2019

Generalized Radial Transport Model for Interpreting Convergent Flow Tracer Tests in Fractured Rock

Md Lal Mamud
University of Mississippi

GENERALIZED RADIAL TRANSPORT MODEL FOR INTERPRETING CONVERGENT FLOW TRACER TESTS IN FRACTURED ROCK

Thesis submitted for partial fulfillment of the requirements for the M.S. in Engineering Science (Hydrology) in the Department of Geology and Geological Engineering
The University of Mississippi
.

by

MD LAL MAMUD

May 2019

ABSTRACT

The double-porosity Generalized Radial Transport (GRT) model is an extension of the generalized radial flow approach developed for hydraulic test interpretation. In both approaches, a flow dimension characterizes the change in flow area versus radial distance from the borehole. The GRT model collapses to a 1D, radial, and spherical advection dispersion equation (ADE) for integer flow dimensions of 1, 2, and 3, respectively. And, the model also transforms to sub-linear, sub-radial, sub-spherical and eventually transform to super-spherical transport model for non-integer flow dimension, n of $0 < n < 1$, $1 < n < 2$, $2 < n < 3$ and $n > 3$ respectively. Non-integer flow dimensions, especially sub-radial, are commonly reported from pumping tests in fractured rock systems and can be linked with aquifer geometry and heterogeneity. We consider the impact of sub-radial flow dimensions on convergent flow tracer tests in fractured rock. In comparison to radial transport, sub-radial transport leads to higher velocities, much earlier arrival times, and higher peak concentrations in breakthrough curves. Faster advective transport leads to less diffusion into fracture-bounded matrix blocks and steeper slopes of late time concentrations. Larger blocks, corresponding to slower diffusion rates, are undersampled. Transport and diffusion parameters estimated from sub-radial tracer tests using a radial ADE will lead to underestimates of dispersivity and diffusive capacity and overestimates of diffusion rate. Then, we compare the results of double porosity, radial transport and GRT interpretations of convergent flow tests conducted in a fractured dolomite in southeastern New Mexico.

DEDICATION

This thesis is dedicated to my mother who always prays for my every success.

ACKNOWLEDGMENTS

I am grateful to Almighty, Omnipresent and Omniscient God for his blessings and kindness to me for successful completion of the thesis work.

I would like to express my deepest gratitude to my advisor, Dr. Robert M. Holt Professor, for his motivation, encouragement, directional supervision, and cordial co-operation all the times. And, I would like to thank Dr. Holt for teaching me all the skills and giving me lots of time to accomplish this thesis.

I would like to acknowledge Dr. Gregg R. Davidson for his directional advices and being always so nice and helpful with me. I want to thank from core of my heart Dr. Andrew O`Reilly for his inspiration, motivation and helping me a lot during the thesis completion. I would like thank Dr. Craig J. Hickey (National Center for Physical Acoustic) for his motivation, direction and being so helpful during completion of my thesis.

I am also thankful to the Department of Geology and Geological Engineering, University of Mississippi for funding and providing all others facility to earn my degree. I would like thank Aubrey Bolen, Samuel S. Zachos and Sherra Jones for helping me a lot during the journey. I want thank my beloved wife for her helps, inspirations and supports during completion of my degree. I would like to thank all of the people who helped me since the beginning: Arindam Mukherjee, Michael Gratzer, Kingsley Abrokwah, Zach Lepchitz, Pratap Bohara and Md Abdus Samad. Last but not the least, I would like to extend my gratitude to all of the other teachers, students, and staff members of the Department of Geology and Geological Engineering, University of Mississippi who helped earn my degree.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Hydraulic test and tracer test responses in a fractured rock aquifer differs from tests performed in porous medium aquifers. The geometry of the subsurface groundwater flow system is assumed to be radial when analyzing hydraulic test data from a well; however, pumping test data from a fractured rock aquifer often indicates a non-radial flow system (Bowman, Roberts, & Holt, 2013). Barker (1988) developed the generalized radial Flow (GRF) model for the fractured rock systems. If the fracture density is large and the distribution is isotropic, then a three-dimensional spherical flow geometry might be considered (Barker, 1988). If the fracture density is low or the system is very anisotropic, a one or two-dimensional flow model is preferable (Barker, 1988). Generally, no presumption about the dimension of the flow system can be made with confidence, because of heterogeneous nature of subsurface. The flow dimension was generalized by Barker to nonintegeral values, while retaining the assumptions of radial-style flow and homogeneity. Barker's (1988) model is GRF model incorporates a geometric term flow dimension to interpret non-radial flow. General use of Barker`s model was hampered by an inability to successfully reproduce non-radial flow characteristics using a commonly used two-dimensional model (Bowman et. al 2013). A physical model linking flow dimension and simple field geometries was developed by Bowman et al. (2013), this model allows direct simulation of sub-radial, non-integer flow dimensions in 2D, non-fractal flow models (Bowman et al., 2013)

Pumping test interpretations using the GRF can be use to determine heterogeneity over large length and time scales but are insensitive to small length scales (e.g., ~10`s of m). Convergent flow tracer tests are ideal to diagnose heterogeneity at small length scales. All current models for interpreting convergent flow tracer tests assume radial flow, but they may be inappropriate in fractured rocks systems due to sub-radial flow.

In this research, a multi-rate, double-porosity Generalized Radial Transport (GRT) model was developed to consider the impact of sub-radial transport on convergent flow tracer tests in fractured rock. Application of the multi-rate, double-porosity GRT model involves development of a model for the injection, resting and finally for the pumping phase of a tracer test. The multi-rate, double-porosity GRT model for pumping phase was solved using a Laplace-domain finite domain approach after following the Haggerty`s (1995, 1998) semi-analytical solution for injection and resting period. This model has been validated using a 1D analytical solution, radial semi-analytical solution and a spherical analytical solution. The sensitivity of the GRT model was evaluated for flow dimension, Peclet number, advective porosity and diffusive porosity. Finally, the GRT was fit to H-11b3 and H-19b7 tracer test data and parameters were estimated using the GRT for these tracer tests. And, estimated parameters using GRT model were compared with the previously estimated parameters using radial model. In comparison with the multi-rate, Double-porosity GRT model, previously used model underestimated longitudinal dispersivity, diffusive capacity and advective porosity, and over estimated the capacity coefficient.

CHAPTER 2

MATHEMATICAL MODEL

2.1 GRT Model

The multi-rate, double-porosity GRT model is an extension of the GRF model of Barker (1988) for hydraulic test interpretation in fractured rock system. The assumptions of the Generalized Radial Flow (GRF) model are: 1) Darcy`s law is applicable throughout the system, 2) flow is n-dimensional radial from a single source into a homogeneous and isotropic system, 3) the source is an n-dimensional sphere, 4) the source has infinitesimal skin, and 5) boreholes in the system have negligible storage capacity and size (Barker, 1988). The GRF is suitable in heterogeneous and extensively fractured-rock aquifers because determination of hydraulic properties from traditional analysis of pumping tests can produce non-plausible results (Bowman et al., 2013). In the GRF and GRT, a flow dimension characterizes the change in flow area versus radial distance from the borehole (Figure 1) (Figure 2). Sub-linear, linear, sub-radial, radial, sub-spherical, spherical and eventually super-spherical flow dimension can be found in fractured subsurface depending on nature of heterogeneity and fractured networks (Figure 1), (Zlotnik & Logan, 1996). Sub-linear flow dimensions can be found where the cross-sectional area of flow is less than the cross-sectional flow area of screen of a pumping well (Figure 1) (Figure 2), (Zlotnik & Logan, 1996). The GRT model transforms to 1D, radial and spherical advection dispersion equation (ADE) for integer flow dimensions, n of 1, 2, and 3 respectively. And, the GRT model also transforms to sub-linear, sub-radial, sub-spherical and eventually

transform to super-spherical advection dispersion equation (ADE) for non-integer flow dimension, n of $0 < n < 1$, $1 < n < 2$, $2 < n < 3$ and $n > 3$, respectively. Non-integer flow dimensions, especially sub-radial flow, are commonly reported from pumping tests in fractured rock systems and can be linked with aquifer geometry and heterogeneity (Bowman et al., 2013).

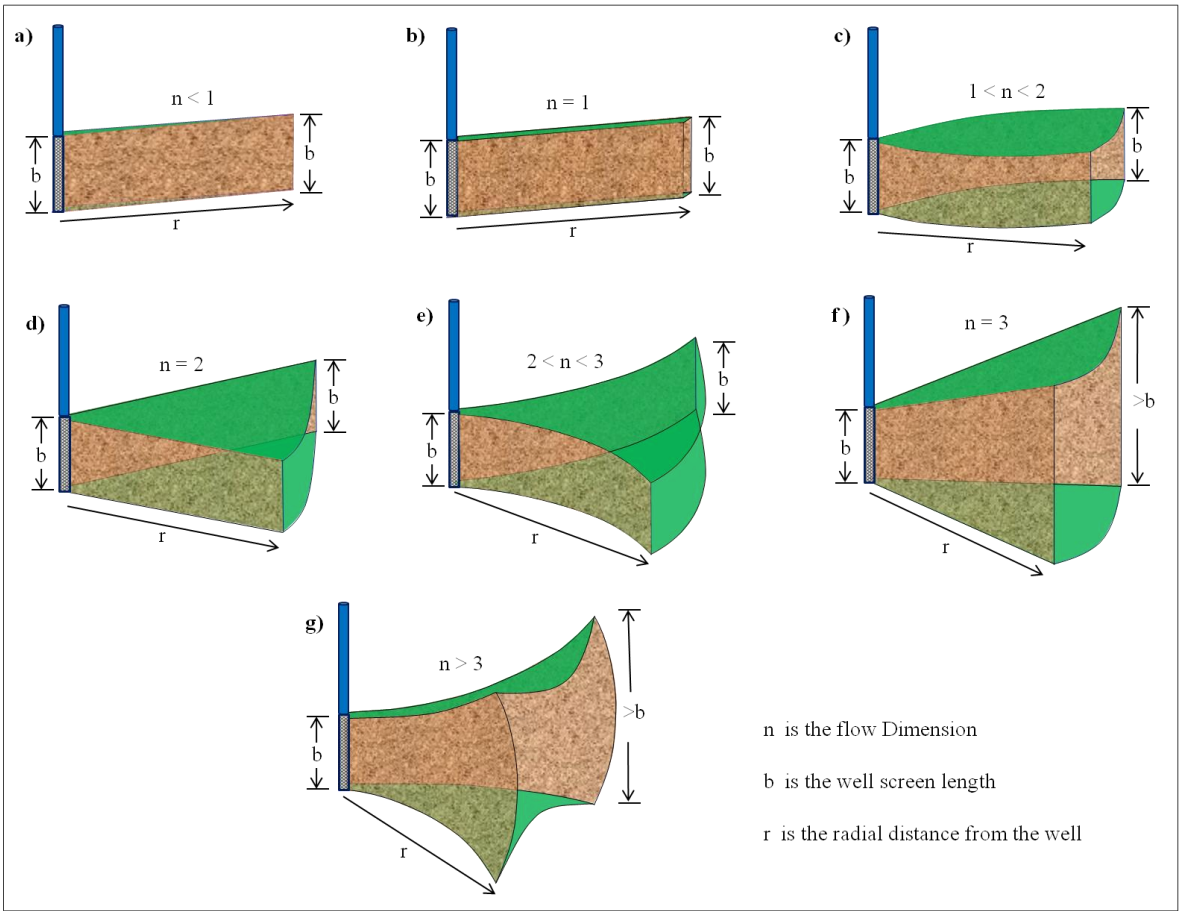

Figure 1. Conceptual Model of Flow Conduit Geometries a) Sub-linear b) linear c) sub-radial d) radial e) sub-spherical f) spherical and g) super-spherical flow conduits

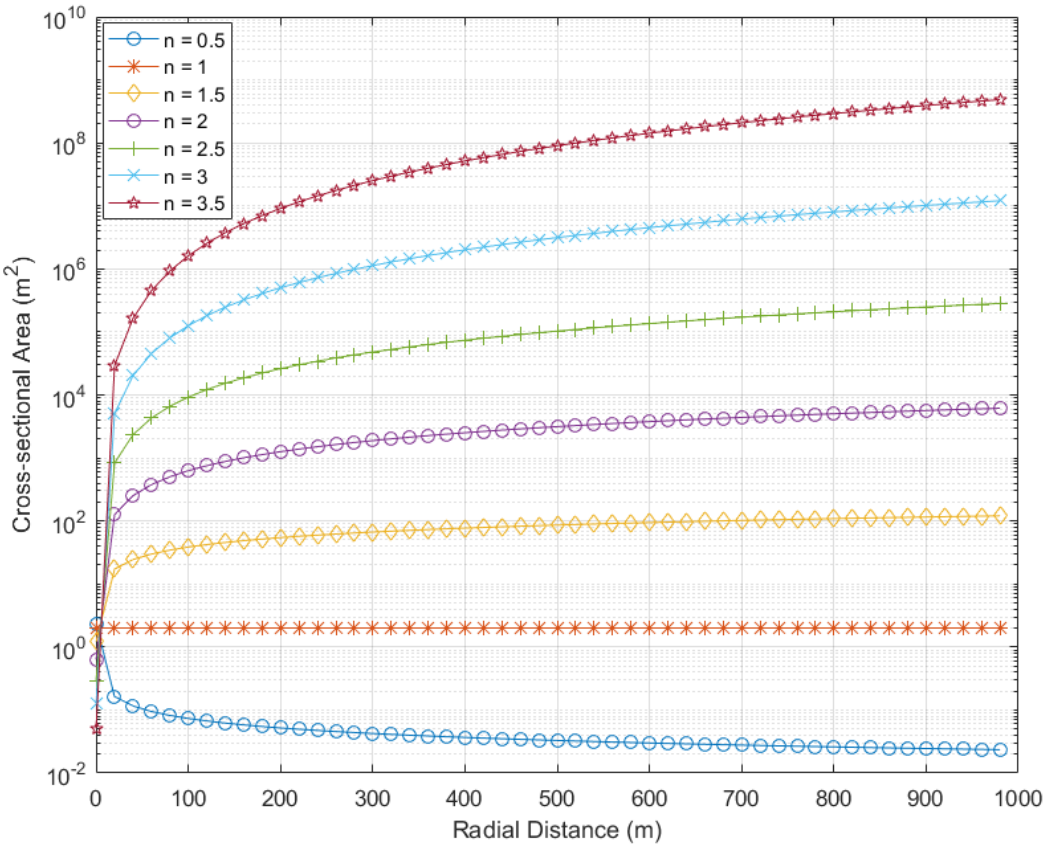

Figure 2. Cross-sectional area of flow conduits vs. radial distance from the pumping well for a fractured aquifer

2.2 Development of the GRT Model

The advective-dispersive-mass-transfer equation for a Two-Well Injection-Withdrawal test involves three steps: 1) developing transport equation for radially divergent transport from the injection well (Figure 3), 2) developing transport equation for resting period and finally (Figure 7), and 3) deriving multi-rate double porosity GRT model for withdrawal of tracer from pumping well (Figure 9). Haggerty`s (1995, 1998) multi-rate model was adopted to derived solution of step 1 and step 2. A Laplace transformed block-centered integrated finite difference model was developed to incorporate flow dimension for the resulting GRT equations. The

Laplace domain solution was numerically inverted using International Mathematics and Statistics Library (IMSL) subroutines to calculate time domain concentrations in each step (International Mathematical and Statistical Libraries, Inc & International Mathematical and Statistical Libraries (U.S.), 1982 .

2.2.1 General Transport Models

The general 1D advection-dispersion-diffusion solute transport model with first order mass transfer model can be given by (Haggerty & Gorelick, 1995):

$$A\frac{\partial C}{\partial t} + B\frac{\partial C^*}{\partial t} = D\frac{\partial^2 C}{\partial x^2} - Q\frac{\partial C}{\partial x} \qquad 1$$

$$\frac{\partial C^*}{\partial t} = \omega\left[\lambda C - C^*\right] \qquad 2$$

where $A[-]$ is the a parameter relates concentration within advective porosity and concentration within a unit volume of the porous medium ; $B[-]$ is a parameter that relates concentration within diffusive porosity and concentration within a unit volume of the porous medium ; $C[ML^{-3}]$ is the solute concentration in the advective porosity ; $C^*[ML^{-3}]$ is the average solute concentration in the diffusive porosity; $D[L^2T^{-1}]$ is the hydrodynamic dispersion coefficient; $Q[L^3T^{-1}]$ is volumetric flux; $\omega[L^{-1}]$is the first order mass transfer rate coefficient; and $\lambda[-]$is the distribution coefficient relating C^* to C ;

Considering variations in aquifer properties, all of the coefficients can be defined as the following to make the single-rate, double-porosity transport model (Haggerty & Gorelick, 1995):

$$A = \phi_f + f_f \rho_b K_{d,f} \qquad 3$$

$$B = (\phi_m) + (f_m)\rho_b (K_{d,m}) \qquad 4$$

$$D = \phi_f \alpha_L v \qquad 5$$

$$Q = \phi_f v \quad (6)$$

$$\omega = \phi_m \alpha' \quad (7)$$

$$\lambda = 1 \quad (8)$$

where, $\phi_f[-]$ is the advective porosity; $\phi_m[-]$ is the diffusive porosity; $f[-]$ is the fraction of the sorbed phase in sorption equilibrium with the mobile phase; $f_m[-]$ is the fraction of the sorbed phase in sorption equilibrium with the immobile phase; $\rho_b[M/L^3]$ is the bulk density of the porous medium; $K_{d,f}[L^3/M]$ is the distribution coefficient in the advective porosity; $K_{d,m}[L^3/M]$ is the distribution coefficient in the diffusive porosity; $\alpha_L[L]$ is the longitudinal dispersivity; $v[LT^{-1}]$ is the pore-water velocity; and $\alpha'[T^{-1}]$ is the first-order mass transfer rate coefficient in diffusive porosity.

Now, the mobile zone and immobile zone retardation factor can be defined with following (Haggerty & Gorelick, 1995):

$$R_f = 1 + \frac{f \rho_b K_d}{\phi_f} \quad (9)$$

$$\left(R_m\right)_j = 1 + \frac{(1-f)\rho_b K_d}{\left(\phi_m\right)_j} \quad (10)$$

where, $R_f[-]$ is a retardation factors in advective porosity; $R_m[-]$ is a retardation factors in matrix block; and $f[-]$ is the fraction of the sorbed phase in sorption equilibrium with the mobile phase.

And, the general 1D single-rate, double-porosity transport model with first-order mass transfer in Cartesian coordinates becomes:

$$\frac{\partial C}{\partial t} + \beta \frac{\partial C^*}{\partial t} = \frac{\alpha_L v}{R_f} \frac{\partial^2 C}{\partial x^2} - \frac{v}{R_f} \frac{\partial C}{\partial x^2} \qquad 11$$

$$\frac{\partial C^*}{\partial t} = \alpha_L \left[C - C^* \right] \qquad 12$$

where the total capacity coefficient of the formation,

$$\beta_{tot} = \frac{R_m \phi_m}{R_f \phi_f} \qquad 13$$

2.2.2 Injection Phase

The flow is radially divergent from the injection well during the injection of a tracer into the fractured aquifer (Figure3). The multi-rate model developed by Haggerty (1995) allows multiple rates of mass transfer to occur simultaneously during solute transport (Figure 4). The mass transfer occurs by diffusion of the solute into a distribution of fracture-bounded (matrix) blocks each with a different diffusion rate coefficient (Figure 5) (Haggerty, 1995) . The diffusion process into the individual matrix block assumed as a lognormal distribution because of different geologic properties, including hydraulic conductivity, sizes of micropores are lognormally or nearly lognormally distributed (Haggerty & Gorelick, 1998). Figure 6 represents a typical lognormal distribution diffusion rate coefficient used in the mass transfer model.

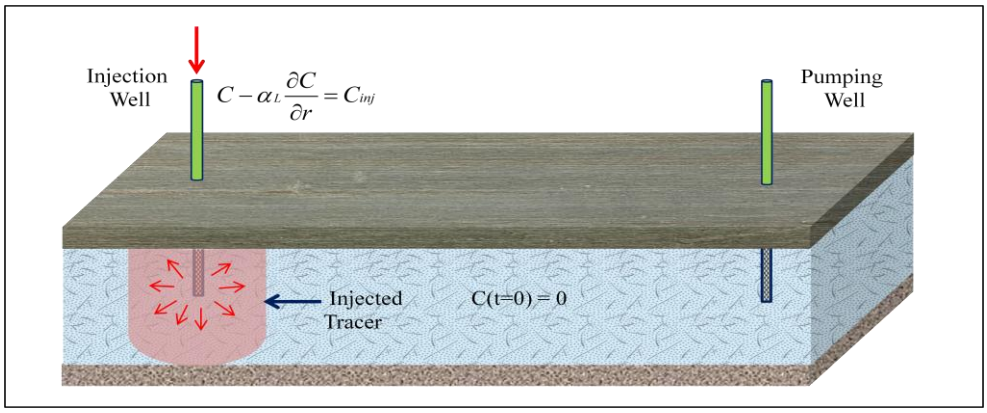

Figure 3. Conceptual Model Domain and Boundary Condition of injection phase for a convergent-flow tracer test

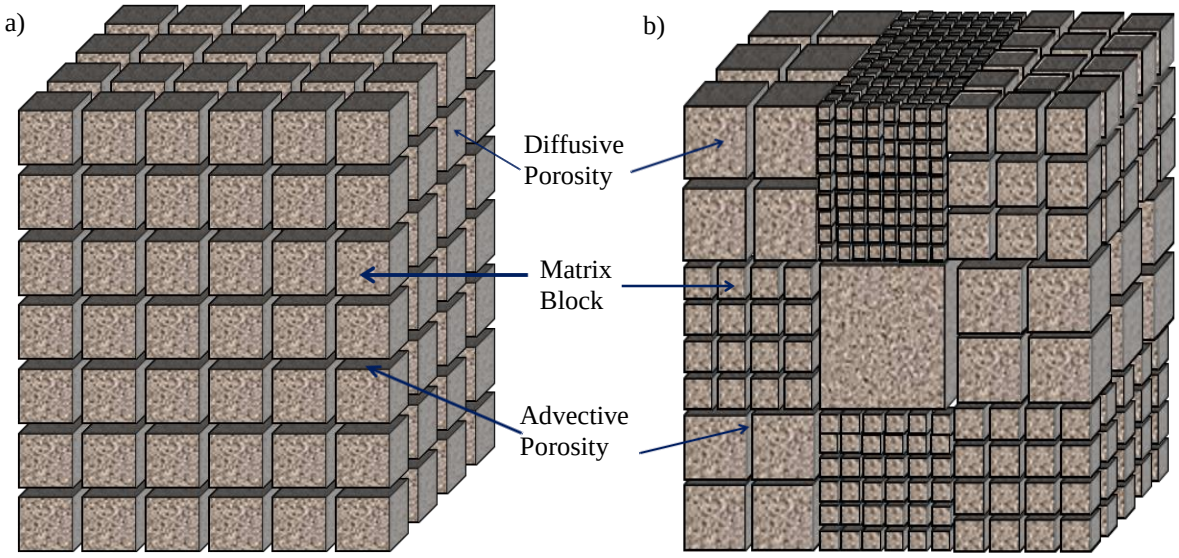

Figure 4. Conceptual Model of a) Single-Rate and b) Multi-Rate Double porosity Medium

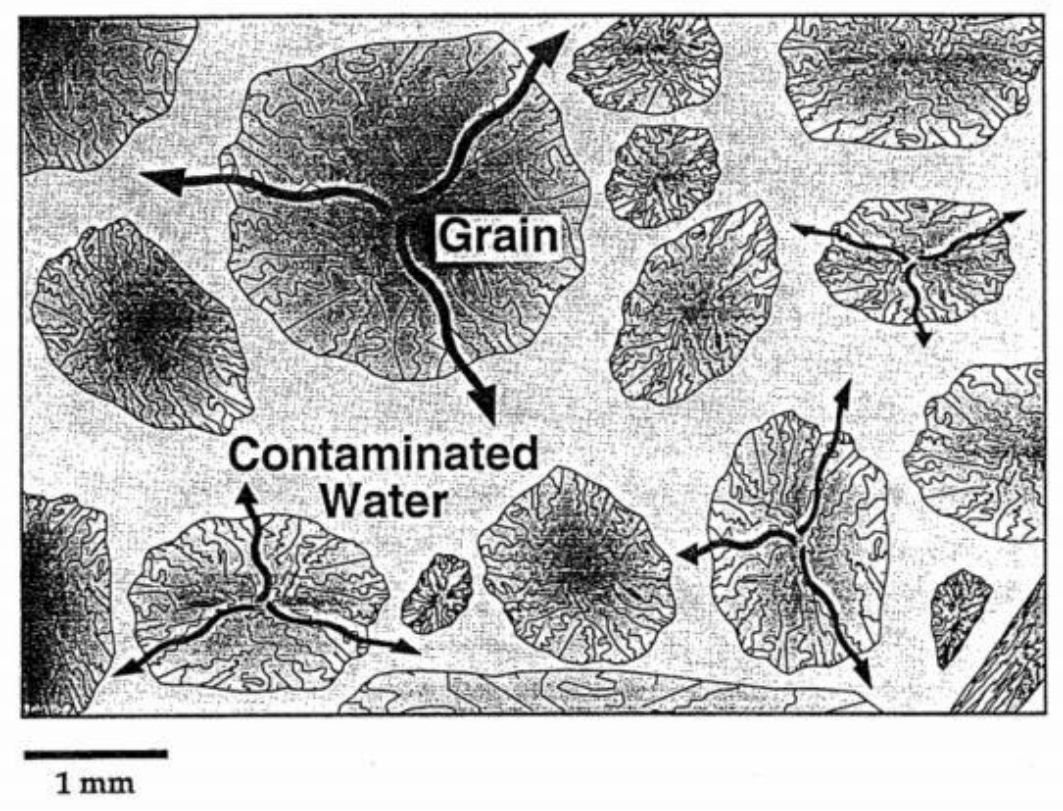

Figure 5. Diffusion Pathway Model (from Haggerty, 1995)

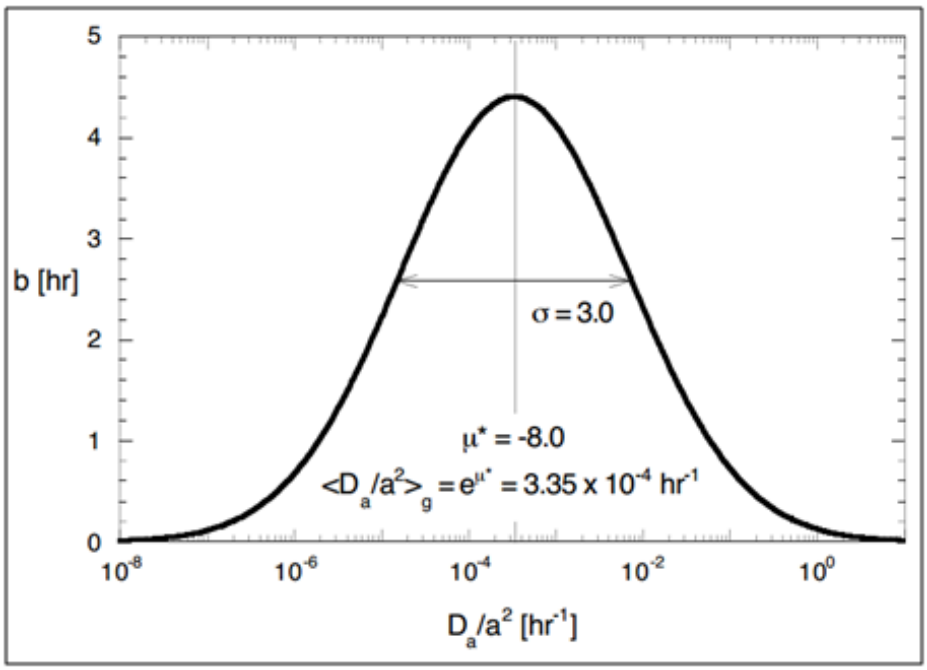

Figure 6. Probability density functions of the diffusion rate coefficient. The geometric mean of the diffusion rate coefficient is given by $\frac{D_a}{a^2}$, where the *a* is block radius. (Modified from Haggerty and Gorelick, 2000).

Transforming Cartesian coordinates into radial coordinates, the advective-dispersive-diffusive transport with multi-rate mass transfer process in the presence of lognormal distribution the equation (11), (12) and (13) becomes as (Haggerty & Gorelick, 1998):

$$\frac{\partial C}{\partial t} + \int_0^\infty b(\alpha_d) \frac{\partial \underline{C}^*(\alpha_d)}{\partial t} d\alpha_d = \frac{1}{r}\frac{\partial}{\partial r}\left(\frac{r\alpha_L v}{R_f}\frac{\partial C}{\partial r}\right) - \frac{v}{R_f}\frac{\partial C}{\partial r} \tag{14}$$

$$b(\alpha_d) = \frac{\beta_{tot}}{\sqrt{2\pi}\sigma\alpha_d} \exp\left\{-\frac{\left[\log(\alpha_d) - \mu^*\right]^2}{2\sigma^2}\right\} \tag{15a}$$

where, $\alpha_d = \frac{D_a}{a^2}$ (15b)

$$\beta_{tot} = \frac{R_m \phi_m}{R_f \phi_f} \tag{15c}$$

and where $C[ML^{-3}]$ is the solute concentration in the advective porosity ; $\underline{C}^*[ML^{-3}]$ is the average solute concentration in a particular pathway in a matrix block; $b(\alpha_d)[T^{-1}]$ is the lognormal probability density function (PDF) of diffusion rate coefficients; $\beta_{tot}[-]$ is the total capacity coefficient of the formation; $\alpha_d[L^{-1}]$ is the diffusion rate coefficient; $\alpha_L[L]$ is the longitudinal dispersivity; $v[LT^{-1}]$ is the pore-water velocity; $R_f[-]$ is the retardation factor in advective porosity; $r[L]$ is the radial coordinate; $t[T]$ is time; σ is standard deviation of the log-transformed diffusion rate coefficient; μ^* is the log of the geometric mean of the diffusion rate coefficients, $D_a[L^2T^{-1}]$ is the apparent diffusion coefficient; $a[L]$ is the length of the diffusion pathway within the matrix; $\phi_m[-]$ is the diffusive porosity (porosity in a matrix block of the formation); $\phi_f[-]$ is the advective porosity, and $R_m[-]$ is the retardation factor in matrix blocks.

The time-derivative of the average solute concentration in a particular pathway within the matrix block is given by (Haggerty, Fleming, & Mckenna, 2000):

$$\frac{\partial \underline{C}^*(\alpha_d)}{\partial t} = \frac{1}{a}\int_0^a b(\alpha_d)\frac{\partial C^*(\alpha_d)}{\partial t}dz, \qquad 0 < \alpha_d < \infty \tag{16a}$$

where $C^*[ML^{-3}]$ is the concentration at a point within the matrix; and $z[L]$ is the coordinate along the pathway. The solution of the equation (16a) is given by (Haggerty et al., 2000):

$$\frac{\partial C^*(\alpha_d)}{\partial t} = D_a \frac{\partial^2 C^*(\alpha_d)}{\partial z} \qquad 0 < \alpha_d < \infty \tag{16b}$$

For multi-rate mass transfer, a series of first-order mass transfer equations with a specific rate and capacity coefficients is precisely and mathematically equivalent to diffusion (Haggerty

& Gorelick, 1995). So, for multi-rate mass transfer we can replace diffusion term using equation (18a) in the equation (14) as follows:

$$\frac{\partial C}{\partial t}+\int_0^{\infty} b(\alpha_m)\frac{\partial C_m^{*}(\alpha_m)}{\partial t}d\alpha_m=\frac{1}{r}\frac{\partial}{\partial r}\left(\frac{r\alpha_L v}{R_f}\frac{\partial C}{\partial r}\right)-\frac{v}{R_f}\frac{\partial C}{\partial r} \tag{17}$$

$$\int_0^{\infty} b(\alpha_d)\frac{\partial C^{*}(\alpha_d)}{\partial t}d\alpha_d=\int_0^{\infty} b(\alpha_m)\frac{\partial C_m^{*}(\alpha_m)}{\partial t}d\alpha_m \tag{18a}$$

$$\frac{\partial C_m^{*}(\alpha_m)}{\partial t}=\alpha_m\left[C-C_m^{*}(\alpha_m)\right] \qquad 0<\alpha_m<\infty \tag{18b}$$

where $\alpha_m[1/T]$ are the diffusion rate coefficients of the matrix blocks; $b(\alpha_m)$ is the probability density function (PDF) describing the distribution of diffusion rate coefficients; and $C_m^{*}[ML^{-3}]$ are the concentrations in a continuous distribution of immobile zones with differing first-order diffusion rate coefficients, α_m.

The right hand side of the equations (18a) and (18b) have discrete form and for specific forms of α_m and $b(\alpha_m)$, are mathematically equivalent to solute transport with diffusion into and out of spheres, cylinders, or layers (Haggerty & Gorelick, 1995). The discrete forms these two equations are:

$$\int_0^{\infty} b(\alpha_d)\frac{\partial C^{*}(\alpha_d)}{\partial t}d\alpha_d=\sum_{j=1}^{\infty}\beta_j\frac{\partial C_{m,j}^{*}}{\partial t} \tag{19a}$$

$$\frac{\partial C_{m,j}^{*}}{\partial t}=\alpha_{m,j}\left[C-C^{*}{}_{m,j}\right] \quad , \qquad j=1,2,.....\infty \tag{19b}$$

The following two series (multi-rate series) make equation (19a) and (19b) equivalent to diffusion into or out of a single one-dimensional pathway (Haggerty et al., 2000);

$$\alpha_{m,j}=\frac{(2j-1)^2\pi^2\alpha_d}{4}, \qquad j=1,2,.....\infty \tag{20a}$$

$$\beta_j = \frac{8\beta_{tot}}{(2j-1)^2 \pi^2}, \qquad j = 1, 2, ...\infty \tag{20b}$$

The equations (20a) and (20b) require truncation for practical use as we are not able to use the entire infinite series and for spherical diffusion the truncated series is calculated precisely up to 35 rate coefficients (Haggerty et al., 2000).

We need to use continuous multi-rate distribution that is equivalent to a lognormal distribution of diffusion rate coefficients. We need to convert $b(\alpha_d)$ to $b(\alpha_m)$ using the intermediate discrete form of $b(\alpha_{m,j})$. The relationship is given by (Thomas, 1986, page 159):

$$\beta(\alpha_{m,j}) = \beta\left(g(\alpha_{m,j})\right)\left|\frac{d\left(g(\alpha_{m,j})\right)}{d\alpha_{m,j}}\right| \tag{21}$$

where, $g(\alpha_{m,j})$ is the inverse of the function given in the equation (20a), i.e.,

$$\alpha_d = g(\alpha_{m,j}) = \frac{4\alpha_{m,j}}{(2j-1)^2 \pi^2}, \qquad j = 1, 2, ...\infty \tag{22}$$

Using equation (15a) and (22) into the equation (21) we get,

$$b(\alpha_{m,j}) = \frac{\beta_{tot}}{\sqrt{2\pi}\sigma\alpha_{m,j}} \exp\left\{ -\frac{\left[ln\left(\frac{4\alpha_{m,j}}{(2j-1)^2 \pi^2} \right) - \mu^* \right]^2}{2\sigma^2} \right\}, \quad j = 1, 2, ...\infty \tag{23}$$

The equation (23) represents an infinite number of distributions. Each of the distribution is continuous on $\alpha_{m,j}$ and we can sum them together in the proportion given in (20b), which represents the weight of each distribution j. Summing up and re-arranging we get (Haggerty et al., 2000):

$$b(\alpha_m) = \sum_{j=1}^{\infty} \left(\frac{8\beta_{tot}}{\sqrt{2\pi^5}\left(2j-1\right)^2 \sigma\alpha_m} \exp\left\{ -\frac{\left[l\mathrm{n}\left(\frac{4\alpha_m}{\left(2j-1\right)^2 \pi^2} \right) - \mu^* \right]^2}{2\sigma^2} \right\} \right) \tag{24}$$

Equation (24) represents the PDF of the multi-rate which is equivalent to the distributed diffusion model with a lognormal distribution of α_d . Mass transfer with either model is equivalent. Either substitution of the equation (24) into (18a) and then (14) or discretization of equation (24) and insert into (19a) and then (14) can be used. In this research discretization of the equation (24) such that both the PDF and associated CDF are exact at each discretization node was used. Mathematically, this can be done as (Haggerty et al., 2000):

$$\beta_j = \int_{\alpha j-1/2}^{\alpha j+1/2} b(\alpha_m) d\alpha_m, \qquad 0 \le \alpha_m < \infty \tag{25}$$

The pore water velocity in the equation (14) is given by (Haggerty et al., 2000):

$$v = \frac{Q}{2\pi r \phi_f b_f} \tag{26}$$

where, $Q[L^3T^{-1}]$ is the injection or pumping rate; $\phi_f[-]$ is the fracture porosity; and $b_f[-]$ is the thickness of the formation. The boundary conditions for (14) and (17) (Figure 3) are (Haggerty et al., 2000):

$$C - \alpha_L \frac{\partial C}{\partial r} = C_{inj} \qquad \text{at} \quad r = r_w \tag{27a}$$

$$\frac{\partial C}{\partial r} = 0 \qquad r \to \infty \tag{27b}$$

where, $r_w[L]$ is the well radius and $C_{inj}[ML^{-3}]$ is the injected concentration. For both advective and dispersive flux across the inlet boundary equation (27a) represents the standard Cauchy boundary (Kreft & Zuber, 1978). The initial conditions of concentration for both matrix and

fracture porosity is zero initially.

All of the equations for injection phase can be nondimensionalized with the following (Haggerty et al., 2000):

$$T = \frac{Qt}{2\pi b_f \phi_f \alpha_L^2 R_f} \tag{28a}$$

$$\omega_m = \alpha_m \frac{2\pi b_f \phi_f \alpha_L^2 R_f}{Q} \tag{28b}$$

$$\rho = \frac{r}{\alpha_L} \tag{28c}$$

$$\rho_w = \frac{r_w}{\alpha_L} \tag{28d}$$

$$\mu = \mu^* - \ln \frac{Q}{2\pi b_f \phi_f \alpha_L^2} \tag{28f}$$

The mass transport equations (17), (18b) and (24) become:

$$\frac{\partial C}{\partial T} + \int_0^\infty b(\omega_m) \frac{\partial C_m^*(\omega_m)}{\partial T} d\omega_m = \frac{1}{\rho}\left(\frac{\partial^2 C}{\partial \rho^2} - \frac{\partial C}{\partial \rho}\right) \tag{29}$$

$$b(\omega_m) = \sum_{j=1}^{\infty} \left(\frac{8\beta_{tot}}{\sqrt{2\pi^5}\,(2j-1)^2\,\sigma\alpha_m} \exp\left\{ -\frac{\left[\ln\left(\frac{4\omega_m}{(2j-1)^2 \pi^2} \right) - \mu \right]^2}{2\sigma^2} \right\} \right) \tag{30a}$$

$$\frac{\partial C_m^*(\alpha_m)}{\partial T} = \omega_m \left[C - C_m^*(\omega_m) \right] \tag{30b}$$

The boundary conditions become:

$$C - \frac{\partial C}{\partial \rho} = C_{inj} \qquad \text{at} \qquad \rho = \rho_w \tag{31a}$$

$$\frac{\partial C}{\partial \rho} = 0 \qquad \text{at} \qquad \rho \to \infty \tag{31b}$$

The Laplace transform of (29) and (30b) are:

$$p\overline{C} + p\int_0^\infty b(\omega_m)\overline{C_m^*(\omega_m)} d\omega_m = \frac{1}{\rho}\left(\frac{\partial^2 \overline{C}}{\partial \rho^2} - \frac{\partial \overline{C}}{\partial \rho}\right) \tag{32}$$

$$p\overline{C_m^*(\omega_m)} = \omega_m\left[\overline{C} - \overline{C_m^*(\omega_m)}\right] \tag{33a}$$

$$\overline{C_m^*(\omega_m)} = \frac{\omega_m \overline{C}}{p+\omega_m} \tag{33b}$$

By plugging $\overline{C_m^*(\omega_m)}$ from (20b) into the equation (32) we get,

$$\frac{\partial^2 \overline{C}}{\partial \rho^2} - \frac{\partial \overline{C}}{\partial \rho} - \rho p\left[1 + \int_0^\infty \frac{b(\omega_m)\omega_m}{p+\omega_m} d\omega_m\right]\overline{C} = 0 \tag{34}$$

The boundary conditions become:

$$\overline{C} - \frac{\partial \overline{C}}{\partial \rho} = \overline{C_{inj}} \qquad \text{at} \quad \rho = \rho_w \tag{35a}$$

$$\frac{\partial \overline{C}}{\partial \rho} = 0 \qquad \text{at} \quad \rho \to \infty \tag{35b}$$

It was assumed that the injected concentration begins at zero, then goes instantaneously to a uniform injection rate of concentration for a given time, and then instantaneously become zero. The Laplace transform of the injected concentration (Haggerty et al., 2000):

$$\overline{C_{inj}} = C_{inj}\frac{\exp\left(pT_0\right) - \exp\left(pT_E\right)}{p} \tag{36}$$

where, p is the Laplace operator and overbar indicates the Laplace transform of a variable. $T_0[-]$ is the beginning dimensionless time for the pulse, and $T_E[-]$ is the ending dimensionless time for the pulse

Equation (34) can be transformed into a homogeneous Airy equation (Chen, 1985). We can define the following variable to simplify (34) (Haggerty et al., 2000):

$$\mathbb{N} = p\left[1 + \int_0^\infty \frac{b(\omega_m)\omega_m}{p+\omega_m} d\omega_m\right] \tag{37}$$

The equation (34) can re-write as:

$$\frac{\partial^2 \overline{C}}{\partial \rho^2} - \frac{\partial \overline{C}}{\partial \rho} - \rho \mathbb{N} \overline{C} = 0 \tag{38}$$

The general solution to the equation (38) (Chen, 1985) is:

$$\overline{C} = A_1 \exp\left(\frac{y}{2}\right) A_i\left(\mathbb{N}^{1/3} y\right) + A_2 \exp\left(\frac{y}{2}\right) B_i\left(\mathbb{N}^{1/3} y\right) \tag{39}$$

where, A_1 and A_2 are functions yet to be determined; $A_i\left(\mathbb{N}^{1/3} y\right)$ and $B_i\left(\mathbb{N}^{1/3} y\right)$ are Airy functions and

$$y = \rho + \frac{1}{4\mathbb{N}} \tag{40}$$

Solving (32) with boundary condition we have the solution (Haggerty et al., 2000):

$$\overline{C} = \overline{C_{inj}} Exp\left(\frac{\rho - \rho_0}{2}\right) \frac{A_i\left(\mathbb{N}^{1/3} y\right)}{\frac{1}{2} A_i\left(\mathbb{N}^{1/3} y_0\right) - \mathbb{N}^{1/3} A_i\left(\mathbb{N}^{1/3} y_0\right)} \tag{41}$$

where, $A_i\left(\mathbb{N}^{1/3} y\right)$ is the derivative of the Airy function and where y_0 indicates the value of y at $\rho = \rho_0$

Using a numerical Laplace inversion algorithm, equations (33b) and (41) can be inverted from Laplace domain to time domain. The de Hoog et. al. (1982) algorithm was employed for inversion as developed in the commercially available International Mathematics and Statistics Libraries (IMSL).

2.2.3 Resting Periods

During the resting period, the advective velocity is assumed to be zero (Haggerty et al., 2000) and injected tracer diffused into matrix blocks and spread with hydrodynamic dispersion (Figure 7). So, the equation (14) becomes:

$$\frac{\partial C}{\partial t} + \int_0^{\infty} b(\alpha_d) \frac{\partial \underline{C}^*(\alpha_d)}{\partial t} d\alpha_d = 0 \tag{42}$$

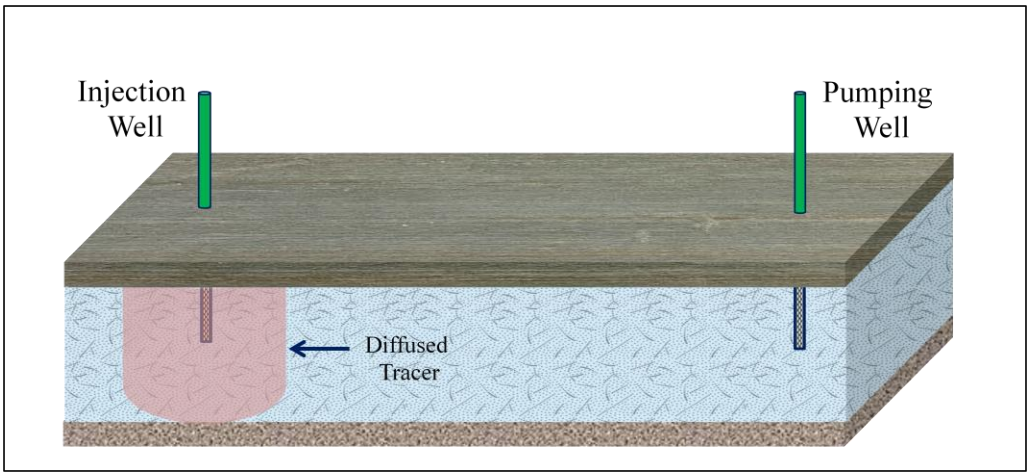

Figure 7. Conceptual Model Domain and Boundary Condition of resting phase for a convergent-flow tracer test

The equations (15a) through (16b) were used from the injection phase. And, for multi-rate of resting phase, the integral part of the equation (44) was replaced with the integral in (17), and replacing (15a), (16a) and (16b) with (18b) and (24). The initial conditions of concentrations in this phase are concentrations at the end of the injection period. The dimensional Laplace domain solution of (42) is

$$\overline{C} = \frac{C_0 + \int_0^{\infty} \frac{b(\alpha_m)\alpha_m}{p+\alpha_m} C_0^*(\alpha_m) d\alpha_m}{p\left[1 + \int_0^{\infty} \frac{b(\alpha_m)\alpha_m}{p+\alpha_m} d\alpha_m\right]} \tag{43}$$

$$\overline{C^*(\alpha_m)} = \frac{\alpha_m}{p+\alpha_m}\overline{C} + \frac{C_0^*(\alpha_m)}{p+\alpha_m} \tag{44}$$

where, $C_0^*(\alpha_m)\left[M / L^3\right]$ are the concentrations in the multi-rate immobile zones at the end of the injection phase; $C_0\left[M / L^3\right]$ are the concentrations in the mobile zones at the end of the injection period. The equation (43) and (44) was inverted to time domain as injection period was done.

2.2.4 Withdrawal from the Pumping Well

After following the resting period, concentration was calculated at the pumping well after converting from polar coordinates relative to injection well (r_{in}, θ_{in}) to coordinates relative to the pumping well (r_{out}, θ_{out}) and azimuthally average the concentrations (Mckenna, Meigs, & Haggerty, 2001).

Figure 8 shows the relationship between polar coordinate systems with respect to two wells. The Cartesian coordinate system that are used to derive the relationship (i.e., $x = r_{out}\cos\theta_{out} = R_0 + r_{in}\cos\theta_{in}$ and $y = r_{out}\sin\theta_{out} = R_0 + r_{in}\sin\theta_{in}$) are also shown in the figure 8.

The transformations of coordinates are (Mckenna et al., 2001):

$$r_{out} = \sqrt{R_0^2 + 2R_0 r_{in}\cos\theta_{in} + r_{in}^2} \tag{45a}$$

$$\theta_{out} = \tan^{-1}\left(\frac{r_{in}\sin\theta_{in}}{R_0 + r_{in}\cos\theta_{in}}\right) \tag{45b}$$

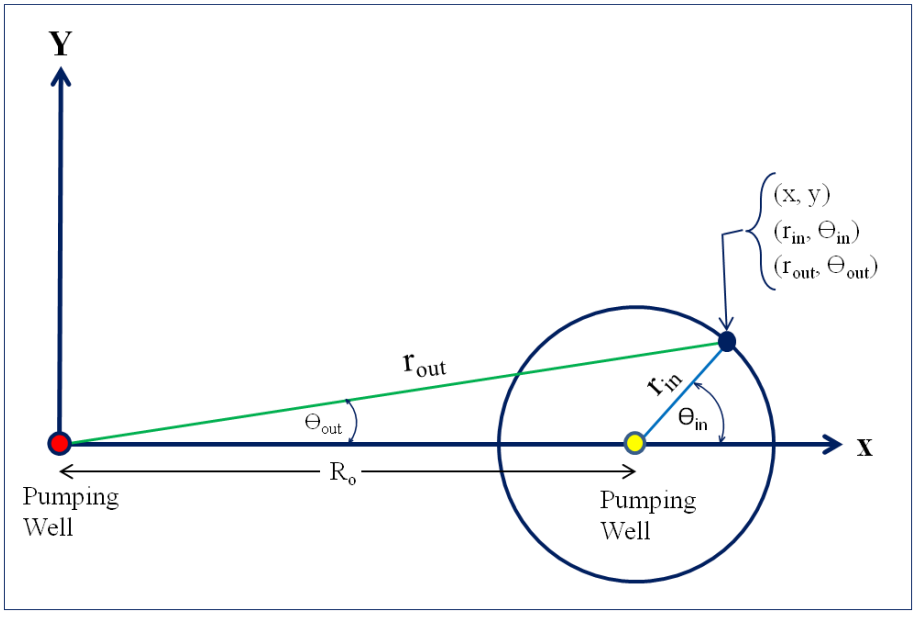

Figure 8. Schematic diagram of polar coordinate transformation (Modified from McKenna et al. 2001)

Azimuthally averaging concentration takes all concentrations at a radial distance r from the pumping well and averages them. If a formation is homogeneous with respect to thickness, porosity, and hydraulic conductivity, then at a radial distance from the withdrawal well solute experience same velocity and similar dispersion coefficient as it moves towards the pumping well. Therefore, transport is simulated in one dimension rather than two and concentration can be averaged (Haggerty et al., 2000). The azimuthally averaged dimensional concentration will be use as initial concentration for the withdrawal phase and can be defined by (Zlotnik & Logan, 1996):

$$C(r_{out}, t_0) = \frac{1}{2\pi} \int_{\pi}^{-\pi} C(r_{out}, \theta_{out}, t_0) d\theta_{out} \tag{46}$$

A geometric term flow dimension was incorporated to investigate sub-radial flow in fractured rock medium in the pumping phase. It was assumed the flow is in a steady state. The cross-sectional area of flow incorporating the flow dimension n is given by (Walker & Roberts, 2003):

$$A(r) = \frac{b^{3-n} 2\pi^{n/2} r^{n-1}}{\Gamma(n/2)} \tag{47}$$

where, $b[L]$ is the thickness of aquifer, n is flow dimension, $r[L]$ is radial distance from he borehole, $A(r)[L^2]$ is the cross-sectional area of flow, $b[L]$ is the thickness of aquifer, $n[-]$ is the flow dimension, and $\Gamma[-]$ is the gamma function. The flow dimension n describes the geometry of the system by defining the rate that the cross- sectional flow area changes with respect to distance from the pumping well.

Groundwater velocity near a pumping well is given by:

$$V(r) = \frac{Q}{\phi_f R_f A(r)} = \frac{Q\Gamma(n/2)}{R_f \phi_f b^{3-n} 2\pi^{n/2} r^{n-1}} = \frac{f(n)}{r^{n-1}} \tag{48}$$

where, $$f(n) = \frac{Q\Gamma(n/2)}{R_f \phi_f b^{3-n} 2\pi^{n/2}} \tag{49}$$

where $V(r)[L^2T^{-1}]$ is the groundwater velocity near a pumping well, $R_f[-]$ is the retardation coefficient in the fractures, $Q[L^3T^{-1}]$ is volumetric discharge rate from the well, and $\phi_f[-]$ is the porosity of fractures,

The dispersion coefficient is given by:

$$D(r) = \alpha_L |V(r)| = \frac{\alpha_L |f(n)|}{r^{n-1}} \tag{50}$$

where $\alpha_L[L]$ is the longitudinal dispersivity. The flux of concentration is given by:

$$q = -D(r)\frac{\partial C}{\partial r} + V(r)C \tag{51}$$

The conservation of mass equation for GRT is given by:

$$\frac{\partial C}{\partial t} = -\frac{1}{r^{n-1}}\frac{\partial}{\partial r}\left(r^{n-1}q\right) \tag{52}$$

A detailed derivation of the single porosity GRT model is given in the APPENDIX A. The single porosity GRT model was obtained by solving equation (52).

$$\frac{\partial C}{\partial t} = \alpha_L \frac{Q\Gamma(n/2)}{R_f \phi b^{3-n} 2\pi^{n/2} r^{n-1}} \frac{\partial^2 C}{\partial r^2} - \frac{Q\Gamma(n/2)}{R_f \phi b^{3-n} 2\pi^{n/2} r^{n-1}} \frac{\partial C}{\partial r} \tag{53}$$

A multi-rate, double porosity GRT model for convergent flow with lognormal

distribution can be written as:

$$\frac{\partial C}{\partial t}+\int_0^{\infty} b(\alpha_m)\frac{\partial C_m^{*}(\alpha_m)}{\partial t}d\alpha_m=\alpha_L\frac{Q\Gamma(n/2)}{R_f\phi b^{3-n}2\pi^{n/2}r^{n-1}}\frac{\partial^2 C}{\partial r^2}-\frac{Q\Gamma(n/2)}{R_f\phi b^{3-n}2\pi^{n/2}r^{n-1}}\frac{\partial C}{\partial r} \tag{54}$$

$$b(\alpha_m)=\sum_{j=1}^{\infty}\left(\frac{8\beta_{tot}}{\sqrt{2\pi^5}(2j-1)^2\sigma\alpha_m}\exp\left\{-\frac{\left[ln\left(\frac{4\alpha_m}{(2j-1)^2\pi^2}\right)-\mu^{*}\right]^2}{2\sigma^2}\right\}\right) \tag{55a}$$

$$\frac{\partial C_m^{*}(\alpha_m)}{\partial t}=\alpha_m\left[C-C_m^{*}(\alpha_m)\right] \qquad 0<\alpha_m<\infty \tag{55b}$$

The boundary conditions for the withdrawal phase are different than that of the injection phase.

The boundary conditions are (Figure 9):

$$\frac{\partial C}{\partial r}=0 \qquad r=r_w \tag{56a}$$

$$C=0 \qquad r\to\infty \tag{56b}$$

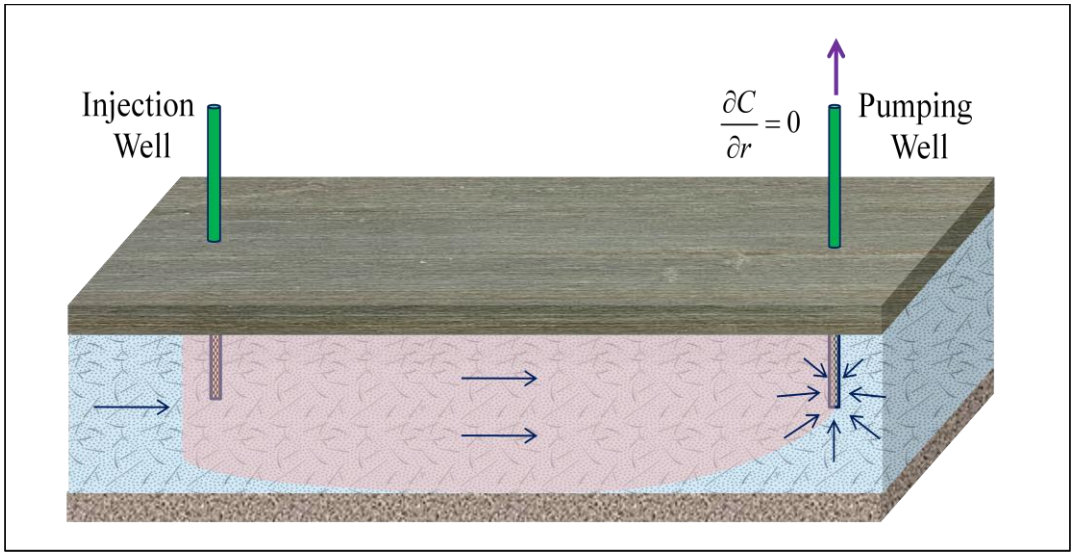

Figure 9. Conceptual Model Domain and Boundary Condition withdrawal phase for a convergent-flow tracer test

The equations (54), (55a) and (55b) for the withdrawal phase can be nondimensionalized using the following definitions:

$$C_D = C / C_{rest} \tag{57a}$$

$$t_D = \frac{Q}{\phi_f L^3 R_f} t \tag{57b}$$

$$\omega_{mp} = \alpha_m \frac{\phi_f L^3}{Q} \tag{57c}$$

$$r_D = r / L \tag{57d}$$

$$r_{D^w} = r_w / L \tag{57e}$$

$$b_D = b / L \tag{57f}$$

$$\mu_p = \mu^* - \log \frac{Q}{\phi_f L^3} \tag{57g}$$

$$P_e = \frac{L}{\alpha_L} \tag{57h}$$

where, $L[L]$ is the distance from the injection well to the pumping well.

