 1988.

Al-Niami, A. N. S., and K. R. Rushton, Analysis of flow against dispersion in porous media, Jour. Hydrology, 33, 87–97, 1977.

Al-Niami, A. N. S., and K. R. Rushton, Radial dispersion to an abstraction well, Jour. Hydrology, 39, 287-300, 1978.

Bastian, W. C., and L. Lapidus, Longitudinal diffusion in ion exchange and chromatographic columns. Finite columns., Jour. Phys. Chem., 60:816–817, 1956.

Chen, C. S., Analytical and approximate solutions to radial dispersion from an injection well to a geological unit with simultaneous diffusion into adjacent strata, Water Resources Research, 21(8), 1069–1076, 1985.

Chen, C. S., Solutions for radionuclide transport from an injection well into a single fracture in a porous formation, Water Resources Research, 22(4), 508–518, 1986.

Chen, C. S., Analytical solutions for radial dispersion with Cauchy boundary at injection well, Water Resources Research, 23(7), 1217–1224, 1987.

Chen, C. S., and Greg D. Woodside, Analytical solution for aquifer decontamination by pumping, Water Resources Research, in press, 1988.

Dowd, R. M., Leaking underground storage tanks, Environ. Sci. Technol., 18:309A, 1984.

Eldor, M., and G. Dagan, Solutions of hydrodynamic dispersion in porous media, Water Resources Research, 7(1), 135–142, 1971.

Gelhar, L. W., and M. A. Collins, General analysis of longitudinal dispersion in nonuniform flow, Water Resources Research, 7(6), 1511–1521, 1971.

Gershon, N. D., and A. Nir, Effects of boundary conditions of models on tracer distribution in flow through porous mediums, Water Resources Research, 5(4):830–839, 1969.

Guvanasen, V., and V. M. Guvanasen, An approximate semianalytical solution for tracer injection tests in a confined aquifer with a radially converging flow field and finite volume of tracer and chase fluid, Water Resources Research, 23(8), 1607–1619, 1987.

Hsieh, P. A., A new formula for the analytical solution of the radial dispersion problem, Water Resources Research, 22(11), 1597–1605, 1986.

Kirkham, D., and S. B. Affleck, Solute travel time to wells, Groundwater, 15(3), 231–242, 1977.

Kumar, N., Unsteady flow against dispersion in finite porous media, Jour. Hydrology,

63, 345–358, 1983.
Lapidus, L., and N. R. Amundson, Mathematics of adsorption in beds. The effect of longitudinal diffusion in ion exchange and chromatographic columns, Jour. Phys. Chem., 56:984–988, 1952.
Mercado, A., Recharge and mixing tests at Yavne 20 well field, Underground Water Storage Study Tech. Rep. 12, Publ. 611, TAHAL–Water Planning for Israel, Ltd., Tel Aviv, 1966.
Oberhettinger, F., and L. Badii, Tables of LaPlace Transforms, Springer–Verlag, 1973.
Ogata, A., and R. B. Banks, A solution of the differential equation of longitudinal dispersion in porous media, U.S.G.S. Prof. Paper 411–A, 1961.
Office of Technology Assessment, Protecting the Nation's Groundwater from Contamination, OTA–0–233, U. S, Government Printing Office, Washington, D. C., 1984.
Philips, K. J., and L. W. Gelhar, Contaminant transport to deep wells, Jour. Hydraulics Div., ASCE, 104, HY6, 807–812, 1978.
Pickens, J. F., and G. E. Grisak, Scale–dependent dispersion in a stratified granular aquifer, Water Resources Research, 17(4), 1191–1211, 1981.
Sauty, J. P., An analysis of hydrodispersive transfer in aquifers, Water Resources Research, 16(1), 145–158, 1980.
Valocchi, A. J., Effect of radial flow on deviations from local equilibrium during sorbing solute transport through homogeneous soils, Water Resources Research, 22(12), 1693–1701.
Van Genuchten, M. Th., and P. J. Wierenga, Mass transfer studies in sorbing porous media I, Analytical solutions, Soil Sci. Soc. of America, 40(4):473–480, 1976.
Wylie, C. R., and L. C. Barrett, Advanced Engineering Mathematics, 5th Ed., McGraw Hill, New York, 1982.

## APPENDIX A

In this appendix, the four unknown coefficients in (14) and (15) are derived using the conditions (12), (13), (16), and (17). Condition (13) indicates that the Green's functions must be bounded and thus $A_1$ must be zero and

$$g_1 = A_2\, exp(-ax) \tag{A1}$$

Using (12),

$$\frac{dg_2}{dx} - (V/2D)g_2 = 0 \qquad at\ x = x_0 \tag{A2}$$

which yields

$$A_3 = A_4\, X\, exp(-2ax_0) \tag{A3}$$

where $X = (\sqrt{p} + \beta)/(\sqrt{p} - \beta)$

Using (A3), $g_2$ becomes

$$g_2 = A_4\, [exp(a(x - 2x_0))X + exp(-ax)] \tag{A4}$$

Application of (16) to (A1) and (A4) yields

$$A_2\, exp(-as) = A_4\, [exp(a(s - 2x_0))X + exp(-as)] \tag{A5}$$

Application of (17) to (A1) and (A4) yields

$$-A_2\, a\, exp(-as) \;-\; A_4\,[a\, exp(a(s-2x_0))X \;-\; a\, exp(-as)] \;=\; -1/D \tag{A6}$$

(A5) and (A6) can now be solved simultaneously to determine $A_2$ and $A_4$ as

$$A_4 \;=\; exp(a(2x_0-s))/(2\sqrt{pD}\,X) \tag{A7}$$

$$A_2 \;=\; [exp(as) \;+\; exp(a(2x_0-s))/X]/(2\,\sqrt{pD}) \tag{A8}$$

Substituting (A7) into (A3) yields

$$A_3 \;=\; exp(-as)/(2\sqrt{pD}) \tag{A9}$$

After substituting (A7), (A8), and (A9) into (14) and (15), the Greens' function are found to be (18) and (19).

## APPENDIX B

In this appendix the integration and LaPlace inversion of (21) is outlined for the initial condition $f(x) = 1 - m(x - x_0)$. The integral to be evaluated is

$$\overline{G} = IG_1 + IG_2 \qquad \text{(B1)}$$

$$IG_1 = (1/2\sqrt{pD}) \int_{x_0}^{x} [exp(a(s-x) + exp(a(2x_0 - s - x)/X][1 - m(s - x_0)]exp(Vs/2D)ds \qquad \text{(B2)}$$

$$IG_2 = (1/2\sqrt{pD}) \int_{x}^{x_1} [exp(a(x-s) + exp(a(2x_0 - s - x)/X][1 - m(s - x_0)]exp(Vs/2D)ds \qquad \text{(B3)}$$

Note that (B2) and (B3) are derived by substituting the expressions (18) and (19) in for $g_1$ and $g_2$ in (21). The integrations in (B2) and (B3) are similar so only the details for (B2) will be outlined. (B2) can be rearranged as

$$\begin{aligned} IG_1 = {} & (1 + mx_0)/2\sqrt{pD} \int_{x_0}^{x} [exp(a(s-x) + Vs/2D) + exp(a(2x_0 - s - x) + Vs/2D)/X]ds \\ & - (m/2\sqrt{pD}) \int_{x_0}^{x} s[exp(a(s-x) + Vs/2D) + exp(a(2x_0 - s - x) + Vs/2D)/X]ds \end{aligned} \qquad \text{(B4)}$$

In (B4) the first integral is a simple exponential integration and is easily performed. The second integral is slightly more detailed but is done with a simple integration by parts. Without showing the lengthy algebraic details, the integration of (B1) can be found to be

$$\overline{G} = I_1 + I_2 + I_3 + I_4 + I_5 + I_6$$

where (B5)

$$I_1 = \frac{(1 - m(x - x_0))\exp(Vx/2D)}{p - V^2/4D} \qquad (B6)$$

$$I_2 = \frac{-mV\exp(Vx/2D)}{(p - V^2/4D)^2} \qquad (B7)$$

$$I_3 = \frac{-0.5\, mD^{1/2}exp(Vx_0/2D)exp(a(x_0 - x))}{p^{1/2}(\beta + p^{1/2})^2} \qquad (B8)$$

$$I_4 = \frac{-0.5\, mD^{1/2}exp(Vx_0/2D)exp(a(x_0 - x))}{p^{1/2}(p - V^2/4D)} \qquad (B9)$$

$$I_5 = \frac{0.5mD^{1/2}exp(Vx_1/2D)exp(a(x - x_1)}{p^{1/2}(\beta - p^{1/2})^2} \qquad (B10)$$

$$I_6 = \frac{0.5mD^{1/2}exp(Vx_1/2D)exp(a(2x_0 - x - x_1)}{p^{1/2}(p - V^2/4D)} \qquad (B11)$$

(B5) is the solution to (6) in the LaPlace domain for the initial condition in Figure (4a). The inversion of the terms $I_1$, $I_2$, $I_3$, and $I_5$ are all published. For the inversion of $I_1$ and $I_2$, see for example Abramowitz and Stegun (1970; p.1022). The inversions of $I_3$ and $I_5$ are given by Oberhettinger and Badii (1973; p. 260, Eq. 5.103). Published inversions of $I_4$ and $I_6$ could not be found. $I_4$ and $I_6$ are both of the form

$$f(p) = b\ \exp(-k\sqrt{p})/(\sqrt{p}(c + p) \qquad (B12)$$

where b, c, and k are constants independent of p. The inversion of (B12) is through the use of the Convolution Theorem which gives the inversion of (B12) in integral form (see for example Wylie and Barrett (1982), pp. 451–452). For (B12) the Convolution Theorem is used by factoring (B12) as

$$f(p) = [b/(c + p)][\exp(-k\sqrt{p})/\sqrt{p}] \tag{B13}$$

The inversions for each term in square brackets in (B13) are all simple exponentials and after application of the Convolution Theorem the inversion of (B12) is found to be

$$L^{-1}(f(p)) = b\int_0^t [exp(c(t-u)exp(-k^2/4u)/\sqrt{\pi u}\,]du \tag{B14}$$

To carry out the integration in (B14) it is rearranged as

$$L^{-1}(f(p)) = bexp(ct)/\sqrt{\pi}\int_0^t [exp(-cu - k^2/4u)/\sqrt{u}]du \tag{B15}$$

Using the variable change $u = z^2$, (B15) is transformed to

$$L^{-1}(f(p)) = 2bexp(ct)/\sqrt{\pi}\int_0^{\sqrt{t}} exp(-cz^2 - k^2/4z^2)dz \tag{B16}$$

The integral in (B16) can be found in Abramowitz and Stegun (1970; p. 304, Eq. 7.4.33). After appropriately substituting the constants in (B9) and (B11) in for b, c, and k in (B16) and making algebraic manipulations, the inversions of $I_4$ and $I_6$ are found to be

$$L^{-1}(I_4 + I_6) = exp(Vx/2D + V^2 t/4D)mD/2V[-exp(V(x_1 - x_0)/D)erfc(k) -$$

$$exp(V(x_0 - x)/D)erfc(m) + erfc(l) + exp(V(x_0 - x)/D)erfc(h)] \qquad \text{(B17)}$$

The outline of the integration and LaPlace inversion of (21) for the initial condition f(x) = 1 - m(x - $x_0$) is thus complete.

## APPENDIX C

In this appendix the numerical evaluation of the radial solution given by (29) is discussed. In the numerical evaluation, the order of integration in (29) is interchanged as in equation (36) of Chen and Woodside (1988). This is done because the terms which depend on the variable s can be easily integrated first. Also, the oscillation of the integrand as a function of x is more regular and thus easier to handle when s is integrated first. The integration is not performed on the complete integrand at once; instead the integrand is factored as shown below to allow for the terms which depend on s to be grouped in the most efficient manner. The integration is thus performed on five different terms and these terms are added up to give the complete answer. The terms used to designate the factored integrand are shown below. Also all input variables, all the important variables used in the program, and the integration technique are explained in the program. The program was written on a MicroVax II with Fortran 77.

Some of the variables in (29) are renamed in the program as follows:

| Variable in (29) | Variable in program |
|---|---|
| s | csi |
| $\psi$ | si |
| $\phi$ | phi |
| $\phi_0$ | phiw |

Terms used to indicate factors in the integrand:

$$H1 = F(csi)Ai(si)$$

$$H2 = F(csi)Bi(si)$$

where F(csi) = csi*exp(csi/2)*f(csi)     f(csi) = initial condition

Note that H1 and H2 are the only terms which depend on csi; two integrations for csi are thus performed. They are:

$$ZINTH1 = \int_{\rho_0}^{\infty} H1dcsi$$

$$ZINTH2 = \int_{\rho_0}^{\infty} H2dcsi$$

The five terms which the integration with respect to x is carried out on are:

$$H3 = 3x^{**}(1/3)\exp(-x^2\tau)Ai(phi)$$

$$H4A = x^{**}(1/3)\exp(-x^2\tau)[1 + (3f_1f_1 - f_2f_2)/(f_1f_1 + f_2f_2)]Bi(phi)$$

$$H5 = x^{**}(1/3)\exp(-x^2\tau)[4f_1f_2/(f_1f_1 + f_2f_2)]Ai(phi)$$

$$H6 = x^{**}(1/3)\exp(-x^2\tau)[4f_1f_2/(f_1f_1 + f_2f_2)]Bi(phi)$$

$$H7 = x^{**}(1/3)\exp(-x^2\tau)[3f_1f_1 - f_2f_2/(f_1f_1 + f_2f_2)]Ai(phi)$$

The five integrations are thus referred to as follows. Note that since ZINTH1 and ZINTH2 are functions of x, these two integrations must be carried out for each x in the integration with respect to x.

$$ZINT1 = \int_0^\infty H3 * ZINTH1dx$$

$$ZINT2 = \int_0^\infty H4A * ZINTH2dx$$

$$ZINT3 = \int_0^\infty H5 * ZINTH2dx$$

$$ZINT4 = \int_0^\infty H6 * ZINTH1dx$$

$$ZINT5 = \int_0^\infty H7 * ZINTH1dx$$

```
*********************************************************************

                          PROGRAM CLN17A

                  CURRENT VERSION AS OF MAY 2, 1988

       THIS PROGRAM PERFORMS THE DOUBLE INTEGRATION FOR THE AQUIFER
       RESTORATION MODEL FOR RADIAL FLOW TO A WITHDRAWAL WELL;
       INTEGRATION IS WITH RESPECT TO CSI FIRST AND THEN IN TERMS OF X;
       TWENTY POINT GAUSSIAN QUADRATURE IS USED FOR BOTH INTEGRATIONS.
       THE INNER INTEGRATION WITH RESPECT TO CSI IS PERFORMED BY USING
       20 POINT GAUSSIAN QUADRATURE FROM ROOT TO ROOT (THE ROOTS ARE FOUND
       USING THE BRENT ALGORITHM).  THE OUTER INTEGRATION WITH RESPECT TO X
       CAN BE PERFORMED WITH 2 OPTIONS.  ONE OPTION IS TO CHOOSE A SPACING
       CALLED AN 'INTEGRATION DISTANCE' AND CALCULATE THE INTEGRAL OVER EACH
       INTEGRATION DISTANCE UNTIL THE CONTRIBUTION DECREASES BELOW A
       CONVERGENCE CRITERIA. IF THIS OPTION IS USED THEN THE PROGRAM CHOOSES
       THE VALUE OF THE INTEGRATION DISTANCE (IT IS THE VARIABLE 'INTDIS' IN
       THE PROGRAM.  THE OTHER OPTION IS FOR THE
       USER TO INPUT A SET OF INTEGRATION DISTANCES WHICH DECREASE IN VALUE
       AND THEN THE PROGRAM CALCULATES THE INTEGRALS FOR EACH INTEGRATION
       DISTANCE UNTIL THE DIFFERENCE FOR TWO DIFFERENT INTEGRATION DISTANCES
       IS LESS THAN A CONVERGENCE CRITERIA.  THIS SECOND OPTION IS THUS MORE
       COMPUTATIONALLY BURDENSOME BECAUSE THE INTEGRALS ARE CALCULATED MORE
       THAN ONE TIME TO COMPARE THE RESULTS FOR DIFFERENT INTEGRATION
       DISTANCES.

       THE PROGRAM WILL WORK FOR R1(DEFINED BELOW) LESS THAN AROUND 80; IF R1
       IS > 55 OR SO THE PROGRAM WILL TAKE A LONG TIME TO RUN SO DONT BE
       SURPRISED.  THIS IS BECAUSE THE LARGER R1 IS THE MORE THE INTEGRAND
       OSCILLATES AND THUS THE SMALLER THE VALUE OF 'INTDIS' MUST BE.


       INPUT VARIABLES                         DESCRIPTION

           RW                                  DIMENSIONLESS WELL RADIUS

           R                                   DIMENSIONLESS RADIAL DISTANCE

           R1                                  DIMENSIONLESS DISTANCE TO
                                               EDGE OF PLUME

           R1A                                 USED IF IFLOW = 2
                                               DISTANCE TO CHANGE IN INITIAL
                                               CONDITION FROM F = 1 TO F =
                                               EXP(-COEF1*(R-R1A)**2)


           ISLOPE                              INDICATOR FOR THE
                                               INITIAL PLUME CONCEN
                                               0==> SLOPE = 0
                                               SO THAT F(R) = 1 RW<R<R1

                                               1==> SLOPE = 1/(R1 - RW)
                                               SO THAT F(R) = 1 -
                                                      SLOPE(R - RW)

                                               2==> F(R)= EXP(-A(R-RW)**2)
                                               A = COEF1 IN THIS PROGRAM
```