Using these nondimensional forms equations (54), (55a) and (55b) become:

$$\frac{\partial C_D}{\partial t_D} + \int_0^\infty b(\omega_{mp}) \frac{\partial C^*_{Dm}(\omega_{mp})}{\partial t_D} d\omega_{mp} = \frac{1}{S_f(n) r_D^{n-1}} \left(\frac{1}{P_e} \frac{\partial^2 C_D}{\partial r_D^2} - \frac{\partial C_D}{\partial r_D} \right) \tag{58}$$

$$b(\omega_{mp}) = \sum_{j=1}^{\infty} \left(\frac{8\beta_{tot}}{\sqrt{2\pi^5}\left(2j-1\right)^2 \sigma \omega_{mp}} \exp\left\{ - \frac{\left[l\mathrm{n}\left(\frac{4\omega_{mp}}{\left(2j-1\right)^2 \pi^2} \right) - \mu_p \right]^2}{2\sigma^2} \right\} \right) \tag{59}$$

$$\frac{\partial C^*_{Dm}(\omega_{mp})}{\partial t_D} = \omega_{mp} \left[C_D - C^*_{Dm}(\omega_{mp}) \right] \tag{59b}$$

$$S_f(n) = \frac{2b_D^{3-n}\pi^{n/2}}{\Gamma(n/2)} \tag{59c}$$

Boundary Conditions are:

$$\frac{\partial C_D}{\partial r_D} = 0 \qquad \text{at } r_D = r_{Dw} \tag{58a}$$

$$C_D = 0 \qquad \text{at } r_D \to \infty \tag{58b}$$

The concentrations at the end of the resting period are initial condition both for fractures and fractures bounded blocks in the withdrawal phase. The Laplace transform of the equation (58) and (59b) with the non-zero initial conditions are:

$$p\overline{C_D} - C_{D,0} + \int_0^\infty b(\omega_{mp})\left[p\overline{C^*_{Dm}(\omega_{mp})} - C^*_{Dm,0}(\omega_{mp})\right] d\omega_{mp} = \frac{1}{S_f(n) r_D^{n-1}}\left(\frac{1}{P_e}\frac{\partial^2 \overline{C_D}}{\partial r_D^2} - \frac{\partial \overline{C_D}}{\partial r_D}\right) \tag{59}$$

$$\overline{C^*_{Dm}(\omega_{mp})} = \frac{\omega_{mp}}{p+\omega_{mp}}\overline{C_D} + \frac{C^*_{Dm,0}(\omega_{mp})}{p+\omega_{mp}} \tag{60}$$

where, $C_{D,0}$ is the dimensionless initial concentration in the fractures , $C^*_{Dm,0}(\omega_{mp})$ is the dimensionless initial concentration in the matrix blocks and overbar indicates the Laplace transform of a variable.

By using (59) and (60), we have the following equation:

$$\begin{aligned} &\frac{1}{P_e}\frac{\partial^2 \overline{C_D}}{\partial r_D^2} - \frac{\partial \overline{C_D}}{\partial r_D} - S_f(n) r_D^{n-1} p\left[1 + \int_0^\infty \frac{b(\omega_{mp})\omega_{mp}}{p+\omega_{mp}} d\omega_{mp}\right]\overline{C_D} \\ &+ S_f(n) r_D^{n-1}\left[C_{D,0} + \int_0^\infty \frac{b(\omega_{mp})\omega_{mp}}{p+\omega_{mp}} C^*_{Dm,0}(\omega_{mp}) d\omega_{mp}\right] = 0 \end{aligned} \tag{61}$$

We can define following variables to simplify the equation (61)

$$\mathbb{R} = p\left[1 + \int_0^\infty \frac{b(\omega_{mp})\omega_{mp}}{p+\omega_{mp}} d\omega_{mp}\right] \tag{62a}$$

$$\mathbb{Z} = \left[C_{D,0} + \int_0^\infty \frac{b(\omega_{mp})\omega_{mp}}{p + \omega_{mp}} C^*_{Dm,0}(\omega_{mp}) d\omega_{mp} \right] \tag{62b}$$

The equation (61) becomes:

$$\frac{1}{P_e}\frac{\partial^2 \overline{C_D}}{\partial r_D^2} - \frac{\partial \overline{C_D}}{\partial r_D} + S_f(n)\left[\mathbb{Z} - \mathbb{R}\overline{C_D}\right] r_D^{n-1} = 0 \tag{63}$$

Using, $A_1 = \frac{1}{P_e\left(r_D^{n-1}\right)}$, $A_2 = \frac{1}{r_D^{n-1}}$ and $A_3 = S_f(n)\mathbb{R}$ (64)

the equation (64) becomes as:

$$A_1 \frac{\partial^2 \overline{C_D}}{\partial r_D^2} - A_2 \frac{\partial \overline{C_D}}{\partial r_D} - A_3 \overline{C_D} = -S_f(n)\mathbb{Z} \tag{65}$$

The Laplace domain solution of (53) does not have an exact analytical solution and the block-centered integrated finite difference approach was adopted to numerically evaluate the solution. Conceptually, for a cell at nondimensional radial distance $r_{D,i}$ from the pumping well (Figure 10) the solution can be written as:

$$\left(\frac{A_1}{(\Delta r)^2} + \frac{A_2}{2\Delta r}\right)\overline{C}_{D_{i-1}} + \left(-\frac{2A_1}{(\Delta r)^2} - A_3\right)\overline{C}_{D_i} + \left(\frac{A_1}{(\Delta r)^2} - \frac{A_2}{2\Delta r}\right)\overline{C}_{D_{i+1}} = -S_f(n)\mathbb{Z} \tag{66}$$

Let, $D_1 = \frac{A_1}{(\Delta r)^2} + \frac{A_2}{2\Delta r}$, $D_2 = -\frac{2A_1}{(\Delta r)^2} - A_3$ and $D_3 = \frac{A_1}{(\Delta r)^2} - \frac{A_2}{2\Delta r}$ (67)

So, the equation becomes as:

$$D_1\overline{C}_{D_{i-1}} + D_2\overline{C}_{D_i} + D_3\overline{C}_{D_{i+1}} = -S_f(n)\mathbb{Z} \tag{68}$$

Equation (68) is the Laplace domain multi-rate, double porosity GRT model. The equation was inverted from Laplace domain to time domain using the de Hoog (1982) numerical algorithm.

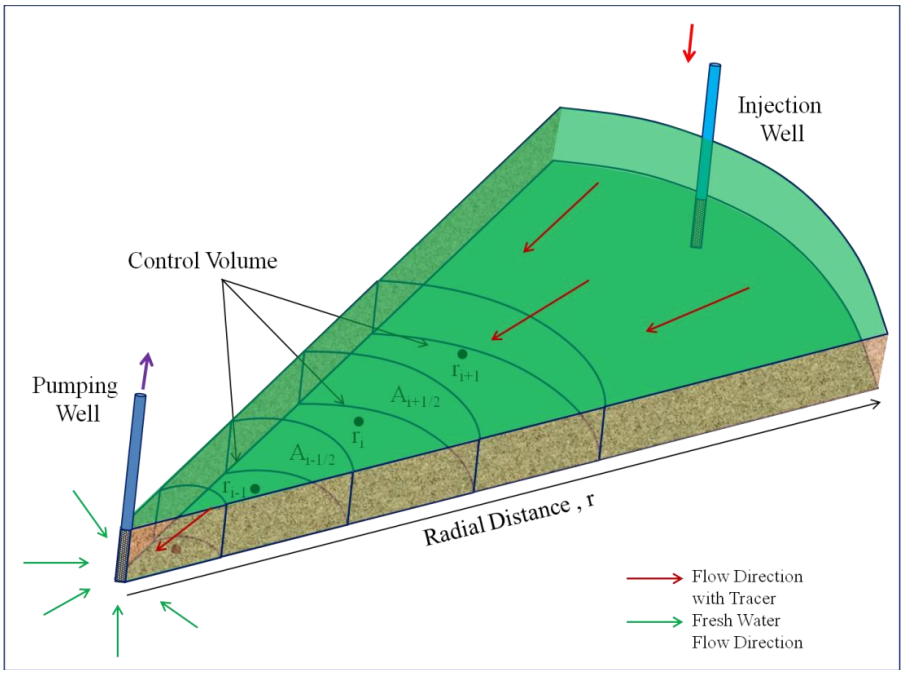

Figure 10. Conceptual model of Control Volume

CHAPTER 3

GRT MODEL VERIFICATION

The multi-rate, double-porosity GRT model is capable of simulating contaminant or tracer transport in subsurface without diffusion, with single-rate diffusion into fracture bounded blocks, and with multi-rate diffusion into fracture bounded blocks. The GRT model also can simulate transport for sub-linear, linear, sub-radial, radial, sub-spherical, spherical and eventually super-spherical flow dimension which can be found in fractured rock aquifer. In this verification chapter, the GRT model was compared and verified with 1D analytical, radial semi-analytical and spherical analytical transport models because transport models with a fractional flow dimension is not available.

3.1. Verification with 1D Model

3.1.1. Continuous Source

The pumping phase part of the multi-rate, double-porosity GRT model was simulated for continuous injection of tracer using flow dimension, n equal to 1 that transforms the GRT model to 1D transport model. The GRT was simulated considering boundary conditions (equation 56a and 56b) and initial condition $C(t=0)=0$. Simulated results were compared with Ogata and Banks (1961) 1D advection-diffusion analytical solution that has the same boundary and initial conditions. And, the breakthrough curve showed that 50% concentration comes at 1 hour like plug-flow as the advective porosity value was 1(Figure 11). The GRT model showed almost perfect matching of breakthrough curve with the analytical solution with given data in Table 1 except some numerical dispersion (Figure 11).

Table 1. Parameters for GRT model verification with 1D analytical solution with continuous source

Parameters	Values
Time, t	1000 hrs
Advective Porosity, ϕ_f	1
Diffusive Porosity, ϕ_m	0
Distance, L	10 m
Longitudinal Dispersivity, α_L	0.1 m
Δr	0.01

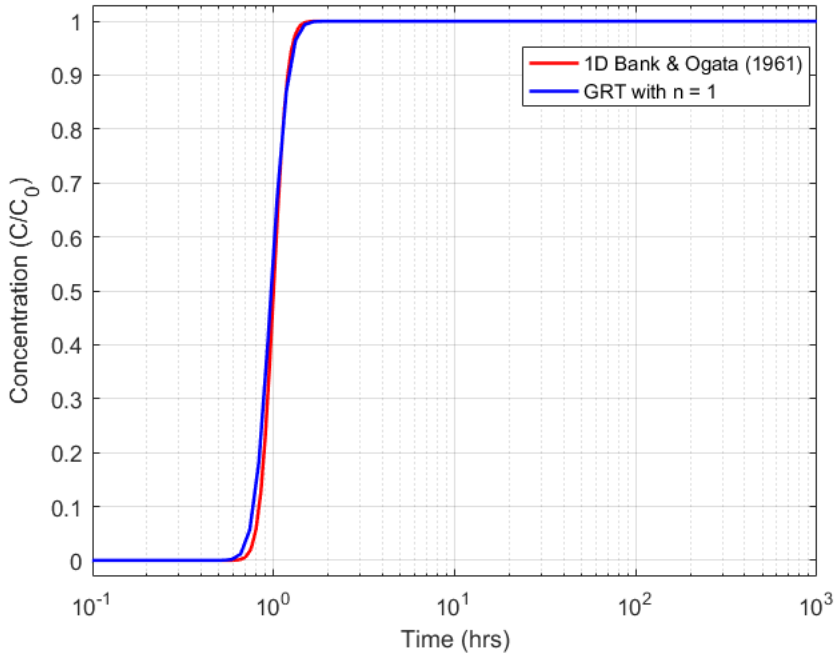

Figure 11. GRT (n =1) verification with 1D Bank and Ogata (1961) Model, Continuous Source

3.1.2. Pulse Type Source

The Haggerty (1995) 1D analytical model was used to verify the GRT model with a flow dimension of the 1. Haggerty (1995) 1D analytical model has similar boundary conditions (equation 56a and 56b) and initial condition as the GRT for a pulse type tracer injection. Simulating GRT model and Haggerty (1995) 1D model with the same input parameters in table

2, GRT model showed the peak concentration come at 1.5 hrs same as the peak concentration arrival time of the Haggerty`s model (Figure 12). Result of GRT model match closely with the Haggerty`s model except some numerical dispersion of the GRT model (Figure 12).

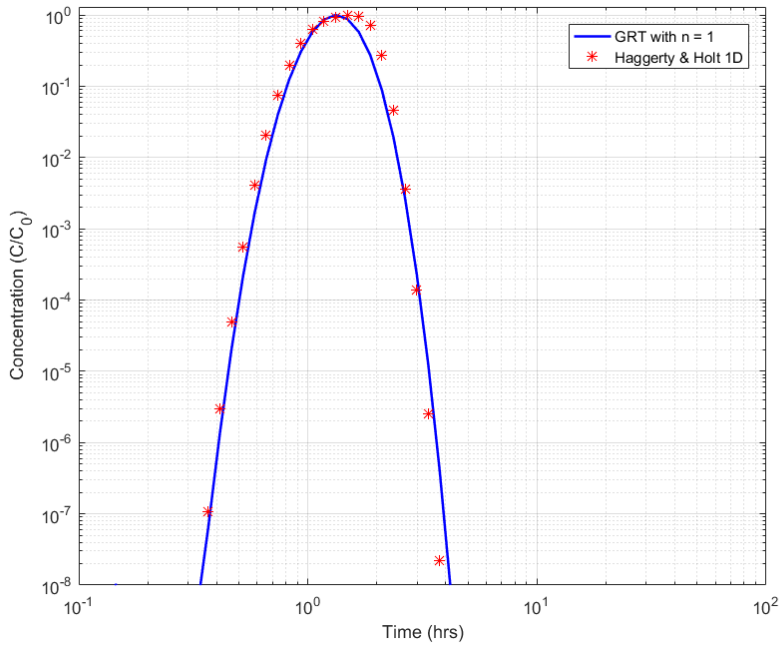

Figure 12. GRT (n =1) verification with 1D Haggerty and Holt (1995) Model, Pulse Type Source

3.2. Verification with Radial Model

The GRT model with flow dimension 2 which transforms the GRT model into a Radial transport model was verified against the Haggerty et. al. (2000) multi-rate, double-porosity semi-analytical radial transport model. Boundary conditions (equation 27a, 27b, 56a and 56b) for injection, resting and pumping phase solution, and initial condition for the GRT and the semi-analytical radial model were same for the pulse type of tracer injection. The GRT model and the semi-analytical radial model were simulated with the similar input parameters both for transport without diffusion and for transport with multi-rate, double porosity (Appendix B1). Diffusive

porosity was set to zero in both models which make both models simulation for only the advective transport and without diffusion. Breakthrough curves from the GRT model match with the semi-analytical radial model both for transport without diffusion and transport with multi-rate, double porosity in term of peak concentration arrivals (Figure 13, 14). Comparing with the semi-analytical radial model the breakthrough curve from the GRT model for transport with multi-rate, double porosity showed some numerical dispersion for later time concentration arrivals (Figure 14)

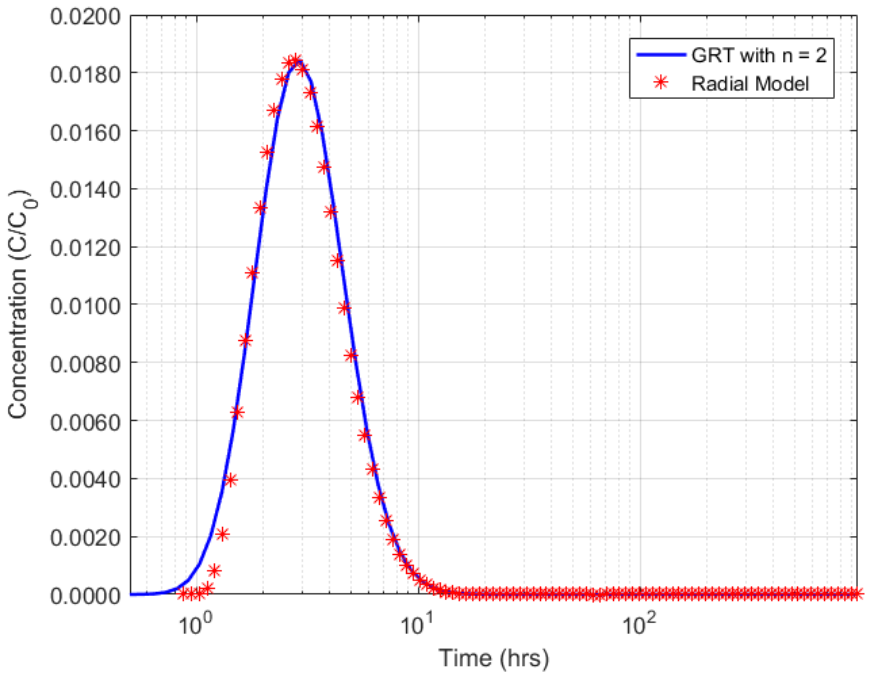

Figure 13. GRT (n =2) verification with Haggerty (1998) Model, transport without diffusion and Pulse Type Source

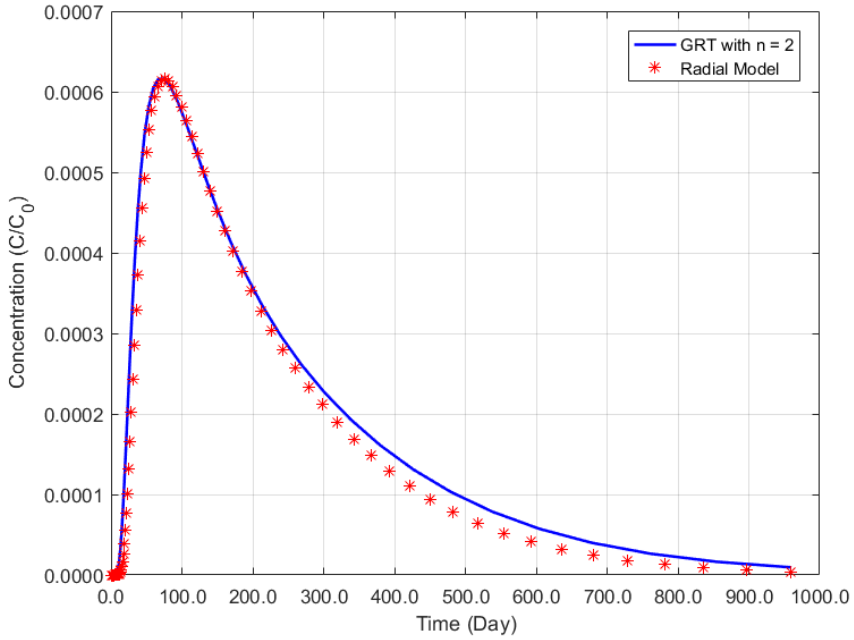

Figure 14. GRT (n =2) verification with Haggerty (1998) Model, multi-Rate Transport and Pulse Type Source

3.3. Verification with Spherical Model

Schroth and Istok (2005) spherical-flow solution for a for single-well injection-withdrawal test was selected to verify developed multi-rate, double-porosity GRT model. Flow dimension was set to 3 for the GRT model. The Schroth and Istok (2005) model was for single-well injection-withdrawal test. The GRT was transformed changing the location of pumping well to the injection well to mimic Schroth and Istok (2005) model.

Boundary conditions (equation 27a, 27b, 56a and 56b) and initial condition were same for the GRT and the Schroth and Istok (2005) model. Simulating both of the model using same data in table 2, breakthrough curve form the GRT model showed very close match with that of the Schroth and Istok (2005) model excepts some minor numerical dispersion (Figure 15).

Table 2. Parameters for GRT model verification with spherical solution with pulse type source

Parameters	Values
Time, t	1000 hrs
Advective Porosity, ϕ_f	0.1
Diffusive Porosity, ϕ_m	0
Radial Distance, r	3.29 m
Longitudinal Dispersivity, α_L	0.02 m
Injection Rate, Qinj	0.5
Pumping Rate, Qext	0.5
Δr	0.01

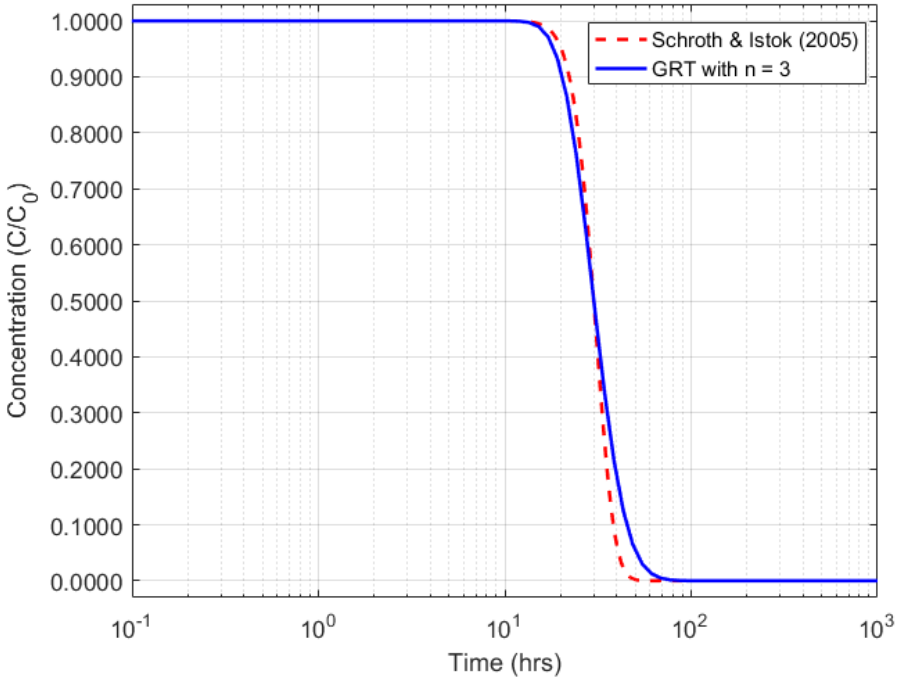

Figure 15. GRT (n =3) verification with Scorth and Istok (2005) Model

Comparing with 1D, radial and spherical transport model, it can be surmised that the multi-rate, double-porosity GRT model can be used for transport modeling with any dimension with a satisfactory level of confidence. Some numerical dispersion of the GRT model is the consequence of large value spatial discretization of finite grid blocks. The model was simulated

using very fine discretization (10^{-5}) to verify whether dispersion as a consequence of larger grid size. Verification showed that numerical dispersion can be reduced by decreasing the size of grid blocks and consequently model will take more time to run.

CHAPTER 4

GRT MODEL SENSITIVITY

The sensitivity and behavior of the GRT model were investigated for varying flow dimension, Peclet number, and capacity coeffient for single-porosity, single-rate double-porosity and multi-rate double porosity transport. A general input parameters file (Appendix B2) was used and parameters of interest were varying to analyze sensitivity.

4.1. Sensitivity to Flow Dimensions

The GRT model collapses to the 1D, radial, and spherical advection-dispersion equation (ADE) for integer flow dimensions, n of 1, 2, and 3 respectively (Figure 1). The GRT model also transforms to a sub-linear, sub-radial, sub-spherical and super-spherical advection dispersion equation (ADE) for non-integer flow dimension, n of $0 < n < 1$, $1 < n < 2$, $2 < n < 3$ and $n > 3$, respectively (Figure 1). In this section, effects of integer and non-integer flow dimension, n of 1, 1.5, 2, 2.5, and 3 on transport have been examined. Larger flow dimensions result in slower transport, and the transport process approaches Local Equilibrium Condition (LEC). The LEC occurs when the characteristic time for diffusion is much smaller than the characteristic time for advection, so diffusion appears to occur instantaneously (Holt, 1997). At the LEC, transport is physically retarded as solutes repeatedly diffuse into and out of fracture-bounded blocks. Resulting breakthrough curves display single-porosity behavior, with all of the porosity (both advective and diffusive porosity) impacting peak arrivals.

4.1.1. Single-Porosity Transport

To simulate single-porosity transport, the total capacity coefficient of the aquifer was set to zero by setting diffusive porosity zero which transformed the multi-rate, double-porosity GRT model to a single-porosity GRT model.

4.1.1.1 Continuous Source

As the flow dimension increases, the average velocity of the groundwater is reduced, due to larger cross-sectional area for flow, and mean arrival of the solute is delayed (Figure 16). The BTC for the flow dimension 1 in the Figure 16 shows that maximum concentration of tracer arrives at 1.5 hrs for flow dimension 1, whereas the arrival time is 40 hrs for flow.

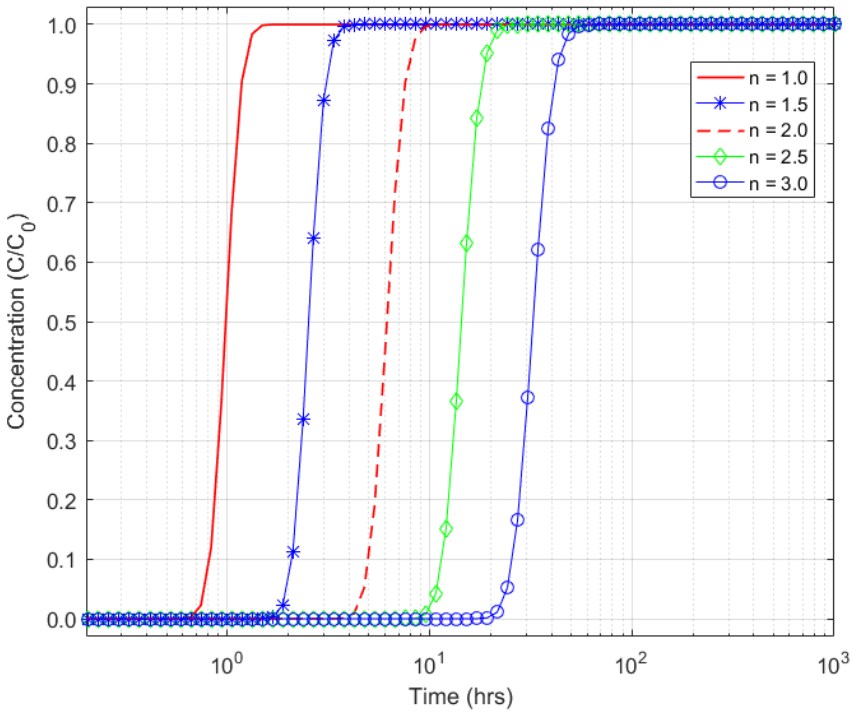

Figure 16. BTCs for different Flow Dimension, without Diffusion and Continuous Source

4.1.1.2. Pulse Type Source

Pulse Type Source breakthrough curves (BTC) (Figure 17) also show increasing peak arrival times with increasing flow dimension (section 4.1.1.1). The BTC for the flow dimension 1 in the figure 17 shows sharp peak while the BTC for flow dimension 3 shows relatively a flat peak. Tracer concentration was diluted in the larger pore space due to larger flow conduit (Figure 1) for flow dimension 3 than that of flow dimension 1 that is the reason of the flat peak.

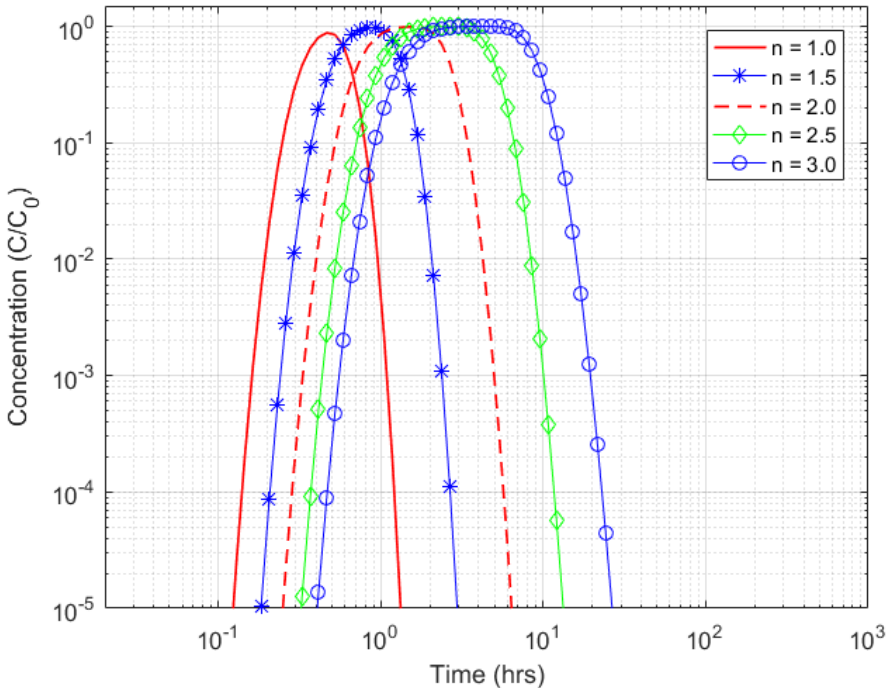

Figure 17. BTCs for different Flow Dimension, without Diffusion and Pulse Type Source

4.1.2. Single-Rate, Double-Porosity Transport

Single-rate, double-porosity transport occurs when all fracture-bounded blocks are of equal size and have the same porosity and retardation coefficient. The capacity coefficient was set to 5 for all of the following simulations. As before, the cross-sectional area for flow increases with flow dimension, decreasing the average velocity and increasing the advective

travel time. As the advective travel time increases, solutes have more time to diffuse into, and out of the fracture-bounded blocks, and the transport processes move closer to the LEC.

4.1.2.1 Continuous Source

Figure 18 shows BTCs for varying flow dimension. As the flow-dimension increases the characteristic time for advective transport increases relative to the characteristic time for diffusion, and the resulting BTCs show the transition from clear double-porosity behavior to apparent single-porosity behavior at the LEC.

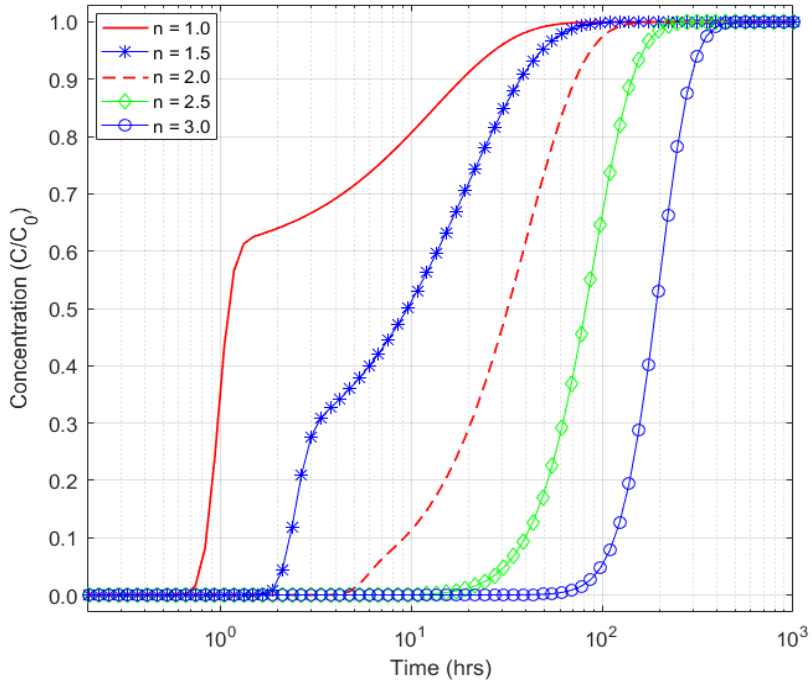

Figure 18. BTCs for different Flow Dimension, Single-Rate Diffusion and Continuous Source

Compare to the BTC of flow dimension 3 in Figure 16, where maximum concentration comes at 40 hrs, with the BTC of flow dimension 3 in Figure 18, where maximum concentration comes at 400 hrs. The delay in the arrival time of the maximum concentration reflects the physical retardation of the BTC at the LEC.

4.1.2.2. Pulse Type Source

BTCs from a pulse type source (Figure 19) also show increasing physical retardation and more impact from diffusive mass transport as the flow dimension increases. Peak arrival times are delayed with increasing flow dimension, as the cross-sectional area of flow increases.

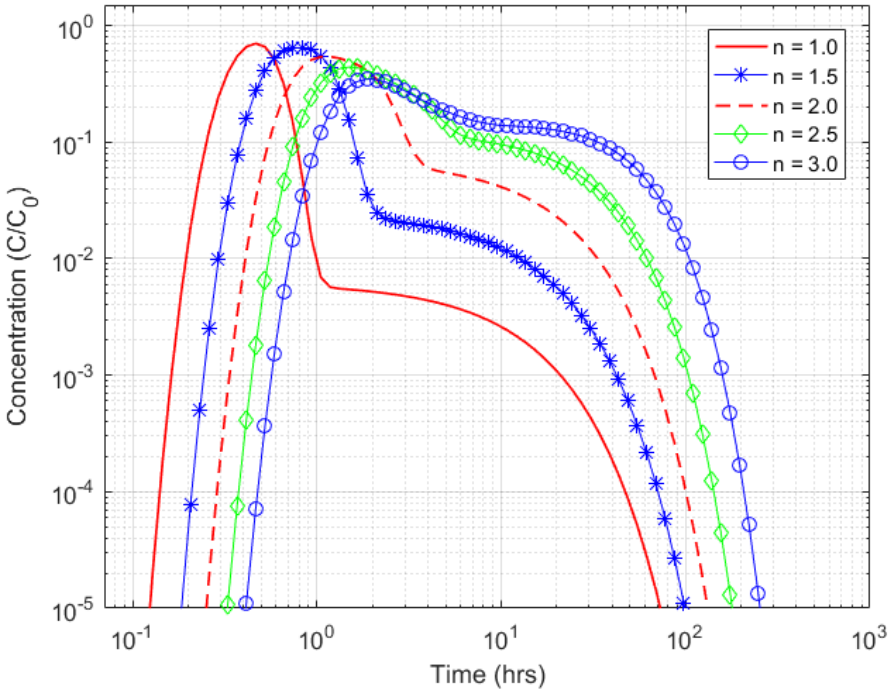

Figure 19. BTC for different Flow Dimension, Single-Rate Diffusion and Pulse Type Source

4.1.3. Multi-Rate, Double-Porosity Transport

Parameters for the following multi-rate, double porosity transport simulations are shown in Table 3. Multi-rate, double-porosity transport differs from single-rate double-porosity transport because there is a distribution of mass-transfer coefficients and capacity coefficients. Because there is a wide range of values for the mass-transfer and capacity coefficients, the double-porosity response shown in the BTCs below is more muted that that of single-rate, double-porosity transport.

Table 3. Parameters of the Multi-Rate, Double-Porosity GRT model

Parameters	Values	units
Injection rate, Q_{in}	0.5	m^3/s
Pumping rate, Q_{out}	1.0	m^3/s
Longitudinal Dispersivity, α_L	0.2	m
Δr	0.01	m
Saturated thickness of aquifer at the injection well, b_1	5	m
Saturated thickness of aquifer at the pumping well, b_1	5	m
Advective Porosity, ϕ_f	0.01	-
Diffusive Porosity, ϕ_m	0.05	-
Log of the geometric mean of the diffusion rate coefficients, μ^*	-6.9078	-
Standard deviation of the log-transformed diffusion rate coefficient, σ	0	-
Free water aqueous diffusion coefficient of solute. D_{aq}	2.62×10^{-6}	m^2/s

BTCs (Figures 20 and 21) from multi-rate, double porosity transport are smoother than those for single-rate, double-porosity transport. As before, increasing the flow dimension leads to slower mean-arrival times and greater influence of diffusive mass transfer (The BTCs move closer to the LEC).

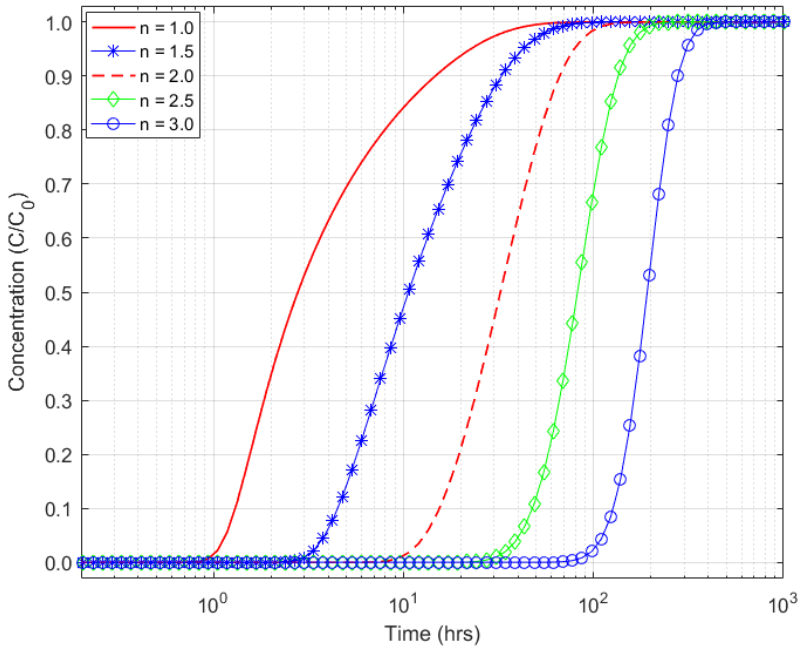

Figure 20. BTC for different Flow Dimension, Multi-Rate Diffusion and Continuous Source

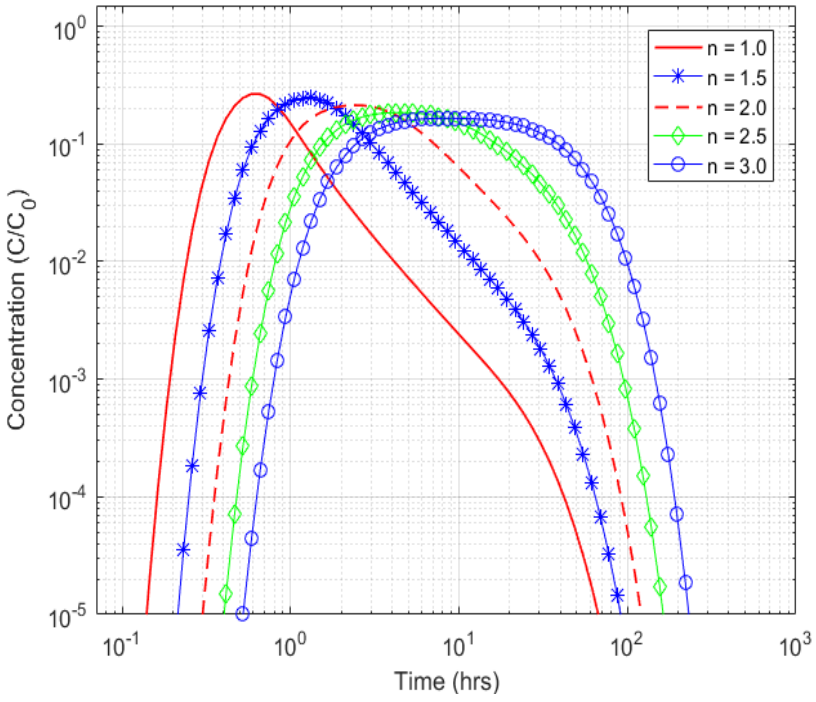

Figure 21. BTC for different Flow Dimension, Multi-Rate Diffusion and Pulse Type Source

4.1.4. Comparison of Single Porosity and Double Porosity Models

Here, single porosity and double porosity models are compared. Figures 22-27 show BTCs for continuous and pulse type sources for flow dimensions of 1, 2, and 3. The average velocity decreases as the flow dimension increases, causing BTCs to move closer to the LEC. Single-rate models show a sharp change in slope due to the presence of only one diffusion rate, while multi-rate models show smoother BTCs reflecting diffusion into a variety of sizes of fracture bounded blocks.

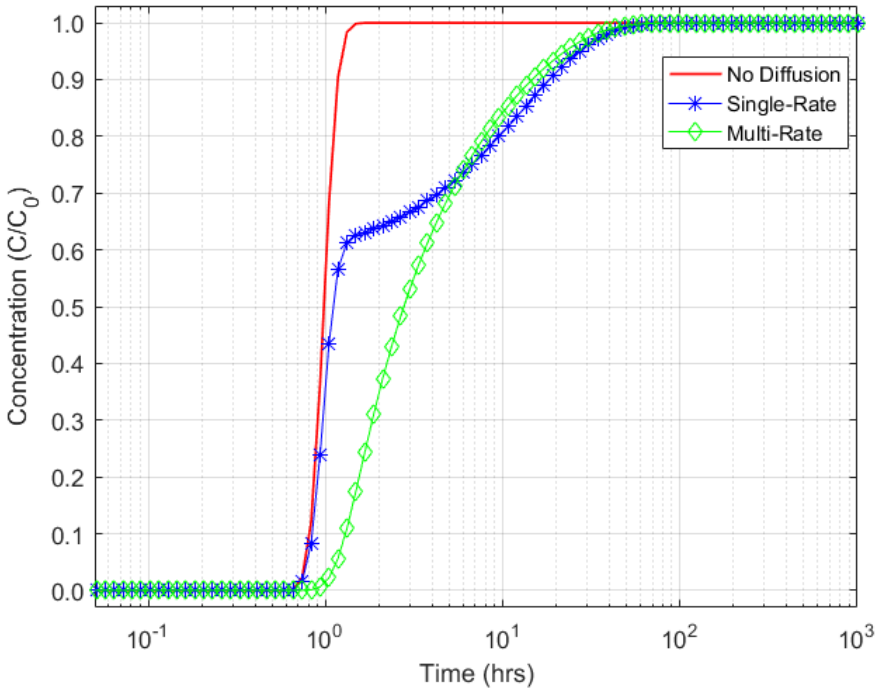

Figure 22. BTC for different types of transport with n = 1 and Continuous Source.

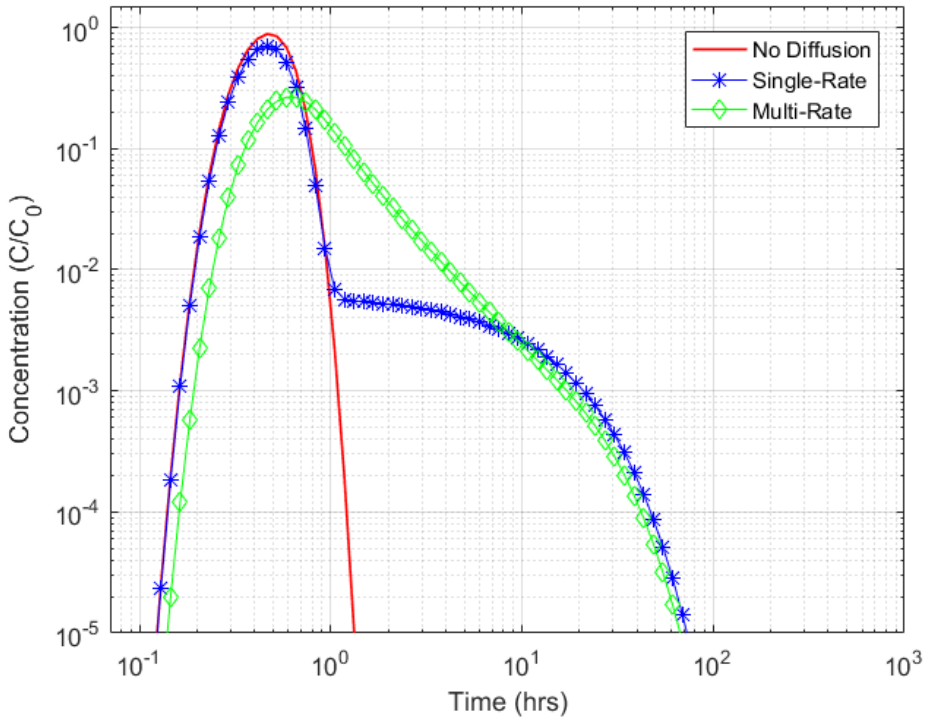

Figure 23. BTC for different types of transport with n = 1 and Pulse Type Source.

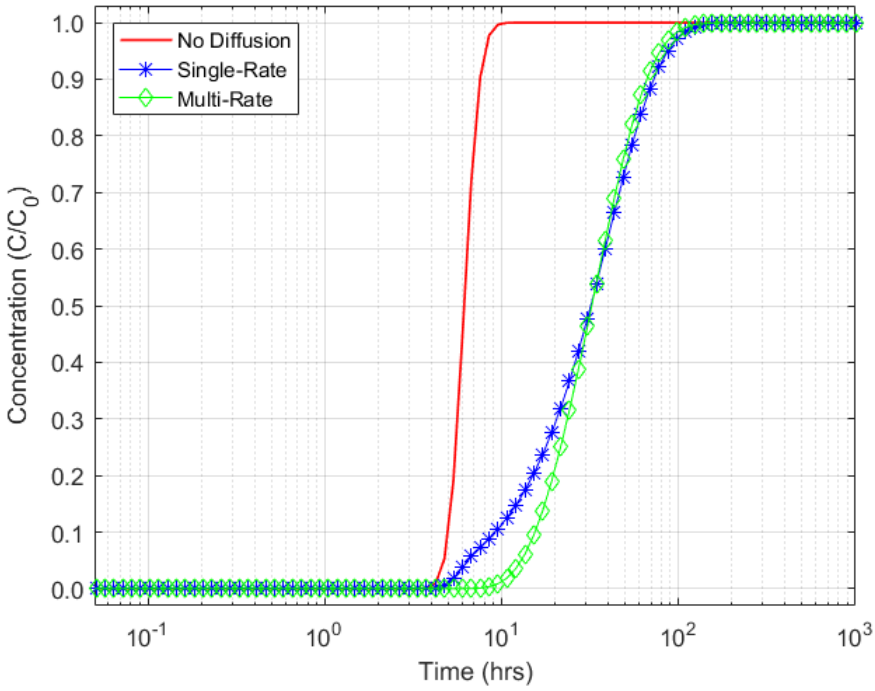

Figure 24. BTC for different types of transport with n = 2 and Continuous Source.

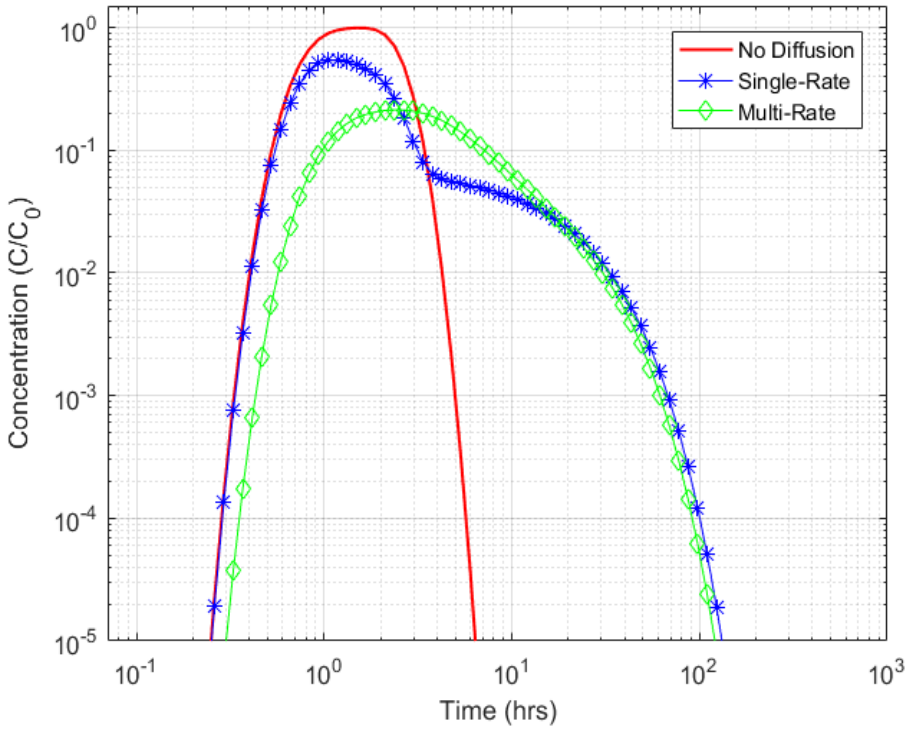

Figure 25. BTC for different types of transport with n = 2 and Pulse Type Source.

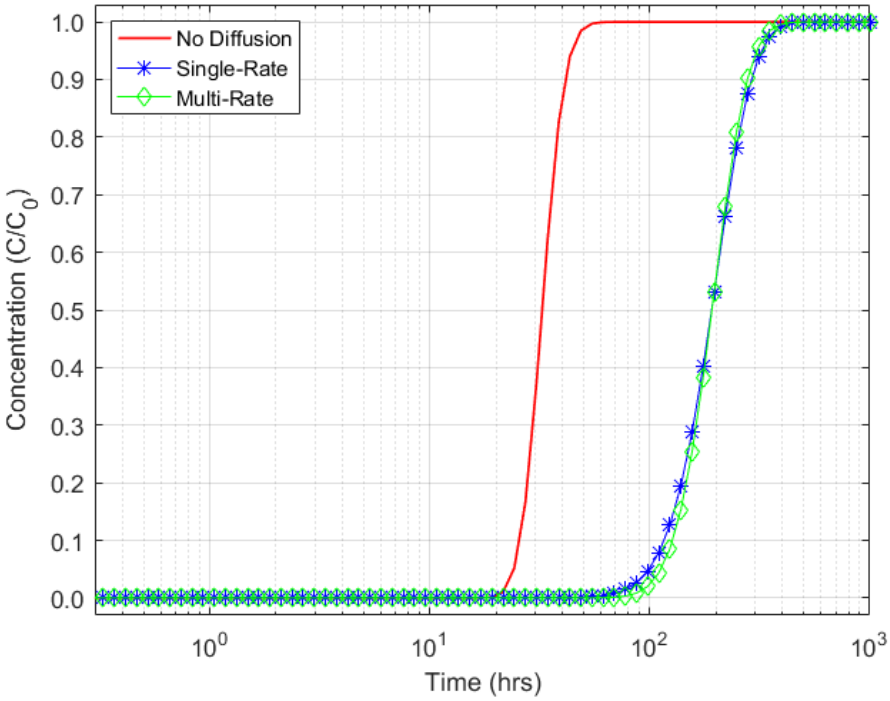

Figure 26. BTC for different types of transport with n = 3 and Continuous Source.

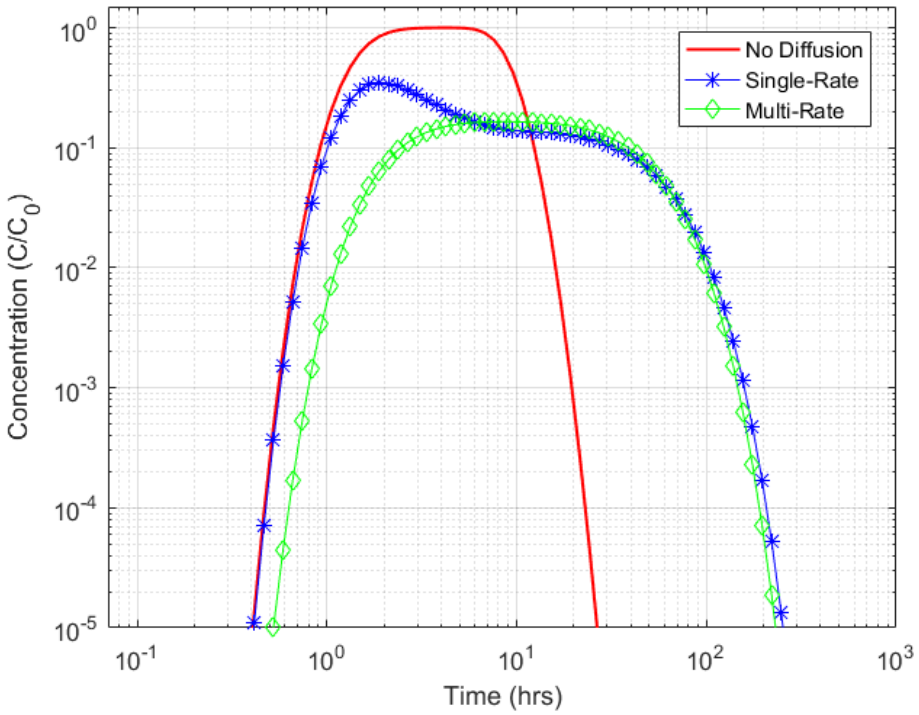

Figure 27. BTC for different types of transport with n = 3 and Pulse Type Source.

4.2. Sensitivity to Peclet Numbers

Peclet number (Pe) for the transport process is the ratio of the travel distance to longitudinal dispersivity (equation 57h). Peclet number can lead more spreading of tracer into the aquifer and can changes advective and or diffusive transport behavior. Smaller Pe values and higher flow dimension may lead to transport under LEA. The distance between the injection and pumping well is 10 m. The Peclet number is smaller means longitudinal dispersivity is higher. In this section, sensitivity and behavior of the GRT model were investigated with for flow dimensions 1, 2 and 3 for transport without diffusion, single-rate and multi-rate transport varying Peclet number from 0.1 to 100 with parameters from Appendix B1

4.2.1. Single-Porosity Transport

4.2.1.1 Continuous Source

The BTC (Figure 28) with Pe 0.1 and flow dimension 1 shows the tracer has already reached the pumping well before pumping by dispersion. The curve with Pe value 100 (Figure 28) shows tracer did not spread out and 50% tracer comes at 1 hr. All other BTCs (Figure 28) show some degree of spreading due to lesser Pe values. And, BTCs for the flow dimension 2 and 3, Figure 29 and 30 respectively show the same trend of spreading for smaller Pe values except delaying tracer arrival due to increasing flow dimensions.

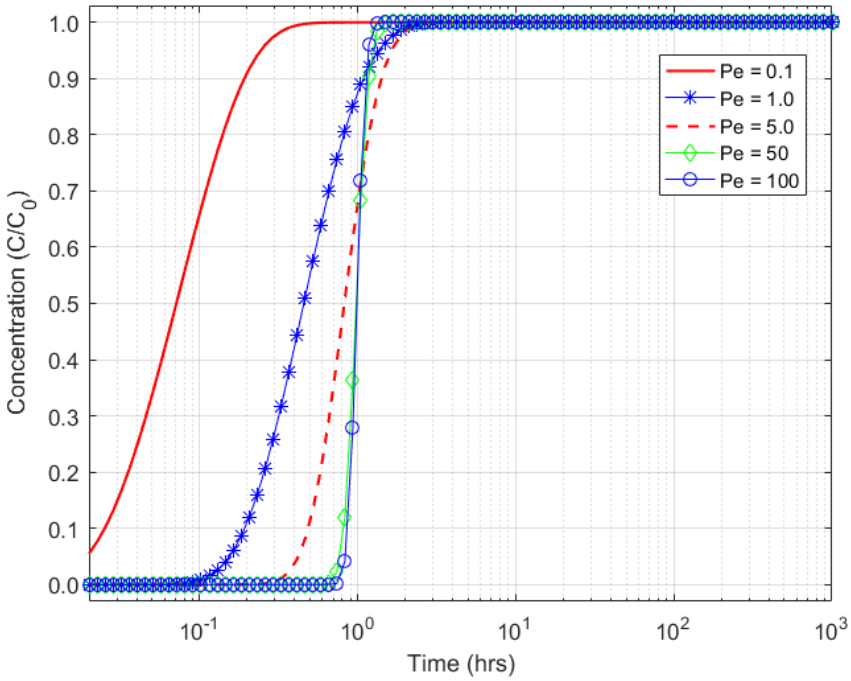

Figure 28. BTC for different Peclet Numbers with n = 1, no diffusion and Continuous Source

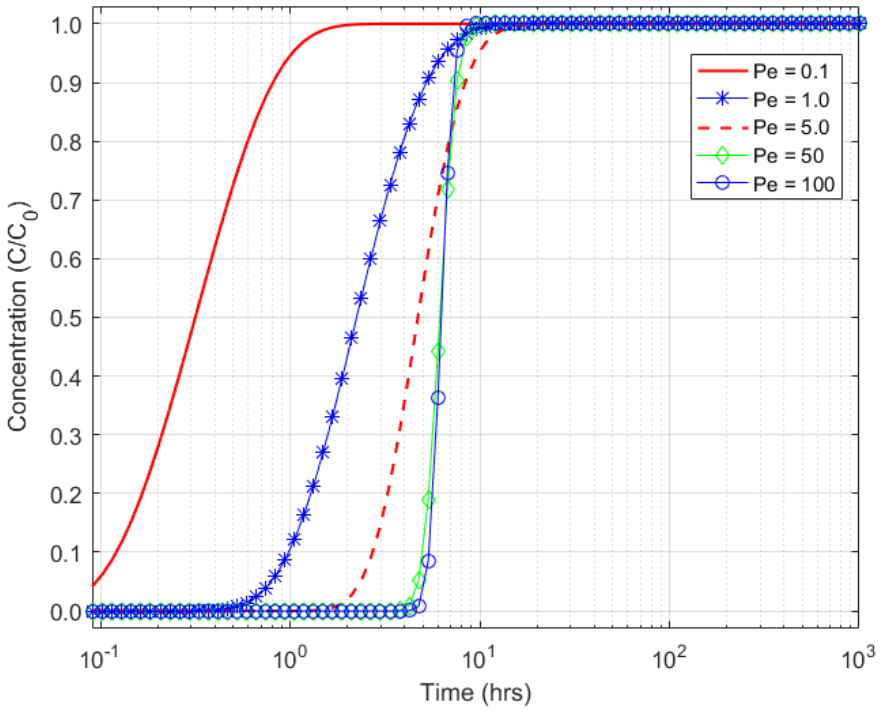

Figure 29. BTC for different Peclet Numbers with n = 2, no diffusion and Continuous Source

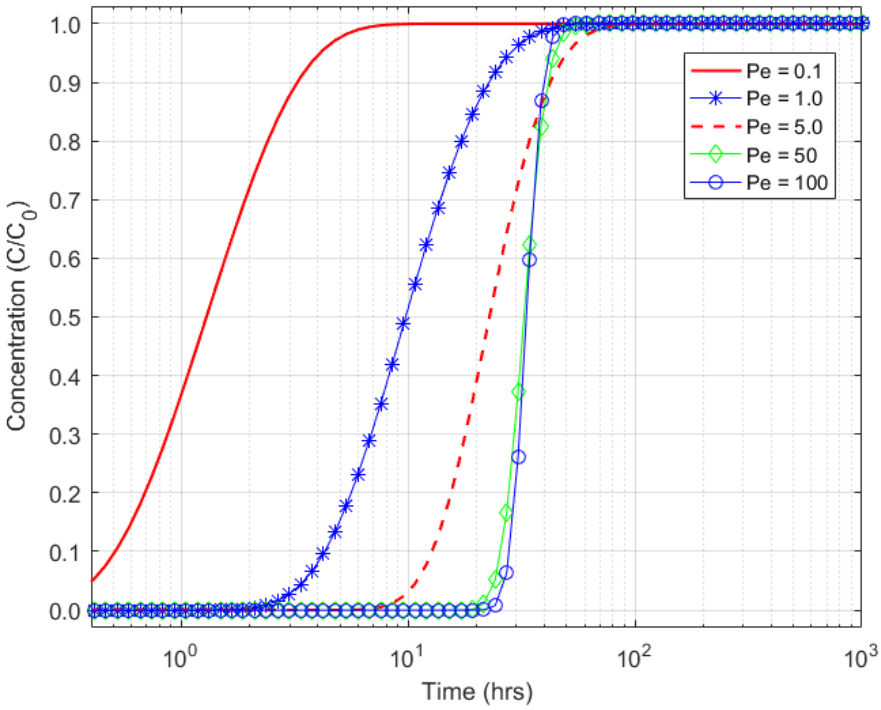

Figure 30. BTC for different Peclet Numbers with n = 3, no diffusion and Continuous Source

4.2.1.2. Pulse Type Source

Similar types of behavior mentioned in sub-section 4.2.1.1 can be seen in BTCs (Figure 31, 32, 33) for pulse type source with flow dimensions 1, 2 and 3. Tracer arrivals time increases with increasing Pe values (Figure 31, 32, 33). BTCs (Figure 33) with higher Pe 50 and 100 show a flat peak because tracer did not spread out due to Pe values effects. The BTCs with the Pe value 0.1 shows little bump either before or after the peak concentration arrival due to transport domination changes from the effects of very smaller Peclet number to advection domination (Figure 31, 32, 33).

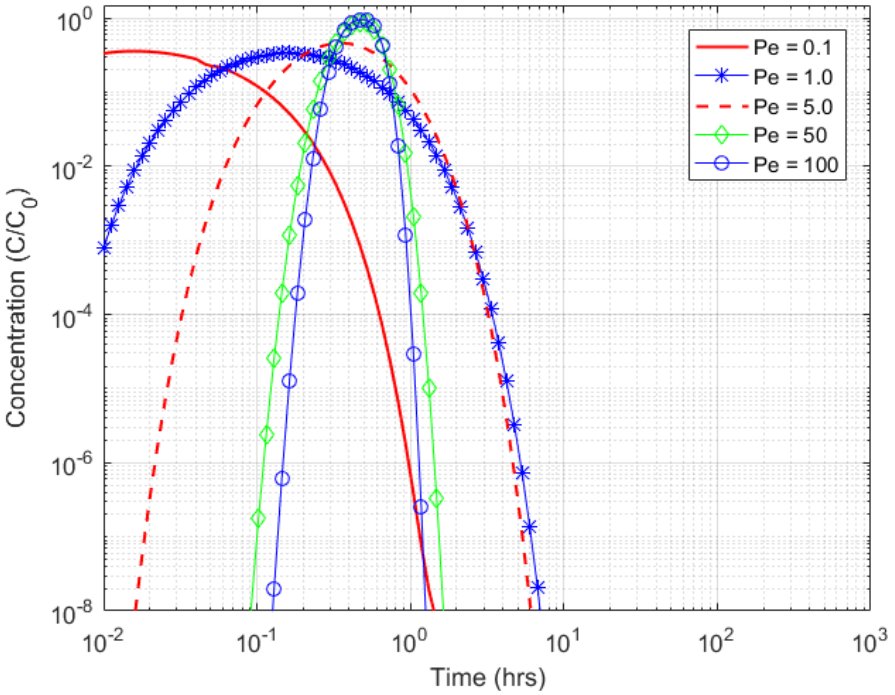

Figure 31. BTC for different Peclet Numbers with n = 1, no diffusion and Pulse Type Source

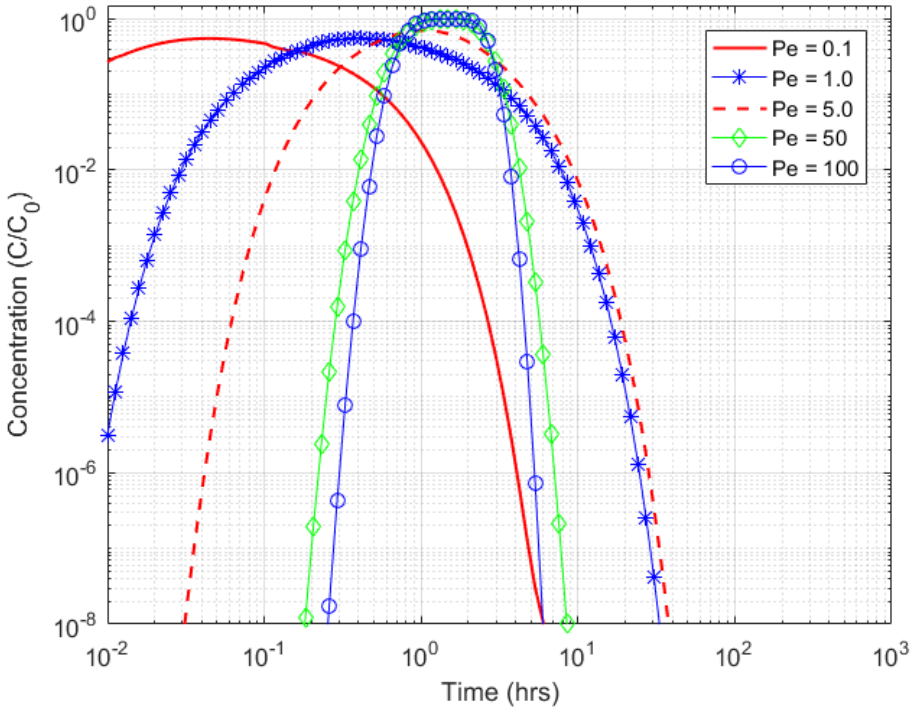

Figure 32. BTC for different Peclet Numbers with n = 2, no diffusion and Pulse Type Source

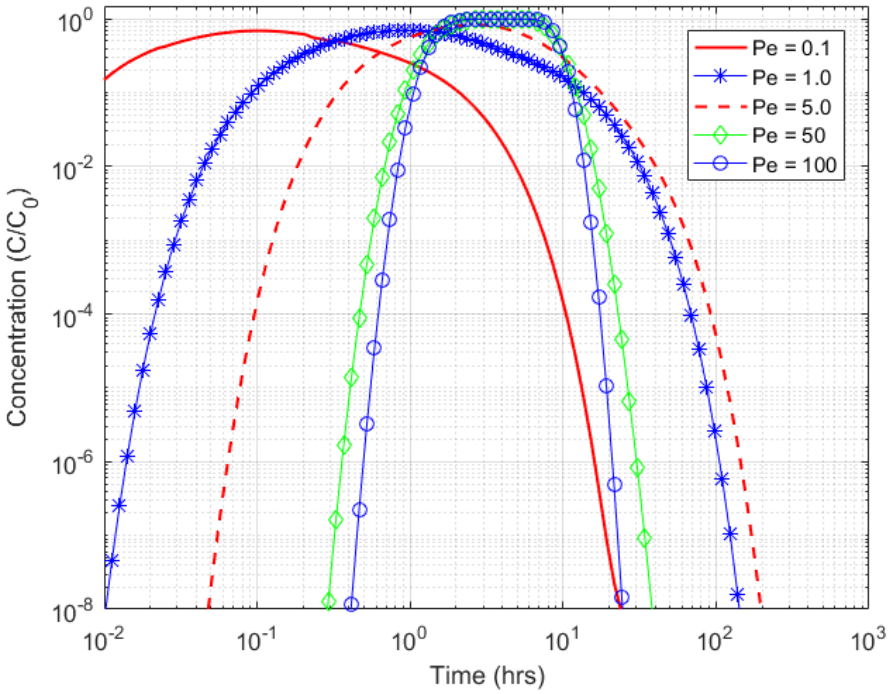

Figure 33. BTC for different Peclet Numbers with n = 3, no diffusion and Pulse Type Source

4.2.2. Single-Rate, Double-Porosity Transport

4.2.2.1 Continuous Source

BTCs (Figure 34) of flow dimension 1 with single-rate transport with continuous source show both effects of Pe values, advection and diffusion. The BTC with Pe value 0.1 (Figure 34) shows effect of longitudinal dispersivity, diffusion and advection at very early time and later diffusion become dominant. Advection becomes dominant for the Pe value 100 at early time where 50% concentration comes at 1 hr and later diffusion dominates transport. BTC with the Pe value 0.1(Figure 35) shows the effects of wider conduit for flow the flow dimension 2 hence concentration dilution, and diffusion. And, tracer behave like more diffusive with increasing Pe values for flow dimension 2 (Figure 35). When the flow dimension is 3 and Pe values are higher than 50, BTCs (Figure 36) show advection dominates transport process at early time.

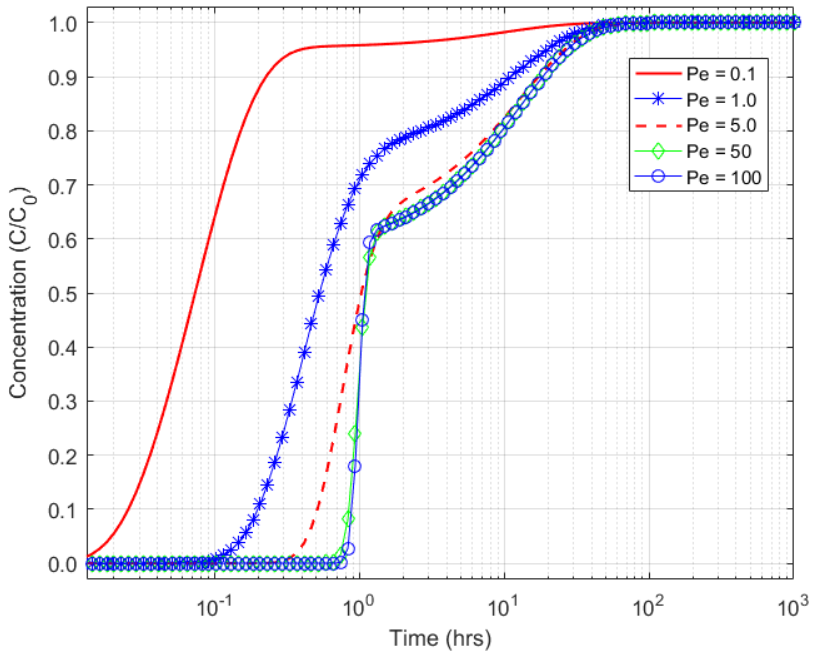

Figure 34. BTC for different Peclet Numbers with n = 1, Single-Rate diffusion and Continuous Source

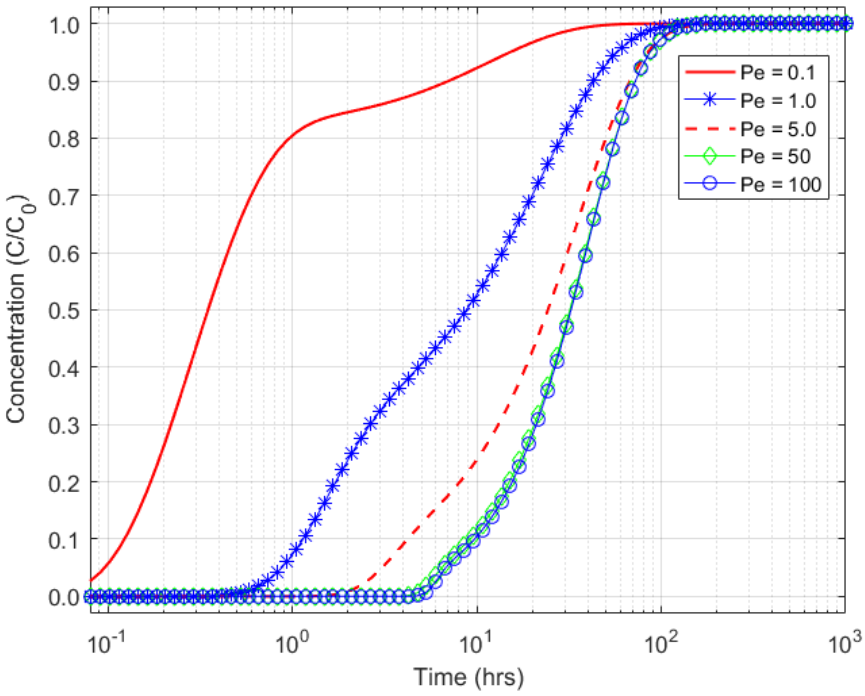

Figure 35. BTC for different Peclet Numbers with n = 2, Single-Rate diffusion and Continuous Source

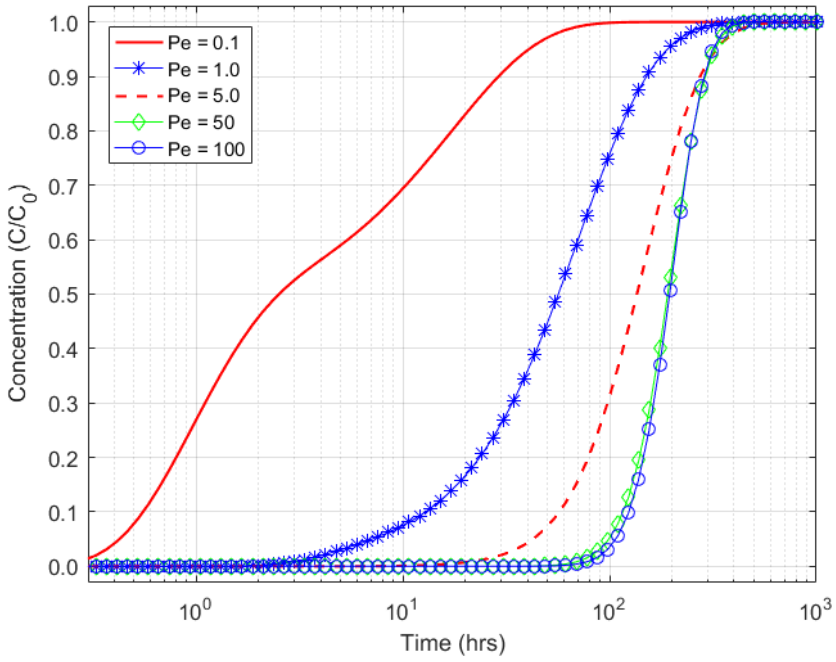

Figure 36. BTC for different Peclet Numbers with n = 3, Single-Rate diffusion and Continuous Source

4.2.2.2. Pulse Type Source

Pulse type source with varying Pe value exhibits effect of advection, diffusion and flow dimension more clearly than continuous source injection. BTCs with single-rate diffusion and flow dimension 1 (Figure 37) show advection dominate transport and with increasing Pe values tracer becomes less spread out. When the flow dimension increases from 1 to 2, advection dominates transport and later diffusion dominates with increasing Pe values (Figure 38). BTCs from transport with single-rate and flow dimension 3 (Figure 38) show more effects of diffusion with increasing Pe values. For all flow dimensions, tracer arrival time increases with increasing Pe values (Figure 37, 38, 39).

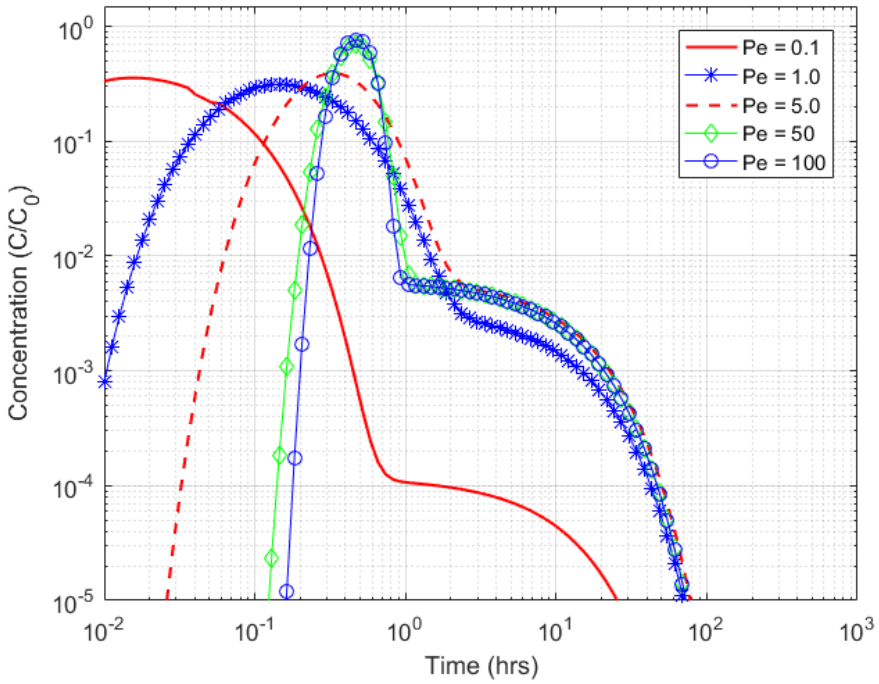

Figure 37. BTC for different Peclet Numbers with n = 1, Single-Rate diffusion and Pulse Type Source

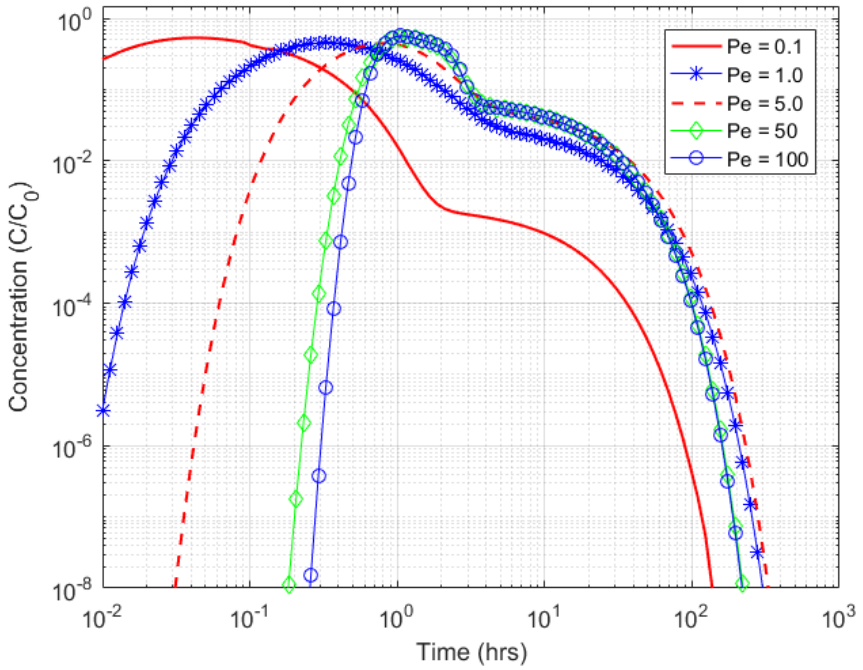

Figure 38. BTC for different Peclet Numbers with n = 2, Single-Rate diffusion and Pulse Type Source

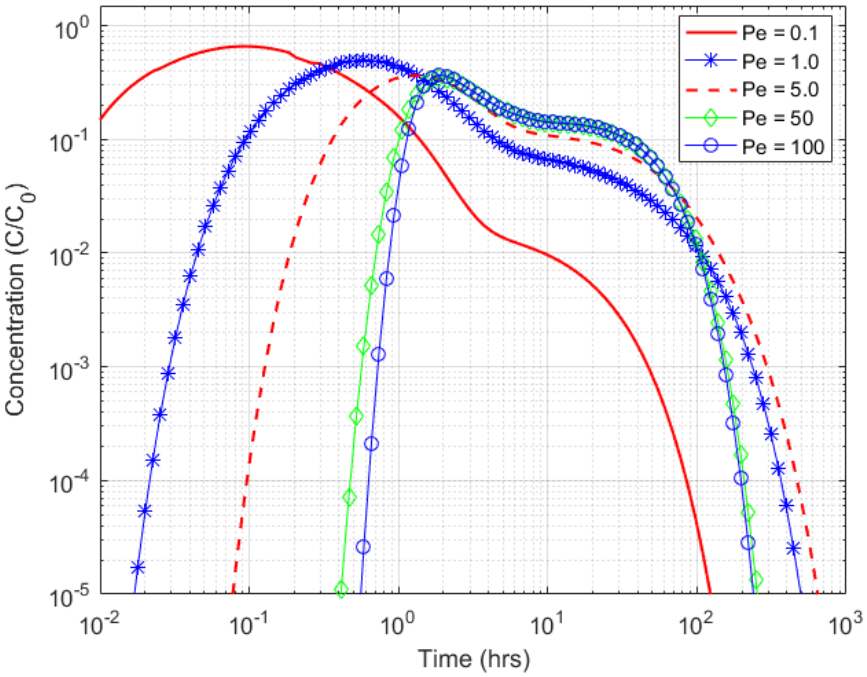

Figure 39. BTC for different Peclet Numbers with n = 3, Single-Rate diffusion and Pulse Type Source

4.2.3. Multi-Rate, Double Porosity Transport

4.2.3.1 Continuous Source

Transport with multi-rate diffusion for continuous source varying Pe values and flow dimension 1, 2, 3 (Figure 40, 41, 42) shows smooth effect of diffusion that compare to the single-rate transport. Tracer arrival times increases with increasing flow dimensions. BTCs with higher Pe value exhibit advection dominate transport for all flow dimensions (Figure 40, 41, 42) and transport approaches to LEA condition.

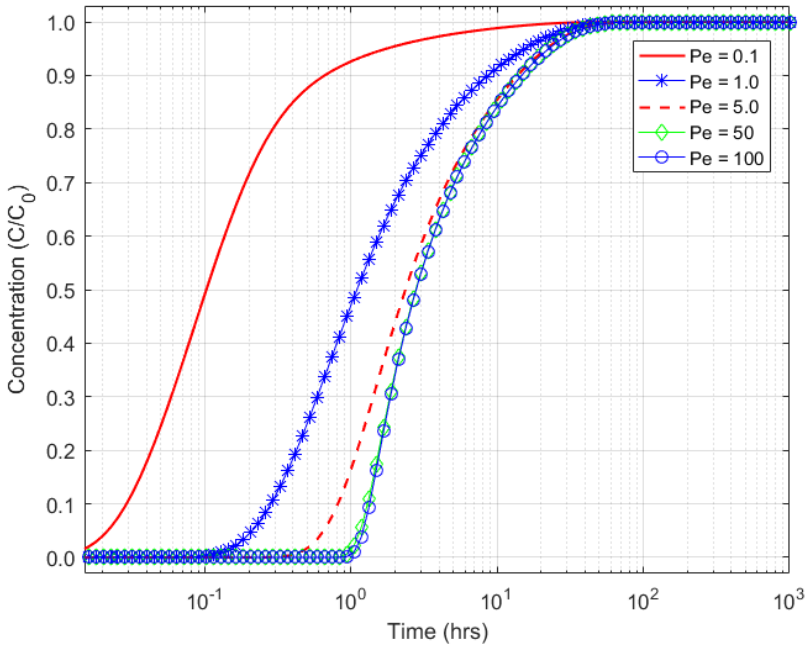

Figure 40. BTC for different Peclet Numbers with n = 1, Multi-Rate diffusion and Continuous Source

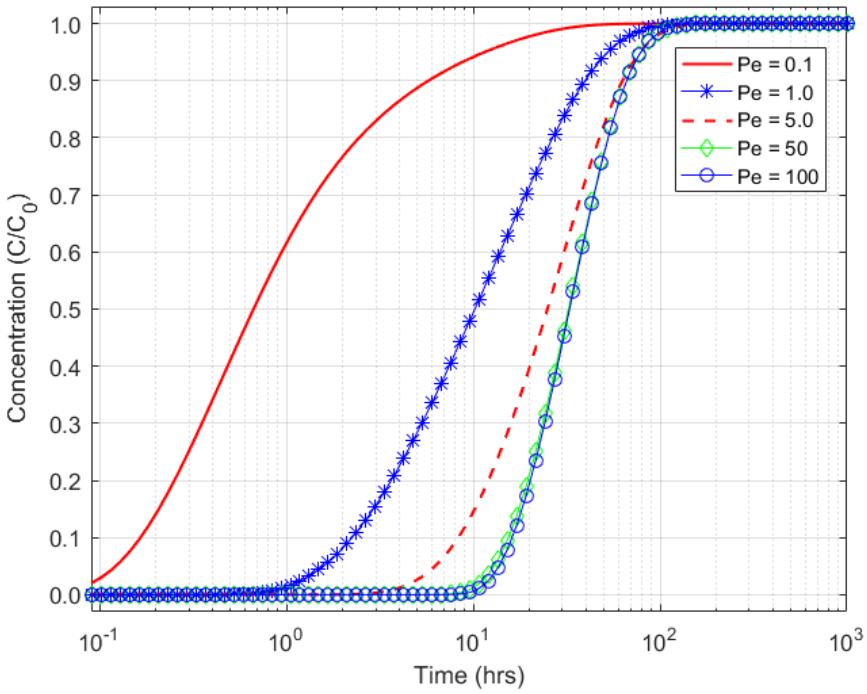

Figure 41. BTC for different Peclet Numbers with n = 2, Multi-Rate diffusion and Continuous Source

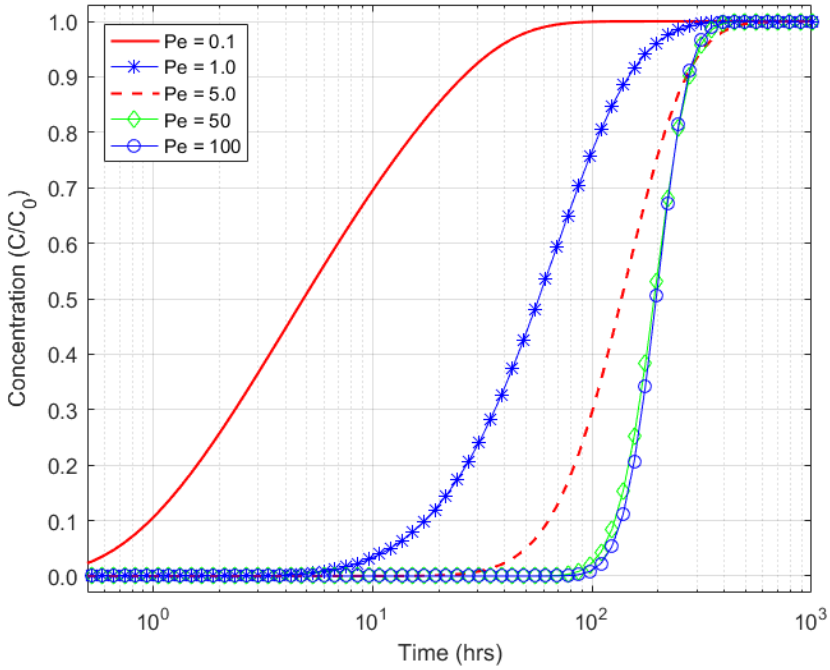

Figure 42. BTC for different Peclet Numbers with n = 3, Multi-Rate diffusion and Continuous Source

4.2.3.2. Pulse Type Source

Like in the sub-section 4.2.3.1, pulse type source and multi-rate, double porosity transport also shows similar behavior with increasing Pe values. BTCs (Figure 43, 44, 45) for the pulse type source show with increasing Pe values tracers spread less in aquifer and smooth changes in BTCs from advection domination to diffusion domination transport.

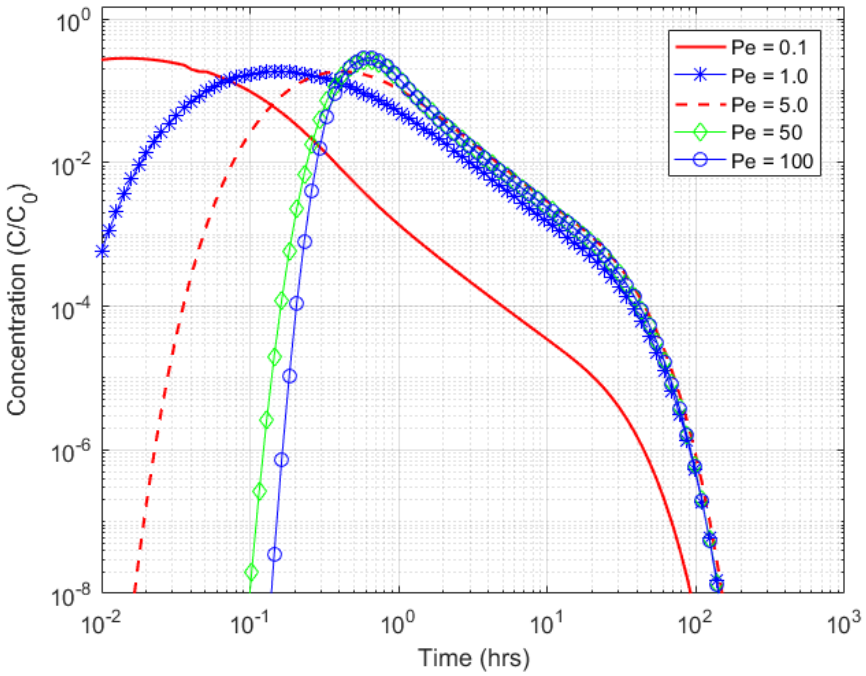

Figure 43. BTC for different Peclet Numbers with n = 1, Multi-Rate diffusion and Pulse Type Source

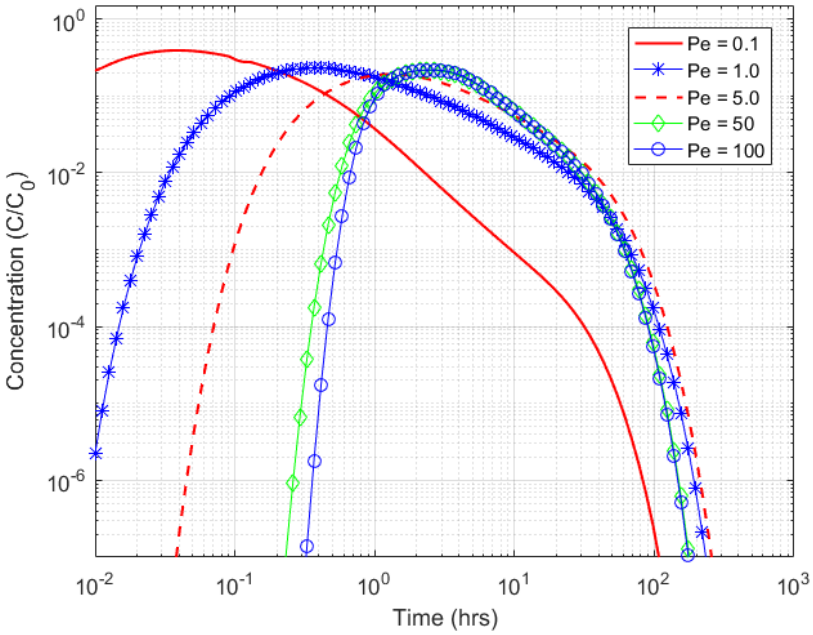

Figure 44. BTC for different Peclet Numbers with n = 2, Multi-Rate diffusion and Pulse Type Source

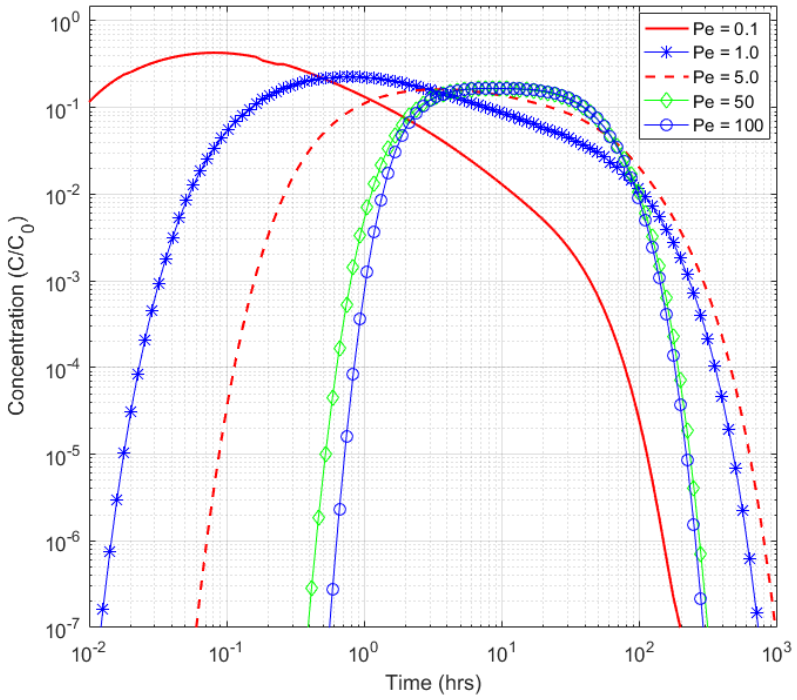

Figure 45. BTC for different Peclet Numbers with n = 3, Multi-Rate diffusion and Pulse Type Source

4.3. Sensitivity to Capacity Coefficient

Sensitivity and behavior of the multi-rate, double-porosity GRT model with pulse type source was investigated for the total capacity coefficient (Beta) with parameter in Appendix B2. In this section, Pe and diffusive porosity were set to 10 and 0.05 respectively. As Beta increases, more solute mass can be stored in the diffusive porosity, creating a larger sink for diffusing solutes, and this leads to later peak arrival times and more impact from diffusive mass transfer (Figures 46, 47, and 48). As the flow dimension increases, the cross-sectional area of flow increases, and the BTCs move closer to the LEC.

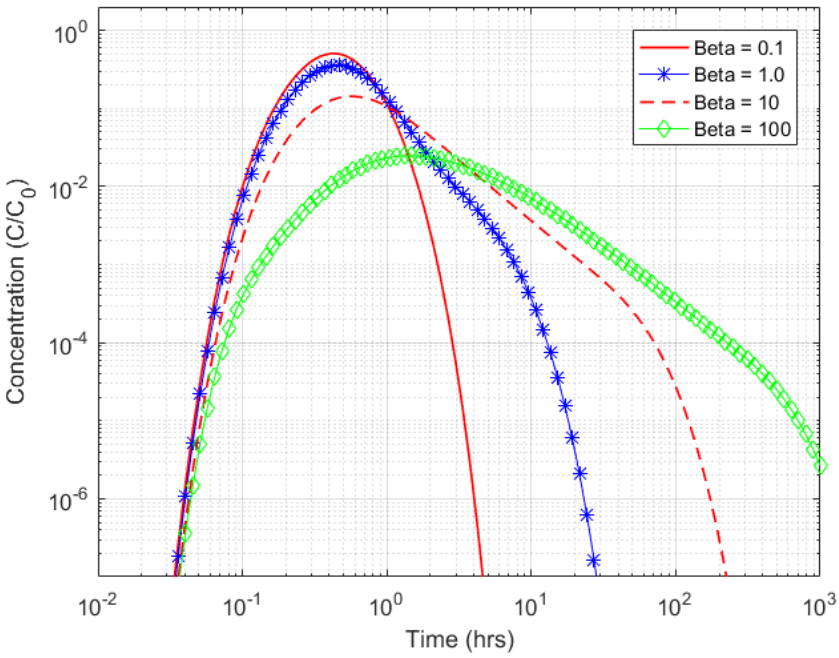

Figure 46. BTCs varying Capacity Coefficients with n = 1, Multi-Rate diffusion and Pulse Type Source

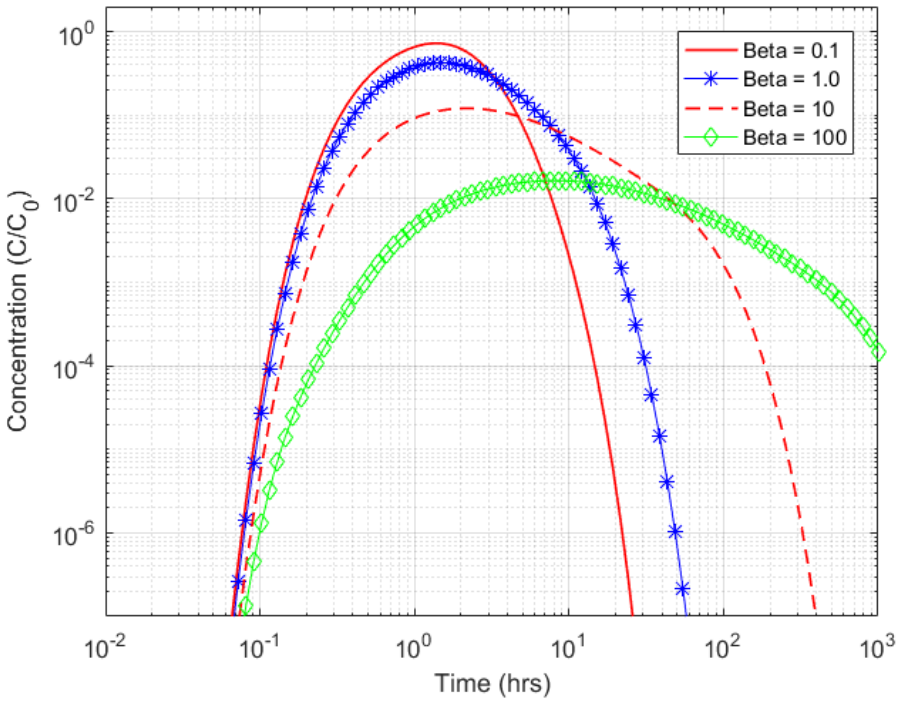

Figure 47. BTCs varying Capacity Coefficients with n = 2, Multi-Rate diffusion and Pulse Type Source

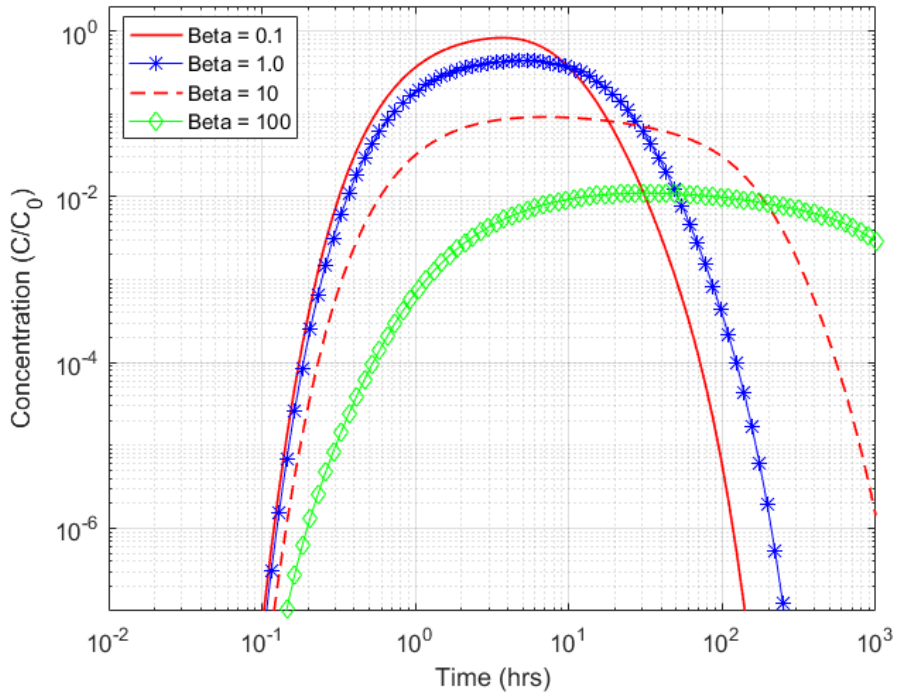

Figure 48. BTCs varying Capacity Coefficients with n = 3, Multi-Rate diffusion and Pulse Type Source

4.3. Generic Comparison with Radial Flow Transport

Generic comparison was done to examine whether radial flow underestimates the peak concentration, allow more time for diffusion, slowdown the flow velocity in breakthrough curve (BTC). The BTCs of transport without diffusion show that radial flow (BTC with n = 2) takes more time for peak concentration arrival (Figure 49) than sub-radial transport (BTC with n = 1.7) due to larger flow conduit. In the case of single-rate (Figure 50) or multi-rate (Figure 51) diffusion transport, radial flow underestimates peak concentration, takes more time for peak arrival and leads more diffusion into matrix blocks. Sub-spherical transport even takes more time than radial flow, allows more time for diffusion and underestimates the peak concentration In comparison to radial transport, sub-radial transport leads to higher velocities, much earlier arrival times, and higher peak concentrations in breakthrough curves. Faster advective transport leads to less diffusion into fracture-bounded matrix blocks and steeper slopes of late time concentrations.

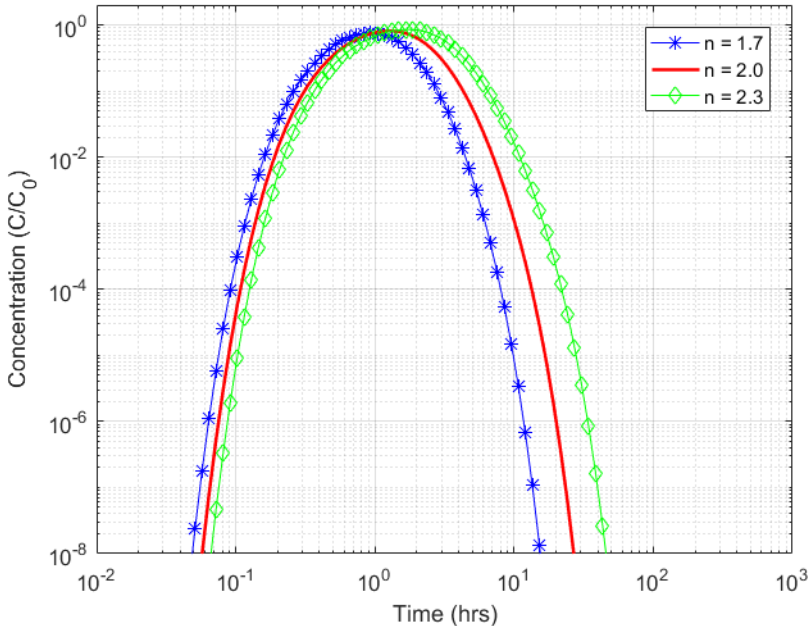

Figure 49. Generic Comparison BTCs without diffusion and Pulse Type Source

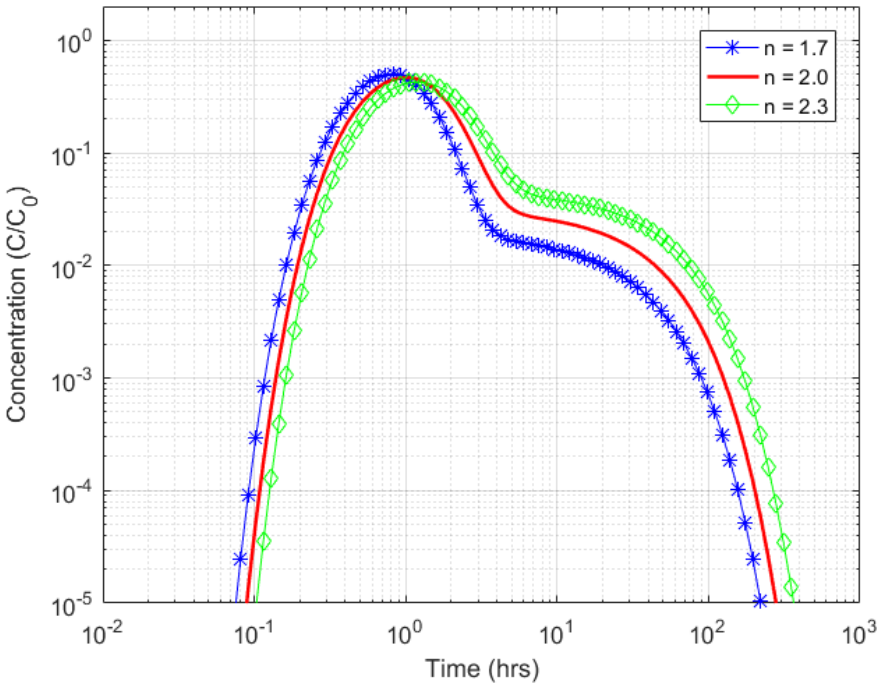

Figure 50. Generic Comparison BTCs with Single-Rate diffusion and Pulse Type Source

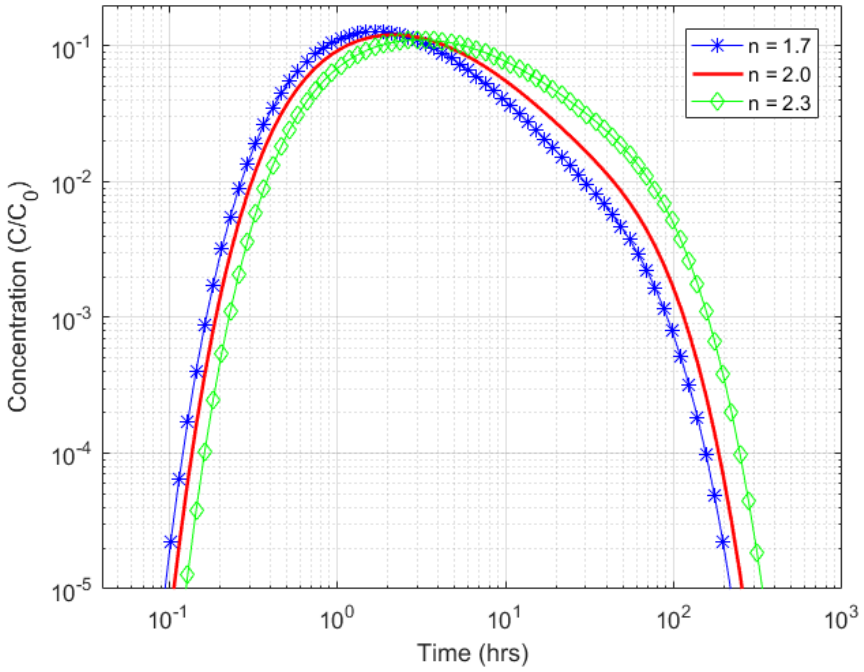

Figure 51. Generic Comparison BTCs with Multi-Rate diffusion and Pulse Type Source

CHAPTER 5

APPLICATIONS OF GRT MODEL

The developed multi-rate, double porosity GRT is used to investigate transport behavior of the Culebra Dolomite to determine flow dimension, longitudinal dispersivity and diffusive capacity, porosity and diffusion rate

3.1 Description of Culebra Dolomite Member

The Culebra Dolomite Member is situated in the vicinity of the Waste Isolation Pilot Plant (WIPP) in the south-eastern part of New Mexico and the western part of Texas, USA (Figure 52). In the vicinity of WIPP the depth of the Culebra Dolomite ranges from 275 m to 215 m below the ground surface. The Culebra Dolomite Member of Rustler Formation occupies an area greater than 25,000 km^2 but was likely originally 100,000 km^2 before erosion (Holt, 1997a).

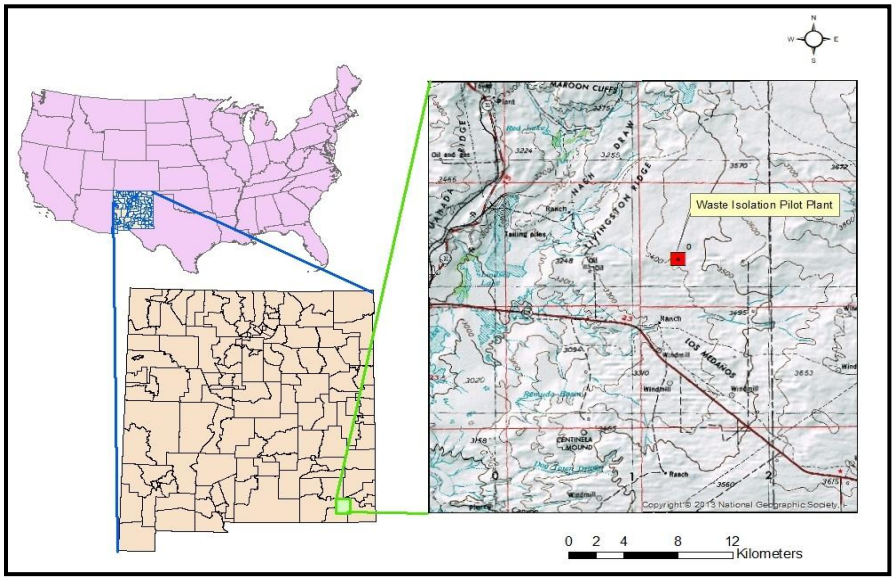

Figure 52. Location map of the study area in the vicinity of Waste Isolation Pilot plant

The Culebra Dolomite Member has been tested for hydraulic properties in 47 locations (Figure 53) in the vicinity of the WIPP; the transmissivity of the Culebra Dolomite varies by six orders of magnitude (Holt, 1997). The transmissivity values range from <4 x 10^{-9} m^2/s at borehole p-18 to 2 x 10^{-3} m^2/s at borehole H-7 (Holt, 1997).

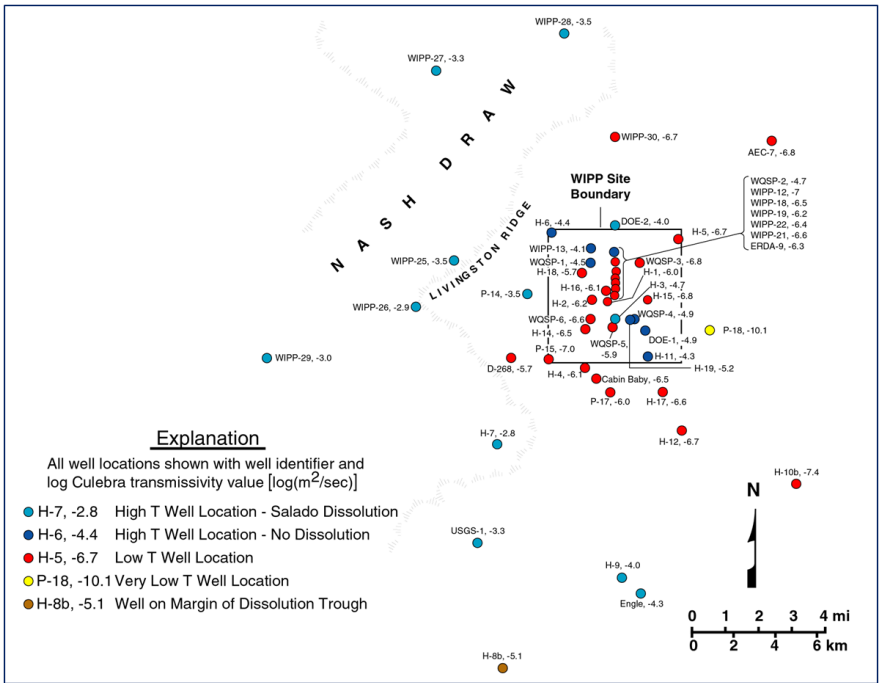

Figure 53. Map of Culebra Transmissivity Distribution and Boreholes (from Holt, 1997)

The Culebra behaves hydraulically as a single-porosity medium during pumping where transmissivities are less than 2 x 10^{-6} m^2/s and behaves as a double-porosity medium where transmissivities are greater than 2 x 10^{-6} m^2/s (Holt, 1997). Tracer tests and hydraulic tests indicate that the Culebra Dolomite is vertically and areally heterogeneous (Figure 54).

The Culebra Dolomite Member is an intensely fractured rock, and it exhibits numerous scales of fracturing with several types of porosity within fractured-bounded blocks. The Culebra unit is important because it is a potential pathway for contaminant transport from the Waste Isolation Pilot Plant (WIPP), a Department of Energy Facility for the disposal of radioactive

waste generated by defense-related activities of the United States, to the environment. This research will interpret transport processes within the Culebra Dolomite Member and improve understanding of the long-term performance of the WIPP facility.

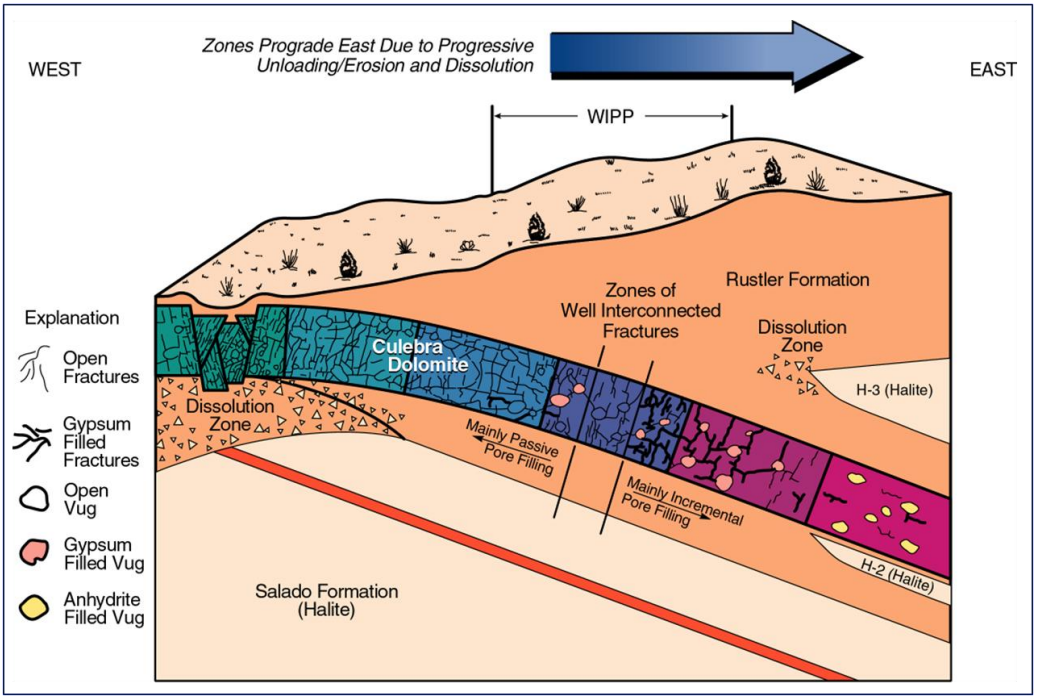

Figure 54. Conceptual Model for Culebra Transmissivity (from Holt, 1997)

3.2 Culebra Tracer Tests

Both single-well injection-withdrawal (SWIW) and multiple well convergent flow (MWCF) tests were conducted in the 7m thick Culebra Dolomite at two sites at H-11 Hydropad and H-19 Hydropad (Meigs & Beauheim, 2001). The H-11 Hydropad was completed in 1988 and comprised of four wells H-11b1, H-11b2, H-11b3, and H-11b4 for SWIW and MWCF tests ((Figure 55) (Meigs & Beauheim, 2001). The Hydropad H-19 was completed in 1995 and comprises of seven wells H-19b0, H-19b2, H-19b3, H-19b4, H-19b5, H-19b6 and H-19b7 (Figure 56).