| | |
|---|---|
| | 3==> F(R) = 1 FOR R <= R1A,<br>EXP(-COEF1*(R-R1A)**2) FOR<br>R >=R1A |
| TIME | DIMENSIONLESS TIME |
| NINTDS | THE NUMBER OF INTEGRATION<br>INTERVALS THAT CAN BE USED<br>FOR THE DX INTEGRATION<br>IF IFLOW = 1 |
| INTDIS | ARRAY FOR THE INTEGRATION<br>DISTANCES; USED<br>IF IFLOW = 1; SET TO ZERO<br>IF IFLOW = 2 |
| EPS2 | CONV. CRITERIA FOR CHANGING<br>THE INTERVAL SPACING (WHEN<br>DIFFERENCE BETWEEN INTEGRATIONS<br>FOR 2 INTERVAL SPACINGS ARE<br>LESS THEN EPS2 THEN STOP);<br>USED IF IFLOW = 1; SET TO ZERO<br>IF IFLOW = 2 |
| IFLOW | FLAG TO DETERMINE INTEGRATION<br>METHOD<br>1 = INTEGR. FOR EACH INTDIS<br>VALUE UNTIL EPS2 IS SATISFIED<br>2 = INTEGR. FOR ONE VALUE OF<br>INTDIS ONLY; NOTE THAT INTDIS<br>FOR THIS OPTION IS CHOSEN BY<br>THE PROGRAM |
| ISKIP | FLAG TO DETERMINE IF ANY<br>TERMS IN THE INTEGRATION ARE<br>TO BE SKIPPED<br>1 = SKIP ZIN2<br>2 = SKIP ZINT2 AND ZINT3<br>3 = SKIP ZINT2, ZINT3,<br>AND ZINT4<br>BE CAREFUL IF YOU DO THIS AND<br>DOUBLE CHECK THAT<br>THE ANSWER IS OKAY<br>BY INTEGRATING THE WHOLE THING<br>ONCE AND COMPARING THE RESULTS.<br>NO GENERAL RULE HAS BEEN FOUND<br>FOR SKIPPING TERMS IN THE INTE-<br>GRATION EXCEPT THAT ZINT2 CAN<br>BE SKIPPED IF R1 < 80. |
| IPRINT1 | FLAG FOR PRINT OPTION<br>1 = PRINT INTERMEDIATE<br>RESULTS; ELSE NO INTEM. RESULTS |

```
C
C        IOUT                                OUTPUT OPTIONS
C                                            1 = CONC VS DIMENSIONLESS RAD.
C                                            FOR GIVEN TIME
C                                            2 = CONC VS VOLUME WITHDRAWN
C                                            OVER VOLUME INJECTED =
C                                                2*TAU/(R1*R1)
C                                            THIS VALUE IS EQUAL TO THE
C                                            NUMBER OF PORE VOLUMES
C                                            WITHDRAWN.
C                                            ONLY USE IOUT = 2 IF ISLOPE
C                                            IS EQUAL TO 0 OR 1
C
C     OTHER IMPORTANT VARIABLES
C
C
C        TOBIG1                              FOR PHIW > TOBIG1, THE
C                                            INTEGRANDS ARE SIMPLIFIED
C                                            USING ASYMPTOTIC VALUES
C
C        XSTART                              LOWER LIMIT FOR THE DX
C                                            INTEGRATION
C
C        EPS1                                CONVERGENCE CRITERIA FOR
C                                            STOPPING INTERVAL BY INTERVAL
C                                            INTEGRATION (WHEN CONTRIBUTION
C                                            IS LESS THAN EPS1 THEN STOP)
C
C
C***************************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 INTDIS

      DIMENSION INTDIS(5)

      COMMON /VALUES/ R, RW, R1, R1A, SLOPE, TIME
      COMMON /TERMS / NINTDS, INTDIS, EPS2
      COMMON /PRINT / IFLOW, ISKIP, IPRINT1, NOUT
      COMMON /SCALE / TOBIG1
      COMMON /ZINITL/ ISLOPE, COEF1

      DATA NIN, NOUT/ 30, 31/

      OPEN (UNIT=NIN,FILE='cln17a.in',STATUS='OLD')
      OPEN (UNIT=NOUT,FILE='cln17a.out',STATUS='NEW')

      READ(NIN,*) RW, R1, R1A, ISLOPE
      READ(NIN,*) COEF1
      READ(NIN,*) NOT, TSTART, DT
      READ(NIN,*) NOR, RSTART, DR
      READ(NIN,*) NINTDS, EPS2
      READ(NIN,*) INTDIS
      READ(NIN,*) IFLOW, ISKIP, IPRINT1
      READ(NIN,*) IOUT

      WRITE(NOUT,900)
      WRITE(NOUT,902)

      IF(IFLOW.EQ.2) WRITE(NOUT,895)
      IF(ISKIP.EQ.1) WRITE(NOUT,880)
```

```
      IF(ISKIP.EQ.2) WRITE(NOUT,882)
      IF(ISKIP.EQ.3) WRITE(NOUT,883)
      IF(ISLOPE.EQ.1)THEN
          SLOPE = 1.D0/(R1 - RW)
      ELSEIF(ISLOPE.EQ.0)THEN
          SLOPE = 0.D0
      ELSEIF(ISLOPE.EQ.2) THEN
          WRITE(NOUT,898)
          WRITE(NOUT,899) COEF1
          WRITE(*,898)
          WRITE(*,899) COEF1
      ELSEIF(ISLOPE.EQ.3) THEN
          WRITE(NOUT,891)
          WRITE(NOUT,892) R1A, COEF1
          WRITE(*,891)
          WRITE(*,892) R1A, COEF1
      ENDIF

      WRITE(NOUT,*) 'RW = ', RW,'  R1 = ',R1
      WRITE(*,*) 'RW = ',RW,'  R1 = ',R1
      WRITE(NOUT,*) 'SLOPE = ',SLOPE
      WRITE(*,*) 'SLOPE = ',SLOPE
      WRITE(NOUT,*) '  EPS2 = ',EPS2

      IF(IOUT.EQ.1) THEN
         WRITE(NOUT,908)
         TIME = TSTART
         DO 10 I=1, NOT
            WRITE(NOUT,920) TIME
            R = RSTART
            DO 20 J=1, NOR
               CALL DINTEGR(CONCEN)
               WRITE(NOUT,930) R, CONCEN
20             R = R + DR
10       TIME = TIME + DT

      ELSEIF(IOUT.EQ.2) THEN
         WRITE(NOUT,910)
         R = RSTART
         DO 30 I=1, NOR
            TIME = TSTART
            WRITE(NOUT,915) R
            DO 40 J=1, NOT
               CALL DINTEGR(CONCEN)
               TR = 2.D0*TIME/(R1*R1)
               WRITE(NOUT,930) TR, CONCEN
40          TIME = TIME + DT
30       R = R + DR
      ENDIF
      STOP

880   FORMAT(1X,'SKIPPING INTEGRATION OF G2A',/)
882   FORMAT(1X,'SKIPPING INTEGRATION OF G2A AND G3',/)
883   FORMAT(1X,'SKIPPING INTEG. OF G2A, G3, AND G4',/)
891   FORMAT(/,1X,'INITIAL COND. F = 1 OR EXP(-COEF1(R-R1A)**2) IF ')
892   FORMAT(1X,'R > R1A=',F9.4,' WITH COEF1 = ',F11.5,/)
895   FORMAT(1X,'INTEGRATING WITH ONE INTDIS ONLY',/)
898   FORMAT(/,1X,'INITIAL COND. F = EXP(-COEF1(R-RW)**2) WHERE')
899   FORMAT(1X,'COEF1 = ',1PE13.6,/)
900   FORMAT(1X,'AQUIFER REST. MODEL FOR RADIAL FLOW; PROGRAM TO')
```

```
902    FORMAT(1X,'PERFORM THE DBL INTEG.; MARCH/APRIL 1988',/)
908    FORMAT(/,6X,' R ',11X,'CONCEN',/)
910    FORMAT(/,3X,'TR',9X,'CONCEN',/)
915    FORMAT(/,4X,'PECLET NUMBER = ',F11.4,/)
920    FORMAT(/,5X,'TIME = ',F10.3,/)
930    FORMAT(2X,F11.4,5X,1PE13.6)

       END

       SUBROUTINE DINTEGR(CONCEN)
C***********************************************************************
C
C      THIS SUBROUTINE PERFORMS THE DOUBLE INTEGRATION
C
C      IMPORTANT VARIABLES:
C
C      L1               VARIABLE USED FOR THE LOWER LIMIT OF INTEGRATION
C                       IN THE INTEGRATION WITH RESPECT TO X
C
C      L2               VARIABLE USED FOR THE UPPER LIMIT OF INTEGRATION
C                       IN THE INTEGRATION WITH RESPECT TO X
C
C      S                VARIABLE USED IN CALCULATING THE NUMERICAL INTE-
C                       GRATION VALUE FOR EACH INTERVAL GAUSSIAN QUADRATURE
C                       IS USED IN
C
C      SUM              VARIABLE WHICH IS INCREMENTED SEQUENTIALLY FOR EACH
C                       INTERVAL GAUSSIAN QUAD. IS USED IN AND IS EQUAL TO
C                       THE VALUE OF THE OUTER INTEGRAL
C
C      DINTG1,2,3,4,5
C                       EACH OF THESE VARIABLES IS THE VALUE OF THE OUTER
C                       INTEGRAL IN ONE INTERVAL OF SPACING INTDIS AND IS
C                       THUS CALCULATED ONCE FOR EACH INTERVAL;  EACH ONE
C                       CORRESPONDS TO A DIFFERENT PART OF THE INTEGRAL DUE
C                       TO THE WAY THE INTEGRAND IS FACTORED: DINTG1 CORRE-
C                       SPONDS WITH ZINT1, DINTG2 WITH ZINT2, DINTG3 WITH
C                       ZINT3, DINTG4 WITH ZINT4 AND DINTG5 GOES WITH ZINT5
C
C      ZINT1,2,3,4,5
C                       EACH VARIABLE REPRESENTS THE VALUE OF THE INTEGRATION
C                       FOR A SPECIFIED TERM IN THE INTEGRAND.
C                       IF IFLOW = 1, THEN WHEN THE DIFFERENCE
C                       BETWEEN TWO SUCCESSIVE VALUES OF ZINT1(OR ZINT2, 3 ETC)
C                       ARE LESS THAN EPS2, THEN THE INTEGRATION FOR THAT TERM
C                       IS COMPLETE.  IF IFLOW = 2, THEN CALCULATE ONLY ONE
C                       VALUE FOR ZINT1 THROUGH ZINT5.
C
C
C      THE VARIABLES AND FUNCTIONS WHICH CALCULATE THE INTEGRAND VALUES WERE
C      GIVEN AT THE BEGINNING OF THIS APPENDIX.  H1 THROUGH H7 ARE SET UP AS
C      FUNCTIONS.  THE SUBROUTINE 'INTEGH1' INTEGRATES H1 AND THE SUBROUTINE
C      'INTEGH2' INTEGRATES H2.  SUBROUTINE 'FROOT1' DETERMINES THE ROOTS OF
C      INTEGRAND H1 AND SUBROUTINE 'FROOT2' DETERMINES THE ROOTS OF H2.
C
C***********************************************************************
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       REAL*8 INTDIS, L1, L2

       DIMENSION Z(10), WT(10), INTDIS(5)
```

```
      DIMENSION ZINT1(5), ZINT2(5), ZINT3(5), ZINT4(5), ZINT5(5)

      COMMON /VALUES/ R, RW, R1, R1A, SLOPE, TIME
      COMMON /TERMS/  NINTDS, INTDIS, EPS2
      COMMON /PRINT/ IFLOW, ISKIP, IPRINT1, NOUT
      COMMON /SCALE / TOBIG1

      DATA TOBIG1, XSTART, EPS1/ 10.D0, 0.05D0, 1.D-10/