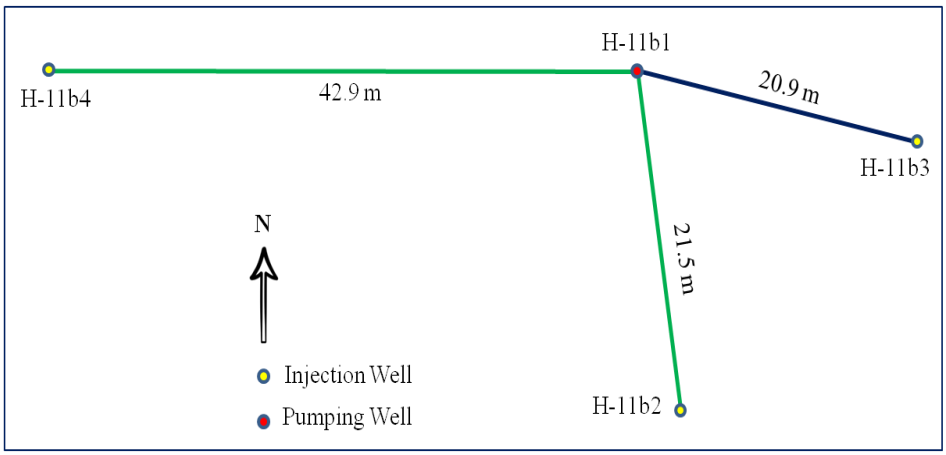

Figure 55. Experimental design and well locations at Hydropads H-11

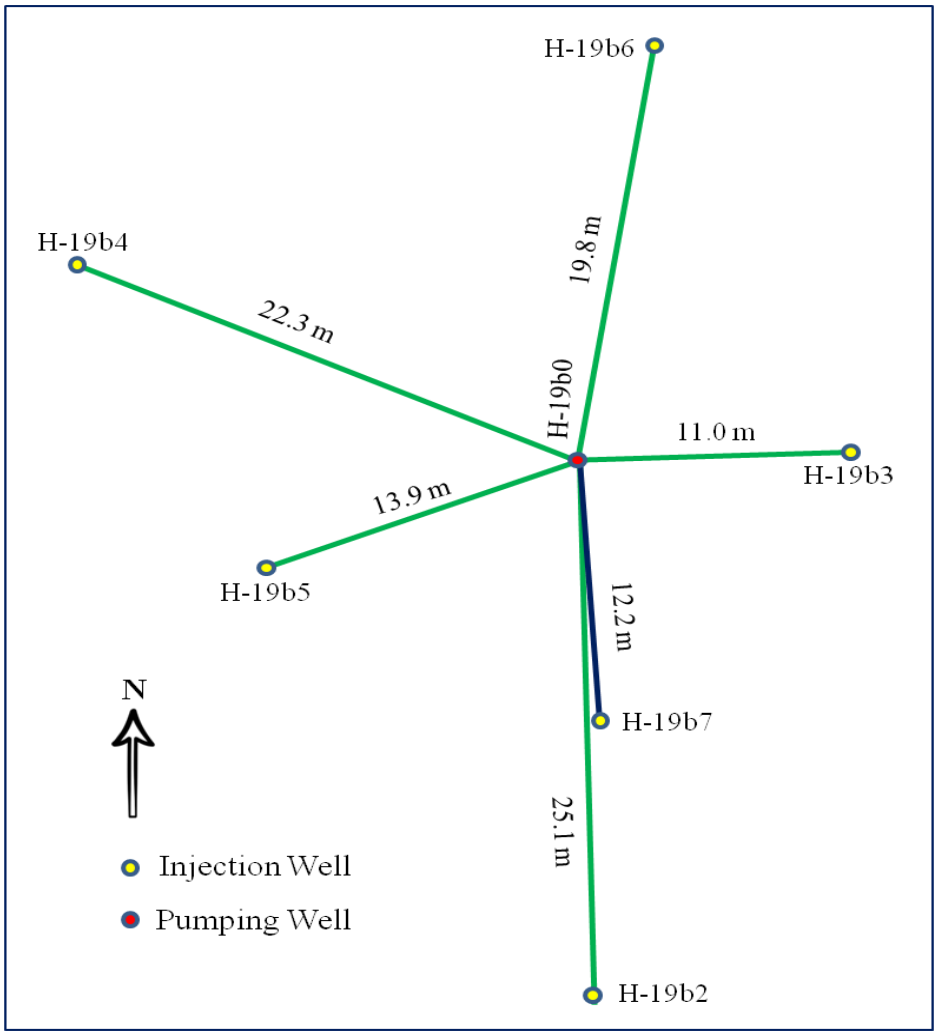

Figure 56. Experimental design and well locations at Hydropads H-19

Numerous benzoic acids were used as conservative tracers to allow the collection of tracer recovery and breakthrough data. After injection, a tracer a chaser solution was injected to displace the tracer from the well into the formation (Meigs et al, 2001). The tracer was pumped back after following a resting period of tracer. Properties of the Culebra Dolomite at the H-11 and H-19 Hydropads are listed in the table 4. Details about the SWIW and MWCF tests are given the table 5 and table 6 respectively.

Table 4. Properties of the Culebra Dolomite at the H-11 and H-19 Hydropads H-11 (from Meigs et al, 2001)

Parameters	H-11 Hydropad	H-19 Hydropad
Field Transmissivity (Full Culebra) (m^2/s)	4.7×10^{-5}	6.8×10^{-6}
Thickness of Full Culebra (m)	7.4	7.4
Thickness of Lower Culebra (m)	4.4	4.4
Mean[1] and Standard Deviation of Log of Core Hydraulic Conductivity (m/s)	-8.40 ± 0.99 $(10)^2$	-9.08 ± 1.12 (20)
Mean[1] and Standard Deviation of Core Porosity	0.16 ± 0.07 (10)	0.15 ± 0.06 (21)
Mean[1] and Standard Deviation of Core Formation Factor	66 ± 37 (4)	110 ± 80 (21)
Mean[1] and Standard Deviation of Calculated Tortuosity	0.11 ± 0.02 (4)	0.09 ± 0.04 (21)

1 Arithmetic average.

2 Numbers in parentheses denote number of samples.

Figure 57 shows tracers recovery curves for SWIW tests and Figure 58 and Figure 59 show tracers recovery curves for MWCF tests (Meigs & Beauheim, 2001) for the different tracers and pumping rate. We will use the MWCF tests data in the multi-rate, double porosity GRT model to determine transport and diffusion parameters of the Culebra Dolomite.

Table 5. Information on the SWIW tracer tests at the H-11 and H-19 Hydropad (from Meigs et. al. 2001)

Test	Pumping Rate, L/s	Injection Date	Culebra Interval	Pause Length, hour	Tracer[a]	Tracer Concentration,[b] g/L	Calculated Aqueous Diffusion Coefficient,[c] m^2/s	Tracer-Injection Rate, L/s	Injected Tracer Volume, L	Chaser-Injection Rate,[d] L/s	Injected Chaser Volume, L	Time to Final Sample, days	Calculated Mass Recovered (Fraction)
H-11 SWIW	0.22	Feb. 6, 1996	full	17.7	2,4-DCBA	8.07 ± 0.40	7.3×10^{-10}	0.12	996	0.13	1920	50	0.98
(H-11b1)		Feb. 6, 1996	full	17.7	3,4-DFBA	5.02 ± 0.32	8.2×10^{-10}	0.13	1010	0.12	910	50	0.97
H-19 SWIW 1	0.24	June 15, 1995	full	17.6	2,4-DCBA	4.94 ± 0.21	7.3×10^{-10}	0.13	997	0.13	2020	32	0.95
(H-19b0)		June 15, 1995	full	17.6	o-TFMBA	1.91 ± 0.04	7.4×10^{-10}	0.13	1005	0.13	1015	32	0.98
H-19 SWIW 2 (H-19b0)	0.27	Dec. 21, 1995	lower	17.7	2,4-DCBA	5.97 ± 0.17	7.3×10^{-10}	0.12	849	0.12	1697	26	0.94

[a]Tracers are as follows: 2,4-DCBA, 2,4-dichlorobenzoic acid; 3,4-DFBA, 3,4-difluorobenzoic acid; and o-TFMBA, ortho-trifluoromethylbenzoic acid.

[b]The concentrations listed result in an initial increase in solution density of between 0.2 and 0.8%, which will decrease rapidly because of mixing with water in the borehole and formation. The temperature of injected solution was colder than the formation water by as much as 10°–20°C for the winter injections and was probably slightly warmer for the summer injections. These temperature differences will also affect the density and viscosity differences between the injected and ambient fluids. Calculations suggest temperature effects should be small and will dissipate rapidly.

[c]Aqueous diffusion coefficient is calculated using the Hyduk and Laudie method as described by *Tucker and Nelken* [1982].

[d]For June 15, 1995, and February 6, 1996, tests, injection sequence consisted of injection of tracer 1 (2,4-DCBA), followed by tracer 2 (o-TFMBA or 3,4-DFBA), followed by chaser (Culebra brine). For tracer 1 listed above, chaser injection rate and volume are calculated as the rate or volume for injection of both tracer 2 and the chaser fluid.

Table 6. Information on the MWCW tracer tests at the H-11 and H-19 Hydropad (from Meigs et. al. 2001)

Test	Pumping Rate, L/s	Path	Injection Date	Average Hydraulic Gradient,[a] m/m	Tracer[b]	Tracer Concentration,[c] g/L	Calculated Aqueous Diffusion Coefficient,[d] m^2/s	Tracer-Injection Rate, L/s	Injected Tracer Volume, L	Chaser-Injection Rate, L/s	Injected Chaser Volume, L	Time to Final Sample, days	Calculated Mass Recovered (Fraction)
H-11 1996 round 1	0.22	b2-b1	Feb. 15, 1996	0.31	2,6-DFBA	10.38 ± 0.05	8.2×10^{-10}	0.068	189	0.060	213	41	0.40
		b3-b1	Feb. 15, 1996	0.30	2,3,4,5-TFBA	10.85 ± 0.24	7.9×10^{-10}	0.096	189	0.098	372	41	0.74
H-11 1996 round 2	0.38	b2-b1	March 14, 1996	0.70	p-TFMBA	10.78 ± 0.11	7.4×10^{-10}	0.072	189	0.062	213	13	0.21
		b3-b1	March 13, 1996	0.72	2,5-DFBA,	10.30 ± 0.15	8.2×10^{-10}	0.095	190	0.097	373	14	0.56
		b3-b1	March 13, 1996	0.72	NaI	10.87 ± 0.03	18.0×10^{-10}	0.095	190	0.097	373	14	0.53
H-19 1995 (preliminary four-well test)	0.24	b2-b0	June 19, 1995	1.4	2,3-DFBA	7.30 ± 0.34	8.2×10^{-10}	0.11	246	0.11	246	37	0.54
		b3-b0	June 20, 1995	3.0	2,3,4,5-TFBA	7.77 ± 0.27	7.9×10^{-10}	0.15	259	0.14	206	37	0.69
		b4-b0	June 19, 1995	1.6	2,6-DFBA	7.06 ± 0.58	8.2×10^{-10}	0.13	265	0.11	255	37	0.41
H-19 1995–1996	0.27	b2-b0	Dec. 22, 1995	1.7	2,3,4-TFBA	8.18 ± 0.25	8.0×10^{-10}	0.13	202	0.15	154	104	0.88
		b3-b0	Dec. 22, 1995	3.7	m-TFMBA	9.52 ± 0.51	7.4×10^{-10}	0.18	198	0.23	173	104	0.88
		b3-b0	Dec. 22, 1995	3.7	NaI	12.71	18.0×10^{-10}	0.18	198	0.23	173	63	0.80
		b4-b0	Dec. 22, 1995	2.1	3,5-DFBA	8.46 ± 1.95	8.2×10^{-10}	0.12	198	0.12	143	104	0.84
		b5(u)-b0	Dec. 20, 1995	3.0	2,3-DCBA	11.45 ± 0.31	7.3×10^{-10}	0.015	147	0.010	105	106	0.18
		b5(l)-b0	Dec. 20, 1995	3.0	2,5-DCBA	13.49 ± 1.26	7.3×10^{-10}	0.011	149	0.009	65	106	0.84
		b6-b0	Dec. 21, 1995	2.3	2,5-DFBA	9.49 ± 0.13	8.2×10^{-10}	0.12	199	0.12	154	106	0.88
		b7-b0	Dec. 21, 1995	3.3	2,4-DFBA	7.58 ± 0.53	8.2×10^{-10}	0.21	198	0.22	168	105	1.03
H-19 1995–1996 round 2	0.25	b3(u)-b0	Jan. 19, 1996	3.5	p-TFMBA	14.13 ± 0.32	7.4×10^{-10}	0.016	132	0.015	69	82	0.13
		b3(l)-b0	Jan. 19, 1996	3.4	o-TFMBA	9.69 ± 0.25	7.4×10^{-10}	0.028	198	0.033	143	82	0.89
		b5-b0	Jan. 19, 1996	2.8	2,4-DCBA	9.85 ± 0.66	7.3×10^{-10}	0.19	199	0.17	169	82	0.83
		b7(u)-b0	Jan. 20, 1996	3.0	PFBA	14.51 ± 0.10	7.7×10^{-10}	0.008	131	0.010	64	81	0.05
		b7(l)-b0	Jan. 20, 1996	3.1	3,5-DCBA	7.67 ± 0.41	7.3×10^{-10}	0.016	197	0.015	139	81	0.90
H-19 1995–1996 round 3	0.16	b3-b0	Feb. 22, 1996	2.0	2,3,4,5-TFBA	9.95 ± 0.34	7.9×10^{-10}	0.100	198	0.12	173	48	0.80
		b6-b0	Feb. 22, 1996	1.3	2,4,6-TCBA	9.87 ± 0.35	6.8×10^{-10}	0.070	197	0.068	153	48	0.74
		b7-b0	Feb. 22, 1996	1.8	2,3,6-TFBA	9.54 ± 0.27	8.0×10^{-10}	0.12	199	0.12	168	48	0.91
		b7-b0	Feb. 22, 1996	1.8	NaI	10.68 ± 1.50	18.0×10^{-10}	0.12	199	0.12	168	48	0.88

[a]Meters of fresh water per meter distance are shown.

[b]Tracers are as follows: x,y-DFBA, x,y-difluorobenzoic acid (e.g., 2,6-DFBA is 2,6-difluorobenzoic acid); 2,3,4,5-TFBA, 2,3,4,5-tetrafluorobenzoic acid; m-TFMBA, o-TFMBA, or p-TFMBA, meta-, ortho-, or para-trifluoromethylbenzoic acid; NaI, sodium iodide; x,y,z-TFBA, x,y,z-trifluorobenzoic acid; x,y-DCBA, x,y-dichlorobenzoic acid; PFBA, pentafluorobenzoic acid; and 2,4,6-TCBA, 2,4,6-trichlorobenzoic acid.

[c]The concentrations listed result in an initial increase in solution density of approximately 0.9% for most injections and vary from 0.7% up to ~2% for the cases where a benzoic acid and iodide were coinjected. The density should decrease rapidly mixing with water in the borehole and formation. The temperature of injected solution was colder than the formation water by as much as 10°–20°C for the winter injections and was probably slightly warmer for the summer injections. These temperature differences will also affect the density and viscosity differences between the injected and ambient fluids. Calculations suggest temperature effects should be small and will dissipate rapidly.

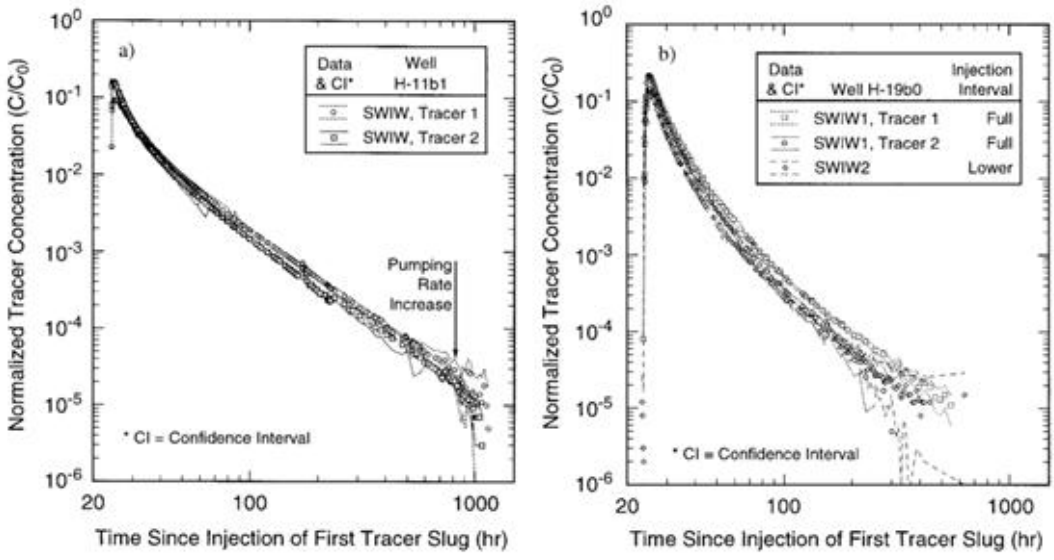

Figure 57. SWIW tracer recovery curves from (a) one test at the H-11 hydropad and (b) Two tests at the H-19 hydropad (from Meigs et al, 2001).

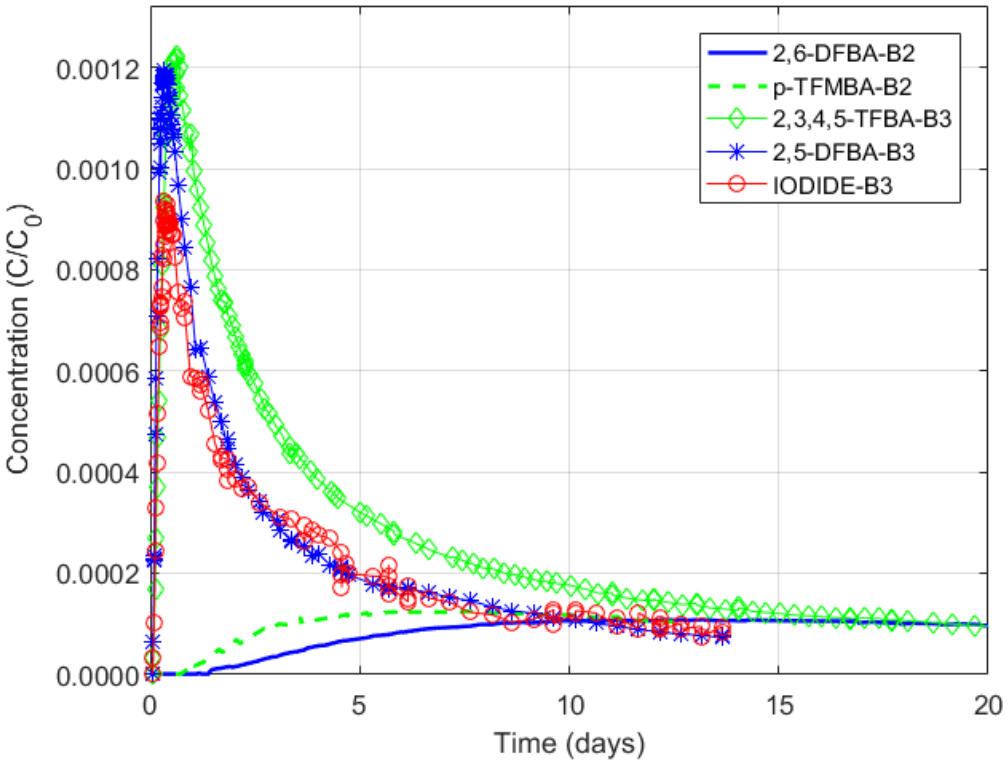

Figure 58. MWCF tracer breakthrough curves for different tracers from the H-11 hydropad

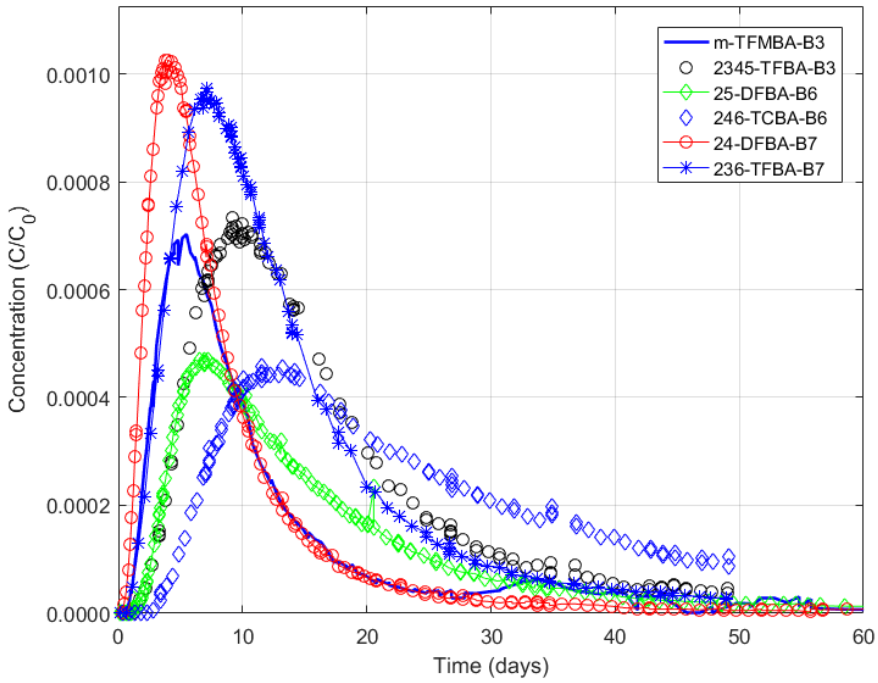

Figure 59. MWCF tracer breakthrough curves for different tracers from the H-19 hydropad

5.3 Fitting GRT model with Tracer Recovery data

The multi-rate, double-porosity GRT model was simulated using data from H-11b3 (Appendix B3) and H-19b7 (Appendix B4) tracer tests. The GRT was simulated in forward mode and parameters were estimated by looking at the optimum fit of calculated breakthrough curve with tracer recovery breakthrough curve. Flow dimension, advective porosity, diffusive porosity and longitudinal dispersivity, total capacity coefficient, log of the geometric mean of the diffusion rate coefficients, and standard deviation of the log-transformed diffusion rate coefficient were estimated.

5.3.1 Fitting with Tracer data H-11b3

The tracer test was conducted for 13.639 days at the H-11b3 injection well. 2, 5-DFBA

with a concentration of 10300 mg/L was injected. The distance between the injection and the H-19b1 pumping well was 20.9 m (Figure 55). The GRT model was simulated with the H-19b3 tracer test parameters for 13.639 days. The simulation result from the GRT model almost perfectly fit with the H-11b3 tracer recovery data (Figure 60) and estimated parameters are given in the table 7. Estimated parameters using the GRT model were compared to previously estimated parameters of the tracer test using a radial model STAMMT-R (Meigs et. al. 2000) and using another radial model SWIFT II/B (Reeves et. al. 1986).

Table 7. Estimated parameters with GRT and Radial Model for the H-11b3 tracer data

Estimated Parameters	Multi-rate GRT Model	Radial Model (SWIFT II/B)	Radial Model STAMMT-R
Flow Dimension, n [-]	1.92	2	2
Advective Porosity, ϕ_f [-]	1.0×10^{-3}	1.7×10^{-4}	6.12×10^{-4}
Diffusive Porosity, ϕ_m [-]	0.17	0.16	0.16
Log of the geometric mean of the diffusion rate coefficients, μ^* [-]	-7.107	-	-17.19
Standard deviation of the log-transformed diffusion rate coefficient, σ [-]	1.05	-	1.12
Longitudinal Dispersivity, α_L (m)	2.74	1.36	3.47
Total Capacity Coefficient, β_{tot} [-]	170	941.17	261.43

In comparison to GRT, both of the radial models underestimated advective porosity and Diffusive porosity; and overestimated the total capacity coefficient. The STAMMT-R overestimated and underestimated the longitudinal dispersivity compared to the GRT model. So, the radial models over estimated the capacity coefficient leads more diffusion into fractures-bounded matrix blocks. The GRT model estimated flow dimension 1.92 for this tracer test which are the characteristics of sub-radial flow.

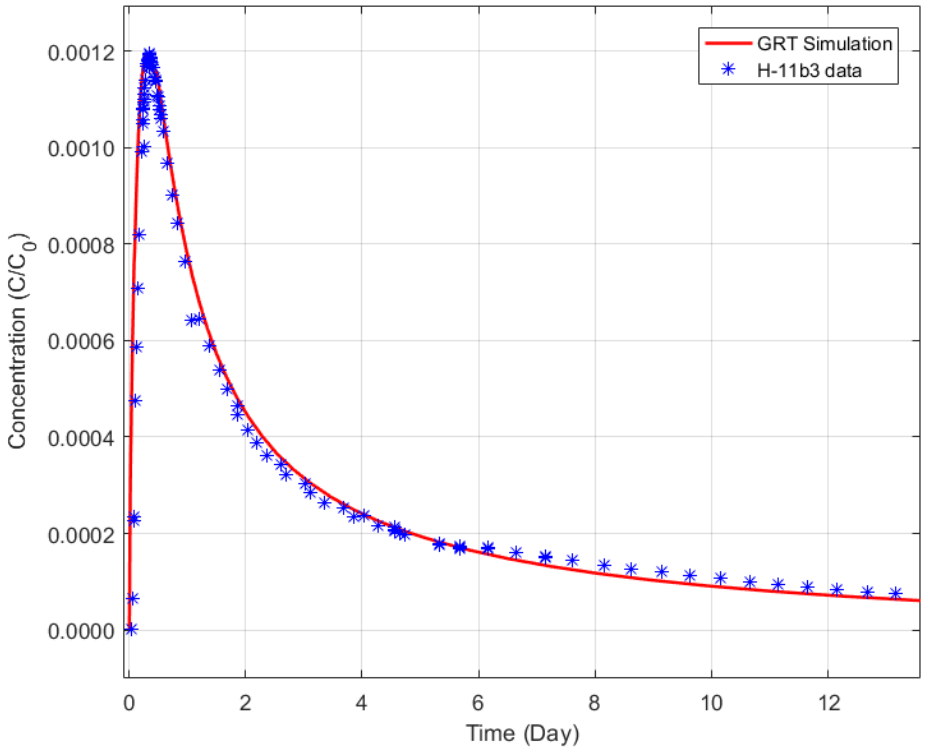

Figure 60. GRT fitting with the H-11b3 Tracer Test Data

5.3.2 Fitting with Tracer data H-19b7

The tracer test at the H-19b7 was conducted for 48.975 days using the non-conservative 2, 3, 6-TFBA with concentration of 9540 mg/L. The tracer test was simulated and fit with the recovery data (Figure 61). Estimated parameters using radial model STAMMT-R (Meigs et. al. 2000), using another radial model SWIFT II/B (Reeves et. al. 1986) and using the GRT are given in the table 8. Unlike the tracer test H-11b3, the radial models underestimated the capacity coefficient compare to that estimated from the GRT model. In comparison to GRT, the radial models overestimated the advective porosity, longitudinal dispersivity and underestimated diffusive porosity. The estimated capacity coefficient 3.15 with that radial implies that tracer did not diffuse much into the fractures-bounded matrix blocks and advection dominates transport

process. But, the tracer recovery data of the H-19b7 shows that tailing back of tracer after the peak arrival which is the effect diffusion (Figure 61).

Table 8. Estimated parameters with GRT and Radial Model for the H-19b7 tracer data

Estimated Parameters	Multi-rate GRT Model	Radial Model (SWIFT II/B)	Radial Model STAMMT-R
Flow Dimension, n [-]	1.87	2	2
Advective Porosity, ϕ_f [-]	3.53 x 10^{-4}	0.0476	1.00 x 10^{-3}
Diffusive Porosity, ϕ_m [-]	0.17	0.15	0.147
Log of the geometric mean of the diffusion rate coefficients, μ^* [-]	-6.45	-	-15.62
Standard deviation of the log-transformed diffusion rate coefficient, σ [-]	4.3	-	4.70
Longitudinal Dispersivity, α_L (m)	0.3	0.61	0.65
Total Capacity Coefficient, β_{tot} [-]	481.59	3.15	147

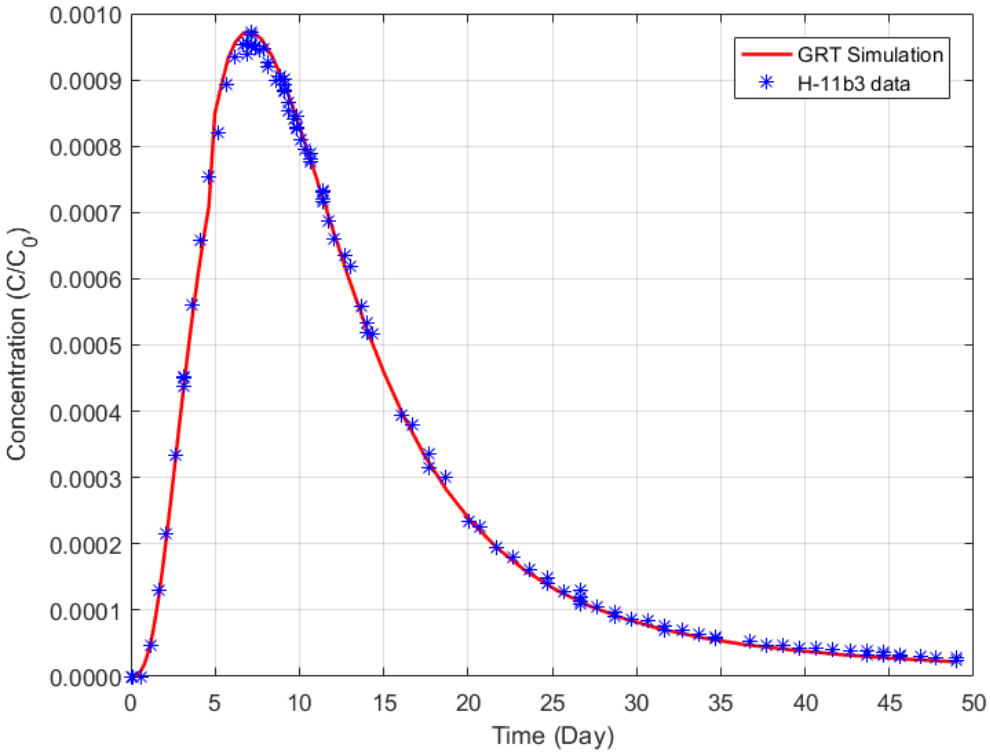

Figure 61. GRT fitting with the H-19b7 Tracer Test Data

The simulated BTC with the GRT model almost perfectly match with the tracer recovery data of the H-19b7 and estimated capacity coefficient is 481.59 which leads more diffusion into fractures-bounded matrix blocks. The estimated flow dimension from the GRT model is 1.87 for this tracer test which indicates the flow is sub-radial.

Radial Flow model underestimates longitudinal dispersivity, diffusive capacity and advective porosity over estimates and sometime underestimates the capacity coefficient. The GRT model fitting with the tracers tests data shows characteristics of sub-radial flow. In comparison to Radial Flow model, the multi-rate, double porosity GRT model better estimates flow dimension, longitudinal dispersivity, advective porosity, diffusive porosity and capacity coefficient.

CHAPTER 6

CONCLUSION

The multi-rate, double porosity GRT model was developed by incorporating a geometric term flow dimension which characterizes the change in flow area versus radial distance from the borehole. The GRT model was solved by Laplace domain block-centered integrated finite difference model for pumping phase solution after following Haggerty`s (1995, 1998) multi-rate semi-analytical model for injection phase and resting phase solution. The multi-rate, double porosity GRT model can simulate transport for integer and non-integer values of flow dimensions. The GRT model transforms to 1D, radial and spherical transport model for integer flow dimensions, n of 1, 2, and 3 respectively. And, the model also transforms to sub-linear, sub-radial, sub-spherical and eventually transform to super-spherical transport model for non-integer flow dimension, n of $0 < n < 1$, $1 < n < 2$, $2 < n < 3$ and $n > 3$ respectively. The GRT model also capable of simulate transport without diffusion, with single-rate diffusion and multi-rate diffusion into matrix block.

The multi-rate, double porosity GRT model with flow dimension 1was validated against 1D analytical model both for continuous and pulse type source, radial semi-analytical solution and spherical analytical model with pulse type source. The GRT model was verified against the Ogata and Banks (1961) 1D analytical for continuous source of injection, the Haggerty (1995) 1D analytical model for pulse type source, the Haggerty et. al. (2000) semi-analytical radial

multi-rate transport model and the Schroth and Istok (2005) for spherical transport. The GRT showed excellent matching of breakthrough curves with 1D, radial and spherical model except some numerical dispersion. Numerical dispersion of the GRT can be eliminated by decreasing grid size and allowing more simulation time.

The sensitivity analysis of the multi-rate, double porosity GRT model showed that larger flow dimensions result in slower transport, and the transport process approaches Local Equilibrium Condition (LEC) where transport is physically retarded as solutes repeatedly diffuse into and out of fracture-bounded blocks. Multi-rate, double-porosity transport differs from single-rate double-porosity transport because there is a distribution of mass-transfer coefficients and capacity coefficients. Smaller Peclet Number shows effect of longitudinal dispersivity, diffusion and advection at very early time and later diffusion become dominant. Multi-rate, double porosity transport showed increasing Peclet number tracers spread less in aquifer and smooth changes in breakthrough curve from advection domination to diffusion domination transport. As Beta increases, more solute mass can be stored in the diffusive porosity, creating a larger sink for diffusing solutes, and this leads to later peak arrival times and more impact from diffusive mass transfer.

The multi-rate, double porosity GRT is used to investigate transport behavior of the Culebra Dolomite to determine dispersivity and diffusive capacity, porosity and diffusion rate. The GRT fitting with the tracer test data H-11b3 and H-19b7 estimated flow dimension 1.92 and 1.87 respectively which are indication of sub-radial flow behavior of the Culebra Dolomite. Interpretation of convergent flow tracer tests in fractured rocks aquifer using the Radial Flow model underestimated longitudinal dispersivity, diffusive capacity and advective porosity, and over estimated the capacity coefficient. In comparison to Radial Flow model, the multi-rate,

double-porosity GRT model showed higher velocities and much earlier arrival time. Faster advective transport of the GRT model due to sub-radial flow leads to less diffusion into fracture-bounded matrix blocks and steeper late-time concentration gradients.

BIBLIOGRAPHY

BIBLIOGRAPHY

Barker, J. A. (1988). A generalized radial flow model for hydraulic tests in fractured rock. *Water Resources Research*, *24*(10), 1796–1804. https://doi.org/10.1029/WR024i010p01796

Bowman, D. O., Roberts, R. M., & Holt, R. M. (2013). Generalized radial flow in synthetic flow systems. *Groundwater*, *51*(5), 768–774. https://doi.org/10.1111/j.1745-6584.2012.01014.x

Chen, C.-S. (1985). 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.

de Hoog, F. R., Knight, J., & Stokes, A. (1982). An improved method for numerical inversion of Laplace transforms. *IAM J. Sci. Stat. Comput.*, *3 (3)*, 357–366.

Haggerty, R. (1995). Aquifer Remediation in the Presence of Rate-Limited Mass Transfer. *PhD Thesis*.

Haggerty, R., Fleming, S. W., & Mckenna, S. A. (2000). STAMMT-R Solute Transport and Multirate Mass Transfer in Radial Coordinates. *New Mexico: Sandia National Laboratories, SAND 99-01*(July).

Haggerty, R., & Gorelick, S. M. (1995). Multiple-rate mass transfer for modeling diffusion and surface in media with pore-scale heterogeneity. Water Resources Research.

Haggerty, R., & Gorelick, S. M. (1998). Modeling Mass Transfer Processes in Soil Columns with Pore-Scale Heterogeneity. *Soil Sci. Soc. Am. J.*, *62*(1), 62–74. https://doi.org/10.2136/sssaj1998.03615995006200010009x

Holt, R. M. (1997a). Conceptual Model for Transport Processes in the Culebra Dolomite Member, Rustler Formation Sandia National Laboratories, (August).

Holt, R. M. (1997b). Conceptual Model for Transport Processes in the Culebra Dolomite Member , Rustler Formation Sandia National Laboratories.

Kreft, A., & Zuber, A. (1978). On the physical meaning of the dispersion equation and its solutions for different initial and boundary conditions. *Chemical Engineering Science*, *33*(11), 1471–1480.

Mathematical, I., & Libraries, S. (1982). *International Mathematical and Statistical Libraries Library Reference Manual*.

Mckenna, S. A., Meigs, L. C., & Haggerty, R. (2001). Tracer tests in a fractured dolomite 3. Double-porosity, multiple-rate mass transfer processes in convergent flow tracer tests. *Water Resources Research*, *37*(5), 1143–1154.

Meigs, L. C., & Beauheim, R. L. (2001). Tracer tests in a fractured dolomite 1. Experimental design and observed tracer recoveries. *Water Resources Research*, *37*(5), 1113–1128. https://doi.org/10.1029/2000WR900335

Meigs, L. C., Beauheim, R. L., & Jones, T. L. (2000). Characteristics of the Culebra, in Interpretations of Tracer Tests Performed in the Culebra Dolomite at the Waste Isolation Pilot Plant Site.

Ogata, A., & Banks, R. B. (1961). A Solution of the Differential Equation of Longitudinal Dispersion in Porous Media. *US Geological Survey, Professional Paper*, *411–A*.

Reeves, M.; Ward, D.S.; Johns, N.D.; Cranwell, R. M. (1986). Theory and implementation for SWIFT II. The Sandia waste-isolation flow and transport model for fractured media.
Schroth, M. H., & Istok, J. D. (2005). Approximate Solution for Solute Transport during Spherical-Flow Push-Pull Tests, *43*(2), 280–284.
Thomas, J. B. (1986). *Introduction to Probability*. New York: Springer-Verlag.
Walker, D. D., & Roberts, R. M. (2003). Flow dimensions corresponding to hydrogeologic conditions. *Water Resources Research*, *39*(12), 1–8. https://doi.org/10.1029/2002WR001511
Zlotnik, V. A., & Logan, J. D. (1996). Boundary conditions for convergent radial tracer tests and Effect of Well Bore Mixing Volume One. *Water Resources*, *32*(7), 2323–2328.

APPENDIX

APPENDIX A
Mathematical Model Derivation

APPENDIX A

Using the equation (48) and (50) in the equation (51), we get:

$$q = -\frac{\alpha_L |f(n)|}{r^{n-1}} \frac{\partial C}{\partial r} + \frac{f(n)}{r^{n-1}} C \tag{A1}$$

Solving the equation (39), we obtained:

$$\frac{\partial C}{\partial t} = -\frac{\partial q}{\partial r} - \left(\frac{n-1}{r}\right) q \tag{A2}$$

By plugging the value of q from (A1) into the 1st term of the right hand side of the equation (A2) and solving for the term, we get:

$$-\frac{\partial q}{\partial r} = \frac{\alpha_L |f(n)|}{r^{n-1}} \frac{\partial^2 C}{\partial r^2} - \alpha_L |f(n)| \frac{\partial C}{\partial r} \frac{\partial}{\partial r}\left(\frac{1}{r^{n-1}}\right) - \frac{f(n)}{r^{n-1}} \frac{\partial C}{\partial r} + f(n) C \frac{\partial}{\partial r}\left(\frac{1}{r^{n-1}}\right) \tag{A3}$$

In the equation (A3),

$$\frac{\partial}{\partial r}\left(\frac{1}{r^{n-1}}\right) = \frac{\partial}{\partial r}\left[r^{-(n-1)}\right] = \frac{n-1}{r^n} \tag{A4}$$

So, the equation (A3) becomes:

$$-\frac{\partial q}{\partial r} = \frac{\alpha_L |f(n)|}{r^{n-1}} \frac{\partial^2 C}{\partial r^2} - \alpha_L |f(n)| \frac{(n-1)}{r^n} \frac{\partial C}{\partial r} - \frac{f(n)}{r^{n-1}} \frac{\partial C}{\partial r} + f(n) \frac{(n-1)}{r^n} C \tag{A5}$$

The 2nd term of the equation (A2) becomes:

$$-\left(\frac{n-1}{r}\right) q = \frac{\alpha_L |f(n)|(n-1)}{r^n} \frac{\partial C}{\partial r} - \frac{f(n)(n-1)}{r^n} C \tag{A6}$$

Now, using the equation (A5) and (A6) into the (A2), we get:

$$\frac{\partial C}{\partial t} = \frac{\alpha_L \left| f(n) \right|}{r^{n-1}} \frac{\partial^2 C}{\partial r^2} - \frac{f(n)}{r^{n-1}} \frac{\partial C}{\partial r} \tag{A7}$$

By plugging the value of $f(n)$ from the equation (49) into the equation (A7), we get the following single porosity GRT model:

$$\frac{\partial C}{\partial t} = \alpha_L \frac{Q\Gamma(n/2)}{R_f \phi b^{3-n} 2\pi^{n/2} r^{n-1}} \frac{\partial^2 C}{\partial r^2} - \frac{Q\Gamma(n/2)}{R_f \phi b^{3-n} 2\pi^{n/2} r^{n-1}} \frac{\partial C}{\partial r} \tag{A8}$$

APPENDIX B

INPUT DATA FILES

Appendix B1: Parameter Input File for Model Verification

```
0                   skipm  !do CFTT simulation? 0 if yes, 1 if no
0.0000d0            Tc0    !start time of solute injection (can be 0), MW [T]
0.3000d0            TcE    !elapsed time from t=0 to end of solute inj., MW [T]
0.5000d0            Timein !elapsed time from t=0 to end of chaser inj., MW [T]
0.5000d0            Qin    !injection rate, MW [L^3/T]
1.000d0             Qout   !pumping rate, MW [L^3/T]
0.80D+00            alphLm !longitudinal dispersivity, MW [L]
3.0d0               rmax   !edge of grid for injection, MW [L]
10.0d0              Ro     !distance from injection to pumping wells, MW [L]
0.01000d0            r0i    !well radius (injection, MW) [L]
1                   MWtime !use (t,C) input file (set=0) or generate times (set=1), MW
1                   MWz    !if MWtime=1: constant time (=0) or constant ln(time) (=1) MW
960                 MWpumpt!if MWtime=1: time from Timein to end of pumping [T], MW
100                 TNM    !number of time vs concentration data points, MW
0                   Trest  ! Resting Time for the CFTT

0.100000d0         r0p    !radius of the pumping well  [L]
4.000d0            b1     !saturated thickness at injection well [L]
4.00d0             b2     !saturated thickness at pumping well [L]
1.d0               Cin    !injection conc. [M/L^3; if Cin=1, conc. dimensionless]
-6.9078D00         mus    !initial guess (parameter estimation) mus [ln(1/T)]
0.000d0            sig    !initial guess (parameter estimation) sig
1.0000D+00         Rm     !mobile zone retardation [-]
1.0000D+00         Rim    !immobile zone retardation [-]
0.20d0             ptot   !maximum permitted total porosity (inversion parameter) [-]
2.628d-6           Daq    !aqueous diffusion coefficient of solute [L^/T]
0.11               tort   !diffusive tortuosity [-]
0.01               dr     ! descritization for pumping phase
2.0                fdn    ! flow dimension, n
0.03               poros  !advective porosity [-]
0.16               pmat   !diffusive porosity


T                 cinject   ! pulse Source (T)
F                 SingR     ! If Single-Rate (T), else (F)
T                 MultR     ! If Multi-Rate (T), else (F)
F                 FixOmBet  ! If Input Omega and Beta (T), else (F)
F                 contsrc   ! Continuous source (T), else (F)
F                 pumpSol   ! Only Pumping Phase Solution (T), else (F)
1.0d-2            tmin      ! Minimum Time
960               tmax      ! Minimum Time
0.0               omegapf   ! Dimensionless Mass Transfer Rate Coeff.
0.0               betompf   ! Dimensionless Capacity Coeff.
0.01              alphar    ! Dimensional Mass Transfer Rate Coeff.


1                       iest   !0=parameter estimation; 1=forward model only
0                       idef   !0=Default distribution; 1 = User-defined distribution
0.0d0                    disc   !
1000                    kmax   !control permissible numerical error, etc.
1.d-5                   relerr !
```

Appendix B2: Parameter Input File GRT Sensitivity Analysis

```
0                   skipm  !do CFTT simulation? 0 if yes, 1 if no
0.0000d0            Tc0    !start time of solute injection (can be 0), MW [T]
0.3000d0            TcE    !elapsed time from t=0 to end of solute inj., MW [T]
0.5000d0            Timein !elapsed time from t=0 to end of chaser inj., MW [T]
0.5000d0            Qin    !injection rate, MW [L^3/T]
1.000d0             Qout   !pumping rate, MW [L^3/T]
1.00D+00            alphLm !longitudinal dispersivity, MW [L]
3.0d0               rmax   !edge of grid for injection, MW [L]
10.0d0              Ro     !distance from injection to pumping wells, MW [L]
0.01000d0           r0i    !well radius (injection, MW) [L]
1                   MWtime !use (t,C) input file (set=0) or generate times (set=1), MW
1                   MWz    !if MWtime=1: constant time (=0) or constant ln(time) (=1) MW
960                 MWpumpt!if MWtime=1: time from Timein to end of pumping [T], MW
100                 TNM    !number of time vs concentration data points, MW
0                   Trest  ! Resting Time for the CFTT

0.100000d0         r0p    !radius of the pumping well  [L]
5.000d0            b1     !saturated thickness at injection well [L]
5.00d0             b2     !saturated thickness at pumping well [L]
1.d0               Cin    !injection conc. [M/L^3; if Cin=1, conc. dimensionless]
-6.9078D00         mus    !initial guess (parameter estimation) mus [ln(1/T)]
0.000d0            sig    !initial guess (parameter estimation) sig
1.0000D+00         Rm     !mobile zone retardation [-]
1.0000D+00         Rim    !immobile zone retardation [-]
0.20d0             ptot   !maximum permitted total porosity (inversion parameter) [-]
2.628d-6           Daq    !aqueous diffusion coefficient of solute [L^/T]
0.11               tort   !diffusive tortuosity [-]
0.01               dr     ! descritization for pumping phase
2.0                fdn    ! flow dimension, n
0.01               poros  !advective porosity [-]
0.05               pmat   !diffusive porosity


T                 cinject   ! pulse Source (T)
F                 SingR     ! If Single-Rate (T), else (F)
T                 MultR     ! If Multi-Rate (T), else (F)
F                 FixOmBet  ! If Input Omega and Beta (T), else (F)
F                 contsrc   ! Continuous source (T), else (F)
F                 pumpSol   ! Only Pumping Phase Solution (T), else (F)
1.0d-2            tmin      ! Minimum Time
960               tmax      ! Minimum Time
0.0               omegapf   ! Dimensionless Mass Transfer Rate Coeff.
0.0               betompf   ! Dimensionless Capacity Coeff.
0.01              alphar    ! Dimensional Mass Transfer Rate Coeff.


1                        iest   !0=parameter estimation; 1=forward model only
0                        idef   !0=Default distribution; 1 = User-defined distribution
0.0d0                     disc   !
1000                     kmax   !control permissible numerical error, etc.
1.d-5                    relerr !
```

Appendix B3: Parameter Input File H-11b3 tracer data

```
0                  skipm  !do CFTT simulation? 0 if yes, 1 if no
0.0000d0           Tc0    !start time of solute injection (can be 0), MW [T]
0.5550d0           TcE    !elapsed time from t=0 to end of solute inj., MW [T]
1.070d0           Timein !elapsed time from t=0 to end of chaser inj., MW [T]
0.3427             Qin    !injection rate, MW [L^3/T]
1.386              Qout   !pumping rate, MW [L^3/T]
2.74       alphLm !longitudinal dispersivity, MW [L]
3.0d0              rmax   !edge of grid for injection, MW [L]
20.9d0            Ro     !distance from injection to pumping wells, MW [L]
0.1000d0           r0i    !well radius (injection, MW) [L]
1                  MWtime !use (t,C) input file (set=0) or generate times (set=1), MW
1                  MWz    !if MWtime=1: constant time (=0) or constant ln(time) (=1) MW
109                MWpumpt!if MWtime=1: time from Timein to end of pumping [T], MW
100                TNM    !number of time vs concentration data points, MW
0                  Trest  ! Resting Time for the CFTT

0.100000d0         r0p    !radius of the pumping well  [L]
7.400d0            b1     !saturated thickness at injection well [L]
7.40d0             b2     !saturated thickness at pumping well [L]
1.d0               Cin    !injection conc. [M/L^3; if Cin=1, conc. dimensionless]
-7.1078D00         mus    !initial guess (parameter estimation) mus [ln(1/T)]
1.05000d0           sig    !initial guess (parameter estimation) sig
1.0000D+00         Rm     !mobile zone retardation [-]
1.0000D+00         Rim    !immobile zone retardation [-]
0.20d0             ptot   !maximum permitted total porosity (inversion parameter) [-]
2.628d-6           Daq    !aqueous diffusion coefficient of solute [L^/T]
0.11               tort   !diffusive tortuosity [-]
0.01               dr
1.92               fdn
0.001              poros  !advective porosity [-]
0.17               pmat   !diffusive porosity


T                 cinject        ! pulse Source (T)
F                 SingR          ! If Single-Rate (T), else (F)
T                 MultR          ! If Multi-Rate (T), else (F)
F                 FixOmBet       ! If Input Omega and Beta (T), else (F)
F                 contsrc        ! Continuous source (T), else (F)
F                 pumpSol        ! Only Pumping Phase Solution (T), else (F)
1.0d-2            tmin           ! Minimum Time
326.15            tmax           ! Minimum Time
0.0               omegapf        ! Dimensionless Mass Transfer Rate Coeff.
0.0               betompf        ! Dimensionless Capacity Coeff.
0.01              alphar         ! Dimensional Mass Transfer Rate Coeff.


1                iest   !0=parameter estimation; 1=forward model only
0                idef   !0=Default distribution; 1 = User-defined distribution
0.0d0            disc   !
1000             kmax   !control permissible numerical error, etc.
1.d-5            relerr !
```

Appendix B4: Parameter Input File H-19b7 tracer data

```
0                   skipm  !do CFTT simulation? 0 if yes, 1 if no
0.0000d0            Tc0    !start time of solute injection (can be 0), MW [T]
0.4720d0            TcE    !elapsed time from t=0 to end of solute inj., MW [T]
0.3920d0            Timein !elapsed time from t=0 to end of chaser inj., MW [T]
0.4212              Qin    !injection rate, MW [L^3/T]
0.576               Qout   !pumping rate, MW [L^3/T]
0.3         alphLm !longitudinal dispersivity, MW [L]
3.0d0               rmax   !edge of grid for injection, MW [L]
12.20d0             Ro     !distance from injection to pumping wells, MW [L]
0.1000d0            r0i    !well radius (injection, MW) [L]
0                   MWtime !use (t,C) input file (set=0) or generate times (set=1), MW
1                   MWz    !if MWtime=1: constant time (=0) or constant ln(time) (=1) MW
110                 MWpumpt!if MWtime=1: time from Timein to end of pumping [T], MW
100                 TNM    !number of time vs concentration data points, MW
0                   Trest  ! Resting Time for the CFTT

0.100000d0         r0p    !radius of the pumping well  [L]
4.40d0             b1     !saturated thickness at injection well [L]
4.40d0             b2     !saturated thickness at pumping well [L]
1.d0               Cin    !injection conc. [M/L^3; if Cin=1, conc. dimensionless]
-6.45       mus    !initial guess (parameter estimation) mus [ln(1/T)]
4.3d0      sig    !initial guess (parameter estimation) sig
1.0000D+00         Rm     !mobile zone retardation [-]
1.0000D+00         Rim    !immobile zone retardation [-]
0.20d0             ptot   !maximum permitted total porosity (inversion parameter) [-]
2.628d-6           Daq    !aqueous diffusion coefficient of solute [L^/T]
0.11               tort   !diffusive tortuosity [-]
0.01               dr
1.87               fdn
0.000353           poros  !advective porosity [-]
0.17               pmat   !diffusive porosity


T                 cinject        ! pulse Source (T)
F                 SingR          ! If Single-Rate (T), else (F)
T                 MultR          ! If Multi-Rate (T), else (F)
F                 FixOmBet       ! If Input Omega and Beta (T), else (F)
F                 contsrc        ! Continuous source (T), else (F)
F                 pumpSol        ! Only Pumping Phase Solution (T), else (F)
0.8               tmin           ! Minimum Time
1172.45           tmax           ! Minimum Time
0.0               omegapf        ! Dimensionless Mass Transfer Rate Coeff.
0.0               betompf        ! Dimensionless Capacity Coeff.
0.01              alphar         ! Dimensional Mass Transfer Rate Coeff.


1                iest   !0=parameter estimation; 1=forward model only
0                idef   !0=Default distribution; 1 = User-defined distribution
0.0d0            disc   !
1000             kmax   !control permissible numerical error, etc.
1.d-6            relerr !
```