      DATA Z /0.076526521133497D0,0.227785851141645D0,
     1       0.373706088715420D0 ,0.510867001950827D0,
     2       0.636053680726515D0 ,0.746331906460151D0,
     3       0.839116971822219D0 ,0.912234428251326D0,
     4       0.963971927277914D0 ,0.993128599185095D0/
     DATA WT/0.152753387130726D0 ,0.149172986472604D0,
     1       0.142096109318382D0 ,0.131688638449177D0,
     2       0.118194531961518D0 ,0.101930119817240D0,
     3       0.083276741576705D0,0.062672048334109D0,
     4       0.040601429800387D0,0.017614007139152D0/

      IF(R1.GT.(65.D0)) XSTART = 0.04D0
C------ IF USING ONLY ONE INTEGRATION DISTANCE FOR THE DX INTEGRATION
C       THEN CHOOSE THE INTDIS
      IF(IFLOW.EQ.2) THEN
         ZINDIS = 0.3D0
         IF(R1.GT.12.D0) ZINDIS = 0.2D0
         IF(R1.GT.19.D0) ZINDIS = 0.1D0
         IF(R1.GT.29.D0) ZINDIS = 0.03D0
         IF(R1.GT.39.D0) ZINDIS = 0.02D0
         IF(R1.GT.49.D0) ZINDIS = 0.01D0
         IF(R1.GT.54.D0) ZINDIS = 0.005D0
         IF(R1.GT.64.D0) ZINDIS = 0.002D0
         IF(R1.GT.74.D0) ZINDIS = 0.001D0
         INTDIS(1) = ZINDIS
      ENDIF

      IF(IPRINT1.NE.1) GOTO 2
      WRITE(NOUT,906)
2     CONTINUE
C***********************************************************************
C-------INTEGRATION OF ZINT1
      WRITE(*,*) 'INTEGRATING ZINT1'
      WRITE(NOUT,*) 'INTEGRATING ZINT1'
      WRITE(NOUT,*)

      S = 0.D0
      CONCEN = 0.D0
      TOTAL = 0.D0
      DO 100 JJ = 1, NINTDS
         L1 = XSTART
         NUMINT = 0
         SUM = 0.D0

105         L2 = L1 + INTDIS(JJ)
            NUMINT = NUMINT + 1

C     PERFORM THE INTEGRATION

            DO 120 J = 1, 2
               DO 130 I = 1, 10
```

```
                IF(J.EQ.2) ZAZA = -Z(I)
                IF(J.EQ.1) ZAZA =  Z(I)
                ZZI = (L2 - L1)*ZAZA/2.D0 + (L2 + L1)/2.D0

                CALL INTEGH1(ZZI,ZINTH1)

                S = S + WT(I)*H3(ZZI)*ZINTH1
.30           CONTINUE
.20         CONTINUE

            DINTG1 = (L2 - L1)*S/2.D0
            S = 0.D0
            SUM = SUM + DINTG1
            IF(IPRINT1.NE.1) GOTO 112
            WRITE(NOUT,920) L2, DINTG1
l12         CONTINUE

            IF((DABS(DINTG1).LT.EPS1).AND.(NUMINT.GT.8))
     1               GOTO 180
            L1 = L2
            GOTO 105

l80        CONTINUE
           ZINT1(JJ) = SUM
           IF(IPRINT1.NE.1) GOTO 188
           WRITE(NOUT,930) ZINT1(JJ)

l88        IF(JJ.EQ.1)THEN
              IF(IFLOW.EQ.2) GOTO 190
              GOTO 100
           ELSE
              IF(DABS(ZINT1(JJ) - ZINT1(JJ-1)).LT.EPS2) GOTO 190
           ENDIF
l00     CONTINUE
        WRITE(*,*) 'FAILED FOR EPS2 WHILE INTEG. ZINT1'
        WRITE(NOUT,*) 'FAILED FOR EPS2 WHILE INTEG. ZINT1'
l90     TOTAL = TOTAL + ZINT1(JJ)

C-----------------------------------------------------------------------
        IF(ISKIP.EQ.1) GOTO 291
        IF(ISKIP.EQ.2) GOTO 291
        IF(ISKIP.EQ.3) GOTO 291
C***********************************************************************
C-------INTEGRATION OF ZINT2
        WRITE(*,*) 'INTEGRATING ZINT2'
        WRITE(NOUT,*) 'INTEGRATING ZINT2'
        WRITE(NOUT,*)

        DO 200 JJ = 1, NINTDS
           L1 = XSTART
           NUMINT = 0
           SUM = 0.D0

205          L2 = L1 + INTDIS(JJ)
             NUMINT = NUMINT + 1

C            PERFORM THE INTEGRATION

             DO 220 J = 1, 2
```

```
            DO 230 I = 1, 10

               IF(J.EQ.2) ZAZA = -Z(I)
               IF(J.EQ.1) ZAZA =  Z(I)
               ZZI = (L2 - L1)*ZAZA/2.D0 + (L2 + L1)/2.D0

               CALL INTEGH2(ZZI,ZINTH2)

               S = S + WT(I)*H4A(ZZI)*ZINTH2

230         CONTINUE
220      CONTINUE

         DINTG2 = (L2 - L1)*S/2.D0
         S = 0.D0
         SUM = SUM + DINTG2
         IF(IPRINT1.NE.1) GOTO 212
         WRITE(NOUT,920) L2, DINTG2
212      CONTINUE

         IF((DABS(DINTG2).LT.EPS1).AND.(NUMINT.GT.8))
    1        GOTO 280
         L1 = L2
         GOTO 205

280    CONTINUE
       ZINT2(JJ) = SUM
       IF(IPRINT1.NE.1) GOTO 288
       WRITE(NOUT,930) ZINT2(JJ)

288    IF(JJ.EQ.1)THEN
          IF(IFLOW.EQ.2) GOTO 290
          GOTO 200
       ELSE
          IF(DABS(ZINT2(JJ) - ZINT2(JJ-1)).LT.EPS2) GOTO 290
       ENDIF
200   CONTINUE
      WRITE(*,*) 'FAILED FOR EPS2 WHILE INTEGRATING ZINT2'
      WRITE(NOUT,*) 'FAILED FOR EPS2 WHILE INTEGRATING ZINT2'
290   TOTAL = TOTAL + ZINT2(JJ)
291   CONTINUE

C-----------------------------------------------------------------------
      IF(ISKIP.EQ.2) GOTO 391
      IF(ISKIP.EQ.3) GOTO 391
C***********************************************************************
C-------INTEGRATION OF ZINT3
      WRITE(*,*) 'INTEGRATING ZINT3'
      WRITE(NOUT,*) 'INTEGRATING ZINT3'
      WRITE(NOUT,*)

      DO 300 JJ = 1, NINTDS
         L1 = XSTART
         NUMINT = 0
         SUM = 0.D0

305      L2 = L1 + INTDIS(JJ)
         NUMINT = NUMINT + 1

C        PERFORM THE INTEGRATION
```

```
            DO 320 J = 1, 2
               DO 330 I = 1, 10

                  IF(J.EQ.2) ZAZA = -Z(I)
                  IF(J.EQ.1) ZAZA =  Z(I)
                  ZZI = (L2 - L1)*ZAZA/2.D0 + (L2 + L1)/2.D0

                  CALL INTEGH2(ZZI,ZINTH2)

                  S = S + WT(I)*H5(ZZI)*ZINTH2

330            CONTINUE
320         CONTINUE

            DINTG3 = (L2 - L1)*S/2.D0
            S = 0.D0
            SUM = SUM + DINTG3
            IF(IPRINT1.NE.1) GOTO 312
            WRITE(NOUT,920) L2, DINTG3
312         CONTINUE

            IF((DABS(DINTG3).LT.EPS1).AND.(NUMINT.GT.8))
     1         GOTO 380
            L1 = L2
            GOTO 305

380       CONTINUE
          ZINT3(JJ) = SUM
          IF(IPRINT1.NE.1) GOTO 388
          WRITE(NOUT,930) ZINT3(JJ)

388       IF(JJ.EQ.1)THEN
             IF(IFLOW.EQ.2) GOTO 390
             GOTO 300
          ELSE
             IF(DABS(ZINT3(JJ) - ZINT3(JJ-1)).LT.EPS2) GOTO 390
          ENDIF
300    CONTINUE
       WRITE(*,*) 'FAILED FOR EPS2 WHILE INTEGRATING ZINT3'
       WRITE(NOUT,*) 'FAILED FOR EPS2 WHILE INTEGRATING ZINT3'
390    TOTAL = TOTAL - ZINT3(JJ)
391    CONTINUE

C-----------------------------------------------------------------------
       IF(ISKIP.EQ.3) GOTO 491
C**********************************************************************
C-------INTEGRATION OF ZINT4
       WRITE(*,*) 'INTEGRATING ZINT4'
       WRITE(NOUT,*) 'INTEGRATING ZINT4'
       WRITE(NOUT,*)

       DO 400 JJ = 1, NINTDS
          L1 = XSTART
          NUMINT = 0
          SUM = 0.D0

405         L2 = L1 + INTDIS(JJ)
            NUMINT = NUMINT + 1
```

```
C             PERFORM THE INTEGRATION

              DO 420 J = 1, 2
                 DO 430 I = 1, 10

                    IF(J.EQ.2) ZAZA = -Z(I)
                    IF(J.EQ.1) ZAZA =  Z(I)
                    ZZI = (L2 - L1)*ZAZA/2.D0 + (L2 + L1)/2.D0

                    CALL INTEGH1(ZZI,ZINTH1)

                    S = S + WT(I)*H6(ZZI)*ZINTH1

430              CONTINUE
420           CONTINUE

              DINTG4 = (L2 - L1)*S/2.D0
              S = 0.D0
              SUM = SUM + DINTG4
              IF(IPRINT1.NE.1) GOTO 412
              WRITE(NOUT,920) L2, DINTG4
412           CONTINUE

              IF((DABS(DINTG4).LT.EPS1).AND.(NUMINT.GT.8))
     1             GOTO 480
              L1 = L2
              GOTO 405

480         CONTINUE
            ZINT4(JJ) = SUM
            IF(IPRINT1.NE.1) GOTO 488
            WRITE(NOUT,930) ZINT4(JJ)

488         IF(JJ.EQ.1)THEN
               IF(IFLOW.EQ.2) GOTO 490
               GOTO 400
            ELSE
               IF(DABS(ZINT4(JJ) - ZINT4(JJ-1)).LT.EPS2) GOTO 490
            ENDIF
400      CONTINUE
         WRITE(*,*) 'FAILED FOR EPS2 WHILE INTEGR. ZINT4'
         WRITE(NOUT,*) 'FAILED FOR EPS2 WHILE INTEGR. ZINT4'
490      TOTAL = TOTAL - ZINT4(JJ)
491      CONTINUE

C***********************************************************************
C-------INTEGRATION OF ZINT5
         WRITE(*,*) 'INTEGRATING ZINT5'
         WRITE(NOUT,*) 'INTEGRATING ZINT5'
         WRITE(NOUT,*)

         DO 500 JJ = 1, NINTDS
            L1 = XSTART
            NUMINT = 0
            SUM = 0.D0

505           L2 = L1 + INTDIS(JJ)
              NUMINT = NUMINT + 1

C             PERFORM THE INTEGRATION
```

```
          DO 520 J = 1, 2
             DO 530 I = 1, 10

                IF(J.EQ.2) ZAZA = -Z(I)
                IF(J.EQ.1) ZAZA =  Z(I)
                ZZI = (L2 - L1)*ZAZA/2.D0 + (L2 + L1)/2.D0

                CALL INTEGH1(ZZI,ZINTH1)

                S = S + WT(I)*H7(ZZI)*ZINTH1

530          CONTINUE
520       CONTINUE

          DINTG5 = (L2 - L1)*S/2.D0
          S = 0.D0
          SUM = SUM + DINTG5
          IF(IPRINT1.NE.1) GOTO 512
          WRITE(NOUT,920) L2, DINTG5
512       CONTINUE

          IF((DABS(DINTG5).LT.EPS1).AND.(NUMINT.GT.8))
     1       GOTO 580
          L1 = L2
          GOTO 505

580     CONTINUE
        ZINT5(JJ) = SUM
        IF(IPRINT1.NE.1) GOTO 588
        WRITE(NOUT,930) ZINT5(JJ)


588     IF(JJ.EQ.1)THEN
           IF(IFLOW.EQ.2) GOTO 590
           GOTO 500
        ELSE
           IF(DABS(ZINT5(JJ) - ZINT5(JJ-1)).LT.EPS2) GOTO 590
        ENDIF
500   CONTINUE
      WRITE(*,*) 'FAILED FOR EPS2 WHILE INTEGR. ZINT5'
      WRITE(NOUT,*) 'FAILED FOR EPS2 WHILE INTEGR. ZINT5'
590   TOTAL = TOTAL - ZINT5(JJ)

      CONCEN = 0.5D0*DEXP(-R/2.D0)*TOTAL
      WRITE(*,*) 'R = ',R,'  TIME = ',TIME
      WRITE(*,*) 'CONCEN = ',CONCEN

      RETURN

906   FORMAT(/,8X,'INTEG. LIMIT',14X,'DINTG1',18X,'ZINT1',/)
920   FORMAT(6X,1PE13.6,11X,1PE13.6)
930   FORMAT(50X,1PE13.6)

      END
```

```
      SUBROUTINE INTEGH1(V,ZINTH1)
C***********************************************************************
C
C       THIS SUBROUTINE INTEGRATES H1 FROM RW TO R1 FROM ROOT TO ROOT
C
C***********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 LL

      COMMON /VALUES/ R, RW, R1, R1A, SLOPE, TIME

      SUM = 0.D0
      ROOTP = RW + 0.0001D0

      CALL FROOT1(V,ROOTP,ROOT,IEND)
      IF(IEND.EQ.1) GOTO 600
      UU = GAUSS1(V,RW,ROOT)
      SUM = SUM + UU

COMMENT: NOW INTEGRATE FROM ROOT TO ROOT UNTIL R1 IS REACHED.

100   LL = ROOT
         ROOTP = ROOT + 0.0001D0
         CALL FROOT1(V,ROOTP,ROOT,IEND)
         IF(IEND.EQ.1) GOTO 500
         UU = GAUSS1(V,LL,ROOT)
         SUM = SUM + UU
      GOTO 100

500   UULAST = GAUSS1(V,LL,R1)
      ZINTH1 = SUM + UULAST

      RETURN

600   R1D2 = R1/2.D0
      ZINTH1 = GAUSS1(V,RW,R1D2) + GAUSS1(V,R1D2,R1)

      RETURN

      END
```

```
      SUBROUTINE INTEGH2(V,ZINTH2)
C***************************************************************************
C
C     THIS SUBROUTINE INTEGRATES H2 FROM RW TO R1 FROM ROOT TO ROOT
C
C***************************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 LL

      COMMON /VALUES/ R, RW, R1, R1A, SLOPE, TIME

      SUM = 0.D0
      ROOTP = RW + 0.0001D0

      CALL FROOT2(V,ROOTP,ROOT,IEND)
      IF(IEND.EQ.1) GOTO 600
      UU = GAUSS2(V,RW,ROOT)
      SUM = SUM + UU

COMMENT: NOW INTEGRATE FROM ROOT TO ROOT UNTIL R1 IS REACHED.

100   LL = ROOT
         ROOTP = ROOT + 0.0001D0
         CALL FROOT2(V,ROOTP,ROOT,IEND)
         IF(IEND.EQ.1) GOTO 500
         UU = GAUSS2(V,LL,ROOT)
         SUM = SUM + UU
      GOTO 100

500   UULAST = GAUSS2(V,LL,R1)
      ZINTH2 = SUM + UULAST

      RETURN

600   R1D2 = R1/2.D0
      ZINTH2 = GAUSS2(V,RW,R1D2) + GAUSS2(V,R1D2,R1)

      RETURN

      END
```

```
      DOUBLE PRECISION FUNCTION H1(V,CSI)
C***********************************************************************
C
C     FUNCTION H1 EVALUATES H1 = F(CSI)*AI(SI)
C
C     WHERE
C
C             F(CSI) = CSI*EXP(CSI/2)*(f(CSI))
C
C     WHERE f(CSI) IS THE INITIAL CONDITION
C
C***********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /VALUES/ R, RW, R1, R1A, SLOPE, TIME
      COMMON /ZINITL/ ISLOPE, COEF1
      COMMON /SCALE/  TOBIG1
      PARAMETER (TWOTH = 2.D0/3.D0, THRHF = 3.D0/2.D0)
      PARAMETER (ONEFR = 1.D0/4.D0)

      DEN = 4.D0*V**(4.D0/3.D0)
      SI = (1.D0 - 4.D0*CSI*V**2)/DEN

C     SELECT THE INITIAL CONDITION

      IF(ISLOPE.EQ.2) THEN
         F = CSI*DEXP(CSI/2.D0)*DEXP(-COEF1*(CSI-RW)**2)
      ELSEIF(ISLOPE.EQ.3) THEN
         IF(CSI.LE.R1A) THEN
            F = CSI*DEXP(CSI/2.D0)
         ELSE
            F = CSI*DEXP(CSI/2.D0)*DEXP(-COEF1*(CSI-R1A)**2)
         ENDIF
      ELSE
         F = CSI*DEXP(CSI/2.D0)*(1.D0 - SLOPE*(CSI-RW))
      ENDIF

      IF(SI.GT.TOBIG1) GOTO 100
      H1 = F*AI(SI,1)
      RETURN

100   H1AA = F*DEXP(-TWOTH*SI**THRHF)/SI**ONEFR
      H1 = H1AA*AI(SI,2)
      RETURN

      END
```

```
      DOUBLE PRECISION FUNCTION H2(V,CSI)
C***********************************************************************
C
C      FUNCTION H2 EVALUATES H2 = F(CSI)*BI(SI)
C
C      WHERE
C
C              F(CSI) = CSI*EXP(CSI/2)*(f(csi))
C
C      WHERE   f(csi) is the initial condition
C
C***********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /VALUES/ R, RW, R1, R1A, SLOPE, TIME
      COMMON /ZINITL/ ISLOPE, COEF1
      COMMON /SCALE/  TOBIG1
      PARAMETER (TWOTH = 2.D0/3.D0, THRHF = 3.D0/2.D0)
      PARAMETER (ONEFR = 1.D0/4.D0)

      DEN = 4.D0*V**(4.D0/3.D0)
      SI = (1.D0 - 4.D0*CSI*V**2)/DEN

C      SELECT THE INITIAL CONDITION

      IF(ISLOPE.EQ.2) THEN
         F = CSI*DEXP(CSI/2.D0)*DEXP(-COEF1*(CSI-RW)**2)
      ELSEIF(ISLOPE.EQ.3) THEN
         IF(CSI.LE.R1A) THEN
            F = CSI*DEXP(CSI/2.D0)
         ELSE
            F = CSI*DEXP(CSI/2.D0)*DEXP(-COEF1*(CSI-R1A)**2)
         ENDIF
      ELSE
         F = CSI*DEXP(CSI/2.D0)*(1.D0 - SLOPE*(CSI-RW))
      ENDIF

      IF(SI.GT.TOBIG1) GOTO 100
      H2 = F*BI(SI,1)
      RETURN

100   H2AA = F*DEXP(TWOTH*SI**THRHF)/SI**ONEFR
      H2 = H2AA*BI(SI,2)
      RETURN

      END
```

```
      DOUBLE PRECISION FUNCTION H3(V)
C***********************************************************************
C
C     FUNCTION H3 EVALUATES H3 =
C
C     (V**ONETH)*DEXP(-V*V*TIME)*3*AI(PHI)
C
C
C***********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /VALUES/ R, RW, R1, R1A, SLOPE, TIME

      PARAMETER (ONETH = 1.D0/3.D0)

      DEN = 4.D0*V**(4.D0/3.D0)
      PHI = (1.D0 - 4.D0*R*V**2)/DEN


      T1 = (V**ONETH)*DEXP(-V*V*TIME)
      H3 = 3.D0*T1*AI(PHI,1)

      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION H4A(V)
C**********************************************************************
C
C     FUNCTION H4A EVALUATES H4A =
C
C   (V**ONETH)*DEXP(-V*V*TIME)*(1 + (3*J*J - K*K)/(J*J + K*K))BI(PHI)
C
C
C     J = F1 = V**2/3*AIP(PHIW) + 0.5*AI(PHIW)
C
C     K = F2 = V**2/3*BIP(PHIW) + 0.5*BI(PHIW)
C
C     THESE VARIABLES ARE USED IN FUNCTIONS H5 THROUGH H7 ALSO
C**********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /VALUES/ R, RW, R1, R1A, SLOPE, TIME
      COMMON /SCALE / TOBIG1

      PARAMETER (ONETH = 1.D0/3.D0, TWOTH = 2.D0/3.D0)
      PARAMETER (THRHF = 3.D0/2.D0, ONEFR = 1.D0/4.D0)

      DEN = 4.D0*V**(4.D0/3.D0)
      PHI  = (1.D0 - 4.D0*R*V**2)/DEN
      PHIW = (1.D0 - 4.D0*RW*V**2)/DEN

      T1 = (V**ONETH)*DEXP(-V*V*TIME)
      IF(PHIW.GT.10.D0) GOTO 100

      F1 = (V**TWOTH)*AIP(PHIW,1) + 0.5D0*AI(PHIW,1)

      F2 = (V**TWOTH)*BIP(PHIW,1) + 0.5D0*BI(PHIW,1)

      T2 = (3.D0*F1*F1 - F2*F2)/(F1*F1 + F2*F2)

      H4A = T1*(T2 + 1.D0)*BI(PHI,1)
      RETURN

100   H4A = 0.D0
      RETURN

      END
```

```
      DOUBLE PRECISION FUNCTION H5(V)
C**********************************************************************
C
C     H5 =
C
C     (V**ONETH)*EXP(-V*V*TIME)*(4*J*K/(J*J + K*K))*AI(PHI)
C
C**********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      COMMON /VALUES/ R, RW, R1, R1A, SLOPE, TIME
      COMMON /SCALE / TOBIG1

      PARAMETER(TWOTH=2.D0/3.D0,ONETH=1.D0/3.D0,ONEFR=1.D0/4.D0)
      PARAMETER (FOURTH = 4.D0/3.D0, THRHF = 3.D0/2.D0)

      DEN = 4.D0*V**(4.D0/3.D0)
      PHI = (1.D0 - 4.D0*R*V*V)/DEN
      PHIW = (1.D0 - 4.D0*RW*V*V)/DEN

      T1 = (V**ONETH)*DEXP(-V*V*TIME)
      IF(PHIW.GT.TOBIG1) GOTO 100

      F1 = (V**TWOTH)*AIP(PHIW,1) + 0.5D0*AI(PHIW,1)

      F2 = (V**TWOTH)*BIP(PHIW,1) + 0.5D0*BI(PHIW,1)

      T2 = 4.D0*F1*F2/(F1*F1 + F2*F2)

      H5 = T1*T2*AI(PHI,1)
      RETURN

100   ARG1 = -FOURTH*PHIW**THRHF
      S1 = (V**TWOTH)*AIP(PHIW,2)*DSQRT(PHIW) + 0.5D0*AI(PHIW,2)

      S2 = (V**TWOTH)*BIP(PHIW,2)*DSQRT(PHIW) + 0.5D0*BI(PHIW,2)

      S3 = 4.D0*S1*DEXP(ARG1)/S2

      IF(PHI.GT.TOBIG1) GOTO 200
      H5 = T1*S3*AI(PHI,1)
      RETURN

200   H5 = (T1*S3*DEXP(-TWOTH*PHI**THRHF)/PHI**ONEFR)*AI(PHI,2)
      RETURN

      END
```

```
      DOUBLE PRECISION FUNCTION H6(V)
C*********************************************************************
C
C       H6 =
C
C       (V**ONETH)*EXP(-V*V*TIME)*(4*J*K/(J*J + K*K))*BI(PHI)
C
C*********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      COMMON /VALUES/ R, RW, R1, R1A, SLOPE, TIME
      COMMON /SCALE / TOBIG1

      PARAMETER(TWOTH=2.D0/3.D0,ONETH=1.D0/3.D0,ONEFR=1.D0/4.D0)
      PARAMETER(FOURTH = 4.D0/3.D0, THRHF = 3.D0/2.D0)

      DEN = 4.D0*V**(4.D0/3.D0)
      PHI = (1.D0 - 4.D0*R*V*V)/DEN
      PHIW = (1.D0 - 4.D0*RW*V*V)/DEN

      T1 = (V**ONETH)*DEXP(-V*V*TIME)