Appendix C

FORTRAN Code

Appendix C: FORTRAN Code

```
      PROGRAM GRTCFTT_FD

!=============================================================================
! GRTCFTT_FD was developed to model Generalized Radial Transport  for
! Interpreting Convergent Flow Tracer Tests using a Laplace Domain
! block-centered integrated Finite Difference Solution.

! Subroutines and functions that are used in Injection and resting phase
! part of the program was modified from STAMMT-R developed by  Roy Haggerty
! and Sean W. Fleming in 1997.
!
! The GRTCFTT_FD was developed and modified by Md Lal Mamud and
! Rbert M. Holt.
! Department of Geology and Geological Engineering
! The University of Mississippi
! May,  2019
!=============================================================================


! Calling MSIMSL
      INCLUDE 'link_fnl_static.h'
      INCLUDE 'link_fnl_shared.h'

      implicit none
      include "dimen.inc"

! Defining parameter types
      integer kmax,iest,iparam(6),i,j,n,is,skipm,idef
      integer TNS,TNM,nexttm
      double precision alphLm,r0i,r0p,Rm,Qin,Qout,poros,b1,b2,concin,
     &  Btot,sig,rmax,Ro,disc,relerr,mrmse,Tc0,TcE,mse,ptot,Qins,
     &  Qouts,Tc0s,TcEs,Timeins,Trest,rmaxs,mustar,alphLs,pmat
      double precision MWtime,MWz,MWdt,SWtime,SWz,SWdt,Tstart,Timein
      double precision tdp(ntm),cdp(ntm),rparam(7),xguess(nx),srmse,
     &  fjac(nt,nx),xscale(nx),fscale(nt),xf(nx),serr(nt),c(ntm),smse,
     &  cs(nts),tdps(nts),cdps(nts),ct,MWpumpt,SWpumpt,totmassins,mmse
      double precision XTXs(nx,nx),covariances(nx,nx),eigenvals(nx),
     &  XTXm(nx,nx),covariancem(nx,nx),eigenvalm(nx),fjacs(nts,nx),
     &  fjacm(ntm,nx),numerator,denominator,corcoefs(nx,nx)
      double precision corcoefm(nx,nx),inverses(nx,nx),inversem(nx,nx)
      double precision weightmat(nt,nt),XtW(nx,nt),XtWX(nx,nx),
     &  covariancesm(nx,nx),corcoefsm(nx,nx),eigenvalsm(nx)
      double precision Daq,tort,fdn,dr,Rim
      double precision omegapf,betompf, alphar
      double precision timeprt
      double precision tmaxL,tminL,tdiff,dlt,tim(maxt),tmin,tmax
        character*40 paramfile,trecovfile,swinputfile,outputbtc,
     &  outputabboti,outputinjdc,outputazavc,outputrestc,outputarcdf,
     &  progress,progressmw,statistics,outputabbotp
      character*40 distribution,cviewinj,cviewavg

      logical pflag,mw,cinject,SingR,MultR,FixOmBet,contsrc,pumpSol
       external obj,AVINT,zairy,zbiry,mtdef,distout
```

```
! Creating Common Blocks to pass variables within function and subroutine
      common/a/r0i,r0p,Qin,Qout,b1,b2,concin,Timein,
     &  Ro,disc,relerr,Tc0,TcE,Rm,Trest,pmat,rmax
      common/aa/cdp,tdp,ptot
      common/c/kmax,idef,skipm
      common/logical/pflag,mw
      common/multipoint/TNM,nexttm,totmassins
      common/progress/progress,progressmw
      common/userdefine/distribution
      common/grt/dr,Rim
      common/grt2/alphLm,poros,mustar,Btot,sig
      common/btcr/tim
      common/grt3/tmin,tmax
      common/grt4/omegapf,betompf, alphar
      common/grtp/fdn
      common/inlogical/cinject
      common/runtyp/SingR,MultR,FixOmBet,contsrc,pumpSol
      common/lal/outputarcdf


! Reading File names from the flname.dat input file
      open (10,file='flname.dat')     ! input file containig file name
      read (10,11) paramfile          ! input file containig parameters
      read (10,11) trecovfile         ! input file containing tracer test data
      read (10,11) outputbtc          ! Output file simulated data
      read (10,11) outputabboti       ! output file for alpha and beta (Injecition)
      read (10,11) outputabbotp       ! output file for alpha and beta (Pumping)
      read (10,11) outputinjdc        ! output file containing injected Conc
      read (10,11) outputazavc        ! output file containing azimuthal avg. Conc
      read (10,11) outputrestc        ! output file containing Conc. after resting
      read (10,11) progress           ! Output file containing estimaed parameters
      read (10,11) progressmw         ! Output file containing Calc Conc. and Obs Conc
      read (10,11) statistics         ! output file containign inversion statistics
      read (10,11) outputarcdf        ! Output containing lognormal distribution radii
      read (10,11) distribution       ! Input file containg user defined distributin


 11   format (A)

 ! Opening input files and creating output file for writing
      open (20,file=paramfile)
      open (21,file=trecovfile)
      open (22,file=outputbtc)
      open (23,file=outputabboti)
      open (73,file=outputabbotp)
      open (24,file=outputinjdc)
      open (25,file=outputazavc)
      open (26,file=outputrestc)


      pflag = .false.

! Reading Input Parameter from the params.dat input file:
      read (20,*) skipm           !do the CFTT simulation? 0 if yes, 1 if no
      read (20,*) Tc0             !start time for conc. inj. (can be 0), MW
      read (20,*) TcE             !elapsed time from t=0 to end of conc. inj., MW
      read (20,*) Timein          !elapsed time from t=0 to end of chaser inj., MW
      read (20,*) Qin             !injection rate, MW
```

```
read (20,*) Qout            !pumping rate, MW
read (20,*) alphLm          !longitudinal dispersivity for MW
read (20,*) rmax            !edge of grid for injection, MW
read (20,*) Ro              !distance from injection to pumping wells
read (20,*) r0i             !well radius (injection)
read (20,*) MWtime          !(t,C) input file (MWtime=0) or generate times
read (20,*) MWz             !if MWtime=1, constant time increment (MWz=0) or
read (20,*) MWpumpt         !if MWtime=1, time from Timein to end of pumping
read (20,*) TNM             !number of time vs concentration data points, MW
read (20,*) Trest           ! Resting time

read (20,*) r0p             !well radius (pumping for MW;)
read (20,*) b1              !saturated thickness at injection well
read (20,*) b2              !saturated thickness at pumping well
read (20,*) concin          !injection concentration
read (20,*) mustar          !intial guess of mus for multirate model)
read (20,*) sig             !initial guess of sigma for multirate model)
read (20,*) Rm              !mobile zone retardation
read (20,*) Rim             !immobile zone retardation [-]
read (20,*) ptot            !maximum total porosity permitted
read (20,*) Daq             !aqueous diffusion coefficient of solute
read (20,*) tort            !tortuosity of secondary (matrix) porosity
read (20,*) dr              !delta r [L]
read (20,*) fdn             !flow dimension n
read (20,*) poros           !primary porosity
read (20,*) pmat            !matrix porosity

read (20,*) cinject         !Logical- injection (T), boundary conc (F)
read (20,*) SingR           !If Single-Rate (T), else (F)
read (20,*) MultR           !If Multi-Rate (T), else (F)
read (20,*) FixOmBet        !If Input Omega and Beta (T), else (F)
read (20,*) contsrc         ! Continuous source (T), else (F)
read (20,*) pumpSol         ! Only Pumping Phase Solution (T), else (F)
read (20,*) tmin            ! Minimum Time
read (20,*) tmax            ! Minimum Time
read (20,*) omegapf         ! Dimensionless Mass Transfer Rate Coeff.
read (20,*) betompf         ! Dimensionless Capacity Coeff.
read (20,*) alphar           ! Dimensional Mass Transfer Rate Coeff.

read (20,*) iest            !0=parameter estimation; 1=forward model only
read (20,*) idef            !0=Logn. distribution; 1=User-defined distribution
read (20,*) disc            !
read (20,*) kmax            !see manual for these last three parameters.
read (20,*) relerr    !


if (skipm.eq.1) then
   Qin=1.0
   Qout=1.0
   alphLm=1.0
endif

Btot = pmat*Rim/poros*Rm

if ((skipm.eq.0).and.(TNM.gt.ntm)) then
  print*,'ERROR: TNM is greater than ntm (nominally 300)'
  print*,'Modify TNM, or recompile (and re-QA) with larger ntm'
  print*,'Aborting STAMMT-R run'
```

```
        stop
      endif


! READING OBSERVED BTC OR GENERATING TIMES INCREMENTED AT (M,S)Wdt INTERVALS
c   * the CFTT case *
      if (skipm.eq.0) then
c     Read the CFTT data:
      if (MWtime.eq.0) then
      do 20,i=1,TNM
        read(21,*) tdp(i),cdp(i)
 20   continue
      if (tdp(1).lt.Timein) then
      print*,'ERROR: first time in 2-well data is < Timein'
      print*,'Aborting run'
      print*,'Modify data and/or input parameter files'
      stop
      endif
c     Pad data from last data point to ntm=300
      nexttm=TNM+1
      do 21,i=nexttm,ntm
        tdp(i)=tdp(TNM)
        cdp(i)=0.01
 21   continue
c     Generate the CFTT data points, if necessary:
      else
        if (MWz.eq.0) then
          MWdt=MWpumpt/TNM
        else
          MWdt=log(MWpumpt)/TNM
        endif
        do 22, i=1,TNM
        ct=dble(i)
        tdp(i)= (Timein+ct*MWdt)*(1.d0-MWz)+MWz*(exp(ct*MWdt)+
     &     Timein-1.d0)
        cdp(i)=0.01
 22   continue
c     Pad "data" from last data point to ntm=300 with first value of tdp
c       to avoid large values of tdp which are not used anyway
      nexttm=TNM+1
      do 23,i=nexttm,ntm
        tdp(i)=tdp(TNM)
        cdp(i)=0.01
 23   continue
      endif
      endif


! Calling FHeader subroutine for writing file headers
      Call FHeader


! Estimation Section Start
         call ERSET(4,0,0)

       if (iest.eq.0) then
        xguess(1) = mustar
        xguess(2) = log(sig)
```

```
         xguess(3) = log(poros)
         xguess(4) = log(alphLm)
        xguess(5) = log(fdn)
        Btot = pmat/poros*Rm

! weighting parameters:
       do 40,i=1,nx
         xscale(i) = 1.d0
 40     continue
! weighting CFTT data:
        do 60,i=1,ntm
          fscale(i) = 1.d0
 60     continue
! weighting CFTT data:
        do 70,i=(ntm+1),nt
          if (skipm.eq.0) then
          fscale(i) = dble(ntm)/dble(nts)
          else
          fscale(i)=1.d0
          endif
 70    continue

        call du4lsf(iparam,rparam)
        iparam(2) = 5

! iparam(4) equal to the maximum number of iterations desired.
       iparam(4)= 10

        call dunlsf(obj,nt,nx,xguess,xscale,fscale,iparam,rparam,xf,
     &      serr,fjac,nt)

        mustar = xf(1)
        sig = exp(xf(2))
         poros = exp(xf(3))
         alphLm = exp(xf(4))
        fdn = exp(xf(5))
        Btot = pmat*Rim/poros*Rm

      endif

      pflag = .true.



! For Forward Simulation Only
! ------------------------------------------------------------------------------
      if(skipm.eq.0) call btc(Btot,mustar,sig,poros,alphLm,fdn,c)


! COMPUTING INVERSION STATISTICS, IF APPLICABLE -----------------------------
      Call inversion

! Lognormal distributions of block radii and diffusion rate coefficients
! for multi-rate runs.
       if ((sig.gt.(0.011d0)).and.(idef.eq.0)) then
         call distout(Daq,tort,mustar,sig)
       endif
```

```
! Clossing all open files------------------------------------------------------------

      close(20)
      close(21)
      close(22)
      close(23)
      close(24)
      close(25)
      close(28)
      close(30)

      stop

      END PROGRAM GRTCFTT_FD



C=! include files:
! ********************************************************************

Dimen.inc file:

c !!!! remember if you change this file to do "make clean"
c otherwise .o files will be kept that were compiled with the old
c include file !!!

      integer nt,nx,nm,nr,dcb,nts,ntm,maxt,maxr
      double precision pi
      parameter(pi=3.14159265358979323 85d0,ntm=108,nts=108,nr=25,nx=5,
     &  nm=35,dcb=50000,nt=ntm+nts,maxt=10000,maxr=1001)

c  ntm = max # data in CFTT problem
c  nts = max # data in single-well problem
c  nr = # grid nodes
c  nx = # of params. in estimation problem
c  nm = # first-order params. in diffusion problem


typscom.inc file:

      integer kmax,iest,iparam(6),i,j,n,is,skipm,idef
      integer TNS,TNM,nexttm
      double precision alphLm,r0i,r0p,Rm,Qin,Qout,poros,b1,b2,concin,
     &  Btot,sig,rmax,Ro,disc,relerr,mrmse,Tc0,TcE,mse,ptot,Qins,
     &  Qouts,Tc0s,TcEs,Timeins,Trest,rmaxs,mustar,alphLs,pmat
      double precision MWtime,MWz,MWdt,SWtime,SWz,SWdt,Tstart,Timein
      double precision tdp(ntm),cdp(ntm),rparam(7),xguess(nx),srmse,
     &  fjac(nt,nx),xscale(nx),fscale(nt),xf(nx),serr(nt),c(ntm),smse,
     &  cs(nts),tdps(nts),cdps(nts),ct,MWpumpt,SWpumpt,totmassins,mmse
      double precision XTXs(nx,nx),covariances(nx,nx),eigenvals(nx),
     &  XTXm(nx,nx),covariancem(nx,nx),eigenvalm(nx),fjacs(nts,nx),
     &  fjacm(ntm,nx),numerator,denominator,corcoefs(nx,nx)
      double precision corcoefm(nx,nx),inverses(nx,nx),inversem(nx,nx)
      double precision weightmat(nt,nt),XtW(nx,nt),XtWX(nx,nx),
     &  covariancesm(nx,nx),corcoefsm(nx,nx),eigenvalsm(nx)
      double precision Daq,tort,fdn,dr,Rim
      double precision omegapf,betompf, alphar
```

```
      double precision timeprt
      double precision tmaxL,tminL,tdiff,dlt,tim(maxt),tmin,tmax
        character*40 parameterfile,mwinputfile,swinputfile,outmw,
     &  outputmt,outputdivmw,outputconvert,outputrest,outputarcdf,
     &  progress,progressmw,statistics,outputabbotp
      character*40 distribution,cviewinj,cviewavg

      logical pflag,mw,cinject,SingR,MultR,FixOmBet,contsrc,pumpSol
       external obj,AVINT,zairy,zbiry,mtdef,distout
      common/a/r0i,r0p,Qin,Qout,b1,b2,concin,Timein,
     &  Ro,disc,relerr,Tc0,TcE,Rm,Trest,pmat,rmax
      common/aa/cdp,tdp,ptot
      common/c/kmax,idef,skipm
      common/logical/pflag,mw
       common/multipoint/TNM,nexttm,totmassins
      common/progress/progress,progressmw
      common/userdefine/distribution
      common/grt/dr,Rim
      common/grt2/alphLm,poros,mustar,Btot,sig
      common/btcr/tim
      common/grt3/tmin,tmax
      common/grt4/omegapf,betompf, alphar
      common/grtp/fdn
      common/inlogical/cinject
      common/runtyp/SingR,MultR,FixOmBet,contsrc,pumpSol
       common/lal/outputarcdf



! External Functions and Subroutines
! ================================================================================

       subroutine FHeader
!      This subroutine setting up headers for all of the output files

! SETTING UP HEADERS FOR SOME OUTPUT FILES-------------------------------------

      write(22,*) 'File-22: cbtc.out contains Simulated Times Vs.
     & Concentrations'
      write(22,*) 'while Pumping from the withdrawal well.'
      write(22,*) '1st column = elapsed time [T]'!
      write(22,*) '2nd column = calculated concentration [M/L^3]'
      write(22,*) '************************************************
     &************'


      write(23,*) 'File-23: abboti.out contains all alpha, Beta and
     & Btot for *Injection Phase*.'


      write(73,*) 'File-73: abbotp.out contains all alpha, Beta and
     & Btot for *Pumping Phase*.'

      write(24,*) 'File-24: cinjd.out contains concentrations at the end
     & of injection.'
      write(24,*) 'The radial coordinates is centered about the
     & *Injection well*'
      write(24,*) 'Column 1: radial distance from "Injection Well" [L]'
```

```
      write(24,*) 'Column 2: concentration in mobile zone [M/L^3]'
      write(24,*) 'Column 3: concentration in s(1)'
      write(24,*) 'Column 4: concentration in s(nm/3)'
      write(24,*) 'Column 5: concentration in s(2*nm/3)'
      write(24,*) 'Column 6: concentration in s(nm)'
      write(24,*) '**************************************************
     &**************************'


      write(25,*) 'File-26: cazav.out contains Azimuthally Averaged
     & concentrations'
      write(25,*) 'after the resting perios, coordiantes centered the
     & *Pupming well*.'
      write(25,*) 'Column 1: radial distance from pumping well [L]'
      write(25,*) 'Column 2: concentration in mobile zone [M/L^3]'
      write(25,*) 'Column 3: concentration in s(1)'
      write(25,*) 'Column 4: concentration in s(nm/3)'
      write(25,*) 'Column 5: concentration in s(2*nm/3)'
      write(25,*) 'Column 6: concentration in s(nm)'
      write(25,*) '**************************************************
     &**************************'


      write(26,*) 'This file contains concentrations at the end of'
      write(26,*) 'the rest period for the CFTTl run.'
      write(26,*) 'Column 1: radial distance from injection well [L]'
      write(26,*) 'Column 2: concentration in mobile zone [M/L^3]'
      write(26,*) 'Column 3: concentration in s(1)'
      write(26,*) 'Column 4: concentration in s(nm/3)'
      write(26,*) 'Column 5: concentration in s(2*nm/3)'
      write(26,*) 'Column 6: concentration in s(nm)'
      write(26,*) '**************************************************
     &**************************'

      END subroutine FHeader


! OBJECTIVE SUBROUTINE FOR ESTIMATION ROUTINE
      subroutine obj(nd,np,xf,serr)
      implicit none
      include "dimen.inc"
      integer kmax,np,nd,i,is,idef,skipm,skips,TNS,TNM,nexttm
      double precision Btot,sig,Rm,Rim,sct,poros,alphLm,ptot,ptm,
     &  alphLs,r0i,r0p,Qin,Qout,b1,b2,concin,Timein,
     &  rmaxs,Ro,disc,relerr,Tc0,Tc0s,TcE,TcEs,Trest,mustar,pmat
      double precision xf(nx),c(ntm),cdp(ntm),tdp(ntm),serr(nt),
     &  rmax,totmassins,smse,mmse,mrmse,srmse,fdn
      character*40,progress,progressmw,progresssw
      common/progress/progress,progressmw
      logical difnan
      external difnan
      common/a/r0i,r0p,Qin,Qout,b1,b2,concin,Timein,
     &  Ro,disc,relerr,Tc0,TcE,Rm,Trest,pmat,rmax
      common/aa/cdp,tdp,ptot
      common/c/kmax,idef,skipm
       common/multipoint/TNM,nexttm,totmassins
```

```
      mustar = xf(1)
      sig = exp(xf(2))
      poros = exp(xf(3))
      alphLm = exp(xf(4))
      fdn = exp(xf(5))
      Btot = pmat/poros*Rm


      open(30,file=progress,access='append')
      write(30,110) 'pmat =    ',pmat
      write(30,110) 'mustar = ',mustar
      write(30,110) 'sig =     ',sig
      write(30,110) 'poros =   ',poros
      write(30,110) 'alphLm = ',alphLm
      write(30,110) 'fdn = ',fdn
 110  format(A,d15.8,A,d15.8)
      close(30)


      if (skipm.eq.0) then
        call btc(Btot,mustar,sig,poros,alphLm,fdn,c)

! Calculating mean errors:
         mmse = 0.d0
         do 10,i=1,TNM
           mmse = mmse + log(cdp(i)/c(i))*log(cdp(i)/c(i))
 10      continue
         mrmse = sqrt(mmse/dble(TNM-nx-1))
        mmse = mrmse*mrmse

         open(41,file=progressmw)
         write(41,*) 'File 41, estfp: Breakthrough curve from the CFTT run,'
         write(41,*) 'Parameter Estimation:'
         write(41,160) '   Btot = ',Btot
        write(41,160) ' mustar = ',mustar
        write(41,160) '    sig = ',sig
        write(41,160) '  poros = ',poros
        write(41,160) ' alphLm = ',alphLm
        write(41,160) '    fdn = ',fdn
         write(41,160) '   RMSE = ',mrmse
 160    format(A,d12.5,A,d12.5)
         write(41,*) '1st column = elapsed time [T]'
         write(41,*) '2nd column = calculated concentration [M/L^3]'
         write(41,*) '3rd column = observed concentration [M/L^3]'
        write(41,*) '************************************************
     &************'
         do 200,i=1,TNM
          write(41,180) tdp(i)+TcE,c(i),cdp(i)
 180      format(3e21.13)
 200    continue
        close(41)
      endif


      sct = 0.d0

      ptm = poros*Rm*(1.d0+Btot)
```

```
      do 70,i=1,ntm
        if(skipm.eq.0) serr(i)=log(c(i)/cdp(i))
        if(skipm.eq.1) serr(i) = 0.d0
         if(i.gt.TNM) serr(i)=0.d0
         if ( difnan(serr(i)) ) serr(i) = 1.d2
           sct = sct + serr(i)*serr(i)
 70   continue

      open(30,file=progress,access='append')
!      write(30,*) 'pmat,mustar,sig,poros,alphLm,fdn,sct:'
!      write(30,'(7d15.8)') pmat,mustar,sig,poros,alphLm,fdn,sct
      write(30,*) '=================================================='
      close(30)

      return
      end



! ================================================================================
       subroutine inversion
      include "dimen.inc"
      include "typscom.inc"


! COMPUTING INVERSION STATISTICS, IF APPLICABLE ------------------------------
       if (iest.eq.0) then


! stats for the CFTT parameter estimation:-----------------------------------
       if (skipm.eq.0) then
         do 71,i=1,ntm
           do 46,j=1,nx
              fjacm(i,j)=fjac(i,j)
 46        continue
 71    continue
       call dmxtxf(ntm,nx,fjacm,ntm,nx,XTXm,nx)
       call dlinrg(nx,XTXm,nx,inversem,nx)
       do 37,i=1,nx
         do 38,j=1,nx
           numerator=inversem(i,j)
           denominator=sqrt(inversem(i,i)*inversem(j,j))
           corcoefm(i,j)=numerator/denominator
 38      continue
 37    continue
! compute covariance matrix:
       do 360,i=1,nx
         do 365,j=1,nx
           covariancem(i,j)=mrmse*mrmse*inversem(i,j)
 365     continue
 360   continue
       call devlsf(nx,covariancem,nx,eigenvalm)
       write(29,*) '2-WELL PARAMETER ESTIMATION STATISTICS'
       write(29,*) 'sensitivity matrix (Jacobian), X:'
       do 47,i=1,ntm
         write(29,41) (fjacm(i,j),j=1,nx)
```

```
 47     continue
        write(29,*) 'XtX (information matrix), square of sensitivity matrix:'
        do 48,i=1,nx
          write(29,41) (XTXm(i,j),j=1,nx)
 48     continue
        write(29,*) 'covariance matrix,(RMSE^2)*(inverse of XtX):'
        do 49,i=1,nx
          write(29,41) (covariancem(i,j),j=1,nx)
 49     continue
        write(29,*) 'correlation coefficients:'
        do 39,i=1,nx
          write(29,41) (corcoefm(i,j),j=1,nx)
 39     continue
        write(29,*) 'eigenvalues of covariance matrix:'
        do 31,i=1,nx
          write(29,*) eigenvalm(i)
 31     continue

        endif

! stats for SIMULTANEOUS  the CFTT inversion, if applicable:
        if (skipm.eq.0.) then
! construct weighting matrix (weightmat) the CFTT RMSE's:
        do 400,i=1,nt
          do 405,j=1,nt
            weightmat(i,j)=0.d0
 405      continue
 400    continue
        do 410,i=1,ntm
          weightmat(i,i)=mrmse*mrmse
 410    continue
        do 415,i=(ntm+1),nt
          weightmat(i,i)=srmse*srmse
 415    continue
c compute XtW, product of transpose of Jacobian with weighting matrix:
        call dmxtyf(nt,nx,fjac,nt,nt,nt,weightmat,nt,nx,nt,XtW,nx)
c compute XtWX, product of XtW with X:
        call dmrrrr(nx,nt,XtW,nx,nt,nx,fjac,nt,nx,nx,XtWX,nx)
c compute covariance matrix, inverse of XtWX:
        call dlinrg(nx,XtWx,nx,covariancesm,nx)
c calculate correlation coefficients:
        do 420,i=1,nx
          do 425,j=1,nx
            numerator=covariancesm(i,j)
            denominator=sqrt(covariancesm(i,i)*covariancesm(j,j))
            corcoefsm(i,j)=numerator/denominator
 425      continue
 420    continue
c compute eigenvalues of covariance matrix:
        call devlsf(nx,covariancesm,nx,eigenvalsm)
c write single well stats to file:
        write(29,*) 'SIMULTANEOUS SWIW/the CFTT PARAM. ESTIMATION STATS.'
        write(29,*) 'sensitivity matrix (Jacobian), X:'
        do 427,i=1,nt
          write(29,41) (fjac(i,j),j=1,nx)
 41     format(5e12.4)
 427    continue
        write(29,*) 'covariance matrix, inverse of XtWX:'
```

```
      do 430,i=1,nx
        write(29,41) (covariancesm(i,j),j=1,nx)
430   continue
      write(29,*) 'correlation coefficients:'
      do 435,i=1,nx
        write(29,41) (corcoefsm(i,j),j=1,nx)
435   continue
      write(29,*) 'eigenvalues of covariance matrix:'
      do 440,i=1,nx
        write(29,*) eigenvalsm(i)
440   continue

      endif

      endif

      END subroutine inversion


!   SUBROUTINE TO CALCULATE THE BREAKTHROUGH CURVE FOR THE CFTT
! ================================================================================
      subroutine btc(Btot,mustar,sig,poros,alphLm,fdn,c)
          implicit none
      include "dimen.inc"
      integer kmax,i,j,k,ii,jj,icbar,lapfn,iercd,n1rty,irule,idef,
     &  skipm,skips,maxi,nexttm,TNM

      double precision alphLm,r0i,r0p,Rm,Qin,Qout,poros,b1,b2,concin,
     &  Btot,mu,sig,rmax,Ro,disc,relerr,step,stepov,jjpp,up,low,Timein,
     &  beti,sum,Tnd1,Tnd2,a1,a2,ri,dnorin,dnordf,derf,rhomx,rho0,Tc0,
     &  TcE,Tc0nd,TcEnd,mass,dcsval,erfac,crt,rhopmn,rhopmx,a,drho,
     &  errabs,errrel,ct,rhop0,Rhod,rhoi,errest,tth,ptot,
     &  Trest,pmat,mustar,hm,rdi(nr),totmassins

      double precision tdp(ntm),cdp(ntm),bet(nm),omega(nm),c(ntm),
     &  Tnd3(ntm),Tnd3s(nts),c1(nr),s1(nr,nm),c2(nr),s2(nr,nm),
     &  alpha(nm),cscoef(4,nr),cscoefm(4,nr,(nm+1)),rho(nr),s(nr),
     &  break(nr),rho2(nr),ervc(8),betom(nm),maxc,rimax,
     &  dimt

      double precision fdn,dr,Rim,stepp,stepovp,jjppp,upp,lowp,
     &  betip,sump,Tnd1p,Tnd2p,a1p,a2p,rhomxp,rho0p,Tc0ndp,TcEndp,
     &  rhopmxp,rhopmnp,drhop,rhop0p,hmp,mup,bdi,bdp,Pe,drd

      double precision betp(nm),omegap(nm),Tnd3p(ntm),
     &  alphap(nm),cscoefp(4,nr),cscoefmp(4,nr,(nm+1)),
     &  betomp(nm),Rhodp,rhop(nr),rho2p(nr)

      double precision tmaxL,tminL,tdiff,dlt,tim(maxt),tmin,tmax
      double precision omegapf,betompf,alphar,omegapc

      double precision Z,sumz,Cic

      logical pflag,mw,cinject,SingR,MultR,FixOmBet,contsrc,pumpSol
! Common from main program ---------------------------------------------------
      common/a/r0i,r0p,Qin,Qout,b1,b2,concin,Timein,
     &  Ro,disc,relerr,Tc0,TcE,Rm,Trest,pmat,rmax
      common/aa/cdp,tdp,ptot
```

```
      common/grt/dr,Rim

! Common blocks created in this btc subrouitne ------------------------------
      common/b/betom,omega,Tnd1,Tnd2,Tnd3,Tnd3s,rhomx,rho0,rhop0,c2,s2,
     &  ri,Tc0nd,TcEnd,rhopmn,rhopmx,Rhod
      common/c/kmax,idef,skipm
      common/d/ii,jj
      common/f/icbar,lapfn
      common/g/alpha,c1,s1
      common/k/cscoefm,break,tth
      common/logical/pflag,mw
      common/l/betp,omegap,betomp,Tnd3p,alphap,cscoefmp
      common/l2/bdi,bdp,Pe,drd,rho2,rhomxp
      common/bet/bet
      common/mss/mass
      common/pm/rho0p,rhop0p
      common/multipoint/TNM,nexttm,totmassins
      common/btcr/tim
      common/grt3/tmin,tmax
      common/grt4/omegapf,betompf, alphar
      common/grt5/omegapc
      common/conp/Z,sumz,Cic
      common/inlogical/cinject
      common/runtyp/SingR,MultR,FixOmBet,contsrc,pumpSol

       external dnorin,dnordf,derf,pushcL,pushsL,pullcLG,
     &  dcsint,dcsval,n1rty,iercd,crt,mtdef


       mw = .true.
      tth = 2.d0/3.d0

      print*,'Non-dimensionalizing parameters for the Injecton Phase.'

! nondimensionalizing parameters for the injection pahse:
! ---------------------------------------------------------------------------
       mw = .true.
      tth = 2.d0/3.d0

c nondimensionalizing mustar to mu:
       mu=mustar-log(Qin/(pi*(b1+b2)*poros*Rm*alphLm*alphLm))

      do 4,i=1,nr
        c1(i) = 0.d0
       c2(i) = 0.d0
        do 3,j=1,nm
         s1(i,j) = 0.d0
         s2(i,j) = 0.d0
 3      continue
 4    continue

      a1 = 1.d0/(2.d0*pi*b1*poros*alphLm*alphLm*Rm)
      rhomx = rmax/alphLm
      rho0 = r0i/alphLm
      rhop0 = r0p/alphLm
      drho = (rhomx-rho0)/dble(nr-1)
      Tnd1 = Qin*Timein*a1
      Tc0nd = Qin*Tc0*a1
```

```
      TcEnd = Qin*TcE*a1
      do 10,i=1,ntm
        Tnd3(i) = Qout*(tdp(i)-Timein)*a1*b1*2.d0/(b1+b2)
 10   continue

      if (idef.eq.0) then

      if (sig.le.1.d-2) then
       hm = 0.d0
       sum = 0.d0
        do 12,i=1,nm-1
          omega(i) = exp(mu)*dble(i*i)*pi*pi
          bet(i) = 6.d0/(dble(i*i)*pi*pi)
         sum = sum + bet(i)
         hm = hm+bet(i)/omega(i)
 12      continue
       omega(nm) = (1.d0-sum)/(1.d0/(15.d0*exp(mu)) - hm)
       bet(nm) = 0.d0
        goto 41
      endif

      a2 = 1.d3*(Rm*(1.d0+Btot)+1.d0)/
     &  (Rm*(1.d0+Btot)*(1.d0+Btot)*(tth*rhomx+1.d0)*pi*pi)
      omega(nm)=Log(a2)
      omega(1) = mu+sig*dnorin(1.d-6)
      step = (omega(nm)-omega(1))/dble(nm-1)
      do 20,i=2,(nm-1)
        omega(i) = omega(1)+step*dble(i-1)
        omega(i) = pi*pi*exp(omega(i))
 20   continue
      omega(1) = pi*pi*exp(omega(1))
      omega(nm) = pi*pi*exp(omega(nm))


! Set up vector of capacity ratios (bet(i)).  See text of
!  Chapter 4 (Haggerty, 1995) to see functions evaluated here.
! ---------------------------------------------------------------------------
      step = exp(step/2.d0)
      stepov = 1.d0/step
      do 25,i=1,nm
        bet(i) = 0.d0
 25   continue
      do 40,j=1,1000
        jjpp = dble(2*j-1)*dble(2*j-1)*pi*pi
        do 30,i=2,(nm-1)
         up = Log((4.d0*omega(i)*step)/jjpp)
         low = Log((4.d0*omega(i)*stepov)/jjpp)
          beti =4.d0*(dErf((mu-low)/(sqrt(2.d0)*sig))-
     &          dErf((mu-up)/(sqrt(2.d0)*sig)))/jjpp
          bet(i) = bet(i) + beti
 30      continue
        up = Log((4.d0*omega(1)*step)/jjpp)
        beti = 4.d0*(1.d0 - dErf((mu-up)/(sqrt(2.d0)*sig)))/jjpp
        bet(1) = bet(1) + beti
        low = Log((4.d0*omega(nm)*stepov)/jjpp)
        beti = 4.d0*(dErf((mu-low)/(sqrt(2.d0)*sig))+ 1.d0)/jjpp
        bet(nm) = bet(nm) + beti
 40   continue
```

```
 41   continue

!  Renormalize bet(i) values so that sum is correct:
! ------------------------------------------------------------------------
      sum = 0.d0
      do 50,i=1,nm
        bet(i) = Btot*bet(i)
       sum = sum + bet(i)
 50   continue
      bet(nm) = bet(nm)+Btot-sum

      else
       call mtdef(btot,mustar,sig,poros)
       do 51,i=1,nm
         omega(i) = alpha(i)/(Qin*a1)
 51     continue
      endif

      do 52,i=1,nm
        betom(i) = bet(i)*omega(i)
 52   continue

! Write to the File-23 abboti.out:  all Alpha, Beta and Btot.
! ------------------------------------------------------------------------------
      write(23,*) 'Column 1: alpha(i) [1/T]'
      write(23,*) 'Column 2: beta(i) [dimensionless]'
      write(23,*) 'Column 3: cummulative sum of beta(i) [dimensionless]'
      write(23,*) 'Last number in column 3 = Btot'
      write(23,*) '**************************************************
     &**********************'
      sum = 0.d0
      do 55,i=1,nm
        sum = sum + bet(i)
        write(23,53) (omega(i)*Qin*a1),bet(i),sum
 53     format(3e25.10)
 55   continue
       do 60,i=1,nr
       rho(i) = rho0 + dble(i-1)*drho
 60   continue

       print*,'   > injecting for the CFTT simulation.'

!  Do inverse Laplace transform for divergent flow from inj. well:
! ---------------------------------------------------------------------------
      do 80,i=1,nr
         icbar = 0
        ri = rho(i)
        call dinlap (pushcL,1,Tnd1,disc,relerr,kmax,c1(i))
       c1(i) = c1(i)*concin
       if(c1(i).lt.0.d0) c1(i) = 0.d0
       do 70,j=1,nm
         jj = j
          lapfn = 0
         call dinlap (pushsL,1,Tnd1,disc,relerr,kmax,s1(i,j))
         s1(i,j) = s1(i,j)*concin
         if(s1(i,j).lt.0.d0) s1(i,j) = 0.d0
 70    continue
```

```
! File-24: write down concentrations at the end of injection.
! ------------------------------------------------------------------------------
       if(pflag)then
        write(24,75) ri*alphLm/Ro,c1(i),s1(i,1),s1(i,nm/3),
     &      s1(i,2*nm/3),s1(i,nm)
 75      format(6d15.5)
       endif

 80   continue


! write to view conc after injection ------------------------------------------
      Do i = 1,nr
          rdi(i) = rho(i)*alphLm
          Do j =1, nm
              write(79,*) rdi(i),c1(i),S1(i,j)
          End Do
      End Do


! checking whether rmax is too big or too small:
! ------------------------------------------------------------------------------
      if(pflag)then
        call calc_mass(c1,s1,rho,b1,poros,alphLm,Rm,bet,mass)
        write(24,*)'Total mass in aquifer = ',mass
      endif

      print*,'Done injection Phase...!!!.'


! Calculate Concentration after Coordinates Transformation respect ot the
! the pumping well after Azimuthal Averaging.
! ==============================================================================

       print*,'    Doing Averaging and Coordinates Transformation'

      call dcsint(nr,rho,c1,break,cscoef)

      do 83,i=1,nr
        do 82,j=1,4
         cscoefm(j,i,1) = cscoef(j,i)
 82     continue
 83   continue
      do 87,k=1,nm
        do 84,j=1,nr
         s(j) = s1(j,k)
 84     continue
        call dcsint(nr,rho,s,break,cscoef)
       do 86,i=1,nr
          do 85,j=1,4
           cscoefm(j,i,(k+1)) = cscoef(j,i)
 85       continue
 86     continue
 87   continue


      Rhod = Ro/alphLm
```

```
      rhopmn = Rhod - rho(nr)
      rhopmx = Rhod + rho(nr)
      A = asin(rho(nr)/Rhod)
      errabs = 1.d-9
      errrel = 1.d-7
      irule = 2

       do 100,i=1,nr
        rho2(i) = rhopmn+(rhopmx-rhopmn)*dble(i-1)/dble(nr-1)
         ri = rho2(i)
         jj = 0

         call dqdag(crt,0.d0,A,errabs,errrel,irule,ct,errest)

         c2(i) = ct*2.d0*b1/(Pi*(b1+b2))
         if(c2(i).lt.0.d0) c2(i) = 0.d0
         do 90,j=1,nm
          jj = j
          call dqdag(crt,0.d0,A,errabs,errrel,irule,ct,errest)

          s2(i,j) = ct*2.d0*b1/(Pi*(b1+b2))
          if(s2(i,j).lt.0.d0) s2(i,j) = 0.d0
 90     continue

! File-26: write down concentrations after azimuthall averaging.
! ------------------------------------------------------------------------------
        if(pflag)then
          write(25,75) (ri*alphLm/Ro),c2(i),s2(i,1),s2(i,nm/3),
     &    s2(i,2*nm/3),s2(i,nm)
        endif
 100   continue

      if(pflag)then
       call calc_mass(c2,s2,rho2,(b1+b2)/2.d0,poros,alphLm,Rm,bet,mass)
        write(25,*)'Total mass in aquifer = ',mass
      endif

      print*,'Done Averaging and Coordinates Transformation...!'

! write to check distribution after averaging---------------------------------------
      Do i = 1,nr
          rdi(i) = rho2(i)*alphLm/Ro
          Do j =1, nm
              write(89,*) rdi(i),c2(i),S2(i,j)
          End Do
      End Do


      print*,'Non-dimensionalizing parameters for the Pumping Phase.'

! Nondimensionalizing Parameters for Pumping Phase
! ==============================================================================

       mup=mustar-log((Qout)/(poros*Ro*Ro*Ro))

      a1p = 1.d0/(poros*Ro*Ro*Ro)
      rhomxp = rmax/Ro
      rho0p = r0i/Ro
```

```
      rhop0p = r0p/Ro
      bdi = b1/Ro
      bdp = b2/Ro
      Pe = Ro/alphLm

      omegapc = (alphar*poros*Rm*Ro*Ro*Ro)/Qout

      drd = dr/Ro
      drhop = (rhomx-rho0)/dble(nr-1)
      Tnd1p = Qin*Timein*a1
      Tc0ndp = Qin*Tc0*a1
      TcEndp = Qin*TcE*a1
      do 101,i=1,ntm
        Tnd3p(i) = Qout*(tdp(i)-Timein)*a1p*bdi*2.d0/(bdi+bdp)
 101  continue

      if (idef.eq.0) then

      if (sig.le.1.d-2) then
       hmp = 0.d0
       sump = 0.d0
        do 102,i=1,nm-1
          omegap(i) = exp(mup)*dble(i*i)*pi*pi
          betp(i) = 6.d0/(dble(i*i)*pi*pi)
         sump = sump + betp(i)
         hmp = hmp+betp(i)/omegap(i)
 102   continue
       omegap(nm) = (1.d0-sump)/(1.d0/(15.d0*exp(mup)) - hmp)
       betp(nm) = 0.d0
        goto 411
      endif


      a2p = 1.d3*(Rm*(1.d0+Btot)+1.d0)/
     &  (Rm*(1.d0+Btot)*(1.d0+Btot)*(tth*rhomxp+1.d0)*pi*pi)
      omegap(nm)=Log(a2)
      omegap(1) = mup+sig*dnorin(1.d-6)
      stepp = (omegap(nm)-omegap(1))/dble(nm-1)
      do 103,i=2,(nm-1)
        omegap(i) = omegap(1)+stepp*dble(i-1)
        omegap(i) = pi*pi*exp(omegap(i))
 103   continue
      omegap(1) = pi*pi*exp(omegap(1))
      omegap(nm) = pi*pi*exp(omegap(nm))


!  Set up vector of capacity ratios (bet(i)).
! ------------------------------------------------------------------------------
      stepp = exp(stepp/2.d0)
      stepovp = 1.d0/stepp
      do 104,i=1,nm
        betp(i) = 0.d0
 104  continue
      do 105,j=1,1000
        jjppp = dble(2*j-1)*dble(2*j-1)*pi*pi
        do 106,i=2,(nm-1)
         upp = Log((4.d0*omegap(i)*stepp)/jjppp)
         lowp = Log((4.d0*omegap(i)*stepovp)/jjppp)
```

```
          betip =4.d0*(dErf((mup-lowp)/(sqrt(2.d0)*sig))-
     &          dErf((mup-upp)/(sqrt(2.d0)*sig)))/jjppp
          betp(i) = betp(i) + betip
 106    continue
        upp = Log((4.d0*omegap(1)*stepp)/jjppp)
        betip = 4.d0*(1.d0 - dErf((mup-upp)/(sqrt(2.d0)*sig)))/jjppp
        betp(1) = betp(1) + betip
        lowp = Log((4.d0*omegap(nm)*stepovp)/jjppp)
        betip = 4.d0*(dErf((mup-lowp)/(sqrt(2.d0)*sig))+ 1.d0)/jjppp
        betp(nm) = betp(nm) + betip
 105  continue

 411   continue

!  Renormalize bet(i) values so that sum is correct.
! ------------------------------------------------------------------------------------
--

      sump = 0.d0
      do 107,i=1,nm
        betp(i) = Btot*betp(i)
       sump = sump + betp(i)
 107  continue
      betp(nm) = betp(nm)+Btot-sump

      else
       call mtdef(Btot,mustar,sig,poros)
       do 108,i=1,nm
         omega(i) = alpha(i)/(Qin*a1)
 108     continue
      endif

      do 109,i=1,nm
        betomp(i) = betp(i)*omegap(i)
 109  continue
      write(73,*) 'Column 1: alpha(i) [1/T]'
      write(73,*) 'Column 2: beta(i) [dimensionless]'
      write(73,*) 'Column 3: cummulative sum of beta(i) [dimensionless]'
      write(73,*) 'Last number in column 3 = Btot'
      write(73,*) '**************************************************
     &**********************'
      sump = 0.d0
      do 111,i=1,nm
        sump = sump + betp(i)
        write(73,54) (omegap(i)*Qin*(a1p/Rm)),betp(i),sump
 54     format(3e25.10)
 111   continue


      print*,'   < Pumping for the CFTT simulation.'


! Pumping Phase Calculation
! ============================================================================


       do 110,i=1,nm
        omegap(i) = omegap(i)*Qin*(bdi+bdp)/(2.d0*Qout*bdi)
```

```
 110  continue

      do 120,i=1,nm
        betomp(i) = betp(i)*omegap(i)
 120  continue


! Cubic Spline Interpolation for pumping -------------------------------------

      do i=1,nr
       rhop(i) = rho0p + dble(i-1)*drhop
      end do

      Rhodp = Ro/Ro
      rhopmnp = Rhodp - rhop(nr)
      rhopmxp = Rhodp + rhop(nr)

       do i=1,nr
        rho2p(i) = rhopmnp+(rhopmxp-rhopmnp)*dble(i-1)/dble(nr-1)
      end do


      call dcsint(nr,rho2p,c2,break,cscoefp)
      do 130,i=1,nr
        do 128,j=1,4
         cscoefmp(j,i,1) = cscoefp(j,i)
 128    continue
 130  continue

      do 170,k=1,nm
        do 140,j=1,nr
         s(j) = s2(j,k)
 140    continue
        call dcsint(nr,rho2p,s,break,cscoefp)
       do 160,i=1,nr
          do 150,j=1,4
           cscoefmp(j,i,(k+1)) = cscoefp(j,i)
 150      continue
 160    continue
 170  continue

 ! Calculate times-----------------------------------------
       tmaxL=dlog10(tmax)
!      tminL=dlog10(tmin+TcE+Timein)
      tminL=dlog10(tmin)
      tdiff=tmaxL-tminL
       dlt=tdiff/(dfloat(TNM-1))
       do i = 1,TNM
         tim(i)=10.0**(tminL+dlt*dfloat(i-1))
      end do


!  Do inverse Laplace transform for pumping period:
      erfac = 1.d1


 180  continue
       call dinlap (pullcLG,TNM,tim,disc,erfac*relerr,kmax,c)
```

```
       if (n1rty(1).ne.0) then
           erfac = erfac*1.d1
          goto 180
       endif

! Writing Output Data file for Continuous Source

      if(contsrc) then
         do i=1,TNM
               write(22,240) tim(i),c(i)
 240          format(2e25.15)
         end do
      end if

! Writing Output Data file and Dimensionalizing Conc by Z (Pumping Phase Sol.)
      if(pumpSol) then
         do i=1,TNM
               write(22,240) tim(i),c(i)/Cic
 241          format(2e25.15)
         end do
      end if

! Writing Output Data file and Dimensionalizing Conc by Z (Three Phase Sol.)
      if(cinject) then
         do i=1,TNM
               write(22,240) tim(i),c(i)/Z
 242          format(2e25.15)
         end do
      end if


       print*,'Congratulations...!!!  Done pumping for the
     & CFTT simulation.......!!!'

        write(*,*) 'CFTT Simulation Summary: '
        write(*,*) '-------------------------------------------------'
!        write(*,*) 'Simulated Time Vs. Conc Data File is cbtc.out'
        write(*,*) '    n =', fdn
        write(*,*) '  Pe =', Pe
        write(*,*) '  AP =', poros
        write(*,*) '  DP =', pmat
        write(*,*) 'Btot =', pmat/poros
        write(*,*) '  mu =', mustar
        write(*,*) 'Sig =', sig

        write(*,*) '************************************************
     &**********************'

       return

      end


! ************************************************************************
      complex*16 function pushcL(s)
      implicit none
      include "dimen.inc"
      integer i,icbar,lapfn,nz,ierr
```

```
      complex*16 s,pp,y,y0,Lcinj,cbar(dcb,2),ppth,Ay0,Apy0,Ay,pph,
     &  pushcL1,Air,AiP
      double precision betom(nm),omega(nm),Tnd1,Tnd2,Tnd3(ntm),rhomx,
     &  rho0,c2(nr),s2(nr,nm),ri,Tc0nd,TcEnd,rhop0,rhopmn,rhopmx,Rhod,
     &  argi,argr,fi,fr,tnd3s(nts)
      common/b/betom,omega,Tnd1,Tnd2,Tnd3,Tnd3s,rhomx,rho0,rhop0,c2,s2,
     &  ri,Tc0nd,TcEnd,rhopmn,rhopmx,Rhod
      common/e/cbar
      common/f/icbar,lapfn
      external zairy

      icbar=icbar+1
      if(icbar.gt.dcb)then
        print*,'icbar gt dcb in pushcL.  Increase dcb'
       stop
      endif
      pp = cmplx(1.d0,0.d0)
      do 10,i=1,nm
       pp = pp + betom(i)/(s+omega(i))
10    continue

      pp = s*pp
      ppth = pp**(1.d0/3.d0)
      pph = sqrt(pp)
      y = ri + 1.d0/(4.d0*pp)
      y0 = rho0 + 1.d0/(4.d0*pp)

      argr = dble(y0*ppth)
      argi = dimag(y0*ppth)
      call zairy(argr,argi,0,2,fr,fi,nz,ierr)
      Ay0 = cmplx(fr,fi)
      call zairy(argr,argi,1,2,fr,fi,nz,ierr)
      Apy0 = cmplx(fr,fi)
      argr = dble(y*ppth)
      argi = dimag(y*ppth)
      call zairy(argr,argi,0,2,fr,fi,nz,ierr)
      Ay = cmplx(fr,fi)

      Lcinj = ( exp(-Tc0nd*s)-exp(-TcEnd*s) )/s
      pushcL = Lcinj *
     &  exp(0.5d0*(ri-rho0)+2.d0/3.d0*pph*(y0**1.5d0-y**1.5d0)) *
     &  Ay/(0.5d0*Ay0-ppth*Apy0)
      cbar(icbar,1) = s
      cbar(icbar,2) = pushcL


      return
      end


! ********************************************************************
      complex*16 function pushsL(s)
      implicit none
      include "dimen.inc"
      integer i,ii,jj,icbar,lapfn
      complex*16 s,cbars,cbar(dcb,2),pushcL
      double precision betom(nm),omega(nm),Tnd1,Tnd2,Tnd3(ntm),rhomx,
     &  rho0,c2(nr),s2(nr,nm),ri,Tc0nd,TcEnd,rhop0,rhopmn,rhopmx,Rhod,
```

```
     &  Tnd3s(nts)
      common/b/betom,omega,Tnd1,Tnd2,Tnd3,Tnd3s,rhomx,rho0,rhop0,c2,s2,
     &  ri,Tc0nd,TcEnd,rhopmn,rhopmx,Rhod
      common/d/ii,jj
      common/e/cbar
      common/f/icbar,lapfn
      external pushcL

c  There must be a better way to do this search!! (At least, I hope so-
c  this is going to be very expensive.)
      lapfn = lapfn+1
      if (cbar(lapfn,1).eq.s) then
        cbars = cbar(lapfn,2)
        goto 20
      endif
      do 10,i=1,icbar
        if (cbar(i,1).eq.s) then
         cbars=cbar(i,2)
         goto 20
       endif
 10   continue
      cbars = pushcL(s)
 20   continue

      pushsL = omega(jj)*cbars/(s+omega(jj))

      return
      end



! ***********************************************************************
      double precision function crt(theta)
      implicit none
      include "dimen.inc"
      integer i,j,ii,jj
      double precision theta,ri,rx,Rhod,dcsval,Tnd1,Tnd2,rhomx,rho0,
     &  rhop0,Tc0nd,TcEnd,rhopmn,rhopmx,tth
      double precision rho(nr),c(nr),cscoefm(nr,4,(nm+1)),cscoef(nr,4),
     &  betom(nm),omega(nm),Tnd3(ntm),c2(nr),s2(nr,nm),break(nr),
     &  Tnd3s(nts)
      common/b/betom,omega,Tnd1,Tnd2,Tnd3,Tnd3s,rhomx,rho0,rhop0,c2,s2,
     &  ri,Tc0nd,TcEnd,rhopmn,rhopmx,Rhod
      common/d/ii,jj
      common/k/cscoefm,break,tth
      external dcsval

      rx = sqrt(ri*ri-2.d0*ri*Rhod*cos(theta)+ Rhod*Rhod)
      if ((rx.gt.rhomx) .or. (rx.lt.rho0)) then
        crt = 0.d0
      else

        if (jj.eq.0) then
          do 20,i=1,nr
            do 10,j=1,4
             cscoef(i,j) = cscoefm(i,j,1)
 10         continue
 20       continue
```

```
        else
          do 40,i=1,nr
            do 30,j=1,4
             cscoef(i,j) = cscoefm(i,j,(jj+1))
30          continue
40        continue
        endif

        crt = dcsval(rx,(nr-1),break,cscoef)
      endif

      return
      end



! ***********************************************************************
      subroutine calc_mass(c,s,rho,b,poros,alphL,Rm,bet,mass)
      implicit none
      include "dimen.inc"
      integer i,j
      double precision c(nr),s(nr,nm),bet(nm),rho(nr),b,poros,mass,a1,
     &  a2,r1,r2,c1,c2,s1,s2,alphL,Rm

      mass = 0.d0
      do 10,i=1,nr-1
        r1 = rho(i)
       r2 = rho(i+1)
       c1 = c(i)
       c2 = c(i+1)
       a1 = (c1 - r1*(c2-c1) / (r2-r1)) * (r2**2 - r1**2)
       a2 = 2.d0*(c2-c1)*(r2**3 - r1**3) / (3.d0*(r2-r1))
       mass = mass + (a1+a2)
10    continue

      do 30,i=1,nr-1
        r1 = rho(i)
       r2 = rho(i+1)
        do 20,j=1,nm
          s1 = s(i,j)
          s2 = s(i+1,j)
          a1 = (s1 - r1*(s2-s1)/(r2-r1)) * (r2**2 - r1**2)
          a2 = 2.d0*(s2-s1)* (r2**3 - r1**3) / (3.d0*(r2-r1))
          mass = mass + bet(j)*(a1+a2)
20      continue
30    continue

       mass = mass*pi*b*poros*alphL*alphL*Rm
      return
      end
```

```
!*******************************************************************************************
       complex*16 function pullcLG(s)
      implicit none
      include "dimen.inc"

      complex*16 s,rr,D2(maxr-1),ssp(nm),F,Cold(maxr-1),
     &  A(3,maxr-1),C(maxr-1),A3,D1(maxr-1),D3(maxr-1)
      integer i,j,k,NR1,nr2,nr3,kmax,idef,skipm
      double precision scfac,gam,cxsi,A1,A2,break(nr),xsi(nr)
      double precision Daq,tort,fdn,dr,Rim,rd,dcsval,sxsi,Cic,
     &  router,rinner,rhomxp,tmin,tmax
      double precision K1,K2,K3,Dload
      double precision Z,nrb, rout, rin,sumz
      double precision omegapf,betompf,alphar,omegapc,sc

      double precision betom(nm),omega(nm),Tnd1,Tnd2,Tnd3,Tnd3s,rhomx,
     &  rho0,rhop0,c2(nr),s2(nr,nm),ri,Tc0nd,TcEnd,rhopmn,rhopmx,Rhod

      double precision betp(nm),omegap(nm),Tnd3p(ntm),betomp(nm),
     &  alphap(nm),cscoefp(nr,4),cscoefmp(nr,4,(nm+1)),mass
      double precision r0i,r0p,Qin,Qout,b1,b2,concin,Timein,
     &  disc,relerr,Tc0,TcE,Rm,Trest,pmat,rmax,alphLm,Ro,poros,
     &  mustar,btot,sig,bdi,bdp,Pe,drd,rho0p,rhop0p,rho2(nr),bet(nm)

      logical pflag,mw,cinject,SingR,MultR,FixOmBet,contsrc,pumpSol

! Common from main program ------------------------------------------------------------
      common/a/r0i,r0p,Qin,Qout,b1,b2,concin,Timein,
     &  Ro,disc,relerr,Tc0,TcE,Rm,Trest,pmat,rmax
      common/c/kmax,idef,skipm
      common/grt/dr,Rim
      common/grtp/fdn
      common/grt2/alphLm,poros,mustar,btot,sig
      common/grt3/tmin,tmax,sc
      common/grt4/omegapf,betompf, alphar
      common/grt5/omegapc
      common/inlogical/cinject
      common/conp/Z,sumz,Cic
      common/runtyp/SingR,MultR,FixOmBet,contsrc,pumpSol


! Common blocks from btc subrouitne --------------------------------
      common/b/betom,omega,Tnd1,Tnd2,Tnd3,Tnd3s,rhomx,rho0,rhop0,c2,s2,
     &  ri,Tc0nd,TcEnd,rhopmn,rhopmx,Rhod
      common/l/betp,omegap,betomp,alphap,cscoefmp
      common/l2/ bdi,bdp,Pe,drd,rho2,rhomxp
      common/mss/mass
      common/pm/rho0p,rhop0p
      external dcsval,dlsacb

   ! LAL