      IF(PHIW.GT.TOBIG1) GOTO 100

      F1 = (V**TWOTH)*AIP(PHIW,1) + 0.5D0*AI(PHIW,1)

      F2 = (V**TWOTH)*BIP(PHIW,1) + 0.5D0*BI(PHIW,1)

      T2 = 4.D0*F1*F2/(F1*F1 + F2*F2)

      H6 = T1*T2*BI(PHI,1)
      RETURN

100   ARG1 = -FOURTH*PHIW**THRHF
      S1 = (V**TWOTH)*AIP(PHIW,2)*DSQRT(PHIW) + 0.5D0*AI(PHIW,2)

      S2 = (V**TWOTH)*BIP(PHIW,2)*DSQRT(PHIW) + 0.5D0*BI(PHIW,2)

      S3 = 4.D0*S1*DEXP(ARG1)/S2

      IF(PHI.GT.TOBIG1) GOTO 200
      H6 = T1*S3*BI(PHI,1)
      RETURN

200   H6 = (T1*S3*DEXP(TWOTH*PHI**THRHF)/PHI**ONEFR)*BI(PHI,2)
      RETURN

      END
```

```
      DOUBLE PRECISION FUNCTION H7(V)
C***********************************************************************
C
C     H7 =
C
C     (V**ONETH)*EXP(-V*V*TIME)*((3*J*J - K*K)/(J*J + K*K))*AI(PHI)
C
C***********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      COMMON /VALUES/ R, RW, R1, R1A, SLOPE, TIME
      COMMON /SCALE / TOBIG1

      PARAMETER(TWOTH=2.D0/3.D0,ONETH=1.D0/3.D0,ONEFR=1.D0/4.D0)
      PARAMETER(FOURTH = 4.D0/3.D0, THRHF = 3.D0/2.D0)

      DEN = 4.D0*V**(4.D0/3.D0)
      PHI = (1.D0 - 4.D0*R*V*V)/DEN
      PHIW = (1.D0 - 4.D0*RW*V*V)/DEN

      T1 = (V**ONETH)*DEXP(-V*V*TIME)
      IF(PHIW.GT.TOBIG1) GOTO 100

      F1 = (V**TWOTH)*AIP(PHIW,1) + 0.5D0*AI(PHIW,1)

      F2 = (V**TWOTH)*BIP(PHIW,1) + 0.5D0*BI(PHIW,1)

      T2 = (3.D0*F1*F1 - F2*F2)/(F1*F1 + F2*F2)

      H7 = T1*T2*AI(PHI,1)
      RETURN

100   IF(PHI.GT.TOBIG1) GOTO 200
      H7 = -T1*AI(PHI,1)
      RETURN

200   H7 = -(T1*DEXP(-TWOTH*PHI**THRHF)/PHI**ONEFR)*AI(PHI,2)
      RETURN

      END
```

```
      SUBROUTINE FROOT1(V,START,ROOT,IEND)
C**********************************************************************
C
C     THIS SUBROUTINE DETERMINES THE ROOTS OF THE INTEGRAND H1;
C     IT FINDS THE INTERVAL IN WHICH THE INTEGRAND CHANGES SIGN AND
C     THEN USES THE FUNCTION 'ZERO' TO DETERMINE THE ROOT.
C
C**********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /VALUES/ R, RW, R1, R1A, SLOPE, TIME

      IEND = 0
      DCSI = 0.01D0
      IF(V.LT.1.5D0)  DCSI = 0.05D0
      IF(V.LT.0.75D0) DCSI = 0.1D0
      IF(V.LT.0.3D0)  DCSI = 0.2D0

      CSIL = START
      FL = H1(V,CSIL)

600   CSIR = CSIL + DCSI
      IF(CSIR.GT.R1) GOTO 900

      FR = H1(V,CSIR)

      SIGN = FL*FR
      IF(SIGN.LT.0.D0) GOTO 800
      CSIL = CSIR
      FL = FR
      GOTO 600

800   ROOT = ZERO1(V,CSIL,CSIR)

      RETURN

900   IEND = 1
      ROOT = R1

      RETURN

      END
```

```
      DOUBLE PRECISION FUNCTION ZERO1(V,A,B)
C***********************************************************************
C
C      FUNCTION SUBROUTINE ZERO.      THIS IS A FORTRAN TRANSLATION
C      OF THE ALGORITHM OF BRENT FOR FINDING THE ZERO OF A FUNCTION
C      WHICH CHANGES SIGN IN A GIVEN INTERVAL. REFERENCE IS
C      'ALGORITHMS FOR MINIMIZATION WITHOUT DERIVATIVES' BY RICHARD
C      P. BRENT, 1973, PRENTICE-HALL.
C      THE ALGORITHM USED HERE IS ON PAGE 188 OF THIS BOOK (IN THE APPENDIX)
C      IT IS WRITTEN SEPARATELY IN THE 'CHIA' SUBDIRECTORY AS 'BRENT.FOR'.
C
C      NOTE: TO CHANGE TO ANOTHER FUNCTION, THERE ARE 3 FCT EVALUATIONS IN
C      THIS ROUTINE (2 AT THE START, 1 AT THE END)
C
C
C*****************************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      REAL*8 MACHEP, M

      DATA MACHEP, T/1.D-17, 1.D-17/
      SA = A
      SB = B
      FA = H1(V,SA)
      FB = H1(V,SB)

10    C = SA
      FC = FA
      E = SB - SA
      D = E

20    IF(DABS(FC).GE.DABS(FB)) GOTO 30
      SA = SB
      SB = C
      C = SA
      FA = FB
      FB = FC
      FC = FA

30    TOL = 2.D0*MACHEP*DABS(SB) + T
      M = 0.5D0*(C - SB)
      IF((DABS(M).LE.TOL).OR.(FB.EQ.0.D0)) GOTO 140
      IF((DABS(E).GE.TOL).AND.(DABS(FA).GT.DABS(FB))) GOTO 40
      E = M
      D = E
      GOTO 100

40    S = FB/FA
      IF(SA.NE.C) GOTO 50
      P = 2.D0*M*S
      Q = 1.D0 - S
      GOTO 60

50    Q = FA/FC
      R = FB/FC
      P = S*(2.D0*M*Q*(Q - R) - (SB - SA)*(R - 1.D0))
      Q = (Q - 1.D0)*(R - 1.D0)*(S - 1.D0)
```

```
60      IF(P.LE.0.D0) GOTO 70
        Q = -Q
        GOTO 80

70      P = -P

80      S = E
        E = D
        IF((2.D0*P.GE.3.D0*M*Q-DABS(TOL*Q)).OR.(P.GE.DABS(0.5D0*S*Q))) GOTO 90
        D = P/Q
        GOTO 100

90      E = M
        D = E

100     SA = SB
        FA = FB
        IF(DABS(D).LE.TOL) GOTO 110
        SB = SB + D
        GOTO 130

110     IF(M.LE.0.D0) GOTO 120
        SB = SB + TOL
        GOTO 130

120     SB = SB - TOL

130     FB = H1(V,SB)
        IF((FB.GT.0.D0).AND.(FC.GT.0.D0)) GOTO 10
        IF((FB.LE.0.D0).AND.(FC.LE.0.D0)) GOTO 10
        GOTO 20

140     ZERO1 = SB

        RETURN
        END
```

```
      DOUBLE PRECISION FUNCTION GAUSS1(V,XA,XB)
C*********************************************************************
C     COMPUTES THE INTEGRAL OF F(X)DX FROM X=XA TO X=XB USING A
C     20-POINT GAUSSIAN QUADRATURE METHOD
C*********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION Z(10),WT(10)

      DATA Z /0.076526521133497D0,0.227785851141650D0,
     1        0.373706088715420D0 ,0.510867001950830D0,
     2        0.636053680726520D0 ,0.746331906460150D0,
     3        0.839116971822220D0 ,0.912234428251330D0,
     4        0.963971927277910D0 ,0.993128599185090D0/
      DATA WT/0.152753387130730D0 ,0.149172986472600D0,
     1        0.142096109318380D0 ,0.131688638449180D0,
     2        0.118194531961520D0 ,0.101930119817240D0,
     3        0.083276741576705D0,0.062672048334109D0,
     4        0.040601429800387D0,0.017614007139152D0/

      NN = 10
      C1 = (XB-XA)/2.0D0
      C2 = (XB+XA)/2.0D0
      SUM = 0.0D0
      DO 1 I=1, NN
              ZZZ1 = Z(I)*C1 + C2
              ZZZ2 = -Z(I)*C1 + C2
1     SUM = SUM + WT(I)*(H1(V,ZZZ1) + H1(V,ZZZ2))

      GAUSS1 = C1*SUM
      RETURN
      END
C
```

```
      SUBROUTINE FROOT2(V,START,ROOT,IEND)
C***********************************************************************
C
C     THIS SUBROUTINE DETERMINES THE ROOTS OF THE INTEGRAND H2;
C     IT FINDS THE INTERVAL IN WHICH THE INTEGRAND CHANGES SIGN AND
C     THEN USES THE FUNCTION 'ZERO' TO DETERMINE THE ROOT.
C
C***********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /VALUES/ R, RW, R1, R1A, SLOPE, TIME

      IEND = 0
      DCSI = 0.01D0
      IF(V.LT.1.5D0)  DCSI = 0.05D0
      IF(V.LT.0.75D0) DCSI = 0.1D0
      IF(V.LT.0.3D0)  DCSI = 0.2D0

      CSIL = START
      FL = H2(V,CSIL)

600   CSIR = CSIL + DCSI
      IF(CSIR.GT.R1) GOTO 900
      FR = H2(V,CSIR)

      SIGN = FL*FR
      IF(SIGN.LT.0.D0) GOTO 800
      CSIL = CSIR
      FL = FR
      GOTO 600

800   ROOT = ZERO2(V,CSIL,CSIR)

      RETURN

900   IEND = 1
      ROOT = R1

      RETURN

      END
```

```
      DOUBLE PRECISION FUNCTION ZERO2(V,A,B)
C*********************************************************************
C
C     FUNCTION SUBROUTINE ZERO.      THIS IS A FORTRAN TRANSLATION
C     OF THE ALGORITHM OF BRENT FOR FINDING THE ZERO OF A FUNCTION
C     WHICH CHANGES SIGN IN A GIVEN INTERVAL. REFERENCE IS
C     'ALGORITHMS FOR MINIMIZATION WITHOUT DERIVATIVES' BY RICHARD
C     P. BRENT, 1973, PRENTICE-HALL.
C     THE ALGORITHM USED HERE IS ON PAGE 188 OF THIS BOOK (IN THE APPENDIX)
C     IT IS WRITTEN SEPARATELY IN THE 'CHIA' SUBDIRECTORY AS 'BRENT.FOR'.
C
C     NOTE: TO CHANGE TO ANOTHER FUNCTION, THERE ARE 3 FCT EVALUATIONS IN
C     THIS ROUTINE (2 AT THE START, 1 AT THE END)
C
C
C****************************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      REAL*8 MACHEP, M

      DATA MACHEP, T/1.D-17, 1.D-17/
      SA = A
      SB = B
      FA = H2(V,SA)
      FB = H2(V,SB)

10    C = SA
      FC = FA
      E = SB - SA
      D = E

20    IF(DABS(FC).GE.DABS(FB)) GOTO 30
      SA = SB
      SB = C
      C = SA
      FA = FB
      FB = FC
      FC = FA

30    TOL = 2.D0*MACHEP*DABS(SB) + T
      M = 0.5D0*(C - SB)
      IF((DABS(M).LE.TOL).OR.(FB.EQ.0.D0)) GOTO 140
      IF((DABS(E).GE.TOL).AND.(DABS(FA).GT.DABS(FB))) GOTO 40
      E = M
      D = E
      GOTO 100

40    S = FB/FA
      IF(SA.NE.C) GOTO 50
      P = 2.D0*M*S
      Q = 1.D0 - S
      GOTO 60

50    Q = FA/FC
      R = FB/FC
      P = S*(2.D0*M*Q*(Q - R) - (SB - SA)*(R - 1.D0))
      Q = (Q - 1.D0)*(R - 1.D0)*(S - 1.D0)
```

```
60      IF(P.LE.0.D0) GOTO 70
        Q = -Q
        GOTO 80

70      P = -P

80      S = E
        E = D
        IF((2.D0*P.GE.3.D0*M*Q-DABS(TOL*Q)).OR.(P.GE.DABS(0.5D0*S*Q))) GOTO 90
        D = P/Q
        GOTO 100

90      E = M
        D = E

100     SA = SB
        FA = FB