! Calculate the scaling factor: ------------------------------------------------------------
---
      gam=gamma(fdn/2.0)
       scfac = 2.0*(pi**(fdn/2.0))*(bdp**(3-fdn))/gam


!  Calculation for the constant R :-----------------------------------------------
```

```
      rr = cmplx(1.d0,0.d0)
      do i=1,nm
          rr = rr + betomp(i)/(s+omegap(i))
      end do
      rr = s*rr

! Determine the node number ------------------------------------
       Nr1 = idnint((Ro/Ro)/dr)
!      NR1 = idnint(((Ro/Ro)*2)/dr)
      nrb = idnint(rhomxp/dr)
      nr3 = (Nr1)
      nr2 = (Nr1*2)

! Calculating Diagonals:
!
================================================================================================
! Calculate Constant A3 (independent of r). You need to comment-out/uncomment following
!      three calculation based on transport type.

! Calculating A3 for Multi-rate Transport
        if(MultR) then
          A3 = scfac*rr
        end if

! Calculating A3 for Single-rate Transport (input Omega and Beta)

        if(FixOmBet) then
            A3 = s*scfac+(betompf*s*omegapf*scfac/(s+omegapf))
        end if

! Calculating A3 for Single-rate Transport (Calculating Omega and Beta)
        if(SingR) then
            A3 = s*scfac+(Btot*s*omegapc*scfac/(s+omegapc))
        end if

! Loop to create diagonals of coefficient matrix  ---------------------------------------
-

      do i=2,nr2-1
          rd = dr*(dfloat(i)-1.0)+rhop0p+dr/2.0
          A1 = 1.0/(Pe*(rd**(fdn-1.0)))
          A2 = 1.0/(rd**(fdn-1.0))

          K1 = A1/(dr**2.0)
          K2 = A2/(2.0*dr)

! Calculate Tridiagonal Matrix ---------------------------------------------------------
          D1(i-1) = K1-K2
          D2(i) = -2.0*K1-A3
          D3(i+1) = K1+K2

      end do

! Calculate first entries of tridiagonal matrix: ---------------------------------
         A1 = 1.0/(Pe*((rhop0p+dr/2.0)**(fdn-1.0)))
         A2 = 1.0/((rhop0p+dr/2.0)**(fdn-1.0))

        K1 = A1/(dr**2.0)
```

```
        K3 = A2/dr

         D3(1) = (0.0,0.0)
        D3(2) = K1+K3
         D2(1) = -K1-K3-A3

! Calculate last entries of tridiagonal matrix (point nr3-1)-------------------------
         rd = dr*(dfloat(nr2)-1.0)+rhop0p+dr/2.0
         A1 = 1.0/(Pe*(rd**(fdn-1.0)))
         A2 = 1.0/(rd**(fdn-1.0))

        K1 = A1/(dr**2.0)
        K3 = A2/dr

        D1(nr2-1) = K1 - (2.0/3.0)*K3
        D1(nr2) = (0.0,0.0)
         D2(nr2) = -3.0*K1 - A3

! Calculate constant for the load vector (Continuous Source) -------------------------
        rd = dr*(dfloat(nr2)-1.0)+rhop0p+dr/2.0
         A1 = 1.0/(Pe*(rd**(fdn-1.0)))
         A2 = 1.0/(rd**(fdn-1.0))

        K1 = A1/(dr**2.0)
        K3 = A2/dr

        Dload = 2.0*K1 + (2.0/3.0)*K3

! Calculate the initial concentration in node nr3 for the  Pulse Source -------------
        ! Calculate the radii of the surfaces bounding nodes nr3
          rout = (nr3+nrb)
          rin = (nr3-nrb)
           router = rhop0p + (rout)*dr
           rinner = rhop0p + (rin)*dr

        ! Calculate the initial concentration if you want to use
        ! only the pumping phase solution
         Cic = fdn/(poros*scfac*((router**fdn)-(rinner**fdn)))

! Setup initial condition for all interior nodes ----------------------------------
        do i=2,nr2-1
          Cold(i)= (0.0,0.0)
        end do

!  Calculation for the constant Z for the load vector:--------------------------------

        xsi = Rho2*alphLm/Ro
!        xsi = rd

        do i=1,nm
           ssp(i) = betomp(i)/(s+omegap(i))
      end do

      do i=1,nr
          do j=1,4
           cscoefp(i,j) = cscoefmp(i,j,1)
          end do
      end do
```

```
        cxsi = dcsval(xsi,(nr-1),break,cscoefp)
        F = cmplx(0.d0,0.d0) + cxsi

        sumz = 0.0
        do i=1,nm
          do j=1,nr
            do k=1,4
             cscoefp(j,k) = cscoefmp(j,k,1+i)
            end do

          end do
          sxsi = dcsval(xsi,(nr-1),break,cscoefp)

          Z = F + ssp(i)*sxsi
          sumz = sumz + Z
        end do

  !     write(*,*) sumz


! Setup initial condition for all interior nodes -------------------------------------
        do i=2,nr2-1
          Cold(i)= (0.0,0.0)
        end do

! Calculate load vector value for first and last node --------------------------------
      ! Calculating Initial and the last node conc. for continuous source
      if(contsrc) then
        Cold(1) = (0.0,0.0)                   ! Load for the first node
        Cold(nr2) = - concin * Dload/s        ! Load for the last node
      end if

      ! Calculating Initial and the last node conc. for pulse type source incorporating
      ! Injection-Resting-Ppumping Phase Solution
      if(cinject) then
          do i = rin,rout
              Cold(1) = (0.0,0.0)             ! Load for the first node
              Cold(i)= - Z * scfac            ! Load for the last nodes
          end do
      end if

      ! Calculating Initial and the last node conc. for pulse type source
      ! Injection-Resting-Ppumping Phase Solution
      if(pumpSol) then
              Cold(1) = (0.0,0.0)             ! Load for the first node
              Cold(i)= - Cic * scfac            ! Load for the last nodes
      end if

!      Calculate concentrations in Laplace space - Solution using LU
!      factorization (must first create coeff. matrix)

      do i=1,nr2
           A(1,i) = D3(i)
           A(2,i) = D2(i)
          A(3,i) = D1(i)
      end do
```

```
      call dlsacb(nr2,A,3,1,1,Cold,1,C)

       pullcLG = C(1)


      return
      end



!**********************************************************************
!    SUBROUTINE TO CALCULATE DISTRIBUTIONS OF DIFFUSION RATE COEFFICIENTS AND
CORRESPONDING
!    BLOCK RADII:
      subroutine distout(Daq,tort,mustar,sig)

       implicit none
       double precision expnt,ratecf(400),a(400),pdf(400),cdf(400),Daq,tort,
     & mustar,sig,pi,part1,part2,dcdf,ratecfo,ratecfn
       integer i,j,k,number,IERR

      character*40 outputarcdf
      common/lal/outputarcdf

       external AVINT


      open (36,file=outputarcdf)

       pi=3.141592654

       number=400

c Caluclating a pdf using known values of mu and sigma and constructed range
c of Da/a^2:
       do 10,i=1,number
         expnt=(dble(i)-(dble(number)/2.d0)-1)*0.1
         ratecf(i)=(10**(expnt))
         a(i)=sqrt(Daq*tort/ratecf(i))
         part1=1/(sqrt(2*pi)*sig*ratecf(i))
         part2=exp(((log(ratecf(i))-mustar)**2)/(-2*(sig**2)))
         pdf(i)=part1*part2
 10    continue

c Performing numerical integration of pdf to get cdf:
c AVINT requires at least 3 abscissas between limits of integration
c lower limit of integration = 1st Da/a^2:
       ratecfo=ratecf(1)
       do 20 i=5,number
c upper limit of integration = current Da/a^2:
          ratecfn=ratecf(i)
          call AVINT(ratecf,pdf,number,ratecfo,ratecfn,dcdf,IERR)
          cdf(i)=dcdf
 20    continue

c Write to file:
       write(36,*) '1st column: block radius [L]'
```

```
       write(36,*) '2nd column: cdf (B/Btot; dimensionless)'
       write(36,*) '3rd column: diffusion rate coefficient, [L^2/T]'
       write(36,*) '4th column: cdf (B/Btot; dimensionless)'
      write(36,*) '**************************************************
     &**************************'
       do 30,i=5,number
         write(36,40) a(i),(1-cdf(i)),ratecf(i),cdf(i)
 30    continue
 40    format(5e12.4)


       return
       end



!**********************************************************************

**************SUBROUTINE AVINT (CALLED BY SUBROUTINES MASSREM AND DISTOUT)***************

*DECK AVINT
      SUBROUTINE AVINT (X, Y, N, XLO, XUP, ANS, IERR)
C***BEGIN PROLOGUE  AVINT
C***PURPOSE  Integrate a function tabulated at arbitrarily spaced
C            abscissas using overlapping parabolas.
C***LIBRARY   SLATEC
C***CATEGORY  H2A1B2
C***TYPE      SINGLE PRECISION (AVINT-S, DAVINT-D)
C***KEYWORDS  INTEGRATION, QUADRATURE, TABULATED DATA
C***AUTHOR  Jones, R. E., (SNLA)
C***DESCRIPTION
C
C     Abstract
C         AVINT integrates a function tabulated at arbitrarily spaced
C         abscissas.  The limits of integration need not coincide
C         with the tabulated abscissas.
C
C         A method of overlapping parabolas fitted to the data is used
C         provided that there are at least 3 abscissas between the
C         limits of integration.  AVINT also handles two special cases.
C         If the limits of integration are equal, AVINT returns a result
C         of zero regardless of the number of tabulated values.
C         If there are only two function values, AVINT uses the
C         trapezoid rule.
C
C     Description of Parameters
C         The user must dimension all arrays appearing in the call list
C              X(N), Y(N).
C
C         Input--
C         X    - real array of abscissas, which must be in increasing
C                order.
C         Y    - real array of functional values. i.e., Y(I)=FUNC(X(I)).
C         N    - the integer number of function values supplied.
C                N .GE. 2 unless XLO = XUP.
C         XLO  - real lower limit of integration.
C         XUP  - real upper limit of integration.
C                Must have XLO .LE. XUP.
```

```
C
C         Output--
C         ANS  - computed approximate value of integral
C         IERR - a status code
C              --normal code
C                =1 means the requested integration was performed.
C              --abnormal codes
C                =2 means XUP was less than XLO.
C                =3 means the number of X(I) between XLO and XUP
C                   (inclusive) was less than 3 and neither of the two
C                   special cases described in the Abstract occurred.
C                   No integration was performed.
C                =4 means the restriction X(I+1) .GT. X(I) was violated.
C                =5 means the number N of function values was .LT. 2.
C                ANS is set to zero if IERR=2,3,4,or 5.
C
C     AVINT is documented completely in SC-M-69-335
C     Original program from "Numerical Integration" by Davis &
C     Rabinowitz.
C     Adaptation and modifications for Sandia Mathematical Program
C     Library by Rondall E. Jones.
C
C***REFERENCES  R. E. Jones, Approximate integrator of functions
C                 tabulated at arbitrarily spaced abscissas,
C                 Report SC-M-69-335, Sandia Laboratories, 1969.
C***ROUTINES CALLED  XERMSG
C***REVISION HISTORY  (YYMMDD)
C   690901  DATE WRITTEN
C   890831  Modified array declarations.  (WRB)
C   890831  REVISION DATE from Version 3.2
C   891214  Prologue converted to Version 4.0 format.  (BAB)
C   900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
C   900326  Removed duplicate information from DESCRIPTION section.
C           (WRB)
C   920501  Reformatted the REFERENCES section.  (WRB)
C***END PROLOGUE  AVINT
C
      DOUBLE PRECISION R3,RP5,SUM,SYL,SYL2,SYL3,SYU,SYU2,SYU3,X1,X2,X3
     1,X12,X13,X23,TERM1,TERM2,TERM3,A,B,C,CA,CB,CC

      DOUBLE PRECISION X(400),Y(400),XLO,XUP,ANS

C***FIRST EXECUTABLE STATEMENT  AVINT

      IERR=1
      ANS =0.0
      IF (XLO-XUP) 3,100,200
    3 IF (N.LT.2) GO TO 215
      DO 5 I=2,N
      IF (X(I).LE.X(I-1)) GO TO 210
      IF (X(I).GT.XUP) GO TO 6
    5 CONTINUE
    6 CONTINUE
      IF (N.GE.3) GO TO 9
C
C     SPECIAL N=2 CASE
      SLOPE = (Y(2)-Y(1))/(X(2)-X(1))
      FL = Y(1) + SLOPE*(XLO-X(1))
```

```
      FR = Y(2) + SLOPE*(XUP-X(2))
      ANS = 0.5*(FL+FR)*(XUP-XLO)
      RETURN
    9 CONTINUE
      IF (X(N-2).LT.XLO)  GO TO 205
      IF (X(3).GT.XUP)    GO TO 205
      I = 1
   10 IF (X(I).GE.XLO) GO TO 15
      I = I+1
      GO TO 10
   15 INLFT = I
      I = N
   20 IF (X(I).LE.XUP) GO TO 25
      I = I-1
      GO TO 20
   25 INRT = I
      IF ((INRT-INLFT).LT.2) GO TO 205
      ISTART = INLFT
      IF (INLFT.EQ.1) ISTART = 2
      ISTOP  = INRT
      IF (INRT.EQ.N)  ISTOP  = N-1
C
      R3 = 3.0D0
      RP5= 0.5D0
      SUM = 0.0
      SYL = XLO
      SYL2= SYL*SYL
      SYL3= SYL2*SYL
C
      DO 50 I=ISTART,ISTOP
      X1 = X(I-1)
      X2 = X(I)
      X3 = X(I+1)
      X12 = X1-X2
      X13 = X1-X3
      X23 = X2-X3
      TERM1 = DBLE(Y(I-1))/(X12*X13)
      TERM2 =-DBLE(Y(I)) /(X12*X23)
      TERM3 = DBLE(Y(I+1))/(X13*X23)
      A = TERM1+TERM2+TERM3
      B = -(X2+X3)*TERM1 - (X1+X3)*TERM2 - (X1+X2)*TERM3
      C = X2*X3*TERM1 + X1*X3*TERM2 + X1*X2*TERM3
      IF (I-ISTART) 30,30,35
   30 CA = A
      CB = B
      CC = C
      GO TO 40
   35 CA = 0.5*(A+CA)
      CB = 0.5*(B+CB)
      CC = 0.5*(C+CC)
   40 SYU = X2
      SYU2= SYU*SYU
      SYU3= SYU2*SYU
      SUM = SUM + CA*(SYU3-SYL3)/R3  + CB*RP5*(SYU2-SYL2) + CC*(SYU-SYL)
      CA  = A
      CB  = B
      CC  = C
      SYL = SYU
```

```
      SYL2= SYU2
      SYL3= SYU3
   50 CONTINUE
      SYU = XUP
      ANS = SUM + CA*(SYU**3-SYL3)/R3 + CB*RP5*(SYU**2-SYL2)
     1  + CC*(SYU-SYL)
  100 RETURN
  200 IERR=2
        write(28,*) 'AVINT error type',IERR
      RETURN
  205 IERR=3
        write(28,*) 'AVINT error type',IERR
      RETURN
  210 IERR=4
        write(28,*) 'AVINT error type',IERR
      RETURN
  215 IERR=5
        write(28,*) 'AVINT error type',IERR
      RETURN
      END

C***********************************************************************
C****************** SUBROUTINES ZBIRY AND ZAIRY ************************
C***********************************************************************

c* ====================================================================
c* NIST Guide to Available Math Software.
c* Fullsource for module ZBIRY from package SLATEC.
c* Retrieved from CAMSUN on Fri Jul 19 13:09:01 1996.
c* ====================================================================
*DECK ZBIRY
      SUBROUTINE ZBIRY (ZR, ZI, ID, KODE, BIR, BII, IERR)
C***BEGIN PROLOGUE  ZBIRY
C***PURPOSE  Compute the Airy function Bi(z) or its derivative dBi/dz
C            for complex argument z.  A scaling option is available
C            to help avoid overflow.
C***LIBRARY   SLATEC
C***CATEGORY  C10D
C***TYPE      COMPLEX (CBIRY-C, ZBIRY-C)
C***KEYWORDS  AIRY FUNCTION, BESSEL FUNCTION OF ORDER ONE THIRD,
C             BESSEL FUNCTION OF ORDER TWO THIRDS
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C                     ***A DOUBLE PRECISION ROUTINE***
C         On KODE=1, ZBIRY computes the complex Airy function Bi(z)
C         or its derivative dBi/dz on ID=0 or ID=1 respectively.
C         On KODE=2, a scaling option exp(abs(Re(zeta)))*Bi(z) or
C         exp(abs(Re(zeta)))*dBi/dz is provided to remove the
C         exponential behavior in both the left and right half planes
C         where zeta=(2/3)*z**(3/2).
C
C         The Airy functions Bi(z) and dBi/dz are analytic in the
C         whole z-plane, and the scaling option does not destroy this
C         property.
C
C         Input
C           ZR     - DOUBLE PRECISION real part of argument Z
```

```
C           ZI     - DOUBLE PRECISION imag part of argument Z
C           ID     - Order of derivative, ID=0 or ID=1
C           KODE   - A parameter to indicate the scaling option
C                    KODE=1  returns
C                            BI=Bi(z)  on ID=0
C                            BI=dBi/dz on ID=1
C                            at z=Z
C                        =2  returns
C                            BI=exp(abs(Re(zeta)))*Bi(z)  on ID=0
C                            BI=exp(abs(Re(zeta)))*dBi/dz on ID=1
C                            at z=Z where zeta=(2/3)*z**(3/2)
C
C         Output
C           BIR    - DOUBLE PRECISION real part of result
C           BII    - DOUBLE PRECISION imag part of result
C           IERR   - Error flag
C                    IERR=0  Normal return     - COMPUTATION COMPLETED
C                    IERR=1  Input error       - NO COMPUTATION
C                    IERR=2  Overflow          - NO COMPUTATION
C                            (Re(Z) too large with KODE=1)
C                    IERR=3  Precision warning - COMPUTATION COMPLETED
C                            (Result has less than half precision)
C                    IERR=4  Precision error   - NO COMPUTATION
C                            (Result has no precision)
C                    IERR=5  Algorithmic error - NO COMPUTATION
C                            (Termination condition not met)
C
C *Long Description:
C
C         Bi(z) and dBi/dz are computed from I Bessel functions by
C
C                Bi(z) =  c*sqrt(z)*( I(-1/3,zeta) + I(1/3,zeta) )
C               dBi/dz =  c*   z   *( I(-2/3,zeta) + I(2/3,zeta) )
C                    c =  1/sqrt(3)
C                 zeta =  (2/3)*z**(3/2)
C
C         when abs(z)>1 and from power series when abs(z)<=1.
C
C         In most complex variable computation, one must evaluate ele-
C         mentary functions.  When the magnitude of Z is large, losses
C         of significance by argument reduction occur.  Consequently, if
C         the magnitude of ZETA=(2/3)*Z**(3/2) exceeds U1=SQRT(0.5/UR),
C         then losses exceeding half precision are likely and an error
C         flag IERR=3 is triggered where UR=MAX(D1MACH(4),1.0D-18) is
C         double precision unit roundoff limited to 18 digits precision.
C         Also, if the magnitude of ZETA is larger than U2=0.5/UR, then
C         all significance is lost and IERR=4.  In order to use the INT
C         function, ZETA must be further restricted not to exceed
C         U3=I1MACH(9)=LARGEST INTEGER.  Thus, the magnitude of ZETA
C         must be restricted by MIN(U2,U3).  In IEEE arithmetic, U1,U2,
C         and U3 are approximately 2.0E+3, 4.2E+6, 2.1E+9 in single
C         precision and 4.7E+7, 2.3E+15, 2.1E+9 in double precision.
C         This makes U2 limiting is single precision and U3 limiting
C         in double precision.  This means that the magnitude of Z
C         cannot exceed approximately 3.4E+4 in single precision and
C         2.1E+6 in double precision.  This also means that one can
C         expect to retain, in the worst cases on 32-bit machines,
C         no digits in single precision and only 6 digits in double
```

```
C         precision.
C
C         The approximate relative error in the magnitude of a complex
C         Bessel function can be expressed as P*10**S where P=MAX(UNIT
C         ROUNDOFF,1.0E-18) is the nominal precision and 10**S repre-
C         sents the increase in error due to argument reduction in the
C         elementary functions.  Here, S=MAX(1,ABS(LOG10(ABS(Z))),
C         ABS(LOG10(FNU))) approximately (i.e., S=MAX(1,ABS(EXPONENT OF
C         ABS(Z),ABS(EXPONENT OF FNU)) ).  However, the phase angle may
C         have only absolute accuracy.  This is most likely to occur
C         when one component (in magnitude) is larger than the other by
C         several orders of magnitude.  If one component is 10**K larger
C         than the other, then one can expect only MAX(ABS(LOG10(P))-K,
C         0) significant digits; or, stated another way, when K exceeds
C         the exponent of P, no significant digits remain in the smaller
C         component.  However, the phase angle retains absolute accuracy
C         because, in complex arithmetic with precision P, the smaller
C         component will not (as a rule) decrease below P times the
C         magnitude of the larger component. In these extreme cases,
C         the principal phase angle is on the order of +P, -P, PI/2-P,
C         or -PI/2+P.
C
C***REFERENCES  1. M. Abramowitz and I. A. Stegun, Handbook of Mathe-
C                  matical Functions, National Bureau of Standards
C                  Applied Mathematics Series 55, U. S. Department
C                  of Commerce, Tenth Printing (1972) or later.
C               2. D. E. Amos, Computation of Bessel Functions of
C                  Complex Argument and Large Order, Report SAND83-0643,
C                  Sandia National Laboratories, Albuquerque, NM, May
C                  1983.
C               3. D. E. Amos, A Subroutine Package for Bessel Functions
C                  of a Complex Argument and Nonnegative Order, Report
C                  SAND85-1018, Sandia National Laboratory, Albuquerque,
C                  NM, May 1985.
C               4. D. E. Amos, A portable package for Bessel functions
C                  of a complex argument and nonnegative order, ACM
C                  Transactions on Mathematical Software, 12 (September
C                  1986), pp. 265-273.
C
C***ROUTINES CALLED  D1MACH, I1MACH, ZABS, ZBINU, ZDIV, ZSQRT
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   890801  REVISION DATE from Version 3.2
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C   920128  Category corrected.  (WRB)
C   920811  Prologue revised.  (DWL)
C   930122  Added ZSQRT to EXTERNAL statement.  (RWC)
C***END PROLOGUE  ZBIRY
C     COMPLEX BI,CONE,CSQ,CY,S1,S2,TRM1,TRM2,Z,ZTA,Z3
      DOUBLE PRECISION AA, AD, AK, ALIM, ATRM, AZ, AZ3, BB, BII, BIR,
     * BK, CC, CK, COEF, CONEI, CONER, CSQI, CSQR, CYI, CYR, C1, C2,
     * DIG, DK, D1, D2, EAA, ELIM, FID, FMR, FNU, FNUL, PI, RL, R1M5,
     * SFAC, STI, STR, S1I, S1R, S2I, S2R, TOL, TRM1I, TRM1R, TRM2I,
     * TRM2R, TTH, ZI, ZR, ZTAI, ZTAR, Z3I, Z3R, D1MACH, ZABS
      INTEGER ID, IERR, K, KODE, K1, K2, NZ, I1MACH
      DIMENSION CYR(2), CYI(2)
      EXTERNAL ZABS, ZSQRT
      DATA TTH, C1, C2, COEF, PI /6.6666666666666667D-01,
```

```
     * 6.14926627446000736D-01,4.48288357353826359D-01,
     * 5.77350269189625765D-01,3.14159265358979324D+00/
      DATA CONER, CONEI /1.0D0,0.0D0/
C***FIRST EXECUTABLE STATEMENT  ZBIRY
      IERR = 0
      NZ=0
      IF (ID.LT.0 .OR. ID.GT.1) IERR=1
      IF (KODE.LT.1 .OR. KODE.GT.2) IERR=1
      IF (IERR.NE.0) RETURN
      AZ = ZABS(ZR,ZI)
      TOL = MAX(D1MACH(4),1.0D-18)
      FID = ID
      IF (AZ.GT.1.0E0) GO TO 70
C-----------------------------------------------------------------------
C     POWER SERIES FOR ABS(Z).LE.1.
C-----------------------------------------------------------------------
      S1R = CONER
      S1I = CONEI
      S2R = CONER
      S2I = CONEI
      IF (AZ.LT.TOL) GO TO 130
      AA = AZ*AZ
      IF (AA.LT.TOL/AZ) GO TO 40
      TRM1R = CONER
      TRM1I = CONEI
      TRM2R = CONER
      TRM2I = CONEI
      ATRM = 1.0D0
      STR = ZR*ZR - ZI*ZI
      STI = ZR*ZI + ZI*ZR
      Z3R = STR*ZR - STI*ZI
      Z3I = STR*ZI + STI*ZR
      AZ3 = AZ*AA
      AK = 2.0D0 + FID
      BK = 3.0D0 - FID - FID
      CK = 4.0D0 - FID
      DK = 3.0D0 + FID + FID
      D1 = AK*DK
      D2 = BK*CK
      AD = MIN(D1,D2)
      AK = 24.0D0 + 9.0D0*FID
      BK = 30.0D0 - 9.0D0*FID
      DO 30 K=1,25
        STR = (TRM1R*Z3R-TRM1I*Z3I)/D1
        TRM1I = (TRM1R*Z3I+TRM1I*Z3R)/D1
        TRM1R = STR
        S1R = S1R + TRM1R
        S1I = S1I + TRM1I
        STR = (TRM2R*Z3R-TRM2I*Z3I)/D2
        TRM2I = (TRM2R*Z3I+TRM2I*Z3R)/D2
        TRM2R = STR
        S2R = S2R + TRM2R
        S2I = S2I + TRM2I
        ATRM = ATRM*AZ3/AD
        D1 = D1 + AK
        D2 = D2 + BK
        AD = MIN(D1,D2)
        IF (ATRM.LT.TOL*AD) GO TO 40
```

```
        AK = AK + 18.0D0
        BK = BK + 18.0D0
   30 CONTINUE
   40 CONTINUE
      IF (ID.EQ.1) GO TO 50
      BIR = C1*S1R + C2*(ZR*S2R-ZI*S2I)
      BII = C1*S1I + C2*(ZR*S2I+ZI*S2R)
      IF (KODE.EQ.1) RETURN
      CALL ZSQRT(ZR, ZI, STR, STI)
      ZTAR = TTH*(ZR*STR-ZI*STI)
      ZTAI = TTH*(ZR*STI+ZI*STR)
      AA = ZTAR
      AA = -ABS(AA)
      EAA = EXP(AA)
      BIR = BIR*EAA
      BII = BII*EAA
      RETURN
   50 CONTINUE
      BIR = S2R*C2
      BII = S2I*C2
      IF (AZ.LE.TOL) GO TO 60
      CC = C1/(1.0D0+FID)
      STR = S1R*ZR - S1I*ZI
      STI = S1R*ZI + S1I*ZR
      BIR = BIR + CC*(STR*ZR-STI*ZI)
      BII = BII + CC*(STR*ZI+STI*ZR)
   60 CONTINUE
      IF (KODE.EQ.1) RETURN
      CALL ZSQRT(ZR, ZI, STR, STI)
      ZTAR = TTH*(ZR*STR-ZI*STI)
      ZTAI = TTH*(ZR*STI+ZI*STR)
      AA = ZTAR
      AA = -ABS(AA)
      EAA = EXP(AA)
      BIR = BIR*EAA
      BII = BII*EAA
      RETURN
C-----------------------------------------------------------------------
C     CASE FOR ABS(Z).GT.1.0
C-----------------------------------------------------------------------
   70 CONTINUE
      FNU = (1.0D0+FID)/3.0D0
C-----------------------------------------------------------------------
C     SET PARAMETERS RELATED TO MACHINE CONSTANTS.
C     TOL IS THE APPROXIMATE UNIT ROUNDOFF LIMITED TO 1.0E-18.
C     ELIM IS THE APPROXIMATE EXPONENTIAL OVER- AND UNDERFLOW LIMIT.
C     EXP(-ELIM).LT.EXP(-ALIM)=EXP(-ELIM)/TOL    AND
C     EXP(ELIM).GT.EXP(ALIM)=EXP(ELIM)*TOL       ARE INTERVALS NEAR
C     UNDERFLOW AND OVERFLOW LIMITS WHERE SCALED ARITHMETIC IS DONE.
C     RL IS THE LOWER BOUNDARY OF THE ASYMPTOTIC EXPANSION FOR LARGE Z.
C     DIG = NUMBER OF BASE 10 DIGITS IN TOL = 10**(-DIG).
C     FNUL IS THE LOWER BOUNDARY OF THE ASYMPTOTIC SERIES FOR LARGE FNU.
C-----------------------------------------------------------------------
      K1 = I1MACH(15)
      K2 = I1MACH(16)
      R1M5 = D1MACH(5)
      K = MIN(ABS(K1),ABS(K2))
      ELIM = 2.303D0*(K*R1M5-3.0D0)
```

```
      K1 = I1MACH(14) - 1
      AA = R1M5*K1
      DIG = MIN(AA,18.0D0)
      AA = AA*2.303D0
      ALIM = ELIM + MAX(-AA,-41.45D0)
      RL = 1.2D0*DIG + 3.0D0
      FNUL = 10.0D0 + 6.0D0*(DIG-3.0D0)
C-----------------------------------------------------------------------
C     TEST FOR RANGE
C-----------------------------------------------------------------------
      AA=0.5D0/TOL
      BB=I1MACH(9)*0.5D0
      AA=MIN(AA,BB)
      AA=AA**TTH
      IF (AZ.GT.AA) GO TO 260
      AA=SQRT(AA)
      IF (AZ.GT.AA) IERR=3
      CALL ZSQRT(ZR, ZI, CSQR, CSQI)
      ZTAR = TTH*(ZR*CSQR-ZI*CSQI)
      ZTAI = TTH*(ZR*CSQI+ZI*CSQR)
C-----------------------------------------------------------------------
C     RE(ZTA).LE.0 WHEN RE(Z).LT.0, ESPECIALLY WHEN IM(Z) IS SMALL
C-----------------------------------------------------------------------
      SFAC = 1.0D0
      AK = ZTAI
      IF (ZR.GE.0.0D0) GO TO 80
      BK = ZTAR
      CK = -ABS(BK)
      ZTAR = CK
      ZTAI = AK
   80 CONTINUE
      IF (ZI.NE.0.0D0 .OR. ZR.GT.0.0D0) GO TO 90
      ZTAR = 0.0D0
      ZTAI = AK
   90 CONTINUE
      AA = ZTAR
      IF (KODE.EQ.2) GO TO 100
C-----------------------------------------------------------------------
C     OVERFLOW TEST
C-----------------------------------------------------------------------
      BB = ABS(AA)
      IF (BB.LT.ALIM) GO TO 100
      BB = BB + 0.25D0*LOG(AZ)
      SFAC = TOL
      IF (BB.GT.ELIM) GO TO 190
  100 CONTINUE
      FMR = 0.0D0
      IF (AA.GE.0.0D0 .AND. ZR.GT.0.0D0) GO TO 110
      FMR = PI
      IF (ZI.LT.0.0D0) FMR = -PI
      ZTAR = -ZTAR
      ZTAI = -ZTAI
  110 CONTINUE
C-----------------------------------------------------------------------
C     AA=FACTOR FOR ANALYTIC CONTINUATION OF I(FNU,ZTA)
C     KODE=2 RETURNS EXP(-ABS(XZTA))*I(FNU,ZTA) FROM CBESI
C-----------------------------------------------------------------------
      CALL ZBINU(ZTAR, ZTAI, FNU, KODE, 1, CYR, CYI, NZ, RL, FNUL, TOL,
```

```
     * ELIM, ALIM)
       IF (NZ.LT.0) GO TO 200
       AA = FMR*FNU
       Z3R = SFAC
       STR = COS(AA)
       STI = SIN(AA)
       S1R = (STR*CYR(1)-STI*CYI(1))*Z3R
       S1I = (STR*CYI(1)+STI*CYR(1))*Z3R
       FNU = (2.0D0-FID)/3.0D0
       CALL ZBINU(ZTAR, ZTAI, FNU, KODE, 2, CYR, CYI, NZ, RL, FNUL, TOL,
     * ELIM, ALIM)
       CYR(1) = CYR(1)*Z3R
       CYI(1) = CYI(1)*Z3R
       CYR(2) = CYR(2)*Z3R
       CYI(2) = CYI(2)*Z3R
C-----------------------------------------------------------------------
C     BACKWARD RECUR ONE STEP FOR ORDERS -1/3 OR -2/3
C-----------------------------------------------------------------------
       CALL ZDIV(CYR(1), CYI(1), ZTAR, ZTAI, STR, STI)
       S2R = (FNU+FNU)*STR + CYR(2)
       S2I = (FNU+FNU)*STI + CYI(2)
       AA = FMR*(FNU-1.0D0)
       STR = COS(AA)
       STI = SIN(AA)
       S1R = COEF*(S1R+S2R*STR-S2I*STI)
       S1I = COEF*(S1I+S2R*STI+S2I*STR)
       IF (ID.EQ.1) GO TO 120
       STR = CSQR*S1R - CSQI*S1I
       S1I = CSQR*S1I + CSQI*S1R
       S1R = STR
       BIR = S1R/SFAC
       BII = S1I/SFAC
       RETURN
  120  CONTINUE
       STR = ZR*S1R - ZI*S1I
       S1I = ZR*S1I + ZI*S1R
       S1R = STR
       BIR = S1R/SFAC
       BII = S1I/SFAC
       RETURN
  130  CONTINUE
       AA = C1*(1.0D0-FID) + FID*C2
       BIR = AA
       BII = 0.0D0
       RETURN
  190  CONTINUE
       IERR=2
       NZ=0
       RETURN
  200  CONTINUE
       IF(NZ.EQ.(-1)) GO TO 190
       NZ=0
       IERR=5
       RETURN
  260  CONTINUE
       IERR=4
       NZ=0
       RETURN
```

```
      END
*DECK ZDIV
      SUBROUTINE ZDIV (AR, AI, BR, BI, CR, CI)
C***BEGIN PROLOGUE  ZDIV
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZBESH, ZBESI, ZBESJ, ZBESK, ZBESY, ZAIRY and
C            ZBIRY
C***LIBRARY   SLATEC
C***TYPE      ALL (ZDIV-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     DOUBLE PRECISION COMPLEX DIVIDE C=A/B.
C
C***SEE ALSO  ZAIRY, ZBESH, ZBESI, ZBESJ, ZBESK, ZBESY, ZBIRY
C***ROUTINES CALLED  ZABS
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C***END PROLOGUE  ZDIV
      DOUBLE PRECISION AR, AI, BR, BI, CR, CI, BM, CA, CB, CC, CD
      DOUBLE PRECISION ZABS
      EXTERNAL ZABS
C***FIRST EXECUTABLE STATEMENT  ZDIV
      BM = 1.0D0/ZABS(BR,BI)
      CC = BR*BM
      CD = BI*BM
      CA = (AR*CC+AI*CD)*BM
      CB = (AI*CC-AR*CD)*BM
      CR = CA
      CI = CB
      RETURN
      END
*DECK ZSQRT
      SUBROUTINE ZSQRT (AR, AI, BR, BI)
C***BEGIN PROLOGUE  ZSQRT
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZBESH, ZBESI, ZBESJ, ZBESK, ZBESY, ZAIRY and
C            ZBIRY
C***LIBRARY   SLATEC
C***TYPE      ALL (ZSQRT-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     DOUBLE PRECISION COMPLEX SQUARE ROOT, B=CSQRT(A)
C
C***SEE ALSO  ZAIRY, ZBESH, ZBESI, ZBESJ, ZBESK, ZBESY, ZBIRY
C***ROUTINES CALLED  ZABS
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C***END PROLOGUE  ZSQRT
      DOUBLE PRECISION AR, AI, BR, BI, ZM, DTHETA, DPI, DRT
      DOUBLE PRECISION ZABS
      EXTERNAL ZABS
      DATA DRT , DPI / 7.071067811865475244008443621D-1,
     1                 3.141592653589793238462643383D+0/
C***FIRST EXECUTABLE STATEMENT  ZSQRT
```

```
      ZM = ZABS(AR,AI)
      ZM = SQRT(ZM)
      IF (AR.EQ.0.0D+0) GO TO 10
      IF (AI.EQ.0.0D+0) GO TO 20
      DTHETA = DATAN(AI/AR)
      IF (DTHETA.LE.0.0D+0) GO TO 40
      IF (AR.LT.0.0D+0) DTHETA = DTHETA - DPI
      GO TO 50
   10 IF (AI.GT.0.0D+0) GO TO 60
      IF (AI.LT.0.0D+0) GO TO 70
      BR = 0.0D+0
      BI = 0.0D+0
      RETURN
   20 IF (AR.GT.0.0D+0) GO TO 30
      BR = 0.0D+0
      BI = SQRT(ABS(AR))
      RETURN
   30 BR = SQRT(AR)
      BI = 0.0D+0
      RETURN
   40 IF (AR.LT.0.0D+0) DTHETA = DTHETA + DPI
   50 DTHETA = DTHETA*0.5D+0
      BR = ZM*COS(DTHETA)
      BI = ZM*SIN(DTHETA)
      RETURN
   60 BR = ZM*DRT
      BI = ZM*DRT
      RETURN
   70 BR = ZM*DRT
      BI = -ZM*DRT
      RETURN
      END
*DECK D1MACH
      DOUBLE PRECISION FUNCTION D1MACH (I)
C***BEGIN PROLOGUE  D1MACH
C***PURPOSE  Return floating point machine dependent constants.
C***LIBRARY   SLATEC
C***CATEGORY  R1
C***TYPE      DOUBLE PRECISION (R1MACH-S, D1MACH-D)
C***KEYWORDS  MACHINE CONSTANTS
C***AUTHOR  Fox, P. A., (Bell Labs)
C           Hall, A. D., (Bell Labs)
C           Schryer, N. L., (Bell Labs)
C***DESCRIPTION
C
C   D1MACH can be used to obtain machine-dependent parameters for the
C   local machine environment.  It is a function subprogram with one
C   (input) argument, and can be referenced as follows:
C
C        D = D1MACH(I)
C
C   where I=1,...,5.  The (output) value of D above is determined by
C   the (input) value of I.  The results for various values of I are
C   discussed below.
C
C   D1MACH( 1) = B**(EMIN-1), the smallest positive magnitude.
C   D1MACH( 2) = B**EMAX*(1 - B**(-T)), the largest magnitude.
C   D1MACH( 3) = B**(-T), the smallest relative spacing.
```

```
C   D1MACH( 4) = B**(1-T), the largest relative spacing.
C   D1MACH( 5) = LOG10(B)
C
C   Assume double precision numbers are represented in the T-digit,
C   base-B form
C
C              sign (B**E)*( (X(1)/B) + ... + (X(T)/B**T) )
C
C   where 0 .LE. X(I) .LT. B for I=1,...,T, 0 .LT. X(1), and
C   EMIN .LE. E .LE. EMAX.
C
C   The values of B, T, EMIN and EMAX are provided in I1MACH as
C   follows:
C   I1MACH(10) = B, the base.
C   I1MACH(14) = T, the number of base-B digits.
C   I1MACH(15) = EMIN, the smallest exponent E.
C   I1MACH(16) = EMAX, the largest exponent E.
C
C   To alter this function for a particular environment, the desired
C   set of DATA statements should be activated by removing the C from
C   column 1.  Also, the values of D1MACH(1) - D1MACH(4) should be
C   checked for consistency with the local operating system.
C
C***REFERENCES  P. A. Fox, A. D. Hall and N. L. Schryer, Framework for
C                 a portable library, ACM Transactions on Mathematical
C                 Software 4, 2 (June 1978), pp. 177-188.
C***ROUTINES CALLED  XERMSG
C***REVISION HISTORY  (YYMMDD)
C   750101  DATE WRITTEN
C   890213  REVISION DATE from Version 3.2
C   891214  Prologue converted to Version 4.0 format.  (BAB)
C   900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
C   900618  Added DEC RISC constants.  (WRB)
C   900723  Added IBM RS 6000 constants.  (WRB)
C   900911  Added SUN 386i constants.  (WRB)
C   910710  Added HP 730 constants.  (SMR)
C   911114  Added Convex IEEE constants.  (WRB)
C   920121  Added SUN -r8 compiler option constants.  (WRB)
C   920229  Added Touchstone Delta i860 constants.  (WRB)
C   920501  Reformatted the REFERENCES section.  (WRB)
C   920625  Added CONVEX -p8 and -pd8 compiler option constants.
C           (BKS, WRB)
C   930201  Added DEC Alpha and SGI constants.  (RWC and WRB)
C***END PROLOGUE  D1MACH
C
      INTEGER SMALL(4)
      INTEGER LARGE(4)
      INTEGER RIGHT(4)
      INTEGER DIVER(4)
      INTEGER LOG10(4)
C
      DOUBLE PRECISION DMACH(5)
      SAVE DMACH
C
      EQUIVALENCE (DMACH(1),SMALL(1))
      EQUIVALENCE (DMACH(2),LARGE(1))
      EQUIVALENCE (DMACH(3),RIGHT(1))
      EQUIVALENCE (DMACH(4),DIVER(1))
```

```
      EQUIVALENCE (DMACH(5),LOG10(1))
C
C     MACHINE CONSTANTS FOR THE AMIGA
C     ABSOFT FORTRAN COMPILER USING THE 68020/68881 COMPILER OPTION
C
C     DATA SMALL(1), SMALL(2) / Z'00100000', Z'00000000' /
C     DATA LARGE(1), LARGE(2) / Z'7FEFFFFF', Z'FFFFFFFF' /
C     DATA RIGHT(1), RIGHT(2) / Z'3CA00000', Z'00000000' /
C     DATA DIVER(1), DIVER(2) / Z'3CB00000', Z'00000000' /
C     DATA LOG10(1), LOG10(2) / Z'3FD34413', Z'509F79FF' /
C
C     MACHINE CONSTANTS FOR THE AMIGA
C     ABSOFT FORTRAN COMPILER USING SOFTWARE FLOATING POINT
C
C     DATA SMALL(1), SMALL(2) / Z'00100000', Z'00000000' /
C     DATA LARGE(1), LARGE(2) / Z'7FDFFFFF', Z'FFFFFFFF' /
C     DATA RIGHT(1), RIGHT(2) / Z'3CA00000', Z'00000000' /
C     DATA DIVER(1), DIVER(2) / Z'3CB00000', Z'00000000' /
C     DATA LOG10(1), LOG10(2) / Z'3FD34413', Z'509F79FF' /
C
C     MACHINE CONSTANTS FOR THE APOLLO
C
C     DATA SMALL(1), SMALL(2) / 16#00100000, 16#00000000 /
C     DATA LARGE(1), LARGE(2) / 16#7FFFFFFF, 16#FFFFFFFF /
C     DATA RIGHT(1), RIGHT(2) / 16#3CA00000, 16#00000000 /
C     DATA DIVER(1), DIVER(2) / 16#3CB00000, 16#00000000 /
C     DATA LOG10(1), LOG10(2) / 16#3FD34413, 16#509F79FF /
C
C     MACHINE CONSTANTS FOR THE BURROUGHS 1700 SYSTEM
C
C     DATA SMALL(1) / ZC00800000 /
C     DATA SMALL(2) / Z000000000 /
C     DATA LARGE(1) / ZDFFFFFFFF /
C     DATA LARGE(2) / ZFFFFFFFFF /
C     DATA RIGHT(1) / ZCC5800000 /
C     DATA RIGHT(2) / Z000000000 /
C     DATA DIVER(1) / ZCC6800000 /
C     DATA DIVER(2) / Z000000000 /
C     DATA LOG10(1) / ZD00E730E7 /
C     DATA LOG10(2) / ZC77800DC0 /
C
C     MACHINE CONSTANTS FOR THE BURROUGHS 5700 SYSTEM
C
C     DATA SMALL(1) / O1771000000000000 /
C     DATA SMALL(2) / O0000000000000000 /
C     DATA LARGE(1) / O0777777777777777 /
C     DATA LARGE(2) / O0007777777777777 /
C     DATA RIGHT(1) / O1461000000000000 /
C     DATA RIGHT(2) / O0000000000000000 /
C     DATA DIVER(1) / O1451000000000000 /
C     DATA DIVER(2) / O0000000000000000 /
C     DATA LOG10(1) / O1157163034761674 /
C     DATA LOG10(2) / O0006677466732724 /
C
C     MACHINE CONSTANTS FOR THE BURROUGHS 6700/7700 SYSTEMS
C
C     DATA SMALL(1) / O1771000000000000 /
C     DATA SMALL(2) / O7770000000000000 /
```

```
C     DATA LARGE(1) / O0777777777777777 /
C     DATA LARGE(2) / O7777777777777777 /
C     DATA RIGHT(1) / O1461000000000000 /
C     DATA RIGHT(2) / O0000000000000000 /
C     DATA DIVER(1) / O1451000000000000 /
C     DATA DIVER(2) / O0000000000000000 /
C     DATA LOG10(1) / O1157163034761674 /
C     DATA LOG10(2) / O0006677466732724 /
C
C     MACHINE CONSTANTS FOR THE CDC 170/180 SERIES USING NOS/VE
C
C     DATA SMALL(1) / Z"3001800000000000" /
C     DATA SMALL(2) / Z"3001000000000000" /
C     DATA LARGE(1) / Z"4FFEFFFFFFFFFFFE" /
C     DATA LARGE(2) / Z"4FFE000000000000" /
C     DATA RIGHT(1) / Z"3FD2800000000000" /
C     DATA RIGHT(2) / Z"3FD2000000000000" /
C     DATA DIVER(1) / Z"3FD3800000000000" /
C     DATA DIVER(2) / Z"3FD3000000000000" /
C     DATA LOG10(1) / Z"3FFF9A209A84FBCF" /
C     DATA LOG10(2) / Z"3FFFF7988F8959AC" /
C
C     MACHINE CONSTANTS FOR THE CDC 6000/7000 SERIES
C
C     DATA SMALL(1) / 00564000000000000000B /
C     DATA SMALL(2) / 00000000000000000000B /
C     DATA LARGE(1) / 37757777777777777777B /
C     DATA LARGE(2) / 37157777777777777777B /
C     DATA RIGHT(1) / 15624000000000000000B /
C     DATA RIGHT(2) / 00000000000000000000B /
C     DATA DIVER(1) / 15634000000000000000B /
C     DATA DIVER(2) / 00000000000000000000B /
C     DATA LOG10(1) / 17164642023241175717B /
C     DATA LOG10(2) / 16367571421742254654B /
C
C     MACHINE CONSTANTS FOR THE CELERITY C1260
C
C     DATA SMALL(1), SMALL(2) / Z'00100000', Z'00000000' /
C     DATA LARGE(1), LARGE(2) / Z'7FEFFFFF', Z'FFFFFFFF' /
C     DATA RIGHT(1), RIGHT(2) / Z'3CA00000', Z'00000000' /
C     DATA DIVER(1), DIVER(2) / Z'3CB00000', Z'00000000' /
C     DATA LOG10(1), LOG10(2) / Z'3FD34413', Z'509F79FF' /
C
C     MACHINE CONSTANTS FOR THE CONVEX
C     USING THE -fn OR -pd8 COMPILER OPTION
C
C     DATA DMACH(1) / Z'0010000000000000' /
C     DATA DMACH(2) / Z'7FFFFFFFFFFFFFFF' /
C     DATA DMACH(3) / Z'3CC0000000000000' /
C     DATA DMACH(4) / Z'3CD0000000000000' /
C     DATA DMACH(5) / Z'3FF34413509F79FF' /
C
C     MACHINE CONSTANTS FOR THE CONVEX
C     USING THE -fi COMPILER OPTION
C
C     DATA DMACH(1) / Z'0010000000000000' /
C     DATA DMACH(2) / Z'7FEFFFFFFFFFFFFF' /
C     DATA DMACH(3) / Z'3CA0000000000000' /
```

```
C     DATA DMACH(4) / Z'3CB0000000000000' /
C     DATA DMACH(5) / Z'3FD34413509F79FF' /
C
C     MACHINE CONSTANTS FOR THE CONVEX
C     USING THE -p8 COMPILER OPTION
C
C     DATA DMACH(1) / Z'0001000000000000000000000000000' /
C     DATA DMACH(2) / Z'7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' /
C     DATA DMACH(3) / Z'3F900000000000000000000000000000' /
C     DATA DMACH(4) / Z'3F910000000000000000000000000000' /
C     DATA DMACH(5) / Z'3FFF34413509F79FEF311F12B35816F9' /
C
C     MACHINE CONSTANTS FOR THE CRAY
C
C     DATA SMALL(1) / 201354000000000000000B /
C     DATA SMALL(2) / 000000000000000000000B /
C     DATA LARGE(1) / 577767777777777777777B /
C     DATA LARGE(2) / 000007777777777777774B /
C     DATA RIGHT(1) / 376434000000000000000B /
C     DATA RIGHT(2) / 000000000000000000000B /
C     DATA DIVER(1) / 376444000000000000000B /
C     DATA DIVER(2) / 000000000000000000000B /
C     DATA LOG10(1) / 377774642023241175717B /
C     DATA LOG10(2) / 000007571421742254654B /
C
C     MACHINE CONSTANTS FOR THE DATA GENERAL ECLIPSE S/200
C     NOTE - IT MAY BE APPROPRIATE TO INCLUDE THE FOLLOWING CARD -
C     STATIC DMACH(5)
C
C     DATA SMALL /    20K, 3*0 /
C     DATA LARGE / 77777K, 3*177777K /
C     DATA RIGHT / 31420K, 3*0 /
C     DATA DIVER / 32020K, 3*0 /
C     DATA LOG10 / 40423K, 42023K, 50237K, 74776K /
C
C     MACHINE CONSTANTS FOR THE DEC ALPHA
C     USING G_FLOAT
C
C     DATA DMACH(1) / '0000000000000010'X /
C     DATA DMACH(2) / 'FFFFFFFFFFFF7FFF'X /
C     DATA DMACH(3) / '0000000000003CC0'X /
C     DATA DMACH(4) / '0000000000003CD0'X /
C     DATA DMACH(5) / '79FF509F44133FF3'X /
C
C     MACHINE CONSTANTS FOR THE DEC ALPHA
C     USING IEEE_FORMAT
C
C     DATA DMACH(1) / '0010000000000000'X /
C     DATA DMACH(2) / '7FEFFFFFFFFFFFFF'X /
C     DATA DMACH(3) / '3CA0000000000000'X /
C     DATA DMACH(4) / '3CB0000000000000'X /
C     DATA DMACH(5) / '3FD34413509F79FF'X /
C
C     MACHINE CONSTANTS FOR THE DEC RISC
C
C     DATA SMALL(1), SMALL(2) / Z'00000000', Z'00100000'/
C     DATA LARGE(1), LARGE(2) / Z'FFFFFFFF', Z'7FEFFFFF'/
C     DATA RIGHT(1), RIGHT(2) / Z'00000000', Z'3CA00000'/
```

```
C     DATA DIVER(1), DIVER(2) / Z'00000000', Z'3CB00000'/
C     DATA LOG10(1), LOG10(2) / Z'509F79FF', Z'3FD34413'/
C
C     MACHINE CONSTANTS FOR THE DEC VAX
C     USING D_FLOATING
C     (EXPRESSED IN INTEGER AND HEXADECIMAL)
C     THE HEX FORMAT BELOW MAY NOT BE SUITABLE FOR UNIX SYSTEMS
C     THE INTEGER FORMAT SHOULD BE OK FOR UNIX SYSTEMS
C
C     DATA SMALL(1), SMALL(2) /        128,           0 /
C     DATA LARGE(1), LARGE(2) /     -32769,          -1 /
C     DATA RIGHT(1), RIGHT(2) /       9344,           0 /
C     DATA DIVER(1), DIVER(2) /       9472,           0 /
C     DATA LOG10(1), LOG10(2) /  546979738,  -805796613 /
C
C     DATA SMALL(1), SMALL(2) / Z00000080, Z00000000 /
C     DATA LARGE(1), LARGE(2) / ZFFFF7FFF, ZFFFFFFFF /
C     DATA RIGHT(1), RIGHT(2) / Z00002480, Z00000000 /
C     DATA DIVER(1), DIVER(2) / Z00002500, Z00000000 /
C     DATA LOG10(1), LOG10(2) / Z209A3F9A, ZCFF884FB /
C
C     MACHINE CONSTANTS FOR THE DEC VAX
C     USING G_FLOATING
C     (EXPRESSED IN INTEGER AND HEXADECIMAL)
C     THE HEX FORMAT BELOW MAY NOT BE SUITABLE FOR UNIX SYSTEMS
C     THE INTEGER FORMAT SHOULD BE OK FOR UNIX SYSTEMS
C
C     DATA SMALL(1), SMALL(2) /         16,           0 /
C     DATA LARGE(1), LARGE(2) /     -32769,          -1 /
C     DATA RIGHT(1), RIGHT(2) /      15552,           0 /
C     DATA DIVER(1), DIVER(2) /      15568,           0 /
C     DATA LOG10(1), LOG10(2) /  1142112243, 2046775455 /
C
C     DATA SMALL(1), SMALL(2) / Z00000010, Z00000000 /
C     DATA LARGE(1), LARGE(2) / ZFFFF7FFF, ZFFFFFFFF /
C     DATA RIGHT(1), RIGHT(2) / Z00003CC0, Z00000000 /
C     DATA DIVER(1), DIVER(2) / Z00003CD0, Z00000000 /
C     DATA LOG10(1), LOG10(2) / Z44133FF3, Z79FF509F /
C
C     MACHINE CONSTANTS FOR THE ELXSI 6400
C     (ASSUMING REAL*8 IS THE DEFAULT DOUBLE PRECISION)
C
C     DATA SMALL(1), SMALL(2) / '00100000'X,'00000000'X /
C     DATA LARGE(1), LARGE(2) / '7FEFFFFF'X,'FFFFFFFF'X /
C     DATA RIGHT(1), RIGHT(2) / '3CB00000'X,'00000000'X /
C     DATA DIVER(1), DIVER(2) / '3CC00000'X,'00000000'X /
C     DATA LOG10(1), LOG10(2) / '3FD34413'X,'509F79FF'X /
C
C     MACHINE CONSTANTS FOR THE HARRIS 220
C
C     DATA SMALL(1), SMALL(2) / '20000000, '00000201 /
C     DATA LARGE(1), LARGE(2) / '37777777, '37777577 /
C     DATA RIGHT(1), RIGHT(2) / '20000000, '00000333 /
C     DATA DIVER(1), DIVER(2) / '20000000, '00000334 /
C     DATA LOG10(1), LOG10(2) / '23210115, '10237777 /
C
C     MACHINE CONSTANTS FOR THE HONEYWELL 600/6000 SERIES
C
```

```
C     DATA SMALL(1), SMALL(2) / O402400000000, O000000000000 /
C     DATA LARGE(1), LARGE(2) / O376777777777, O777777777777 /
C     DATA RIGHT(1), RIGHT(2) / O604400000000, O000000000000 /
C     DATA DIVER(1), DIVER(2) / O606400000000, O000000000000 /
C     DATA LOG10(1), LOG10(2) / O776464202324, O117571775714 /
C
C     MACHINE CONSTANTS FOR THE HP 730
C
C     DATA DMACH(1) / Z'0010000000000000' /
C     DATA DMACH(2) / Z'7FEFFFFFFFFFFFFF' /
C     DATA DMACH(3) / Z'3CA0000000000000' /
C     DATA DMACH(4) / Z'3CB0000000000000' /
C     DATA DMACH(5) / Z'3FD34413509F79FF' /
C
C     MACHINE CONSTANTS FOR THE HP 2100
C     THREE WORD DOUBLE PRECISION OPTION WITH FTN4
C
C     DATA SMALL(1), SMALL(2), SMALL(3) / 40000B,       0,       1 /
C     DATA LARGE(1), LARGE(2), LARGE(3) / 77777B, 177777B, 177776B /
C     DATA RIGHT(1), RIGHT(2), RIGHT(3) / 40000B,       0,    265B /
C     DATA DIVER(1), DIVER(2), DIVER(3) / 40000B,       0,    276B /
C     DATA LOG10(1), LOG10(2), LOG10(3) / 46420B,  46502B,  77777B /
C
C     MACHINE CONSTANTS FOR THE HP 2100
C     FOUR WORD DOUBLE PRECISION OPTION WITH FTN4
C
C     DATA SMALL(1), SMALL(2) /  40000B,       0 /
C     DATA SMALL(3), SMALL(4) /       0,       1 /
C     DATA LARGE(1), LARGE(2) /  77777B, 177777B /
C     DATA LARGE(3), LARGE(4) / 177777B, 177776B /
C     DATA RIGHT(1), RIGHT(2) /  40000B,       0 /
C     DATA RIGHT(3), RIGHT(4) /       0,    225B /
C     DATA DIVER(1), DIVER(2) /  40000B,       0 /
C     DATA DIVER(3), DIVER(4) /       0,    227B /
C     DATA LOG10(1), LOG10(2) /  46420B,  46502B /
C     DATA LOG10(3), LOG10(4) /  76747B, 176377B /
C
C     MACHINE CONSTANTS FOR THE HP 9000
C
C     DATA SMALL(1), SMALL(2) / 00040000000B, 00000000000B /
C     DATA LARGE(1), LARGE(2) / 17737777777B, 37777777777B /
C     DATA RIGHT(1), RIGHT(2) / 07454000000B, 00000000000B /
C     DATA DIVER(1), DIVER(2) / 07460000000B, 00000000000B /
C     DATA LOG10(1), LOG10(2) / 07764642023B, 12047674777B /
C
C     MACHINE CONSTANTS FOR THE IBM 360/370 SERIES,
C     THE XEROX SIGMA 5/7/9, THE SEL SYSTEMS 85/86, AND
C     THE PERKIN ELMER (INTERDATA) 7/32.
C
C     DATA SMALL(1), SMALL(2) / Z00100000, Z00000000 /
C     DATA LARGE(1), LARGE(2) / Z7FFFFFFF, ZFFFFFFFF /
C     DATA RIGHT(1), RIGHT(2) / Z33100000, Z00000000 /
C     DATA DIVER(1), DIVER(2) / Z34100000, Z00000000 /
C     DATA LOG10(1), LOG10(2) / Z41134413, Z509F79FF /
C
C     MACHINE CONSTANTS FOR THE IBM PC
C     ASSUMES THAT ALL ARITHMETIC IS DONE IN DOUBLE PRECISION
C     ON 8088, I.E., NOT IN 80 BIT FORM FOR THE 8087.
```

```
C
C     DATA SMALL(1) / 2.23D-308  /
C     DATA LARGE(1) / 1.79D+308  /
C     DATA RIGHT(1) / 1.11D-16   /
C     DATA DIVER(1) / 2.22D-16   /
C     DATA LOG10(1) / 0.301029995663981195D0 /
C
C     MACHINE CONSTANTS FOR THE IBM RS 6000
C
C     DATA DMACH(1) / Z'0010000000000000' /
C     DATA DMACH(2) / Z'7FEFFFFFFFFFFFFF' /
C     DATA DMACH(3) / Z'3CA0000000000000' /
C     DATA DMACH(4) / Z'3CB0000000000000' /
C     DATA DMACH(5) / Z'3FD34413509F79FF' /
C
C     MACHINE CONSTANTS FOR THE INTEL i860
C
C     DATA DMACH(1) / Z'0010000000000000' /
C     DATA DMACH(2) / Z'7FEFFFFFFFFFFFFF' /
C     DATA DMACH(3) / Z'3CA0000000000000' /
C     DATA DMACH(4) / Z'3CB0000000000000' /
C     DATA DMACH(5) / Z'3FD34413509F79FF' /
C
C     MACHINE CONSTANTS FOR THE PDP-10 (KA PROCESSOR)
C
C     DATA SMALL(1), SMALL(2) / "033400000000, "000000000000 /
C     DATA LARGE(1), LARGE(2) / "377777777777, "344777777777 /
C     DATA RIGHT(1), RIGHT(2) / "113400000000, "000000000000 /
C     DATA DIVER(1), DIVER(2) / "114400000000, "000000000000 /
C     DATA LOG10(1), LOG10(2) / "177464202324, "144117571776 /
C
C     MACHINE CONSTANTS FOR THE PDP-10 (KI PROCESSOR)
C
C     DATA SMALL(1), SMALL(2) / "000400000000, "000000000000 /