        IF(DABS(D).LE.TOL) GOTO 110
        SB = SB + D
        GOTO 130

110     IF(M.LE.0.D0) GOTO 120
        SB = SB + TOL
        GOTO 130

120     SB = SB - TOL

130     FB = H2(V,SB)
        IF((FB.GT.0.D0).AND.(FC.GT.0.D0)) GOTO 10
        IF((FB.LE.0.D0).AND.(FC.LE.0.D0)) GOTO 10
        GOTO 20

140     ZERO2 = SB

        RETURN
        END
```

```
         DOUBLE PRECISION FUNCTION GAUSS2(V,XA,XB)
C**********************************************************************
C       COMPUTES THE INTEGRAL OF F(X)DX FROM X=XA TO X=XB USING A
C       20-POINT GAUSSIAN QUADRATURE METHOD
C**********************************************************************
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        DIMENSION Z(10),WT(10)

        DATA Z /0.076526521133497D0,0.227785851141645D0,
       1        0.373706088715420D0 ,0.510867001950827D0,
       2        0.636053680726515D0 ,0.746331906460151D0,
       3        0.839116971822219D0 ,0.912234428251326D0,
       4        0.963971927277914D0 ,0.993128599185095D0/
        DATA WT/0.152753387130726D0 ,0.149172986472604D0,
       1        0.142096109318382D0 ,0.131688638449177D0,
       2        0.118194531961518D0 ,0.101930119817240D0,
       3        0.0832767415767048D0,0.0626720483341091D0,
       4        0.0406014298003869D0,0.0176140071391521D0/

        NN = 10
        C1 = (XB-XA)/2.0D0
        C2 = (XB+XA)/2.0D0
        SUM = 0.0D0
        DO 1 I=1, NN
                ZZZ1 = Z(I)*C1 + C2
                ZZZ2 = -Z(I)*C1 + C2
1       SUM = SUM + WT(I)*(H2(V,ZZZ1) + H2(V,ZZZ2))

        GAUSS2 = C1*SUM

        RETURN
        END
C
```

```
      DOUBLE PRECISION FUNCTION AI(Z,IOPT)
C************************************************************************
C     THIS FUNCTION SUBROUTINE COMPUTES THE AIRY FUNCTION AI(Z).
C
C     FOR POSITIVE ARGEMENTS, A SCALING OPTION IS AVAILABLE.
C     IF IOPT=1, THE RESULT IS NOT SCALED.
C     IF IOPT=2, THE RESULT IS THE FUNCTION VALUE MULTIPLIED BY
C        (Z**0.25)*EXP(U), WHERE U=(2./3.)*(Z**1.5)
C
C     FOR NON-POSITIVE ARGUMENTS, NO SCALING OPTION IS AVAILABLE.  IF
C     IOPT IS SET TO 2, IT IS IGNORED, AND A WARNING IS PRINTED.
C************************************************************************

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DATA C1,C2,PI/.35502 80538 878D0,.25881 94037 928D0,
     1   3.14159 26535 90D0/
      DATA COEF1,COEF2,COEF3,COEF4,COEF5,COEF6/9.555526226877D-29,
     1   4.235605597020D-32,1.661021802753D-35,1.013578212294D-29,
     2   4.309431174718D-33,1.624974047782D-36/
      PID4=PI/4.D0
      PIRT2=DSQRT(PI)*2.D0
      TWTHRD=2.D0/3.D0
      IF (Z.GT.4.8D0) GOTO 100
      IF (Z.LT.-5.D0) GOTO 200
C-------COMPUTE AI(Z) FOR -5.0 < Z < 4.8
      P=Z**3
      F=1.D0+P*( 1.666666666667D-01+P*( 5.555555555556D-03+
     1       P*( 7.716049382716D-05+P*( 5.845491956603D-07+
     2       P*( 2.783567598382D-09+P*( 9.096626138505D-12+
     3       P*( 2.165863366311D-14+P*( 3.923665518679D-17+
     4       P*( 5.589267120625D-20+P*( 6.424444966235D-23+
     5       P*( 6.083754702874D-26+P*( 4.828376748313D-29+
     6       P*( 3.258014000211D-32+P*( 1.891994192922D-35+
     7       P*(       1.0D-10*COEF1+P*(       1.0D-10*COEF2+
     8       P*(       1.0D-10* COEF3))))))))))))))))
      G=Z*(1.D0+P*( 8.333333333333D-02+P*( 1.984126984127D-03+
     1           P*( 2.204585537919D-05+P*( 1.413195857640D-07+
     2           P*( 5.888316073501D-10+P*( 1.721729846053D-12+
     3           P*( 3.726687978470D-15+P*( 6.211146630783D-18+
     4           P*( 8.215802421670D-21+P*( 8.834196152333D-24+
     5           P*( 7.873615109032D-27+P*( 5.911122454228D-30+
     6           P*( 3.789181060403D-33+P*( 2.098106899448D-36+
     7           P*(       1.0D-10*COEF4+P*(       1.0D-10*COEF5+
     8           P*(       1.0D-10*COEF6)))))))))))))))))
      AI=C1*F-C2*G
      IF (IOPT.EQ.2) GOTO 20
      RETURN
20    IF (Z.GT.0.D0) GOTO 30
      WRITE (30,900)
      RETURN
30    U=TWTHRD*Z**1.5D0
      AI=(Z**0.25D0)*DEXP(U)*AI
      RETURN
C-------COMPUTE AI(Z) FOR Z > 4.8
100   U=TWTHRD*Z**1.5D0
      P=1.D0/U
      A=1.D0+P*(-6.944444444444D-02+P*( 3.713348765432D-02+
     1       P*(-3.799305912780D-02+P*( 5.764919041267D-02+
```

```
     2        P*(-1.160990640255D-01+P*( 2.915913992307D-01+
     3        P*(-8.776669695100D-01+P*( 3.079453030173D+00+
     4        P*(-1.234157333235D+01+P*( 5.562278536591D+01+
     5        P*(-2.784650807776D+02+P*( 1.533169432013D+03+
     6        P*(-9.207206599726D+03))))))))))))
      IF (IOPT.EQ.2) GOTO 130
      AI=A*DEXP(-U)/PIRT2/(Z**0.25D0)
      RETURN
130   AI=A/PIRT2
      RETURN
C-------COMPUTE AI(Z) FOR Z < -5.0
200   ZN=-Z
      UN=TWTHRD*ZN**1.5D0
      W=UN+PID4
      P=1.D0/(UN**2)
      A=1.D0+P*(-3.713348765432D-02+P*( 5.764919041267D-02+
     1        P*(-2.915913992307D-01+P*( 3.079453030173D+00+
     2        P*(-5.562278536591D+01+P*( 1.533169432013D+03+
     3        P*(-5.989251356587D+04+P*( 3.148257417867D+06))))))))
      B=(1.D0/UN)*( 6.944444444444D-02+P*(-3.799305912780D-02+
     1          P*( 1.160990640255D-01+P*(-8.776669695100D-01+
     2          P*( 1.234157333235D+01+P*(-2.784650807776D+02+
     3          P*( 9.207206599726D+03+P*(-4.195248751165D+05))))))))
      AI=(DSIN(W)*A-DCOS(W)*B)/DSQRT(PI*DSQRT(ZN))
      IF (IOPT.EQ.2) WRITE (30,900)
      RETURN
900   FORMAT ('***WARNING*** IOPT=2 IS IGNORED FOR A NON-POSITIVE ',
     1        'ARGUMENT OF AI(Z).')
      END

C
C
C


      DOUBLE PRECISION FUNCTION BI(Z,IOPT)
C***********************************************************************
C     THIS FUNCTION SUBROUTINE COMPUTES THE AIRY FUNCTION BI(Z).
C
C     FOR POSITIVE ARGEMENTS, A SCALING OPTION IS AVAILABLE.
C     IF IOPT=1, THE RESULT IS NOT SCALED.
C     IF IOPT=2, THE RESULT IS THE FUNCTION VALUE MULTIPLIED BY
C        (Z**0.25)*EXP(-U), WHERE U=(2./3.)*(Z**1.5)
C
C     FOR NON-POSITIVE ARGEMENTS, NO SCALING OPTION IS AVAILABLE.  IF
C     IOPT IS SET TO 2, IT IS IGNORED, BUT A WARNING IS PRINTED.
C***********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DATA D1,D2,PI/.61492 66274 460D0,.44828 83573 538D0,
     1   3.14159 26535 90D0/
      DATA COEF1,COEF2,COEF3,COEF4,COEF5,COEF6/9.555526226877D-29,
     1   4.235605597020D-32,1.661021802753D-35,1.013578212294D-29,
     2   4.309431174718D-33,1.624974047782D-36/
      PID4=PI/4.D0
      PIRT=DSQRT(PI)
      TWTHRD=2.D0/3.D0
      IF (Z.GT.4.8D0) GOTO 100
      IF (Z.LT.-5.D0) GOTO 200
C-------COMPUTE BI(Z) FOR -5.0 < Z < 4.8
      P=Z**3
```

```
      F=1.D0+P*( 1.666666666667D-01+P*( 5.555555555556D-03+
     1         P*( 7.716049382716D-05+P*( 5.845491956603D-07+
     2         P*( 2.783567598382D-09+P*( 9.096626138505D-12+
     3         P*( 2.165863366311D-14+P*( 3.923665518679D-17+
     4         P*( 5.589267120625D-20+P*( 6.424444966235D-23+
     5         P*( 6.083754702874D-26+P*( 4.828376748313D-29+
     6         P*( 3.258014000211D-32+P*( 1.891994192922D-35+
     7         P*(       1.0D-10*COEF1+P*(       1.0D-10*COEF2+
     8         P*(       1.0D-10*COEF3)))))))))))))))
      G=Z*(1.D0+P*( 8.333333333333D-02+P*( 1.984126984127D-03+
     1            P*( 2.204585537919D-05+P*( 1.413195857640D-07+
     2            P*( 5.888316073501D-10+P*( 1.721729846053D-12+
     3            P*( 3.726687978470D-15+P*( 6.211146630783D-18+
     4            P*( 8.215802421670D-21+P*( 8.834196152333D-24+
     5            P*( 7.873615109032D-27+P*( 5.911122454228D-30+
     6            P*( 3.789181060403D-33+P*( 2.098106899448D-36+
     7            P*(       1.0D-10*COEF4+P*(       1.0D-10*COEF5+
     8            P*(       1.0D-10*COEF6))))))))))))))))
      BI=D1*F+D2*G
      IF (IOPT.EQ.2) GOTO 20
      RETURN
20    IF (Z.GT.0.D0) GOTO 30
      WRITE (30,900)
      RETURN
30    U=TWTHRD*Z**1.5D0
      BI=(Z**0.25D0)*DEXP(-U)*BI
      RETURN
C-------COMPUTE BI(Z) FOR Z > 4.8
100   U=TWTHRD*Z**1.5D0
      P=1.D0/U
      A=1.D0+P*( 6.944444444444D-02+P*( 3.713348765432D-02+
     1         P*( 3.799305912780D-02+P*( 5.764919041267D-02+
     2         P*( 1.160990640255D-01+P*( 2.915913992307D-01+
     3         P*( 8.776669695100D-01+P*( 3.079453030173D+00+
     4         P*( 1.234157333235D+01+P*( 5.562278536591D+01+
     5         P*( 2.784650807776D+02+P*( 1.533169432013D+03+
     6         P*( 9.207206599726D+03)))))))))))
      IF (IOPT.EQ.2) GOTO 130
      BI=A*DEXP(U)/PIRT/(Z**0.25D0)
      RETURN
130   BI=A/PIRT
      RETURN
C-------COMPUTE BI(Z) FOR Z < -5.0
200   ZN=-Z