C     DATA LARGE(1), LARGE(2) / "377777777777, "377777777777 /
C     DATA RIGHT(1), RIGHT(2) / "103400000000, "000000000000 /
C     DATA DIVER(1), DIVER(2) / "104400000000, "000000000000 /
C     DATA LOG10(1), LOG10(2) / "177464202324, "476747767461 /
C
C     MACHINE CONSTANTS FOR PDP-11 FORTRAN SUPPORTING
C     32-BIT INTEGERS (EXPRESSED IN INTEGER AND OCTAL).
C
C     DATA SMALL(1), SMALL(2) /    8388608,           0 /
C     DATA LARGE(1), LARGE(2) / 2147483647,          -1 /
C     DATA RIGHT(1), RIGHT(2) /  612368384,           0 /
C     DATA DIVER(1), DIVER(2) /  620756992,           0 /
C     DATA LOG10(1), LOG10(2) / 1067065498, -2063872008 /
C
C     DATA SMALL(1), SMALL(2) / O00040000000, O00000000000 /
C     DATA LARGE(1), LARGE(2) / O17777777777, O37777777777 /
C     DATA RIGHT(1), RIGHT(2) / O04440000000, O00000000000 /
C     DATA DIVER(1), DIVER(2) / O04500000000, O00000000000 /
C     DATA LOG10(1), LOG10(2) / O07746420232, O20476747770 /
C
C     MACHINE CONSTANTS FOR PDP-11 FORTRAN SUPPORTING
C     16-BIT INTEGERS (EXPRESSED IN INTEGER AND OCTAL).
C
C     DATA SMALL(1), SMALL(2) /    128,      0 /
```

```
C     DATA SMALL(3), SMALL(4) /      0,      0 /
C     DATA LARGE(1), LARGE(2) /  32767,     -1 /
C     DATA LARGE(3), LARGE(4) /     -1,     -1 /
C     DATA RIGHT(1), RIGHT(2) /   9344,      0 /
C     DATA RIGHT(3), RIGHT(4) /      0,      0 /
C     DATA DIVER(1), DIVER(2) /   9472,      0 /
C     DATA DIVER(3), DIVER(4) /      0,      0 /
C     DATA LOG10(1), LOG10(2) /  16282,   8346 /
C     DATA LOG10(3), LOG10(4) / -31493, -12296 /
C
C     DATA SMALL(1), SMALL(2) / O000200, O000000 /
C     DATA SMALL(3), SMALL(4) / O000000, O000000 /
C     DATA LARGE(1), LARGE(2) / O077777, O177777 /
C     DATA LARGE(3), LARGE(4) / O177777, O177777 /
C     DATA RIGHT(1), RIGHT(2) / O022200, O000000 /
C     DATA RIGHT(3), RIGHT(4) / O000000, O000000 /
C     DATA DIVER(1), DIVER(2) / O022400, O000000 /
C     DATA DIVER(3), DIVER(4) / O000000, O000000 /
C     DATA LOG10(1), LOG10(2) / O037632, O020232 /
C     DATA LOG10(3), LOG10(4) / O102373, O147770 /
C
C     MACHINE CONSTANTS FOR THE SILICON GRAPHICS
C
      DATA SMALL(1), SMALL(2) / Z'00100000', Z'00000000' /
      DATA LARGE(1), LARGE(2) / Z'7FEFFFFF', Z'FFFFFFFF' /
      DATA RIGHT(1), RIGHT(2) / Z'3CA00000', Z'00000000' /
      DATA DIVER(1), DIVER(2) / Z'3CB00000', Z'00000000' /
      DATA LOG10(1), LOG10(2) / Z'3FD34413', Z'509F79FF' /
C
C     MACHINE CONSTANTS FOR THE SUN
C
c     DATA DMACH(1) / Z'0010000000000000' /
c     DATA DMACH(2) / Z'7FEFFFFFFFFFFFFF' /
c     DATA DMACH(3) / Z'3CA0000000000000' /
c     DATA DMACH(4) / Z'3CB0000000000000' /
c     DATA DMACH(5) / Z'3FD34413509F79FF' /
C
C     MACHINE CONSTANTS FOR THE SUN
C     USING THE -r8 COMPILER OPTION
C
C     DATA DMACH(1) / Z'00010000000000000000000000000000' /
C     DATA DMACH(2) / Z'7FFEFFFFFFFFFFFFFFFFFFFFFFFFFFFF' /
C     DATA DMACH(3) / Z'3F8E0000000000000000000000000000' /
C     DATA DMACH(4) / Z'3F8F0000000000000000000000000000' /
C     DATA DMACH(5) / Z'3FFD34413509F79FEF311F12B35816F9' /
C
C     MACHINE CONSTANTS FOR THE SUN 386i
C
C     DATA SMALL(1), SMALL(2) / Z'FFFFFFFD', Z'000FFFFF' /
C     DATA LARGE(1), LARGE(2) / Z'FFFFFFB0', Z'7FEFFFFF' /
C     DATA RIGHT(1), RIGHT(2) / Z'000000B0', Z'3CA00000' /
C     DATA DIVER(1), DIVER(2) / Z'FFFFFFCB', Z'3CAFFFFF'
C     DATA LOG10(1), LOG10(2) / Z'509F79E9', Z'3FD34413' /
C
C     MACHINE CONSTANTS FOR THE UNIVAC 1100 SERIES FTN COMPILER
C
C     DATA SMALL(1), SMALL(2) / O000040000000, O000000000000 /
C     DATA LARGE(1), LARGE(2) / O377777777777, O777777777777 /
```

```
C     DATA RIGHT(1), RIGHT(2) / O170540000000, O000000000000 /
C     DATA DIVER(1), DIVER(2) / O170640000000, O000000000000 /
C     DATA LOG10(1), LOG10(2) / O177746420232, O411757177572 /
C
C***FIRST EXECUTABLE STATEMENT  D1MACH
      IF (I .LT. 1 .OR. I .GT. 5) CALL XERMSG ('SLATEC', 'D1MACH',
     +   'I OUT OF BOUNDS', 1, 2)
C
      D1MACH = DMACH(I)
      RETURN
C
      END
*DECK I1MACH
      INTEGER FUNCTION I1MACH (I)
C***BEGIN PROLOGUE  I1MACH
C***PURPOSE  Return integer machine dependent constants.
C***LIBRARY   SLATEC
C***CATEGORY  R1
C***TYPE      INTEGER (I1MACH-I)
C***KEYWORDS  MACHINE CONSTANTS
C***AUTHOR  Fox, P. A., (Bell Labs)
C           Hall, A. D., (Bell Labs)
C           Schryer, N. L., (Bell Labs)
C***DESCRIPTION
C
C   I1MACH can be used to obtain machine-dependent parameters for the
C   local machine environment.  It is a function subprogram with one
C   (input) argument and can be referenced as follows:
C
C        K = I1MACH(I)
C
C   where I=1,...,16.  The (output) value of K above is determined by
C   the (input) value of I.  The results for various values of I are
C   discussed below.
C
C   I/O unit numbers:
C     I1MACH( 1) = the standard input unit.
C     I1MACH( 2) = the standard output unit.
C     I1MACH( 3) = the standard punch unit.
C     I1MACH( 4) = the standard error message unit.
C
C   Words:
C     I1MACH( 5) = the number of bits per integer storage unit.
C     I1MACH( 6) = the number of characters per integer storage unit.
C
C   Integers:
C     assume integers are represented in the S-digit, base-A form
C
C                sign ( X(S-1)*A**(S-1) + ... + X(1)*A + X(0) )
C
C                where 0 .LE. X(I) .LT. A for I=0,...,S-1.
C     I1MACH( 7) = A, the base.
C     I1MACH( 8) = S, the number of base-A digits.
C     I1MACH( 9) = A**S - 1, the largest magnitude.
C
C   Floating-Point Numbers:
C     Assume floating-point numbers are represented in the T-digit,
C     base-B form
```

```
C                sign (B**E)*( (X(1)/B) + ... + (X(T)/B**T) )
C
C                where 0 .LE. X(I) .LT. B for I=1,...,T,
C                0 .LT. X(1), and EMIN .LE. E .LE. EMAX.
C     I1MACH(10) = B, the base.
C
C   Single-Precision:
C     I1MACH(11) = T, the number of base-B digits.
C     I1MACH(12) = EMIN, the smallest exponent E.
C     I1MACH(13) = EMAX, the largest exponent E.
C
C   Double-Precision:
C     I1MACH(14) = T, the number of base-B digits.
C     I1MACH(15) = EMIN, the smallest exponent E.
C     I1MACH(16) = EMAX, the largest exponent E.
C
C   To alter this function for a particular environment, the desired
C   set of DATA statements should be activated by removing the C from
C   column 1.  Also, the values of I1MACH(1) - I1MACH(4) should be
C   checked for consistency with the local operating system.
C
C***REFERENCES  P. A. Fox, A. D. Hall and N. L. Schryer, Framework for
C                 a portable library, ACM Transactions on Mathematical
C                 Software 4, 2 (June 1978), pp. 177-188.
C***ROUTINES CALLED  (NONE)
C***REVISION HISTORY  (YYMMDD)
C   750101  DATE WRITTEN
C   891012  Added VAX G-floating constants.  (WRB)
C   891012  REVISION DATE from Version 3.2
C   891214  Prologue converted to Version 4.0 format.  (BAB)
C   900618  Added DEC RISC constants.  (WRB)
C   900723  Added IBM RS 6000 constants.  (WRB)
C   901009  Correct I1MACH(7) for IBM Mainframes. Should be 2 not 16.
C           (RWC)
C   910710  Added HP 730 constants.  (SMR)
C   911114  Added Convex IEEE constants.  (WRB)
C   920121  Added SUN -r8 compiler option constants.  (WRB)
C   920229  Added Touchstone Delta i860 constants.  (WRB)
C   920501  Reformatted the REFERENCES section.  (WRB)
C   920625  Added Convex -p8 and -pd8 compiler option constants.
C           (BKS, WRB)
C   930201  Added DEC Alpha and SGI constants.  (RWC and WRB)
C   930618  Corrected I1MACH(5) for Convex -p8 and -pd8 compiler
C           options.  (DWL, RWC and WRB).
C***END PROLOGUE  I1MACH
C
      INTEGER IMACH(16),OUTPUT
      SAVE IMACH
      EQUIVALENCE (IMACH(4),OUTPUT)
C
C     MACHINE CONSTANTS FOR THE AMIGA
C     ABSOFT COMPILER
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          5 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         32 /
```

```
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         31 /
C     DATA IMACH( 9) / 2147483647 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -126 /
C     DATA IMACH(13) /        127 /
C     DATA IMACH(14) /         53 /
C     DATA IMACH(15) /      -1022 /
C     DATA IMACH(16) /       1023 /
C
C     MACHINE CONSTANTS FOR THE APOLLO
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          6 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         32 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         31 /
C     DATA IMACH( 9) / 2147483647 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -125 /
C     DATA IMACH(13) /        129 /
C     DATA IMACH(14) /         53 /
C     DATA IMACH(15) /      -1021 /
C     DATA IMACH(16) /       1025 /
C
C     MACHINE CONSTANTS FOR THE BURROUGHS 1700 SYSTEM
C
C     DATA IMACH( 1) /          7 /
C     DATA IMACH( 2) /          2 /
C     DATA IMACH( 3) /          2 /
C     DATA IMACH( 4) /          2 /
C     DATA IMACH( 5) /         36 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         33 /
C     DATA IMACH( 9) / Z1FFFFFFFF /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -256 /
C     DATA IMACH(13) /        255 /
C     DATA IMACH(14) /         60 /
C     DATA IMACH(15) /       -256 /
C     DATA IMACH(16) /        255 /
C
C     MACHINE CONSTANTS FOR THE BURROUGHS 5700 SYSTEM
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          7 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         48 /
C     DATA IMACH( 6) /          6 /
```

```
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         39 /
C     DATA IMACH( 9) / O0007777777777777 /
C     DATA IMACH(10) /          8 /
C     DATA IMACH(11) /         13 /
C     DATA IMACH(12) /        -50 /
C     DATA IMACH(13) /         76 /
C     DATA IMACH(14) /         26 /
C     DATA IMACH(15) /        -50 /
C     DATA IMACH(16) /         76 /
C
C     MACHINE CONSTANTS FOR THE BURROUGHS 6700/7700 SYSTEMS
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          7 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         48 /
C     DATA IMACH( 6) /          6 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         39 /
C     DATA IMACH( 9) / O0007777777777777 /
C     DATA IMACH(10) /          8 /
C     DATA IMACH(11) /         13 /
C     DATA IMACH(12) /        -50 /
C     DATA IMACH(13) /         76 /
C     DATA IMACH(14) /         26 /
C     DATA IMACH(15) /     -32754 /
C     DATA IMACH(16) /      32780 /
C
C     MACHINE CONSTANTS FOR THE CDC 170/180 SERIES USING NOS/VE
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          7 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         64 /
C     DATA IMACH( 6) /          8 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         63 /
C     DATA IMACH( 9) / 9223372036854775807 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         47 /
C     DATA IMACH(12) /      -4095 /
C     DATA IMACH(13) /       4094 /
C     DATA IMACH(14) /         94 /
C     DATA IMACH(15) /      -4095 /
C     DATA IMACH(16) /       4094 /
C
C     MACHINE CONSTANTS FOR THE CDC 6000/7000 SERIES
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          7 /
C     DATA IMACH( 4) /    6LOUTPUT/
C     DATA IMACH( 5) /         60 /
C     DATA IMACH( 6) /         10 /
C     DATA IMACH( 7) /          2 /
```

```
C     DATA IMACH( 8) /         48 /
C     DATA IMACH( 9) / 00007777777777777777B /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         47 /
C     DATA IMACH(12) /       -929 /
C     DATA IMACH(13) /       1070 /
C     DATA IMACH(14) /         94 /
C     DATA IMACH(15) /       -929 /
C     DATA IMACH(16) /       1069 /
C
C     MACHINE CONSTANTS FOR THE CELERITY C1260
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          6 /
C     DATA IMACH( 4) /          0 /
C     DATA IMACH( 5) /         32 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         31 /
C     DATA IMACH( 9) / Z'7FFFFFFF' /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -126 /
C     DATA IMACH(13) /        127 /
C     DATA IMACH(14) /         53 /
C     DATA IMACH(15) /      -1022 /
C     DATA IMACH(16) /       1023 /
C
C     MACHINE CONSTANTS FOR THE CONVEX
C     USING THE -fn COMPILER OPTION
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          7 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         32 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         31 /
C     DATA IMACH( 9) / 2147483647 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -127 /
C     DATA IMACH(13) /        127 /
C     DATA IMACH(14) /         53 /
C     DATA IMACH(15) /      -1023 /
C     DATA IMACH(16) /       1023 /
C
C     MACHINE CONSTANTS FOR THE CONVEX
C     USING THE -fi COMPILER OPTION
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          7 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         32 /
C     DATA IMACH( 6) /          4 /
```

```
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         31 /
C     DATA IMACH( 9) / 2147483647 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -125 /
C     DATA IMACH(13) /        128 /
C     DATA IMACH(14) /         53 /
C     DATA IMACH(15) /      -1021 /
C     DATA IMACH(16) /       1024 /
C
C     MACHINE CONSTANTS FOR THE CONVEX
C     USING THE -p8 COMPILER OPTION
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          7 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         64 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         63 /
C     DATA IMACH( 9) / 9223372036854775807 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         53 /
C     DATA IMACH(12) /      -1023 /
C     DATA IMACH(13) /       1023 /
C     DATA IMACH(14) /        113 /
C     DATA IMACH(15) /     -16383 /
C     DATA IMACH(16) /      16383 /
C
C     MACHINE CONSTANTS FOR THE CONVEX
C     USING THE -pd8 COMPILER OPTION
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          7 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         64 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         63 /
C     DATA IMACH( 9) / 9223372036854775807 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         53 /
C     DATA IMACH(12) /      -1023 /
C     DATA IMACH(13) /       1023 /
C     DATA IMACH(14) /         53 /
C     DATA IMACH(15) /      -1023 /
C     DATA IMACH(16) /       1023 /
C
C     MACHINE CONSTANTS FOR THE CRAY
C     USING THE 46 BIT INTEGER COMPILER OPTION
C
C     DATA IMACH( 1) /        100 /
C     DATA IMACH( 2) /        101 /
C     DATA IMACH( 3) /        102 /
C     DATA IMACH( 4) /        101 /
```

```
C     DATA IMACH( 5) /         64 /
C     DATA IMACH( 6) /          8 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         46 /
C     DATA IMACH( 9) / 1777777777777777B /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         47 /
C     DATA IMACH(12) /      -8189 /
C     DATA IMACH(13) /       8190 /
C     DATA IMACH(14) /         94 /
C     DATA IMACH(15) /      -8099 /
C     DATA IMACH(16) /       8190 /
C
C     MACHINE CONSTANTS FOR THE CRAY
C     USING THE 64 BIT INTEGER COMPILER OPTION
C
C     DATA IMACH( 1) /        100 /
C     DATA IMACH( 2) /        101 /
C     DATA IMACH( 3) /        102 /
C     DATA IMACH( 4) /        101 /
C     DATA IMACH( 5) /         64 /
C     DATA IMACH( 6) /          8 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         63 /
C     DATA IMACH( 9) / 777777777777777777B /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         47 /
C     DATA IMACH(12) /      -8189 /
C     DATA IMACH(13) /       8190 /
C     DATA IMACH(14) /         94 /
C     DATA IMACH(15) /      -8099 /
C     DATA IMACH(16) /       8190 /
C
C     MACHINE CONSTANTS FOR THE DATA GENERAL ECLIPSE S/200
C
C     DATA IMACH( 1) /         11 /
C     DATA IMACH( 2) /         12 /
C     DATA IMACH( 3) /          8 /
C     DATA IMACH( 4) /         10 /
C     DATA IMACH( 5) /         16 /
C     DATA IMACH( 6) /          2 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         15 /
C     DATA IMACH( 9) /      32767 /
C     DATA IMACH(10) /         16 /
C     DATA IMACH(11) /          6 /
C     DATA IMACH(12) /        -64 /
C     DATA IMACH(13) /         63 /
C     DATA IMACH(14) /         14 /
C     DATA IMACH(15) /        -64 /
C     DATA IMACH(16) /         63 /
C
C     MACHINE CONSTANTS FOR THE DEC ALPHA
C     USING G_FLOAT
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          5 /
```

```
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         32 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         31 /
C     DATA IMACH( 9) / 2147483647 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -127 /
C     DATA IMACH(13) /        127 /
C     DATA IMACH(14) /         53 /
C     DATA IMACH(15) /      -1023 /
C     DATA IMACH(16) /       1023 /
C
C     MACHINE CONSTANTS FOR THE DEC ALPHA
C     USING IEEE_FLOAT
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          6 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         32 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         31 /
C     DATA IMACH( 9) / 2147483647 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -125 /
C     DATA IMACH(13) /        128 /
C     DATA IMACH(14) /         53 /
C     DATA IMACH(15) /      -1021 /
C     DATA IMACH(16) /       1024 /
C
C     MACHINE CONSTANTS FOR THE DEC RISC
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          6 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         32 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         31 /
C     DATA IMACH( 9) / 2147483647 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -125 /
C     DATA IMACH(13) /        128 /
C     DATA IMACH(14) /         53 /
C     DATA IMACH(15) /      -1021 /
C     DATA IMACH(16) /       1024 /
C
C     MACHINE CONSTANTS FOR THE DEC VAX
C     USING D_FLOATING
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
```

```
C     DATA IMACH( 3) /          5 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         32 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         31 /
C     DATA IMACH( 9) / 2147483647 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -127 /
C     DATA IMACH(13) /        127 /
C     DATA IMACH(14) /         56 /
C     DATA IMACH(15) /       -127 /
C     DATA IMACH(16) /        127 /
C
C     MACHINE CONSTANTS FOR THE DEC VAX
C     USING G_FLOATING
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          5 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         32 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         31 /
C     DATA IMACH( 9) / 2147483647 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -127 /
C     DATA IMACH(13) /        127 /
C     DATA IMACH(14) /         53 /
C     DATA IMACH(15) /      -1023 /
C     DATA IMACH(16) /       1023 /
C
C     MACHINE CONSTANTS FOR THE ELXSI 6400
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          6 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         32 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         32 /
C     DATA IMACH( 9) / 2147483647 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -126 /
C     DATA IMACH(13) /        127 /
C     DATA IMACH(14) /         53 /
C     DATA IMACH(15) /      -1022 /
C     DATA IMACH(16) /       1023 /
C
C     MACHINE CONSTANTS FOR THE HARRIS 220
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
```

```
C     DATA IMACH( 3) /          0 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         24 /
C     DATA IMACH( 6) /          3 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         23 /
C     DATA IMACH( 9) /    8388607 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         23 /
C     DATA IMACH(12) /       -127 /
C     DATA IMACH(13) /        127 /
C     DATA IMACH(14) /         38 /
C     DATA IMACH(15) /       -127 /
C     DATA IMACH(16) /        127 /
C
C     MACHINE CONSTANTS FOR THE HONEYWELL 600/6000 SERIES
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /         43 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         36 /
C     DATA IMACH( 6) /          6 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         35 /
C     DATA IMACH( 9) / O377777777777 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         27 /
C     DATA IMACH(12) /       -127 /
C     DATA IMACH(13) /        127 /
C     DATA IMACH(14) /         63 /
C     DATA IMACH(15) /       -127 /
C     DATA IMACH(16) /        127 /
C
C     MACHINE CONSTANTS FOR THE HP 730
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          6 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         32 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         31 /
C     DATA IMACH( 9) / 2147483647 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -125 /
C     DATA IMACH(13) /        128 /
C     DATA IMACH(14) /         53 /
C     DATA IMACH(15) /      -1021 /
C     DATA IMACH(16) /       1024 /
C
C     MACHINE CONSTANTS FOR THE HP 2100
C     3 WORD DOUBLE PRECISION OPTION WITH FTN4
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
```

```
C     DATA IMACH( 3) /          4 /
C     DATA IMACH( 4) /          1 /
C     DATA IMACH( 5) /         16 /
C     DATA IMACH( 6) /          2 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         15 /
C     DATA IMACH( 9) /      32767 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         23 /
C     DATA IMACH(12) /       -128 /
C     DATA IMACH(13) /        127 /
C     DATA IMACH(14) /         39 /
C     DATA IMACH(15) /       -128 /
C     DATA IMACH(16) /        127 /
C
C     MACHINE CONSTANTS FOR THE HP 2100
C     4 WORD DOUBLE PRECISION OPTION WITH FTN4
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          4 /
C     DATA IMACH( 4) /          1 /
C     DATA IMACH( 5) /         16 /
C     DATA IMACH( 6) /          2 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         15 /
C     DATA IMACH( 9) /      32767 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         23 /
C     DATA IMACH(12) /       -128 /
C     DATA IMACH(13) /        127 /
C     DATA IMACH(14) /         55 /
C     DATA IMACH(15) /       -128 /
C     DATA IMACH(16) /        127 /
C
C     MACHINE CONSTANTS FOR THE HP 9000
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          6 /
C     DATA IMACH( 4) /          7 /
C     DATA IMACH( 5) /         32 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         32 /
C     DATA IMACH( 9) / 2147483647 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -126 /
C     DATA IMACH(13) /        127 /
C     DATA IMACH(14) /         53 /
C     DATA IMACH(15) /      -1015 /
C     DATA IMACH(16) /       1017 /
C
C     MACHINE CONSTANTS FOR THE IBM 360/370 SERIES,
C     THE XEROX SIGMA 5/7/9, THE SEL SYSTEMS 85/86, AND
C     THE PERKIN ELMER (INTERDATA) 7/32.
C
```

```
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          7 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         32 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         31 /
C     DATA IMACH( 9) /  Z7FFFFFFF /
C     DATA IMACH(10) /         16 /
C     DATA IMACH(11) /          6 /
C     DATA IMACH(12) /        -64 /
C     DATA IMACH(13) /         63 /
C     DATA IMACH(14) /         14 /
C     DATA IMACH(15) /        -64 /
C     DATA IMACH(16) /         63 /
C
C     MACHINE CONSTANTS FOR THE IBM PC
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          0 /
C     DATA IMACH( 4) /          0 /
C     DATA IMACH( 5) /         32 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         31 /
C     DATA IMACH( 9) / 2147483647 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -125 /
C     DATA IMACH(13) /        127 /
C     DATA IMACH(14) /         53 /
C     DATA IMACH(15) /      -1021 /
C     DATA IMACH(16) /       1023 /
C
C     MACHINE CONSTANTS FOR THE IBM RS 6000
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          6 /
C     DATA IMACH( 4) /          0 /
C     DATA IMACH( 5) /         32 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         31 /
C     DATA IMACH( 9) / 2147483647 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -125 /
C     DATA IMACH(13) /        128 /
C     DATA IMACH(14) /         53 /
C     DATA IMACH(15) /      -1021 /
C     DATA IMACH(16) /       1024 /
C
C     MACHINE CONSTANTS FOR THE INTEL i860
C
C     DATA IMACH( 1) /          5 /
```

```
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          6 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         32 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         31 /
C     DATA IMACH( 9) / 2147483647 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -125 /
C     DATA IMACH(13) /        128 /
C     DATA IMACH(14) /         53 /
C     DATA IMACH(15) /      -1021 /
C     DATA IMACH(16) /       1024 /
C
C     MACHINE CONSTANTS FOR THE PDP-10 (KA PROCESSOR)
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          5 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         36 /
C     DATA IMACH( 6) /          5 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         35 /
C     DATA IMACH( 9) / "377777777777 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         27 /
C     DATA IMACH(12) /       -128 /
C     DATA IMACH(13) /        127 /
C     DATA IMACH(14) /         54 /
C     DATA IMACH(15) /       -101 /
C     DATA IMACH(16) /        127 /
C
C     MACHINE CONSTANTS FOR THE PDP-10 (KI PROCESSOR)
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          5 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         36 /
C     DATA IMACH( 6) /          5 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         35 /
C     DATA IMACH( 9) / "377777777777 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         27 /
C     DATA IMACH(12) /       -128 /
C     DATA IMACH(13) /        127 /
C     DATA IMACH(14) /         62 /
C     DATA IMACH(15) /       -128 /
C     DATA IMACH(16) /        127 /
C
C     MACHINE CONSTANTS FOR PDP-11 FORTRAN SUPPORTING
C     32-BIT INTEGER ARITHMETIC.
C
C     DATA IMACH( 1) /          5 /
```

```
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          5 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         32 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         31 /
C     DATA IMACH( 9) / 2147483647 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -127 /
C     DATA IMACH(13) /        127 /
C     DATA IMACH(14) /         56 /
C     DATA IMACH(15) /       -127 /
C     DATA IMACH(16) /        127 /
C
C     MACHINE CONSTANTS FOR PDP-11 FORTRAN SUPPORTING
C     16-BIT INTEGER ARITHMETIC.
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          5 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         16 /
C     DATA IMACH( 6) /          2 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         15 /
C     DATA IMACH( 9) /      32767 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -127 /
C     DATA IMACH(13) /        127 /
C     DATA IMACH(14) /         56 /
C     DATA IMACH(15) /       -127 /
C     DATA IMACH(16) /        127 /
C
C     MACHINE CONSTANTS FOR THE SILICON GRAPHICS
C
      DATA IMACH( 1) /          5 /
      DATA IMACH( 2) /          6 /
      DATA IMACH( 3) /          6 /
      DATA IMACH( 4) /          6 /
      DATA IMACH( 5) /         32 /
      DATA IMACH( 6) /          4 /
      DATA IMACH( 7) /          2 /
      DATA IMACH( 8) /         31 /
      DATA IMACH( 9) / 2147483647 /
      DATA IMACH(10) /          2 /
      DATA IMACH(11) /         24 /
      DATA IMACH(12) /       -125 /
      DATA IMACH(13) /        128 /
      DATA IMACH(14) /         53 /
      DATA IMACH(15) /      -1021 /
      DATA IMACH(16) /       1024 /
C
C     MACHINE CONSTANTS FOR THE SUN
C
c     DATA IMACH( 1) /          5 /
```

```
c     DATA IMACH( 2) /          6 /
c     DATA IMACH( 3) /          6 /
c     DATA IMACH( 4) /          6 /
c     DATA IMACH( 5) /         32 /
c     DATA IMACH( 6) /          4 /
c     DATA IMACH( 7) /          2 /
c     DATA IMACH( 8) /         31 /
c     DATA IMACH( 9) / 2147483647 /
c     DATA IMACH(10) /          2 /
c     DATA IMACH(11) /         24 /
c     DATA IMACH(12) /       -125 /
c     DATA IMACH(13) /        128 /
c     DATA IMACH(14) /         53 /
c     DATA IMACH(15) /      -1021 /
c     DATA IMACH(16) /       1024 /
C
C     MACHINE CONSTANTS FOR THE SUN
C     USING THE -r8 COMPILER OPTION
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          6 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         32 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         31 /
C     DATA IMACH( 9) / 2147483647 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         53 /
C     DATA IMACH(12) /      -1021 /
C     DATA IMACH(13) /       1024 /
C     DATA IMACH(14) /        113 /
C     DATA IMACH(15) /     -16381 /
C     DATA IMACH(16) /      16384 /
C
C     MACHINE CONSTANTS FOR THE UNIVAC 1100 SERIES FTN COMPILER
C
C     DATA IMACH( 1) /          5 /
C     DATA IMACH( 2) /          6 /
C     DATA IMACH( 3) /          1 /
C     DATA IMACH( 4) /          6 /
C     DATA IMACH( 5) /         36 /
C     DATA IMACH( 6) /          4 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         35 /
C     DATA IMACH( 9) / O377777777777 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         27 /
C     DATA IMACH(12) /       -128 /
C     DATA IMACH(13) /        127 /
C     DATA IMACH(14) /         60 /
C     DATA IMACH(15) /      -1024 /
C     DATA IMACH(16) /       1023 /
C
C     MACHINE CONSTANTS FOR THE Z80 MICROPROCESSOR
C
C     DATA IMACH( 1) /          1 /
```

```
C     DATA IMACH( 2) /          1 /
C     DATA IMACH( 3) /          0 /
C     DATA IMACH( 4) /          1 /
C     DATA IMACH( 5) /         16 /
C     DATA IMACH( 6) /          2 /
C     DATA IMACH( 7) /          2 /
C     DATA IMACH( 8) /         15 /
C     DATA IMACH( 9) /      32767 /
C     DATA IMACH(10) /          2 /
C     DATA IMACH(11) /         24 /
C     DATA IMACH(12) /       -127 /
C     DATA IMACH(13) /        127 /
C     DATA IMACH(14) /         56 /
C     DATA IMACH(15) /       -127 /
C     DATA IMACH(16) /        127 /
C
C***FIRST EXECUTABLE STATEMENT  I1MACH
      IF (I .LT. 1  .OR.  I .GT. 16) GO TO 10
C
      I1MACH = IMACH(I)
      RETURN
C
   10 CONTINUE
      WRITE (UNIT = OUTPUT, FMT = 9000)
 9000 FORMAT ('1ERROR    1 IN I1MACH - I OUT OF BOUNDS')
C
C     CALL FDUMP
C
      STOP
      END
*DECK XERMSG
      SUBROUTINE XERMSG (LIBRAR, SUBROU, MESSG, NERR, LEVEL)
C***BEGIN PROLOGUE  XERMSG
C***PURPOSE  Process error messages for SLATEC and other libraries.
C***LIBRARY   SLATEC (XERROR)
C***CATEGORY  R3C
C***TYPE      ALL (XERMSG-A)
C***KEYWORDS  ERROR MESSAGE, XERROR
C***AUTHOR  Fong, Kirby, (NMFECC at LLNL)
C***DESCRIPTION
C
C   XERMSG processes a diagnostic message in a manner determined by the
C   value of LEVEL and the current value of the library error control
C   flag, KONTRL.  See subroutine XSETF for details.
C
C    LIBRAR   A character constant (or character variable) with the name
C             of the library.  This will be 'SLATEC' for the SLATEC
C             Common Math Library.  The error handling package is
C             general enough to be used by many libraries
C             simultaneously, so it is desirable for the routine that
C             detects and reports an error to identify the library name
C             as well as the routine name.
C
C    SUBROU   A character constant (or character variable) with the name
C             of the routine that detected the error.  Usually it is the
C             name of the routine that is calling XERMSG.  There are
C             some instances where a user callable library routine calls
C             lower level subsidiary routines where the error is
```

```
C             detected.  In such cases it may be more informative to
C             supply the name of the routine the user called rather than
C             the name of the subsidiary routine that detected the
C             error.
C
C    MESSG    A character constant (or character variable) with the text
C             of the error or warning message.  In the example below,
C             the message is a character constant that contains a
C             generic message.
C
C                   CALL XERMSG ('SLATEC', 'MMPY',
C                  *'THE ORDER OF THE MATRIX EXCEEDS THE ROW DIMENSION',
C                  *3, 1)
C
C             It is possible (and is sometimes desirable) to generate a
C             specific message--e.g., one that contains actual numeric
C             values.  Specific numeric values can be converted into
C             character strings using formatted WRITE statements into
C             character variables.  This is called standard Fortran
C             internal file I/O and is exemplified in the first three
C             lines of the following example.  You can also catenate
C             substrings of characters to construct the error message.
C             Here is an example showing the use of both writing to
C             an internal file and catenating character strings.
C
C                   CHARACTER*5 CHARN, CHARL
C                   WRITE (CHARN,10) N
C                   WRITE (CHARL,10) LDA
C                10 FORMAT(I5)
C                   CALL XERMSG ('SLATEC', 'MMPY', 'THE ORDER'//CHARN//
C                  *   ' OF THE MATRIX EXCEEDS ITS ROW DIMENSION OF'//
C                  *   CHARL, 3, 1)
C
C             There are two subtleties worth mentioning.  One is that
C             the // for character catenation is used to construct the
C             error message so that no single character constant is
C             continued to the next line.  This avoids confusion as to
C             whether there are trailing blanks at the end of the line.
C             The second is that by catenating the parts of the message
C             as an actual argument rather than encoding the entire
C             message into one large character variable, we avoid
C             having to know how long the message will be in order to
C             declare an adequate length for that large character
C             variable.  XERMSG calls XERPRN to print the message using
C             multiple lines if necessary.  If the message is very long,
C             XERPRN will break it into pieces of 72 characters (as
C             requested by XERMSG) for printing on multiple lines.
C             Also, XERMSG asks XERPRN to prefix each line with ' *  '
C             so that the total line length could be 76 characters.
C             Note also that XERPRN scans the error message backwards
C             to ignore trailing blanks.  Another feature is that
C             the substring '$$' is treated as a new line sentinel
C             by XERPRN.  If you want to construct a multiline
C             message without having to count out multiples of 72
C             characters, just use '$$' as a separator.  '$$'
C             obviously must occur within 72 characters of the
C             start of each line to have its intended effect since
C             XERPRN is asked to wrap around at 72 characters in
```

```
C             addition to looking for '$$'.
C
C    NERR     An integer value that is chosen by the library routine's
C             author.  It must be in the range -99 to 999 (three
C             printable digits).  Each distinct error should have its
C             own error number.  These error numbers should be described
C             in the machine readable documentation for the routine.
C             The error numbers need be unique only within each routine,
C             so it is reasonable for each routine to start enumerating
C             errors from 1 and proceeding to the next integer.
C
C    LEVEL    An integer value in the range 0 to 2 that indicates the
C             level (severity) of the error.  Their meanings are
C
C            -1  A warning message.  This is used if it is not clear
C                that there really is an error, but the user's attention
C                may be needed.  An attempt is made to only print this
C                message once.
C
C             0  A warning message.  This is used if it is not clear
C                that there really is an error, but the user's attention
C                may be needed.
C
C             1  A recoverable error.  This is used even if the error is
C                so serious that the routine cannot return any useful
C                answer.  If the user has told the error package to
C                return after recoverable errors, then XERMSG will
C                return to the Library routine which can then return to
C                the user's routine.  The user may also permit the error
C                package to terminate the program upon encountering a
C                recoverable error.
C
C             2  A fatal error.  XERMSG will not return to its caller
C                after it receives a fatal error.  This level should
C                hardly ever be used; it is much better to allow the
C                user a chance to recover.  An example of one of the few
C                cases in which it is permissible to declare a level 2
C                error is a reverse communication Library routine that
C                is likely to be called repeatedly until it integrates
C                across some interval.  If there is a serious error in
C                the input such that another step cannot be taken and
C                the Library routine is called again without the input
C                error having been corrected by the caller, the Library
C                routine will probably be called forever with improper
C                input.  In this case, it is reasonable to declare the
C                error to be fatal.
C
C    Each of the arguments to XERMSG is input; none will be modified by
C    XERMSG.  A routine may make multiple calls to XERMSG with warning
C    level messages; however, after a call to XERMSG with a recoverable
C    error, the routine should return to the user.  Do not try to call
C    XERMSG with a second recoverable error after the first recoverable
C    error because the error package saves the error number.  The user
C    can retrieve this error number by calling another entry point in
C    the error handling package and then clear the error number when
C    recovering from the error.  Calling XERMSG in succession causes the
C    old error number to be overwritten by the latest error number.
C    This is considered harmless for error numbers associated with
```

```
C    warning messages but must not be done for error numbers of serious
C    errors.  After a call to XERMSG with a recoverable error, the user
C    must be given a chance to call NUMXER or XERCLR to retrieve or
C    clear the error number.
C***REFERENCES  R. E. Jones and D. K. Kahaner, XERROR, the SLATEC
C                 Error-handling Package, SAND82-0800, Sandia
C                 Laboratories, 1982.
C***ROUTINES CALLED  FDUMP, J4SAVE, XERCNT, XERHLT, XERPRN, XERSVE
C***REVISION HISTORY  (YYMMDD)
C   880101  DATE WRITTEN
C   880621  REVISED AS DIRECTED AT SLATEC CML MEETING OF FEBRUARY 1988.
C           THERE ARE TWO BASIC CHANGES.
C           1.  A NEW ROUTINE, XERPRN, IS USED INSTEAD OF XERPRT TO
C               PRINT MESSAGES.  THIS ROUTINE WILL BREAK LONG MESSAGES
C               INTO PIECES FOR PRINTING ON MULTIPLE LINES.  '$$' IS
C               ACCEPTED AS A NEW LINE SENTINEL.  A PREFIX CAN BE
C               ADDED TO EACH LINE TO BE PRINTED.  XERMSG USES EITHER
C               ' ***' OR ' *  ' AND LONG MESSAGES ARE BROKEN EVERY
C               72 CHARACTERS (AT MOST) SO THAT THE MAXIMUM LINE
C               LENGTH OUTPUT CAN NOW BE AS GREAT AS 76.
C           2.  THE TEXT OF ALL MESSAGES IS NOW IN UPPER CASE SINCE THE
C               FORTRAN STANDARD DOCUMENT DOES NOT ADMIT THE EXISTENCE
C               OF LOWER CASE.
C   880708  REVISED AFTER THE SLATEC CML MEETING OF JUNE 29 AND 30.
C           THE PRINCIPAL CHANGES ARE
C           1.  CLARIFY COMMENTS IN THE PROLOGUES
C           2.  RENAME XRPRNT TO XERPRN
C           3.  REWORK HANDLING OF '$$' IN XERPRN TO HANDLE BLANK LINES
C               SIMILAR TO THE WAY FORMAT STATEMENTS HANDLE THE /
C               CHARACTER FOR NEW RECORDS.
C   890706  REVISED WITH THE HELP OF FRED FRITSCH AND REG CLEMENS TO
C           CLEAN UP THE CODING.
C   890721  REVISED TO USE NEW FEATURE IN XERPRN TO COUNT CHARACTERS IN
C           PREFIX.
C   891013  REVISED TO CORRECT COMMENTS.
C   891214  Prologue converted to Version 4.0 format.  (WRB)
C   900510  Changed test on NERR to be -9999999 < NERR < 99999999, but
C           NERR .ne. 0, and on LEVEL to be -2 < LEVEL < 3.  Added
C           LEVEL=-1 logic, changed calls to XERSAV to XERSVE, and
C           XERCTL to XERCNT.  (RWC)
C   920501  Reformatted the REFERENCES section.  (WRB)
C***END PROLOGUE  XERMSG
      CHARACTER*(*) LIBRAR, SUBROU, MESSG
      CHARACTER*8 XLIBR, XSUBR
      CHARACTER*72  TEMP
      CHARACTER*20  LFIRST
C***FIRST EXECUTABLE STATEMENT  XERMSG
      LKNTRL = J4SAVE (2, 0, .FALSE.)
      MAXMES = J4SAVE (4, 0, .FALSE.)
C
C       LKNTRL IS A LOCAL COPY OF THE CONTROL FLAG KONTRL.
C       MAXMES IS THE MAXIMUM NUMBER OF TIMES ANY PARTICULAR MESSAGE
C          SHOULD BE PRINTED.
C
C       WE PRINT A FATAL ERROR MESSAGE AND TERMINATE FOR AN ERROR IN
C          CALLING XERMSG.  THE ERROR NUMBER SHOULD BE POSITIVE,
C          AND THE LEVEL SHOULD BE BETWEEN 0 AND 2.
C
```

```
      IF (NERR.LT.-9999999 .OR. NERR.GT.99999999 .OR. NERR.EQ.0 .OR.
     *   LEVEL.LT.-1 .OR. LEVEL.GT.2) THEN
         CALL XERPRN (' ***', -1, 'FATAL ERROR IN...$$ ' //
     *      'XERMSG -- INVALID ERROR NUMBER OR LEVEL$$ '//
     *      'JOB ABORT DUE TO FATAL ERROR.', 72)
         CALL XERSVE (' ', ' ', ' ', 0, 0, 0, KDUMMY)
         CALL XERHLT (' ***XERMSG -- INVALID INPUT')
         RETURN
      ENDIF
C
C       RECORD THE MESSAGE.
C
      I = J4SAVE (1, NERR, .TRUE.)
      CALL XERSVE (LIBRAR, SUBROU, MESSG, 1, NERR, LEVEL, KOUNT)
C
C       HANDLE PRINT-ONCE WARNING MESSAGES.
C
      IF (LEVEL.EQ.-1 .AND. KOUNT.GT.1) RETURN
C
C       ALLOW TEMPORARY USER OVERRIDE OF THE CONTROL FLAG.
C
      XLIBR  = LIBRAR
      XSUBR  = SUBROU
      LFIRST = MESSG
      LERR   = NERR
      LLEVEL = LEVEL
      CALL XERCNT (XLIBR, XSUBR, LFIRST, LERR, LLEVEL, LKNTRL)
C
      LKNTRL = MAX(-2, MIN(2,LKNTRL))
      MKNTRL = ABS(LKNTRL)
C
C       SKIP PRINTING IF THE CONTROL FLAG VALUE AS RESET IN XERCNT IS
C       ZERO AND THE ERROR IS NOT FATAL.
C
      IF (LEVEL.LT.2 .AND. LKNTRL.EQ.0) GO TO 30
      IF (LEVEL.EQ.0 .AND. KOUNT.GT.MAXMES) GO TO 30
      IF (LEVEL.EQ.1 .AND. KOUNT.GT.MAXMES .AND. MKNTRL.EQ.1) GO TO 30
      IF (LEVEL.EQ.2 .AND. KOUNT.GT.MAX(1,MAXMES)) GO TO 30
C
C       ANNOUNCE THE NAMES OF THE LIBRARY AND SUBROUTINE BY BUILDING A
C       MESSAGE IN CHARACTER VARIABLE TEMP (NOT EXCEEDING 66 CHARACTERS)
C       AND SENDING IT OUT VIA XERPRN.  PRINT ONLY IF CONTROL FLAG
C       IS NOT ZERO.
C
      IF (LKNTRL .NE. 0) THEN
         TEMP(1:21) = 'MESSAGE FROM ROUTINE '
         I = MIN(LEN(SUBROU), 16)
         TEMP(22:21+I) = SUBROU(1:I)
         TEMP(22+I:33+I) = ' IN LIBRARY '
         LTEMP = 33 + I
         I = MIN(LEN(LIBRAR), 16)
         TEMP(LTEMP+1:LTEMP+I) = LIBRAR (1:I)
         TEMP(LTEMP+I+1:LTEMP+I+1) = '.'
         LTEMP = LTEMP + I + 1
         CALL XERPRN (' ***', -1, TEMP(1:LTEMP), 72)
      ENDIF
C
C       IF LKNTRL IS POSITIVE, PRINT AN INTRODUCTORY LINE BEFORE
```

```
C       PRINTING THE MESSAGE.  THE INTRODUCTORY LINE TELLS THE CHOICE
C       FROM EACH OF THE FOLLOWING THREE OPTIONS.
C       1.  LEVEL OF THE MESSAGE
C              'INFORMATIVE MESSAGE'
C              'POTENTIALLY RECOVERABLE ERROR'
C              'FATAL ERROR'
C       2.  WHETHER CONTROL FLAG WILL ALLOW PROGRAM TO CONTINUE
C              'PROG CONTINUES'
C              'PROG ABORTED'
C       3.  WHETHER OR NOT A TRACEBACK WAS REQUESTED.  (THE TRACEBACK
C           MAY NOT BE IMPLEMENTED AT SOME SITES, SO THIS ONLY TELLS
C           WHAT WAS REQUESTED, NOT WHAT WAS DELIVERED.)
C              'TRACEBACK REQUESTED'
C              'TRACEBACK NOT REQUESTED'
C       NOTICE THAT THE LINE INCLUDING FOUR PREFIX CHARACTERS WILL NOT
C       EXCEED 74 CHARACTERS.
C       WE SKIP THE NEXT BLOCK IF THE INTRODUCTORY LINE IS NOT NEEDED.
C
      IF (LKNTRL .GT. 0) THEN
C
C       THE FIRST PART OF THE MESSAGE TELLS ABOUT THE LEVEL.
C
         IF (LEVEL .LE. 0) THEN
            TEMP(1:20) = 'INFORMATIVE MESSAGE,'
            LTEMP = 20
         ELSEIF (LEVEL .EQ. 1) THEN
            TEMP(1:30) = 'POTENTIALLY RECOVERABLE ERROR,'
            LTEMP = 30
         ELSE
            TEMP(1:12) = 'FATAL ERROR,'
            LTEMP = 12
         ENDIF
C
C       THEN WHETHER THE PROGRAM WILL CONTINUE.
C
         IF ((MKNTRL.EQ.2 .AND. LEVEL.GE.1) .OR.
     *        (MKNTRL.EQ.1 .AND. LEVEL.EQ.2)) THEN
            TEMP(LTEMP+1:LTEMP+14) = ' PROG ABORTED,'
            LTEMP = LTEMP + 14
         ELSE
            TEMP(LTEMP+1:LTEMP+16) = ' PROG CONTINUES,'
            LTEMP = LTEMP + 16
         ENDIF
C
C       FINALLY TELL WHETHER THERE SHOULD BE A TRACEBACK.
C
         IF (LKNTRL .GT. 0) THEN
            TEMP(LTEMP+1:LTEMP+20) = ' TRACEBACK REQUESTED'
            LTEMP = LTEMP + 20
         ELSE
            TEMP(LTEMP+1:LTEMP+24) = ' TRACEBACK NOT REQUESTED'
            LTEMP = LTEMP + 24
         ENDIF
         CALL XERPRN (' ***', -1, TEMP(1:LTEMP), 72)
      ENDIF
C
C       NOW SEND OUT THE MESSAGE.
C
```

```
      CALL XERPRN (' *  ', -1, MESSG, 72)
C
C       IF LKNTRL IS POSITIVE, WRITE THE ERROR NUMBER AND REQUEST A
C          TRACEBACK.
C
      IF (LKNTRL .GT. 0) THEN
         WRITE (TEMP, '(''ERROR NUMBER = '', I8)') NERR
         DO 10 I=16,22
            IF (TEMP(I:I) .NE. ' ') GO TO 20
   10    CONTINUE
C
   20    CALL XERPRN (' *  ', -1, TEMP(1:15) // TEMP(I:23), 72)
         CALL FDUMP
      ENDIF
C
C       IF LKNTRL IS NOT ZERO, PRINT A BLANK LINE AND AN END OF MESSAGE.
C
      IF (LKNTRL .NE. 0) THEN
         CALL XERPRN (' *  ', -1, ' ', 72)
         CALL XERPRN (' ***', -1, 'END OF MESSAGE', 72)
         CALL XERPRN ('    ',  0, ' ', 72)
      ENDIF
C
C       IF THE ERROR IS NOT FATAL OR THE ERROR IS RECOVERABLE AND THE
C       CONTROL FLAG IS SET FOR RECOVERY, THEN RETURN.
C
   30 IF (LEVEL.LE.0 .OR. (LEVEL.EQ.1 .AND. MKNTRL.LE.1)) RETURN
C
C       THE PROGRAM WILL BE STOPPED DUE TO AN UNRECOVERED ERROR OR A
C       FATAL ERROR.  PRINT THE REASON FOR THE ABORT AND THE ERROR
C       SUMMARY IF THE CONTROL FLAG AND THE MAXIMUM ERROR COUNT PERMIT.
C
      IF (LKNTRL.GT.0 .AND. KOUNT.LT.MAX(1,MAXMES)) THEN
         IF (LEVEL .EQ. 1) THEN
            CALL XERPRN
     *         (' ***', -1, 'JOB ABORT DUE TO UNRECOVERED ERROR.', 72)
         ELSE
            CALL XERPRN(' ***', -1, 'JOB ABORT DUE TO FATAL ERROR.', 72)
         ENDIF
         CALL XERSVE (' ', ' ', ' ', -1, 0, 0, KDUMMY)