      UN=TWTHRD*ZN**1.5D0
      W=UN+PID4
      P=1.D0/(UN**2)
      A=1.D0+P*(-3.713348765432D-02+P*( 5.764919041267D-02+
     1         P*(-2.915913992307D-01+P*( 3.079453030173D+00+
     2         P*(-5.562278536591D+01+P*( 1.533169432013D+03+
     3         P*(-5.989251356587D+04+P*( 3.148257417867D+06)))))))
      B=(1.D0/UN)*( 6.944444444444D-02+P*(-3.799305912780D-02+
     1            P*( 1.160990640255D-01+P*(-8.776669695100D-01+
     2            P*( 1.234157333235D+01+P*(-2.784650807776D+02+
     3            P*( 9.207206599726D+03+P*(-4.195248751165D+05)))))))
      BI=(DCOS(W)*A+DSIN(W)*B)/DSQRT(PI*DSQRT(ZN))
      IF (IOPT.EQ.2) WRITE (30,900)
      RETURN
900   FORMAT ('***WARNING*** IOPT=2 IS IGNORED FOR A NON-POSITIVE ',
     1        'ARGUMENT OF BI(Z).')
```

```
      END
C
C
C
C
C
      DOUBLE PRECISION FUNCTION AIP(Z,IOPT)
C***********************************************************************
C     THIS FUNCTION SUBROUTINE COMPUTES AIP(Z), THE FIRST DERIVATIVE OF
C     THE AIRY FUNCTION AI(Z).
C
C     FOR POSITIVE ARGEMENTS, A SCALING OPTION IS AVAILABLE.
C     IF IOPT=1, THE RESULT IS NOT SCALED.
C     IF IOPT=2, THE RESULT IS THE FUNCTION VALUE MULTIPLIED BY
C        EXP(U)/(Z**0.25), WHERE U=(2./3.)*(Z**1.5)
C
C     FOR NON-POSITIVE ARGUMENTS, NO SCALING OPTION IS AVAILABLE.  IF
C     IOPT IS SET TO 2, IT IS IGNORED, AND A WARNING IS PRINTED.
C***********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DATA C1,C2,PI/.35502 80538 878D0,.25881 94037 928D0,
     1   3.14159 26535 90D0/
      DATA COEF1,COEF2,COEF3,COEF4,COEF5,COEF6/4.066181373139D-30,
     1   1.694242238808D-33,6.268006802841D-37,4.662459776551D-28,
     2   2.111621275612D-31,8.449865048466D-35/
      PID4=PI/4.D0
      PIRT2=DSQRT(PI)*2.D0
      TWTHRD=2.D0/3.D0
      IF (Z.GT.4.8D0) GOTO 100
      IF (Z.LT.-5.D0) GOTO 200
C-------COMPUTE AIP(Z) FOR -5.0 > Z > 4.8
      P=Z**3
      ZZD2=Z**2/2.D0
      F=ZZD2*(1.D0+P*( 6.666666666667D-02+P*( 1.388888888889D-03+
     1             P*( 1.402918069585D-05+P*( 8.350702795147D-08+
     2             P*( 3.274785409862D-10+P*( 9.096626138505D-13+
     3             P*( 1.883359448966D-15+P*( 3.018204245137D-18+
     4             P*( 3.854666979741D-21+P*( 4.015278103897D-24+
     5             P*( 3.476431258785D-27+P*( 2.541250920165D-30+
     6             P*( 1.589275122054D-33+P*( 8.599973604190D-37+
     7             P*(      1.0D-10*COEF1+P*(      1.0D-10*COEF2+
     8             P*(      1.0D-10*COEF3)))))))))))))))))
      G=1.D0+P*( 3.333333333333D-01+P*( 1.388888888889D-02+
     1       P*( 2.204585537919D-04+P*( 1.837154614932D-06+
     2       P*( 9.421305717602D-09+P*( 3.271286707501D-11+
     3       P*( 8.198713552633D-14+P*( 1.552786657696D-16+
     4       P*( 2.300424678068D-19+P*( 2.738600807224D-22+
     5       P*( 2.677029137071D-25+P*( 2.187115308064D-28+
     6       P*( 1.515672424161D-31+P*( 9.021859667626D-35+
     7       P*(      1.0D-10*COEF4+P*(      1.0D-10*COEF5+
     8       P*(      1.0D-10*COEF6)))))))))))))))))
      AIP=C1*F-C2*G
      IF (IOPT.EQ.2) GOTO 20
      RETURN
20    IF (Z.GT.0.D0) GOTO 30
      WRITE (30,900)
      RETURN
30    U=TWTHRD*Z**1.5D0
```

```
      AIP=DEXP(U)*AIP/(Z**0.25D0)
      RETURN
C-------COMPUTE AIP(Z) FOR Z > 4.8
100   U=TWTHRD*Z**1.5D0
      P=1.D0/U
      A=1.D0+P*( 9.722222222222D-02+P*(-4.388503086420D-02+
     1       P*( 4.246283078989D-02+P*(-6.266216349203D-02+
     2       P*( 1.241058960273D-01+P*(-3.082537649011D-01+
     3       P*( 9.204799924129D-01+P*(-3.210493584649D+00+
     4       P*( 1.280729308074D+01+P*(-5.750830351391D+01+
     5       P*( 2.870332371092D+02+P*(-1.576357303337D+03+
     6       P*( 9.446354823095D+03)))))))))))
      IF (IOPT.EQ.2) GOTO 130
      AIP=-A*DEXP(-U)*(Z**0.25D0)/PIRT2
      RETURN
130   AIP=-A/PIRT2
      RETURN
C-------COMPUTE AIP(Z) FOR Z < -5.0
200   ZN=-Z
      UN=TWTHRD*ZN**1.5D0
      W=UN+PID4
      P=1.D0/(UN**2)
      A=1.D0+P*( 4.388503086420D-02+P*(-6.266216349203D-02+
     1       P*( 3.082537649011D-01+P*(-3.210493584649D+00+
     2       P*( 5.750830351391D+01+P*(-1.576357303337D+03+
     3       P*( 6.133570666385D+04+P*(-3.214536521401D+06)))))))
      B=(1.D0/UN)*(-9.722222222222D-02+P*( 4.246283078989D-02+
     1          P*(-1.241058960273D-01+P*( 9.204799924129D-01+
     2          P*(-1.280729308074D+01+P*( 2.870332371092D+02+
     3          P*(-9.446354823095D+03+P*( 4.289524004000D+05)))))))
      AIP=-(DCOS(W)*A+DSIN(W)*B)*DSQRT(DSQRT(ZN)/PI)
      IF (IOPT.EQ.2) WRITE (30,900)
      RETURN
900   FORMAT ('***WARNING*** IOPT=2 IS IGNORED FOR A NON-POSITIVE ',
     1        'ARGUMENT OF AIP(Z).')
      END


C
C
C
C



      DOUBLE PRECISION FUNCTION BIP(Z,IOPT)
C**********************************************************************
C     THIS FUNCTION SUBROUTINE COMPUTES BIP(Z),THE FIRST DERIVATIVE OF
C     THE AIRY FUNCTION BI(Z).
C
C     FOR POSITIVE ARGEMENTS, A SCALING OPTION IS AVAILABLE.
C     IF IOPT=1, THE RESULT IS NOT SCALED.
C     IF IOPT=2, THE RESULT IS THE FUNCTION VALUE MULTIPLIED BY
C        EXP(-U)/(Z**0.25), WHERE U=(2./3.)*(Z**1.5)
C
C     FOR NON-POSITIVE ARGEMENTS, NO SCALING OPTION IS AVAILABLE.  IF
C     IOPT IS SET TO 2, IT IS IGNORED, BUT A WARNING IS PRINTED.
C**********************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DATA D1,D2,PI/.61492 66274 460D0,.44828 83573 538D0,
     1   3.14159 26535 90D0/
      DATA COEF1,COEF2,COEF3,COEF4,COEF5,COEF6/4.066181373139D-30,
```

```
     2   1.694242238808D-33,6.268006802841D-37,4.662459776551D-28,
     3   2.111621275612D-31,8.449865048466D-35/
      PID4=PI/4.D0
      PIRT=DSQRT(PI)
      TWTHRD=2.D0/3.D0
      IF (Z.GT.4.8D0) GOTO 100
      IF (Z.LT.-5.D0) GOTO 200
C--------COMPUTE BIP(Z) FOR -5.0 < Z < 4.8
      P=Z**3
      ZZD2=Z**2/2.D0
      F=ZZD2*(1.D0+P*( 6.666666666667D-02+P*( 1.388888888889D-03+
     1           P*( 1.402918069585D-05+P*( 8.350702795147D-08+
     2           P*( 3.274785409862D-10+P*( 9.096626138505D-13+
     3           P*( 1.883359448966D-15+P*( 3.018204245137D-18+
     4           P*( 3.854666979741D-21+P*( 4.015278103897D-24+
     5           P*( 3.476431258785D-27+P*( 2.541250920165D-30+
     6           P*( 1.589275122054D-33+P*( 8.599973604190D-37+
     7           P*(      1.0D-10*COEF1+P*(      1.0D-10*COEF2+
     8           P*(      1.0D-10*COEF3))))))))))))))))))
      G=1.D0+P*( 3.333333333333D-01+P*( 1.388888888889D-02+
     1      P*( 2.204585537919D-04+P*( 1.837154614932D-06+
     2      P*( 9.421305717602D-09+P*( 3.271286707501D-11+
     3      P*( 8.198713552633D-14+P*( 1.552786657696D-16+
     4      P*( 2.300424678068D-19+P*( 2.738600807224D-22+
     5      P*( 2.677029137071D-25+P*( 2.187115308064D-28+
     6      P*( 1.515672424161D-31+P*( 9.021859667626D-35+
     7      P*(      1.0D-10*COEF4+P*(      1.0D-10*COEF5+
     8      P*(      1.0D-10*COEF6)))))))))))))))))
      BIP=D1*F+D2*G
      IF (IOPT.EQ.2) GOTO 20
      RETURN
20    IF (Z.GT.0.D0) GOTO 30
      WRITE (30,900)
      RETURN
30    U=TWTHRD*Z**1.5D0
      BIP=BIP*DEXP(-U)/(Z**0.25D0)
      RETURN
C-------COMPUTE BIP(Z) FOR Z > 4.8
100   U=TWTHRD*Z**1.5D0
      P=1.D0/U
      A=1.D0+P*(-9.722222222222D-02+P*(-4.388503086420D-02+
     1      P*(-4.246283078989D-02+P*(-6.266216349203D-02+
     2      P*(-1.241058960273D-01+P*(-3.082537649011D-01+
     3      P*(-9.204799924129D-01+P*(-3.210493584649D+00+
     4      P*(-1.280729308074D+01+P*(-5.750830351391D+01+
     5      P*(-2.870332371092D+02+P*(-1.576357303337D+03+
     6      P*(-9.446354823095D+03)))))))))))))
      IF (IOPT.EQ.2) GOTO 130
      BIP=A*DEXP(U)*(Z**0.25D0)/PIRT
      RETURN
130   BIP=A/PIRT
      RETURN
C-------COMPUTE BIP(Z) FOR Z < -5.0
200   ZN=-Z
      UN=TWTHRD*ZN**1.5D0
      W=UN+PID4
      P=1.D0/(UN**2)
      A=1.D0+P*( 4.388503086420D-02+P*(-6.266216349203D-02+
     1      P*( 3.082537649011D-01+P*(-3.210493584649D+00+
     2      P*( 5.750830351391D+01+P*(-1.576357303337D+03+
```

```
     3       P*( 6.133570666385D+04+P*(-3.214536521401D+06)))))))
      B=(1.D0/UN)*(-9.722222222222D-02+P*( 4.246283078989D-02+
     1          P*(-1.241058960273D-01+P*( 9.204799924129D-01+
     2          P*(-1.280729308074D+01+P*( 2.870332371092D+02+
     3          P*(-9.446354823095D+03+P*( 4.289524004000D+05)))))))
      BIP=(DSIN(W)*A-DCOS(W)*B)*DSQRT(DSQRT(ZN)/PI)
      IF (IOPT.EQ.2) WRITE (30,900)
      RETURN
900   FORMAT ('***WARNING*** IOPT=2 IS IGNORED FOR A NON-POSITIVE ',
     1        'ARGUMENT OF BIP(Z).')

      END
```