         CALL XERHLT (' ')
      ELSE
         CALL XERHLT (MESSG)
      ENDIF
      RETURN
      END
*DECK XERPRN
      SUBROUTINE XERPRN (PREFIX, NPREF, MESSG, NWRAP)
C***BEGIN PROLOGUE  XERPRN
C***SUBSIDIARY
C***PURPOSE  Print error messages processed by XERMSG.
C***LIBRARY   SLATEC (XERROR)
C***CATEGORY  R3C
C***TYPE      ALL (XERPRN-A)
C***KEYWORDS  ERROR MESSAGES, PRINTING, XERROR
C***AUTHOR  Fong, Kirby, (NMFECC at LLNL)
C***DESCRIPTION
C
```

```
C This routine sends one or more lines to each of the (up to five)
C logical units to which error messages are to be sent.  This routine
C is called several times by XERMSG, sometimes with a single line to
C print and sometimes with a (potentially very long) message that may
C wrap around into multiple lines.
C
C PREFIX  Input argument of type CHARACTER.  This argument contains
C         characters to be put at the beginning of each line before
C         the body of the message.  No more than 16 characters of
C         PREFIX will be used.
C
C NPREF   Input argument of type INTEGER.  This argument is the number
C         of characters to use from PREFIX.  If it is negative, the
C         intrinsic function LEN is used to determine its length.  If
C         it is zero, PREFIX is not used.  If it exceeds 16 or if
C         LEN(PREFIX) exceeds 16, only the first 16 characters will be
C         used.  If NPREF is positive and the length of PREFIX is less
C         than NPREF, a copy of PREFIX extended with blanks to length
C         NPREF will be used.
C
C MESSG   Input argument of type CHARACTER.  This is the text of a
C         message to be printed.  If it is a long message, it will be
C         broken into pieces for printing on multiple lines.  Each line
C         will start with the appropriate prefix and be followed by a
C         piece of the message.  NWRAP is the number of characters per
C         piece; that is, after each NWRAP characters, we break and
C         start a new line.  In addition the characters '$$' embedded
C         in MESSG are a sentinel for a new line.  The counting of
C         characters up to NWRAP starts over for each new line.  The
C         value of NWRAP typically used by XERMSG is 72 since many
C         older error messages in the SLATEC Library are laid out to
C         rely on wrap-around every 72 characters.
C
C NWRAP   Input argument of type INTEGER.  This gives the maximum size
C         piece into which to break MESSG for printing on multiple
C         lines.  An embedded '$$' ends a line, and the count restarts
C         at the following character.  If a line break does not occur
C         on a blank (it would split a word) that word is moved to the
C         next line.  Values of NWRAP less than 16 will be treated as
C         16.  Values of NWRAP greater than 132 will be treated as 132.
C         The actual line length will be NPREF + NWRAP after NPREF has
C         been adjusted to fall between 0 and 16 and NWRAP has been
C         adjusted to fall between 16 and 132.
C
C***REFERENCES  R. E. Jones and D. K. Kahaner, XERROR, the SLATEC
C                 Error-handling Package, SAND82-0800, Sandia
C                 Laboratories, 1982.
C***ROUTINES CALLED  I1MACH, XGETUA
C***REVISION HISTORY  (YYMMDD)
C   880621  DATE WRITTEN
C   880708  REVISED AFTER THE SLATEC CML SUBCOMMITTEE MEETING OF
C           JUNE 29 AND 30 TO CHANGE THE NAME TO XERPRN AND TO REWORK
C           THE HANDLING OF THE NEW LINE SENTINEL TO BEHAVE LIKE THE
C           SLASH CHARACTER IN FORMAT STATEMENTS.
C   890706  REVISED WITH THE HELP OF FRED FRITSCH AND REG CLEMENS TO
C           STREAMLINE THE CODING AND FIX A BUG THAT CAUSED EXTRA BLANK
C           LINES TO BE PRINTED.
C   890721  REVISED TO ADD A NEW FEATURE.  A NEGATIVE VALUE OF NPREF
```

```
C           CAUSES LEN(PREFIX) TO BE USED AS THE LENGTH.
C   891013  REVISED TO CORRECT ERROR IN CALCULATING PREFIX LENGTH.
C   891214  Prologue converted to Version 4.0 format.  (WRB)
C   900510  Added code to break messages between words.  (RWC)
C   920501  Reformatted the REFERENCES section.  (WRB)
C***END PROLOGUE  XERPRN
      CHARACTER*(*) PREFIX, MESSG
      INTEGER NPREF, NWRAP
      CHARACTER*148 CBUFF
      INTEGER IU(5), NUNIT
      CHARACTER*2 NEWLIN
      PARAMETER (NEWLIN = '$$')
C***FIRST EXECUTABLE STATEMENT  XERPRN
      CALL XGETUA(IU,NUNIT)
C
C       A ZERO VALUE FOR A LOGICAL UNIT NUMBER MEANS TO USE THE STANDARD
C       ERROR MESSAGE UNIT INSTEAD.  I1MACH(4) RETRIEVES THE STANDARD
C       ERROR MESSAGE UNIT.
C
      N = I1MACH(4)
      DO 10 I=1,NUNIT
         IF (IU(I) .EQ. 0) IU(I) = N
   10 CONTINUE
C
C       LPREF IS THE LENGTH OF THE PREFIX.  THE PREFIX IS PLACED AT THE
C       BEGINNING OF CBUFF, THE CHARACTER BUFFER, AND KEPT THERE DURING
C       THE REST OF THIS ROUTINE.
C
      IF ( NPREF .LT. 0 ) THEN
         LPREF = LEN(PREFIX)
      ELSE
         LPREF = NPREF
      ENDIF
      LPREF = MIN(16, LPREF)
      IF (LPREF .NE. 0) CBUFF(1:LPREF) = PREFIX
C
C       LWRAP IS THE MAXIMUM NUMBER OF CHARACTERS WE WANT TO TAKE AT ONE
C       TIME FROM MESSG TO PRINT ON ONE LINE.
C
      LWRAP = MAX(16, MIN(132, NWRAP))
C
C       SET LENMSG TO THE LENGTH OF MESSG, IGNORE ANY TRAILING BLANKS.
C
      LENMSG = LEN(MESSG)
      N = LENMSG
      DO 20 I=1,N
         IF (MESSG(LENMSG:LENMSG) .NE. ' ') GO TO 30
         LENMSG = LENMSG - 1
   20 CONTINUE
   30 CONTINUE
C
C       IF THE MESSAGE IS ALL BLANKS, THEN PRINT ONE BLANK LINE.
C
      IF (LENMSG .EQ. 0) THEN
         CBUFF(LPREF+1:LPREF+1) = ' '
         DO 40 I=1,NUNIT
            WRITE(IU(I), '(A)') CBUFF(1:LPREF+1)
   40    CONTINUE
```

```
         RETURN
      ENDIF
C
C       SET NEXTC TO THE POSITION IN MESSG WHERE THE NEXT SUBSTRING
C       STARTS.  FROM THIS POSITION WE SCAN FOR THE NEW LINE SENTINEL.
C       WHEN NEXTC EXCEEDS LENMSG, THERE IS NO MORE TO PRINT.
C       WE LOOP BACK TO LABEL 50 UNTIL ALL PIECES HAVE BEEN PRINTED.
C
C       WE LOOK FOR THE NEXT OCCURRENCE OF THE NEW LINE SENTINEL.  THE
C       INDEX INTRINSIC FUNCTION RETURNS ZERO IF THERE IS NO OCCURRENCE
C       OR IF THE LENGTH OF THE FIRST ARGUMENT IS LESS THAN THE LENGTH
C       OF THE SECOND ARGUMENT.
C
C       THERE ARE SEVERAL CASES WHICH SHOULD BE CHECKED FOR IN THE
C       FOLLOWING ORDER.  WE ARE ATTEMPTING TO SET LPIECE TO THE NUMBER
C       OF CHARACTERS THAT SHOULD BE TAKEN FROM MESSG STARTING AT
C       POSITION NEXTC.
C
C       LPIECE .EQ. 0   THE NEW LINE SENTINEL DOES NOT OCCUR IN THE
C                       REMAINDER OF THE CHARACTER STRING.  LPIECE
C                       SHOULD BE SET TO LWRAP OR LENMSG+1-NEXTC,
C                       WHICHEVER IS LESS.
C
C       LPIECE .EQ. 1   THE NEW LINE SENTINEL STARTS AT MESSG(NEXTC:
C                       NEXTC).  LPIECE IS EFFECTIVELY ZERO, AND WE
C                       PRINT NOTHING TO AVOID PRODUCING UNNECESSARY
C                       BLANK LINES.  THIS TAKES CARE OF THE SITUATION
C                       WHERE THE LIBRARY ROUTINE HAS A MESSAGE OF
C                       EXACTLY 72 CHARACTERS FOLLOWED BY A NEW LINE
C                       SENTINEL FOLLOWED BY MORE CHARACTERS.  NEXTC
C                       SHOULD BE INCREMENTED BY 2.
C
C       LPIECE .GT. LWRAP+1  REDUCE LPIECE TO LWRAP.
C
C       ELSE            THIS LAST CASE MEANS 2 .LE. LPIECE .LE. LWRAP+1
C                       RESET LPIECE = LPIECE-1.  NOTE THAT THIS
C                       PROPERLY HANDLES THE END CASE WHERE LPIECE .EQ.
C                       LWRAP+1.  THAT IS, THE SENTINEL FALLS EXACTLY
C                       AT THE END OF A LINE.
C
      NEXTC = 1
   50 LPIECE = INDEX(MESSG(NEXTC:LENMSG), NEWLIN)
      IF (LPIECE .EQ. 0) THEN
C
C       THERE WAS NO NEW LINE SENTINEL FOUND.
C
         IDELTA = 0
         LPIECE = MIN(LWRAP, LENMSG+1-NEXTC)
         IF (LPIECE .LT. LENMSG+1-NEXTC) THEN
            DO 52 I=LPIECE+1,2,-1
               IF (MESSG(NEXTC+I-1:NEXTC+I-1) .EQ. ' ') THEN
                  LPIECE = I-1
                  IDELTA = 1
                  GOTO 54
               ENDIF
   52       CONTINUE
         ENDIF
   54    CBUFF(LPREF+1:LPREF+LPIECE) = MESSG(NEXTC:NEXTC+LPIECE-1)
```

```
         NEXTC = NEXTC + LPIECE + IDELTA
      ELSEIF (LPIECE .EQ. 1) THEN
C
C       WE HAVE A NEW LINE SENTINEL AT MESSG(NEXTC:NEXTC+1).
C       DON'T PRINT A BLANK LINE.
C
         NEXTC = NEXTC + 2
         GO TO 50
      ELSEIF (LPIECE .GT. LWRAP+1) THEN
C
C       LPIECE SHOULD BE SET DOWN TO LWRAP.
C
         IDELTA = 0
         LPIECE = LWRAP
         DO 56 I=LPIECE+1,2,-1
            IF (MESSG(NEXTC+I-1:NEXTC+I-1) .EQ. ' ') THEN
               LPIECE = I-1
               IDELTA = 1
               GOTO 58
            ENDIF
   56    CONTINUE
   58    CBUFF(LPREF+1:LPREF+LPIECE) = MESSG(NEXTC:NEXTC+LPIECE-1)
         NEXTC = NEXTC + LPIECE + IDELTA
      ELSE
C
C       IF WE ARRIVE HERE, IT MEANS 2 .LE. LPIECE .LE. LWRAP+1.
C       WE SHOULD DECREMENT LPIECE BY ONE.
C
         LPIECE = LPIECE - 1
         CBUFF(LPREF+1:LPREF+LPIECE) = MESSG(NEXTC:NEXTC+LPIECE-1)
         NEXTC  = NEXTC + LPIECE + 2
      ENDIF
C
C       PRINT
C
      DO 60 I=1,NUNIT
         WRITE(IU(I), '(A)') CBUFF(1:LPREF+LPIECE)
   60 CONTINUE
C
      IF (NEXTC .LE. LENMSG) GO TO 50
      RETURN
      END
*DECK XERSVE
      SUBROUTINE XERSVE (LIBRAR, SUBROU, MESSG, KFLAG, NERR, LEVEL,
     +   ICOUNT)
C***BEGIN PROLOGUE  XERSVE
C***SUBSIDIARY
C***PURPOSE  Record that an error has occurred.
C***LIBRARY   SLATEC (XERROR)
C***CATEGORY  R3
C***TYPE      ALL (XERSVE-A)
C***KEYWORDS  ERROR, XERROR
C***AUTHOR  Jones, R. E., (SNLA)
C***DESCRIPTION
C
C *Usage:
C
C        INTEGER  KFLAG, NERR, LEVEL, ICOUNT
```

```
C        CHARACTER * (len) LIBRAR, SUBROU, MESSG
C
C        CALL XERSVE (LIBRAR, SUBROU, MESSG, KFLAG, NERR, LEVEL, ICOUNT)
C
C *Arguments:
C
C        LIBRAR :IN    is the library that the message is from.
C        SUBROU :IN    is the subroutine that the message is from.
C        MESSG  :IN    is the message to be saved.
C        KFLAG  :IN    indicates the action to be performed.
C                      when KFLAG > 0, the message in MESSG is saved.
C                      when KFLAG=0 the tables will be dumped and
C                      cleared.
C                      when KFLAG < 0, the tables will be dumped and
C                      not cleared.
C        NERR   :IN    is the error number.
C        LEVEL  :IN    is the error severity.
C        ICOUNT :OUT   the number of times this message has been seen,
C                      or zero if the table has overflowed and does not
C                      contain this message specifically.  When KFLAG=0,
C                      ICOUNT will not be altered.
C
C *Description:
C
C   Record that this error occurred and possibly dump and clear the
C   tables.
C
C***REFERENCES  R. E. Jones and D. K. Kahaner, XERROR, the SLATEC
C                Error-handling Package, SAND82-0800, Sandia
C                Laboratories, 1982.
C***ROUTINES CALLED  I1MACH, XGETUA
C***REVISION HISTORY  (YYMMDD)
C   800319  DATE WRITTEN
C   861211  REVISION DATE from Version 3.2
C   891214  Prologue converted to Version 4.0 format.  (BAB)
C   900413  Routine modified to remove reference to KFLAG.  (WRB)
C   900510  Changed to add LIBRARY NAME and SUBROUTINE to calling
C           sequence, use IF-THEN-ELSE, make number of saved entries
C           easily changeable, changed routine name from XERSAV to
C           XERSVE.  (RWC)
C   910626  Added LIBTAB and SUBTAB to SAVE statement.  (BKS)
C   920501  Reformatted the REFERENCES section.  (WRB)
C***END PROLOGUE  XERSVE
      PARAMETER (LENTAB=10)
      INTEGER LUN(5)
      CHARACTER*(*) LIBRAR, SUBROU, MESSG
      CHARACTER*8  LIBTAB(LENTAB), SUBTAB(LENTAB), LIB, SUB
      CHARACTER*20 MESTAB(LENTAB), MES
      DIMENSION NERTAB(LENTAB), LEVTAB(LENTAB), KOUNT(LENTAB)
      SAVE LIBTAB, SUBTAB, MESTAB, NERTAB, LEVTAB, KOUNT, KOUNTX, NMSG
      DATA KOUNTX/0/, NMSG/0/
C***FIRST EXECUTABLE STATEMENT  XERSVE
C
      IF (KFLAG.LE.0) THEN
C
C        Dump the table.
C
         IF (NMSG.EQ.0) RETURN
```

```
C
C        Print to each unit.
C
         CALL XGETUA (LUN, NUNIT)
         DO 20 KUNIT = 1,NUNIT
            IUNIT = LUN(KUNIT)
            IF (IUNIT.EQ.0) IUNIT = I1MACH(4)
C
C           Print the table header.
C
            WRITE (IUNIT,9000)
C
C           Print body of table.
C
            DO 10 I = 1,NMSG
               WRITE (IUNIT,9010) LIBTAB(I), SUBTAB(I), MESTAB(I),
     *            NERTAB(I),LEVTAB(I),KOUNT(I)
   10       CONTINUE
C
C           Print number of other errors.
C
            IF (KOUNTX.NE.0) WRITE (IUNIT,9020) KOUNTX
            WRITE (IUNIT,9030)
   20    CONTINUE
C
C        Clear the error tables.
C
         IF (KFLAG.EQ.0) THEN
            NMSG = 0
            KOUNTX = 0
         ENDIF
      ELSE
C
C        PROCESS A MESSAGE...
C        SEARCH FOR THIS MESSG, OR ELSE AN EMPTY SLOT FOR THIS MESSG,
C        OR ELSE DETERMINE THAT THE ERROR TABLE IS FULL.
C
         LIB = LIBRAR
         SUB = SUBROU
         MES = MESSG
         DO 30 I = 1,NMSG
            IF (LIB.EQ.LIBTAB(I) .AND. SUB.EQ.SUBTAB(I) .AND.
     *         MES.EQ.MESTAB(I) .AND. NERR.EQ.NERTAB(I) .AND.
     *         LEVEL.EQ.LEVTAB(I)) THEN
               KOUNT(I) = KOUNT(I) + 1
               ICOUNT = KOUNT(I)
               RETURN
            ENDIF
   30    CONTINUE
C
         IF (NMSG.LT.LENTAB) THEN
C
C           Empty slot found for new message.
C
            NMSG = NMSG + 1
            LIBTAB(I) = LIB
            SUBTAB(I) = SUB
            MESTAB(I) = MES
```

```
            NERTAB(I) = NERR
            LEVTAB(I) = LEVEL
            KOUNT (I) = 1
            ICOUNT    = 1
         ELSE
C
C           Table is full.
C
            KOUNTX = KOUNTX+1
            ICOUNT = 0
         ENDIF
      ENDIF
      RETURN
C
C     Formats.
C
 9000 FORMAT ('0          ERROR MESSAGE SUMMARY' /
     +    ' LIBRARY    SUBROUTINE MESSAGE START             NERR',
     +    '     LEVEL     COUNT')
 9010 FORMAT (1X,A,3X,A,3X,A,3I10)
 9020 FORMAT ('0OTHER ERRORS NOT INDIVIDUALLY TABULATED = ', I10)
 9030 FORMAT (1X)
      END
*DECK XGETUA
      SUBROUTINE XGETUA (IUNITA, N)
C***BEGIN PROLOGUE  XGETUA
C***PURPOSE  Return unit number(s) to which error messages are being
C            sent.
C***LIBRARY   SLATEC (XERROR)
C***CATEGORY  R3C
C***TYPE      ALL (XGETUA-A)
C***KEYWORDS  ERROR, XERROR
C***AUTHOR  Jones, R. E., (SNLA)
C***DESCRIPTION
C
C     Abstract
C        XGETUA may be called to determine the unit number or numbers
C        to which error messages are being sent.
C        These unit numbers may have been set by a call to XSETUN,
C        or a call to XSETUA, or may be a default value.
C
C     Description of Parameters
C      --Output--
C        IUNIT - an array of one to five unit numbers, depending
C                on the value of N.  A value of zero refers to the
C                default unit, as defined by the I1MACH machine
C                constant routine.  Only IUNIT(1),...,IUNIT(N) are
C                defined by XGETUA.  The values of IUNIT(N+1),...,
C                IUNIT(5) are not defined (for N .LT. 5) or altered
C                in any way by XGETUA.
C        N     - the number of units to which copies of the
C                error messages are being sent.  N will be in the
C                range from 1 to 5.
C
C***REFERENCES  R. E. Jones and D. K. Kahaner, XERROR, the SLATEC
C                 Error-handling Package, SAND82-0800, Sandia
C                 Laboratories, 1982.
C***ROUTINES CALLED  J4SAVE
```

```
C***REVISION HISTORY  (YYMMDD)
C   790801  DATE WRITTEN
C   861211  REVISION DATE from Version 3.2
C   891214  Prologue converted to Version 4.0 format.  (BAB)
C   920501  Reformatted the REFERENCES section.  (WRB)
C***END PROLOGUE  XGETUA
      DIMENSION IUNITA(5)
C***FIRST EXECUTABLE STATEMENT  XGETUA
      N = J4SAVE(5,0,.FALSE.)
      DO 30 I=1,N
         INDEX = I+4
         IF (I.EQ.1) INDEX = 3
         IUNITA(I) = J4SAVE(INDEX,0,.FALSE.)
   30 CONTINUE
      RETURN
      END
*DECK ZABS
      DOUBLE PRECISION FUNCTION ZABS (ZR, ZI)
C***BEGIN PROLOGUE  ZABS
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZBESH, ZBESI, ZBESJ, ZBESK, ZBESY, ZAIRY and
C            ZBIRY
C***LIBRARY   SLATEC
C***TYPE      ALL (ZABS-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     ZABS COMPUTES THE ABSOLUTE VALUE OR MAGNITUDE OF A DOUBLE
C     PRECISION COMPLEX VARIABLE CMPLX(ZR,ZI)
C
C***SEE ALSO  ZAIRY, ZBESH, ZBESI, ZBESJ, ZBESK, ZBESY, ZBIRY
C***ROUTINES CALLED  (NONE)
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C***END PROLOGUE  ZABS
      DOUBLE PRECISION ZR, ZI, U, V, Q, S
C***FIRST EXECUTABLE STATEMENT  ZABS
      U = ABS(ZR)
      V = ABS(ZI)
      S = U + V
C------------------------------------------------------------------------
C     S*1.0D0 MAKES AN UNNORMALIZED UNDERFLOW ON CDC MACHINES INTO A
C     TRUE FLOATING ZERO
C------------------------------------------------------------------------
      S = S*1.0D+0
      IF (S.EQ.0.0D+0) GO TO 20
      IF (U.GT.V) GO TO 10
      Q = U/V
      ZABS = V*SQRT(1.D+0+Q*Q)
      RETURN
   10 Q = V/U
      ZABS = U*SQRT(1.D+0+Q*Q)
      RETURN
   20 ZABS = 0.0D+0
      RETURN
      END
*DECK ZBINU
```

```
      SUBROUTINE ZBINU (ZR, ZI, FNU, KODE, N, CYR, CYI, NZ, RL, FNUL,
     +   TOL, ELIM, ALIM)
C***BEGIN PROLOGUE  ZBINU
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZAIRY, ZBESH, ZBESI, ZBESJ, ZBESK and ZBIRY
C***LIBRARY   SLATEC
C***TYPE      ALL (CBINU-A, ZBINU-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     ZBINU COMPUTES THE I FUNCTION IN THE RIGHT HALF Z PLANE
C
C***SEE ALSO  ZAIRY, ZBESH, ZBESI, ZBESJ, ZBESK, ZBIRY
C***ROUTINES CALLED  ZABS, ZASYI, ZBUNI, ZMLRI, ZSERI, ZUOIK, ZWRSK
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C***END PROLOGUE  ZBINU
      DOUBLE PRECISION ALIM, AZ, CWI, CWR, CYI, CYR, DFNU, ELIM, FNU,
     * FNUL, RL, TOL, ZEROI, ZEROR, ZI, ZR, ZABS
      INTEGER I, INW, KODE, N, NLAST, NN, NUI, NW, NZ
      DIMENSION CYR(N), CYI(N), CWR(2), CWI(2)
      EXTERNAL ZABS
      DATA ZEROR,ZEROI / 0.0D0, 0.0D0 /
C***FIRST EXECUTABLE STATEMENT  ZBINU
      NZ = 0
      AZ = ZABS(ZR,ZI)
      NN = N
      DFNU = FNU + (N-1)
      IF (AZ.LE.2.0D0) GO TO 10
      IF (AZ*AZ*0.25D0.GT.DFNU+1.0D0) GO TO 20
   10 CONTINUE
C-----------------------------------------------------------------------
C     POWER SERIES
C-----------------------------------------------------------------------
      CALL ZSERI(ZR, ZI, FNU, KODE, NN, CYR, CYI, NW, TOL, ELIM, ALIM)
      INW = ABS(NW)
      NZ = NZ + INW
      NN = NN - INW
      IF (NN.EQ.0) RETURN
      IF (NW.GE.0) GO TO 120
      DFNU = FNU + (NN-1)
   20 CONTINUE
      IF (AZ.LT.RL) GO TO 40
      IF (DFNU.LE.1.0D0) GO TO 30
      IF (AZ+AZ.LT.DFNU*DFNU) GO TO 50
C-----------------------------------------------------------------------
C     ASYMPTOTIC EXPANSION FOR LARGE Z
C-----------------------------------------------------------------------
   30 CONTINUE
      CALL ZASYI(ZR, ZI, FNU, KODE, NN, CYR, CYI, NW, RL, TOL, ELIM,
     * ALIM)
      IF (NW.LT.0) GO TO 130
      GO TO 120
   40 CONTINUE
      IF (DFNU.LE.1.0D0) GO TO 70
   50 CONTINUE
C-----------------------------------------------------------------------
```

```
C     OVERFLOW AND UNDERFLOW TEST ON I SEQUENCE FOR MILLER ALGORITHM
C-----------------------------------------------------------------------
      CALL ZUOIK(ZR, ZI, FNU, KODE, 1, NN, CYR, CYI, NW, TOL, ELIM,
     * ALIM)
      IF (NW.LT.0) GO TO 130
      NZ = NZ + NW
      NN = NN - NW
      IF (NN.EQ.0) RETURN
      DFNU = FNU+(NN-1)
      IF (DFNU.GT.FNUL) GO TO 110
      IF (AZ.GT.FNUL) GO TO 110
   60 CONTINUE
      IF (AZ.GT.RL) GO TO 80
   70 CONTINUE
C-----------------------------------------------------------------------
C     MILLER ALGORITHM NORMALIZED BY THE SERIES
C-----------------------------------------------------------------------
      CALL ZMLRI(ZR, ZI, FNU, KODE, NN, CYR, CYI, NW, TOL)
      IF(NW.LT.0) GO TO 130
      GO TO 120
   80 CONTINUE
C-----------------------------------------------------------------------
C     MILLER ALGORITHM NORMALIZED BY THE WRONSKIAN
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     OVERFLOW TEST ON K FUNCTIONS USED IN WRONSKIAN
C-----------------------------------------------------------------------
      CALL ZUOIK(ZR, ZI, FNU, KODE, 2, 2, CWR, CWI, NW, TOL, ELIM,
     * ALIM)
      IF (NW.GE.0) GO TO 100
      NZ = NN
      DO 90 I=1,NN
        CYR(I) = ZEROR
        CYI(I) = ZEROI
   90 CONTINUE
      RETURN
  100 CONTINUE
      IF (NW.GT.0) GO TO 130
      CALL ZWRSK(ZR, ZI, FNU, KODE, NN, CYR, CYI, NW, CWR, CWI, TOL,
     * ELIM, ALIM)
      IF (NW.LT.0) GO TO 130
      GO TO 120
  110 CONTINUE
C-----------------------------------------------------------------------
C     INCREMENT FNU+NN-1 UP TO FNUL, COMPUTE AND RECUR BACKWARD
C-----------------------------------------------------------------------
      NUI = FNUL-DFNU + 1
      NUI = MAX(NUI,0)
      CALL ZBUNI(ZR, ZI, FNU, KODE, NN, CYR, CYI, NW, NUI, NLAST, FNUL,
     * TOL, ELIM, ALIM)
      IF (NW.LT.0) GO TO 130
      NZ = NZ + NW
      IF (NLAST.EQ.0) GO TO 120
      NN = NLAST
      GO TO 60
  120 CONTINUE
      RETURN
  130 CONTINUE
```

```
      NZ = -1
      IF(NW.EQ.(-2)) NZ=-2
      RETURN
      END
*DECK ZBUNI
      SUBROUTINE ZBUNI (ZR, ZI, FNU, KODE, N, YR, YI, NZ, NUI, NLAST,
     +   FNUL, TOL, ELIM, ALIM)
C***BEGIN PROLOGUE  ZBUNI
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZBESI and ZBESK
C***LIBRARY   SLATEC
C***TYPE      ALL (CBUNI-A, ZBUNI-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     ZBUNI COMPUTES THE I BESSEL FUNCTION FOR LARGE ABS(Z).GT.
C     FNUL AND FNU+N-1.LT.FNUL. THE ORDER IS INCREASED FROM
C     FNU+N-1 GREATER THAN FNUL BY ADDING NUI AND COMPUTING
C     ACCORDING TO THE UNIFORM ASYMPTOTIC EXPANSION FOR I(FNU,Z)
C     ON IFORM=1 AND THE EXPANSION FOR J(FNU,Z) ON IFORM=2
C
C***SEE ALSO  ZBESI, ZBESK
C***ROUTINES CALLED  D1MACH, ZABS, ZUNI1, ZUNI2
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C***END PROLOGUE  ZBUNI
C     COMPLEX CSCL,CSCR,CY,RZ,ST,S1,S2,Y,Z
      DOUBLE PRECISION ALIM, AX, AY, CSCLR, CSCRR, CYI, CYR, DFNU,
     * ELIM, FNU, FNUI, FNUL, GNU, RAZ, RZI, RZR, STI, STR, S1I, S1R,
     * S2I, S2R, TOL, YI, YR, ZI, ZR, ZABS, ASCLE, BRY, C1R, C1I, C1M,
     * D1MACH
      INTEGER I, IFLAG, IFORM, K, KODE, N, NL, NLAST, NUI, NW, NZ
      DIMENSION YR(N), YI(N), CYR(2), CYI(2), BRY(3)
      EXTERNAL ZABS
C***FIRST EXECUTABLE STATEMENT  ZBUNI
      NZ = 0
      AX = ABS(ZR)*1.7321D0
      AY = ABS(ZI)
      IFORM = 1
      IF (AY.GT.AX) IFORM = 2
      IF (NUI.EQ.0) GO TO 60
      FNUI = NUI
      DFNU = FNU + (N-1)
      GNU = DFNU + FNUI
      IF (IFORM.EQ.2) GO TO 10
C-----------------------------------------------------------------------
C     ASYMPTOTIC EXPANSION FOR I(FNU,Z) FOR LARGE FNU APPLIED IN
C     -PI/3.LE.ARG(Z).LE.PI/3
C-----------------------------------------------------------------------
      CALL ZUNI1(ZR, ZI, GNU, KODE, 2, CYR, CYI, NW, NLAST, FNUL, TOL,
     * ELIM, ALIM)
      GO TO 20
   10 CONTINUE
C-----------------------------------------------------------------------
C     ASYMPTOTIC EXPANSION FOR J(FNU,Z*EXP(M*HPI)) FOR LARGE FNU
C     APPLIED IN PI/3.LT.ABS(ARG(Z)).LE.PI/2 WHERE M=+I OR -I
C     AND HPI=PI/2
```

```
C-----------------------------------------------------------------------
      CALL ZUNI2(ZR, ZI, GNU, KODE, 2, CYR, CYI, NW, NLAST, FNUL, TOL,
     * ELIM, ALIM)
   20 CONTINUE
      IF (NW.LT.0) GO TO 50
      IF (NW.NE.0) GO TO 90
      STR = ZABS(CYR(1),CYI(1))
C-----------------------------------------------------------------------
C     SCALE BACKWARD RECURRENCE, BRY(3) IS DEFINED BUT NEVER USED
C-----------------------------------------------------------------------
      BRY(1)=1.0D+3*D1MACH(1)/TOL
      BRY(2) = 1.0D0/BRY(1)
      BRY(3) = BRY(2)
      IFLAG = 2
      ASCLE = BRY(2)
      CSCLR = 1.0D0
      IF (STR.GT.BRY(1)) GO TO 21
      IFLAG = 1
      ASCLE = BRY(1)
      CSCLR = 1.0D0/TOL
      GO TO 25
   21 CONTINUE
      IF (STR.LT.BRY(2)) GO TO 25
      IFLAG = 3
      ASCLE=BRY(3)
      CSCLR = TOL
   25 CONTINUE
      CSCRR = 1.0D0/CSCLR
      S1R = CYR(2)*CSCLR
      S1I = CYI(2)*CSCLR
      S2R = CYR(1)*CSCLR
      S2I = CYI(1)*CSCLR
      RAZ = 1.0D0/ZABS(ZR,ZI)
      STR = ZR*RAZ
      STI = -ZI*RAZ
      RZR = (STR+STR)*RAZ
      RZI = (STI+STI)*RAZ
      DO 30 I=1,NUI
        STR = S2R
        STI = S2I
        S2R = (DFNU+FNUI)*(RZR*STR-RZI*STI) + S1R
        S2I = (DFNU+FNUI)*(RZR*STI+RZI*STR) + S1I
        S1R = STR
        S1I = STI
        FNUI = FNUI - 1.0D0
        IF (IFLAG.GE.3) GO TO 30
        STR = S2R*CSCRR
        STI = S2I*CSCRR
        C1R = ABS(STR)
        C1I = ABS(STI)
        C1M = MAX(C1R,C1I)
        IF (C1M.LE.ASCLE) GO TO 30
        IFLAG = IFLAG+1
        ASCLE = BRY(IFLAG)
        S1R = S1R*CSCRR
        S1I = S1I*CSCRR
        S2R = STR
        S2I = STI
```

```
        CSCLR = CSCLR*TOL
        CSCRR = 1.0D0/CSCLR
        S1R = S1R*CSCLR
        S1I = S1I*CSCLR
        S2R = S2R*CSCLR
        S2I = S2I*CSCLR
   30 CONTINUE
      YR(N) = S2R*CSCRR
      YI(N) = S2I*CSCRR
      IF (N.EQ.1) RETURN
      NL = N - 1
      FNUI = NL
      K = NL
      DO 40 I=1,NL
        STR = S2R
        STI = S2I
        S2R = (FNU+FNUI)*(RZR*STR-RZI*STI) + S1R
        S2I = (FNU+FNUI)*(RZR*STI+RZI*STR) + S1I
        S1R = STR
        S1I = STI
        STR = S2R*CSCRR
        STI = S2I*CSCRR
        YR(K) = STR
        YI(K) = STI
        FNUI = FNUI - 1.0D0
        K = K - 1
        IF (IFLAG.GE.3) GO TO 40
        C1R = ABS(STR)
        C1I = ABS(STI)
        C1M = MAX(C1R,C1I)
        IF (C1M.LE.ASCLE) GO TO 40
        IFLAG = IFLAG+1
        ASCLE = BRY(IFLAG)
        S1R = S1R*CSCRR
        S1I = S1I*CSCRR
        S2R = STR
        S2I = STI
        CSCLR = CSCLR*TOL
        CSCRR = 1.0D0/CSCLR
        S1R = S1R*CSCLR
        S1I = S1I*CSCLR
        S2R = S2R*CSCLR
        S2I = S2I*CSCLR
   40 CONTINUE
      RETURN
   50 CONTINUE
      NZ = -1
      IF(NW.EQ.(-2)) NZ=-2
      RETURN
   60 CONTINUE
      IF (IFORM.EQ.2) GO TO 70
C-----------------------------------------------------------------------
C     ASYMPTOTIC EXPANSION FOR I(FNU,Z) FOR LARGE FNU APPLIED IN
C     -PI/3.LE.ARG(Z).LE.PI/3
C-----------------------------------------------------------------------
      CALL ZUNI1(ZR, ZI, FNU, KODE, N, YR, YI, NW, NLAST, FNUL, TOL,
     * ELIM, ALIM)
      GO TO 80
```

```
   70 CONTINUE
C-----------------------------------------------------------------------
C     ASYMPTOTIC EXPANSION FOR J(FNU,Z*EXP(M*HPI)) FOR LARGE FNU
C     APPLIED IN PI/3.LT.ABS(ARG(Z)).LE.PI/2 WHERE M=+I OR -I
C     AND HPI=PI/2
C-----------------------------------------------------------------------
      CALL ZUNI2(ZR, ZI, FNU, KODE, N, YR, YI, NW, NLAST, FNUL, TOL,
     * ELIM, ALIM)
   80 CONTINUE
      IF (NW.LT.0) GO TO 50
      NZ = NW
      RETURN
   90 CONTINUE
      NLAST = N
      RETURN
      END
*DECK ZMLRI
      SUBROUTINE ZMLRI (ZR, ZI, FNU, KODE, N, YR, YI, NZ, TOL)
C***BEGIN PROLOGUE  ZMLRI
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZBESI and ZBESK
C***LIBRARY   SLATEC
C***TYPE      ALL (CMLRI-A, ZMLRI-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     ZMLRI COMPUTES THE I BESSEL FUNCTION FOR RE(Z).GE.0.0 BY THE
C     MILLER ALGORITHM NORMALIZED BY A NEUMANN SERIES.
C
C***SEE ALSO  ZBESI, ZBESK
C***ROUTINES CALLED  D1MACH, DGAMLN, ZABS, ZEXP, ZLOG, ZMLT
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C   930122  Added ZEXP and ZLOG to EXTERNAL statement.  (RWC)
C***END PROLOGUE  ZMLRI
C     COMPLEX CK,CNORM,CONE,CTWO,CZERO,PT,P1,P2,RZ,SUM,Y,Z
      DOUBLE PRECISION ACK, AK, AP, AT, AZ, BK, CKI, CKR, CNORMI,
     * CNORMR, CONEI, CONER, FKAP, FKK, FLAM, FNF, FNU, PTI, PTR, P1I,
     * P1R, P2I, P2R, RAZ, RHO, RHO2, RZI, RZR, SCLE, STI, STR, SUMI,
     * SUMR, TFNF, TOL, TST, YI, YR, ZEROI, ZEROR, ZI, ZR, DGAMLN,
     * D1MACH, ZABS
      INTEGER I, IAZ, IDUM, IFNU, INU, ITIME, K, KK, KM, KODE, M, N, NZ
      DIMENSION YR(N), YI(N)
      EXTERNAL ZABS, ZEXP, ZLOG
      DATA ZEROR,ZEROI,CONER,CONEI / 0.0D0, 0.0D0, 1.0D0, 0.0D0 /
C***FIRST EXECUTABLE STATEMENT  ZMLRI
      SCLE = D1MACH(1)/TOL
      NZ=0
      AZ = ZABS(ZR,ZI)
      IAZ = AZ
      IFNU = FNU
      INU = IFNU + N - 1
      AT = IAZ + 1.0D0
      RAZ = 1.0D0/AZ
      STR = ZR*RAZ
      STI = -ZI*RAZ
      CKR = STR*AT*RAZ
```

```
      CKI = STI*AT*RAZ
      RZR = (STR+STR)*RAZ
      RZI = (STI+STI)*RAZ
      P1R = ZEROR
      P1I = ZEROI
      P2R = CONER
      P2I = CONEI
      ACK = (AT+1.0D0)*RAZ
      RHO = ACK + SQRT(ACK*ACK-1.0D0)
      RHO2 = RHO*RHO
      TST = (RHO2+RHO2)/((RHO2-1.0D0)*(RHO-1.0D0))
      TST = TST/TOL
C-----------------------------------------------------------------------
C     COMPUTE RELATIVE TRUNCATION ERROR INDEX FOR SERIES
C-----------------------------------------------------------------------
      AK = AT
      DO 10 I=1,80
        PTR = P2R
        PTI = P2I
        P2R = P1R - (CKR*PTR-CKI*PTI)
        P2I = P1I - (CKI*PTR+CKR*PTI)
        P1R = PTR
        P1I = PTI
        CKR = CKR + RZR
        CKI = CKI + RZI
        AP = ZABS(P2R,P2I)
        IF (AP.GT.TST*AK*AK) GO TO 20
        AK = AK + 1.0D0
   10 CONTINUE
      GO TO 110
   20 CONTINUE
      I = I + 1
      K = 0
      IF (INU.LT.IAZ) GO TO 40
C-----------------------------------------------------------------------
C     COMPUTE RELATIVE TRUNCATION ERROR FOR RATIOS
C-----------------------------------------------------------------------
      P1R = ZEROR
      P1I = ZEROI
      P2R = CONER
      P2I = CONEI
      AT = INU + 1.0D0
      STR = ZR*RAZ
      STI = -ZI*RAZ
      CKR = STR*AT*RAZ
      CKI = STI*AT*RAZ
      ACK = AT*RAZ
      TST = SQRT(ACK/TOL)
      ITIME = 1
      DO 30 K=1,80
        PTR = P2R
        PTI = P2I
        P2R = P1R - (CKR*PTR-CKI*PTI)
        P2I = P1I - (CKR*PTI+CKI*PTR)
        P1R = PTR
        P1I = PTI
        CKR = CKR + RZR
        CKI = CKI + RZI
```

```
        AP = ZABS(P2R,P2I)
        IF (AP.LT.TST) GO TO 30
        IF (ITIME.EQ.2) GO TO 40
        ACK = ZABS(CKR,CKI)
        FLAM = ACK + SQRT(ACK*ACK-1.0D0)
        FKAP = AP/ZABS(P1R,P1I)
        RHO = MIN(FLAM,FKAP)
        TST = TST*SQRT(RHO/(RHO*RHO-1.0D0))
        ITIME = 2
   30 CONTINUE
      GO TO 110
   40 CONTINUE
C-----------------------------------------------------------------------
C     BACKWARD RECURRENCE AND SUM NORMALIZING RELATION
C-----------------------------------------------------------------------
      K = K + 1
      KK = MAX(I+IAZ,K+INU)
      FKK = KK
      P1R = ZEROR
      P1I = ZEROI
C-----------------------------------------------------------------------
C     SCALE P2 AND SUM BY SCLE
C-----------------------------------------------------------------------
      P2R = SCLE
      P2I = ZEROI
      FNF = FNU - IFNU
      TFNF = FNF + FNF
      BK = DGAMLN(FKK+TFNF+1.0D0,IDUM) - DGAMLN(FKK+1.0D0,IDUM) -
     * DGAMLN(TFNF+1.0D0,IDUM)
      BK = EXP(BK)
      SUMR = ZEROR
      SUMI = ZEROI
      KM = KK - INU
      DO 50 I=1,KM
        PTR = P2R
        PTI = P2I
        P2R = P1R + (FKK+FNF)*(RZR*PTR-RZI*PTI)
        P2I = P1I + (FKK+FNF)*(RZI*PTR+RZR*PTI)
        P1R = PTR
        P1I = PTI
        AK = 1.0D0 - TFNF/(FKK+TFNF)
        ACK = BK*AK
        SUMR = SUMR + (ACK+BK)*P1R
        SUMI = SUMI + (ACK+BK)*P1I
        BK = ACK
        FKK = FKK - 1.0D0
   50 CONTINUE
      YR(N) = P2R
      YI(N) = P2I
      IF (N.EQ.1) GO TO 70
      DO 60 I=2,N
        PTR = P2R
        PTI = P2I
        P2R = P1R + (FKK+FNF)*(RZR*PTR-RZI*PTI)
        P2I = P1I + (FKK+FNF)*(RZI*PTR+RZR*PTI)
        P1R = PTR
        P1I = PTI
        AK = 1.0D0 - TFNF/(FKK+TFNF)
```

```
        ACK = BK*AK
        SUMR = SUMR + (ACK+BK)*P1R
        SUMI = SUMI + (ACK+BK)*P1I
        BK = ACK
        FKK = FKK - 1.0D0
        M = N - I + 1
        YR(M) = P2R
        YI(M) = P2I
   60 CONTINUE
   70 CONTINUE
      IF (IFNU.LE.0) GO TO 90
      DO 80 I=1,IFNU
        PTR = P2R
        PTI = P2I
        P2R = P1R + (FKK+FNF)*(RZR*PTR-RZI*PTI)
        P2I = P1I + (FKK+FNF)*(RZR*PTI+RZI*PTR)
        P1R = PTR
        P1I = PTI
        AK = 1.0D0 - TFNF/(FKK+TFNF)
        ACK = BK*AK
        SUMR = SUMR + (ACK+BK)*P1R
        SUMI = SUMI + (ACK+BK)*P1I
        BK = ACK
        FKK = FKK - 1.0D0
   80 CONTINUE
   90 CONTINUE
      PTR = ZR
      PTI = ZI
      IF (KODE.EQ.2) PTR = ZEROR
      CALL ZLOG(RZR, RZI, STR, STI, IDUM)
      P1R = -FNF*STR + PTR
      P1I = -FNF*STI + PTI
      AP = DGAMLN(1.0D0+FNF,IDUM)
      PTR = P1R - AP
      PTI = P1I
C-----------------------------------------------------------------------
C     THE DIVISION CEXP(PT)/(SUM+P2) IS ALTERED TO AVOID OVERFLOW
C     IN THE DENOMINATOR BY SQUARING LARGE QUANTITIES
C-----------------------------------------------------------------------
      P2R = P2R + SUMR
      P2I = P2I + SUMI
      AP = ZABS(P2R,P2I)
      P1R = 1.0D0/AP
      CALL ZEXP(PTR, PTI, STR, STI)
      CKR = STR*P1R
      CKI = STI*P1R
      PTR = P2R*P1R
      PTI = -P2I*P1R
      CALL ZMLT(CKR, CKI, PTR, PTI, CNORMR, CNORMI)
      DO 100 I=1,N
        STR = YR(I)*CNORMR - YI(I)*CNORMI
        YI(I) = YR(I)*CNORMI + YI(I)*CNORMR
        YR(I) = STR
  100 CONTINUE
      RETURN
  110 CONTINUE
      NZ=-2
      RETURN
```

```
      END
*DECK ZMLT
      SUBROUTINE ZMLT (AR, AI, BR, BI, CR, CI)
C***BEGIN PROLOGUE  ZMLT
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZBESH, ZBESI, ZBESJ, ZBESK, ZBESY, ZAIRY and
C            ZBIRY
C***LIBRARY   SLATEC
C***TYPE      ALL (ZMLT-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     DOUBLE PRECISION COMPLEX MULTIPLY, C=A*B.
C
C***SEE ALSO  ZAIRY, ZBESH, ZBESI, ZBESJ, ZBESK, ZBESY, ZBIRY
C***ROUTINES CALLED  (NONE)
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C***END PROLOGUE  ZMLT
      DOUBLE PRECISION AR, AI, BR, BI, CR, CI, CA, CB
C***FIRST EXECUTABLE STATEMENT  ZMLT
      CA = AR*BR - AI*BI
      CB = AR*BI + AI*BR
      CR = CA
      CI = CB
      RETURN
      END
*DECK ZSERI
      SUBROUTINE ZSERI (ZR, ZI, FNU, KODE, N, YR, YI, NZ, TOL, ELIM,
     +   ALIM)
C***BEGIN PROLOGUE  ZSERI
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZBESI and ZBESK
C***LIBRARY   SLATEC
C***TYPE      ALL (CSERI-A, ZSERI-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     ZSERI COMPUTES THE I BESSEL FUNCTION FOR REAL(Z).GE.0.0 BY
C     MEANS OF THE POWER SERIES FOR LARGE ABS(Z) IN THE
C     REGION ABS(Z).LE.2*SQRT(FNU+1). NZ=0 IS A NORMAL RETURN.
C     NZ.GT.0 MEANS THAT THE LAST NZ COMPONENTS WERE SET TO ZERO
C     DUE TO UNDERFLOW. NZ.LT.0 MEANS UNDERFLOW OCCURRED, BUT THE
C     CONDITION ABS(Z).LE.2*SQRT(FNU+1) WAS VIOLATED AND THE
C     COMPUTATION MUST BE COMPLETED IN ANOTHER ROUTINE WITH N=N-ABS(NZ).
C
C***SEE ALSO  ZBESI, ZBESK
C***ROUTINES CALLED  D1MACH, DGAMLN, ZABS, ZDIV, ZLOG, ZMLT, ZUCHK
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C   930122  Added ZLOG to EXTERNAL statement.  (RWC)
C***END PROLOGUE  ZSERI
C     COMPLEX AK1,CK,COEF,CONE,CRSC,CSCL,CZ,CZERO,HZ,RZ,S1,S2,Y,Z
      DOUBLE PRECISION AA, ACZ, AK, AK1I, AK1R, ALIM, ARM, ASCLE, ATOL,
     * AZ, CKI, CKR, COEFI, COEFR, CONEI, CONER, CRSCR, CZI, CZR, DFNU,
     * ELIM, FNU, FNUP, HZI, HZR, RAZ, RS, RTR1, RZI, RZR, S, SS, STI,
```

```
     * STR, S1I, S1R, S2I, S2R, TOL, YI, YR, WI, WR, ZEROI, ZEROR, ZI,
     * ZR, DGAMLN, D1MACH, ZABS
       INTEGER I, IB, IDUM, IFLAG, IL, K, KODE, L, M, N, NN, NZ, NW
       DIMENSION YR(N), YI(N), WR(2), WI(2)
       EXTERNAL ZABS, ZLOG
       DATA ZEROR,ZEROI,CONER,CONEI / 0.0D0, 0.0D0, 1.0D0, 0.0D0 /
C***FIRST EXECUTABLE STATEMENT  ZSERI
       NZ = 0
       AZ = ZABS(ZR,ZI)
       IF (AZ.EQ.0.0D0) GO TO 160
       ARM = 1.0D+3*D1MACH(1)
       RTR1 = SQRT(ARM)
       CRSCR = 1.0D0
       IFLAG = 0
       IF (AZ.LT.ARM) GO TO 150
       HZR = 0.5D0*ZR
       HZI = 0.5D0*ZI
       CZR = ZEROR
       CZI = ZEROI
       IF (AZ.LE.RTR1) GO TO 10
       CALL ZMLT(HZR, HZI, HZR, HZI, CZR, CZI)
    10 CONTINUE
       ACZ = ZABS(CZR,CZI)
       NN = N
       CALL ZLOG(HZR, HZI, CKR, CKI, IDUM)
    20 CONTINUE
       DFNU = FNU + (NN-1)
       FNUP = DFNU + 1.0D0
C-----------------------------------------------------------------------
C     UNDERFLOW TEST
C-----------------------------------------------------------------------
       AK1R = CKR*DFNU
       AK1I = CKI*DFNU
       AK = DGAMLN(FNUP,IDUM)
       AK1R = AK1R - AK
       IF (KODE.EQ.2) AK1R = AK1R - ZR
       IF (AK1R.GT.(-ELIM)) GO TO 40
    30 CONTINUE
       NZ = NZ + 1
       YR(NN) = ZEROR
       YI(NN) = ZEROI
       IF (ACZ.GT.DFNU) GO TO 190
       NN = NN - 1
       IF (NN.EQ.0) RETURN
       GO TO 20
    40 CONTINUE
       IF (AK1R.GT.(-ALIM)) GO TO 50
       IFLAG = 1
       SS = 1.0D0/TOL
       CRSCR = TOL
       ASCLE = ARM*SS
    50 CONTINUE
       AA = EXP(AK1R)
       IF (IFLAG.EQ.1) AA = AA*SS
       COEFR = AA*COS(AK1I)
       COEFI = AA*SIN(AK1I)
       ATOL = TOL*ACZ/FNUP
       IL = MIN(2,NN)
```

```
      DO 90 I=1,IL
        DFNU = FNU + (NN-I)
        FNUP = DFNU + 1.0D0
        S1R = CONER
        S1I = CONEI
        IF (ACZ.LT.TOL*FNUP) GO TO 70
        AK1R = CONER
        AK1I = CONEI
        AK = FNUP + 2.0D0
        S = FNUP
        AA = 2.0D0
   60   CONTINUE
        RS = 1.0D0/S
        STR = AK1R*CZR - AK1I*CZI
        STI = AK1R*CZI + AK1I*CZR
        AK1R = STR*RS
        AK1I = STI*RS
        S1R = S1R + AK1R
        S1I = S1I + AK1I
        S = S + AK
        AK = AK + 2.0D0
        AA = AA*ACZ*RS
        IF (AA.GT.ATOL) GO TO 60
   70   CONTINUE
        S2R = S1R*COEFR - S1I*COEFI
        S2I = S1R*COEFI + S1I*COEFR
        WR(I) = S2R
        WI(I) = S2I
        IF (IFLAG.EQ.0) GO TO 80
        CALL ZUCHK(S2R, S2I, NW, ASCLE, TOL)
        IF (NW.NE.0) GO TO 30
   80   CONTINUE
        M = NN - I + 1
        YR(M) = S2R*CRSCR
        YI(M) = S2I*CRSCR
        IF (I.EQ.IL) GO TO 90
        CALL ZDIV(COEFR, COEFI, HZR, HZI, STR, STI)
        COEFR = STR*DFNU
        COEFI = STI*DFNU
   90 CONTINUE
      IF (NN.LE.2) RETURN
      K = NN - 2
      AK = K
      RAZ = 1.0D0/AZ
      STR = ZR*RAZ
      STI = -ZI*RAZ
      RZR = (STR+STR)*RAZ
      RZI = (STI+STI)*RAZ
      IF (IFLAG.EQ.1) GO TO 120
      IB = 3
  100 CONTINUE
      DO 110 I=IB,NN
        YR(K) = (AK+FNU)*(RZR*YR(K+1)-RZI*YI(K+1)) + YR(K+2)
        YI(K) = (AK+FNU)*(RZR*YI(K+1)+RZI*YR(K+1)) + YI(K+2)
        AK = AK - 1.0D0
        K = K - 1
  110 CONTINUE
      RETURN
```

```
C-----------------------------------------------------------------------
C     RECUR BACKWARD WITH SCALED VALUES
C-----------------------------------------------------------------------
  120 CONTINUE
C-----------------------------------------------------------------------
C     EXP(-ALIM)=EXP(-ELIM)/TOL=APPROX. ONE PRECISION ABOVE THE
C     UNDERFLOW LIMIT = ASCLE = D1MACH(1)*SS*1.0D+3
C-----------------------------------------------------------------------
      S1R = WR(1)
      S1I = WI(1)
      S2R = WR(2)
      S2I = WI(2)
      DO 130 L=3,NN
        CKR = S2R
        CKI = S2I
        S2R = S1R + (AK+FNU)*(RZR*CKR-RZI*CKI)
        S2I = S1I + (AK+FNU)*(RZR*CKI+RZI*CKR)
        S1R = CKR
        S1I = CKI
        CKR = S2R*CRSCR
        CKI = S2I*CRSCR
        YR(K) = CKR
        YI(K) = CKI
        AK = AK - 1.0D0
        K = K - 1
        IF (ZABS(CKR,CKI).GT.ASCLE) GO TO 140
  130 CONTINUE
      RETURN
  140 CONTINUE
      IB = L + 1
      IF (IB.GT.NN) RETURN
      GO TO 100
  150 CONTINUE
      NZ = N
      IF (FNU.EQ.0.0D0) NZ = NZ - 1
  160 CONTINUE
      YR(1) = ZEROR
      YI(1) = ZEROI
      IF (FNU.NE.0.0D0) GO TO 170
      YR(1) = CONER
      YI(1) = CONEI
  170 CONTINUE
      IF (N.EQ.1) RETURN
      DO 180 I=2,N
        YR(I) = ZEROR
        YI(I) = ZEROI
  180 CONTINUE
      RETURN
C-----------------------------------------------------------------------
C     RETURN WITH NZ.LT.0 IF ABS(Z*Z/4).GT.FNU+N-NZ-1 COMPLETE
C     THE CALCULATION IN CBINU WITH N=N-ABS(NZ)
C-----------------------------------------------------------------------
  190 CONTINUE
      NZ = -NZ
      RETURN
      END
*DECK ZUCHK
      SUBROUTINE ZUCHK (YR, YI, NZ, ASCLE, TOL)
```

```
C***BEGIN PROLOGUE  ZUCHK
C***SUBSIDIARY
C***PURPOSE  Subsidiary to SERI, ZUOIK, ZUNK1, ZUNK2, ZUNI1, ZUNI2 and
C            ZKSCL
C***LIBRARY   SLATEC
C***TYPE      ALL (CUCHK-A, ZUCHK-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C      Y ENTERS AS A SCALED QUANTITY WHOSE MAGNITUDE IS GREATER THAN
C      EXP(-ALIM)=ASCLE=1.0E+3*D1MACH(1)/TOL. THE TEST IS MADE TO SEE
C      IF THE MAGNITUDE OF THE REAL OR IMAGINARY PART WOULD UNDERFLOW
C      WHEN Y IS SCALED (BY TOL) TO ITS PROPER VALUE. Y IS ACCEPTED
C      IF THE UNDERFLOW IS AT LEAST ONE PRECISION BELOW THE MAGNITUDEf
C      OF THE LARGEST COMPONENT; OTHERWISE THE PHASE ANGLE DOES NOT HAVE
C      ABSOLUTE ACCURACY AND AN UNDERFLOW IS ASSUMED.
C
C***SEE ALSO  SERI, ZKSCL, ZUNI1, ZUNI2, ZUNK1, ZUNK2, ZUOIK
C***ROUTINES CALLED  (NONE)
C***REVISION HISTORY  (YYMMDD)
C   ??????  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C***END PROLOGUE  ZUCHK
C
C     COMPLEX Y
      DOUBLE PRECISION ASCLE, SS, ST, TOL, WR, WI, YR, YI
      INTEGER NZ
C***FIRST EXECUTABLE STATEMENT  ZUCHK
      NZ = 0
      WR = ABS(YR)
      WI = ABS(YI)
      ST = MIN(WR,WI)
      IF (ST.GT.ASCLE) RETURN
      SS = MAX(WR,WI)
      ST = ST/TOL
      IF (SS.LT.ST) NZ = 1
      RETURN
      END
*DECK ZUNI1
      SUBROUTINE ZUNI1 (ZR, ZI, FNU, KODE, N, YR, YI, NZ, NLAST, FNUL,
     +   TOL, ELIM, ALIM)
C***BEGIN PROLOGUE  ZUNI1
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZBESI and ZBESK
C***LIBRARY   SLATEC
C***TYPE      ALL (CUNI1-A, ZUNI1-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     ZUNI1 COMPUTES I(FNU,Z)  BY MEANS OF THE UNIFORM ASYMPTOTIC
C     EXPANSION FOR I(FNU,Z) IN -PI/3.LE.ARG Z.LE.PI/3.
C
C     FNUL IS THE SMALLEST ORDER PERMITTED FOR THE ASYMPTOTIC
C     EXPANSION. NLAST=0 MEANS ALL OF THE Y VALUES WERE SET.
C     NLAST.NE.0 IS THE NUMBER LEFT TO BE COMPUTED BY ANOTHER
C     FORMULA FOR ORDERS FNU TO FNU+NLAST-1 BECAUSE FNU+NLAST-1.LT.FNUL.
C     Y(I)=CZERO FOR I=NLAST+1,N
C
```

```
C***SEE ALSO  ZBESI, ZBESK
C***ROUTINES CALLED  D1MACH, ZABS, ZUCHK, ZUNIK, ZUOIK
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C***END PROLOGUE  ZUNI1
C     COMPLEX CFN,CONE,CRSC,CSCL,CSR,CSS,CWRK,CZERO,C1,C2,PHI,RZ,SUM,S1,
C    *S2,Y,Z,ZETA1,ZETA2
      DOUBLE PRECISION ALIM, APHI, ASCLE, BRY, CONER, CRSC,
     * CSCL, CSRR, CSSR, CWRKI, CWRKR, C1R, C2I, C2M, C2R, ELIM, FN,
     * FNU, FNUL, PHII, PHIR, RAST, RS1, RZI, RZR, STI, STR, SUMI,
     * SUMR, S1I, S1R, S2I, S2R, TOL, YI, YR, ZEROI, ZEROR, ZETA1I,
     * ZETA1R, ZETA2I, ZETA2R, ZI, ZR, CYR, CYI, D1MACH, ZABS
      INTEGER I, IFLAG, INIT, K, KODE, M, N, ND, NLAST, NN, NUF, NW, NZ
      DIMENSION BRY(3), YR(N), YI(N), CWRKR(16), CWRKI(16), CSSR(3),
     * CSRR(3), CYR(2), CYI(2)
      EXTERNAL ZABS
      DATA ZEROR,ZEROI,CONER / 0.0D0, 0.0D0, 1.0D0 /
C***FIRST EXECUTABLE STATEMENT  ZUNI1
      NZ = 0
      ND = N
      NLAST = 0
C-----------------------------------------------------------------------
C     COMPUTED VALUES WITH EXPONENTS BETWEEN ALIM AND ELIM IN MAG-
C     NITUDE ARE SCALED TO KEEP INTERMEDIATE ARITHMETIC ON SCALE,
C     EXP(ALIM)=EXP(ELIM)*TOL
C-----------------------------------------------------------------------
      CSCL = 1.0D0/TOL
      CRSC = TOL
      CSSR(1) = CSCL
      CSSR(2) = CONER
      CSSR(3) = CRSC
      CSRR(1) = CRSC
      CSRR(2) = CONER
      CSRR(3) = CSCL
      BRY(1) = 1.0D+3*D1MACH(1)/TOL
C-----------------------------------------------------------------------
C     CHECK FOR UNDERFLOW AND OVERFLOW ON FIRST MEMBER
C-----------------------------------------------------------------------
      FN = MAX(FNU,1.0D0)
      INIT = 0
      CALL ZUNIK(ZR, ZI, FN, 1, 1, TOL, INIT, PHIR, PHII, ZETA1R,
     * ZETA1I, ZETA2R, ZETA2I, SUMR, SUMI, CWRKR, CWRKI)
      IF (KODE.EQ.1) GO TO 10
      STR = ZR + ZETA2R
      STI = ZI + ZETA2I
      RAST = FN/ZABS(STR,STI)
      STR = STR*RAST*RAST
      STI = -STI*RAST*RAST
      S1R = -ZETA1R + STR
      S1I = -ZETA1I + STI
      GO TO 20
   10 CONTINUE
      S1R = -ZETA1R + ZETA2R
      S1I = -ZETA1I + ZETA2I
   20 CONTINUE
      RS1 = S1R
      IF (ABS(RS1).GT.ELIM) GO TO 130
```

```
   30 CONTINUE
      NN = MIN(2,ND)
      DO 80 I=1,NN
        FN = FNU + (ND-I)
        INIT = 0
        CALL ZUNIK(ZR, ZI, FN, 1, 0, TOL, INIT, PHIR, PHII, ZETA1R,
     *   ZETA1I, ZETA2R, ZETA2I, SUMR, SUMI, CWRKR, CWRKI)
        IF (KODE.EQ.1) GO TO 40
        STR = ZR + ZETA2R
        STI = ZI + ZETA2I
        RAST = FN/ZABS(STR,STI)
        STR = STR*RAST*RAST
        STI = -STI*RAST*RAST
        S1R = -ZETA1R + STR
        S1I = -ZETA1I + STI + ZI
        GO TO 50
   40   CONTINUE
        S1R = -ZETA1R + ZETA2R
        S1I = -ZETA1I + ZETA2I
   50   CONTINUE
C-----------------------------------------------------------------------
C     TEST FOR UNDERFLOW AND OVERFLOW
C-----------------------------------------------------------------------
        RS1 = S1R
        IF (ABS(RS1).GT.ELIM) GO TO 110
        IF (I.EQ.1) IFLAG = 2
        IF (ABS(RS1).LT.ALIM) GO TO 60
C-----------------------------------------------------------------------
C     REFINE  TEST AND SCALE
C-----------------------------------------------------------------------
        APHI = ZABS(PHIR,PHII)
        RS1 = RS1 + LOG(APHI)
        IF (ABS(RS1).GT.ELIM) GO TO 110
        IF (I.EQ.1) IFLAG = 1
        IF (RS1.LT.0.0D0) GO TO 60
        IF (I.EQ.1) IFLAG = 3
   60   CONTINUE
C-----------------------------------------------------------------------
C     SCALE S1 IF ABS(S1).LT.ASCLE
C-----------------------------------------------------------------------
        S2R = PHIR*SUMR - PHII*SUMI
        S2I = PHIR*SUMI + PHII*SUMR
        STR = EXP(S1R)*CSSR(IFLAG)
        S1R = STR*COS(S1I)
        S1I = STR*SIN(S1I)
        STR = S2R*S1R - S2I*S1I
        S2I = S2R*S1I + S2I*S1R
        S2R = STR
        IF (IFLAG.NE.1) GO TO 70
        CALL ZUCHK(S2R, S2I, NW, BRY(1), TOL)
        IF (NW.NE.0) GO TO 110
   70   CONTINUE
        CYR(I) = S2R
        CYI(I) = S2I
        M = ND - I + 1
        YR(M) = S2R*CSRR(IFLAG)
        YI(M) = S2I*CSRR(IFLAG)
   80 CONTINUE
```

```
      IF (ND.LE.2) GO TO 100
      RAST = 1.0D0/ZABS(ZR,ZI)
      STR = ZR*RAST
      STI = -ZI*RAST
      RZR = (STR+STR)*RAST
      RZI = (STI+STI)*RAST
      BRY(2) = 1.0D0/BRY(1)
      BRY(3) = D1MACH(2)
      S1R = CYR(1)
      S1I = CYI(1)
      S2R = CYR(2)
      S2I = CYI(2)
      C1R = CSRR(IFLAG)
      ASCLE = BRY(IFLAG)
      K = ND - 2
      FN = K
      DO 90 I=3,ND
        C2R = S2R
        C2I = S2I
        S2R = S1R + (FNU+FN)*(RZR*C2R-RZI*C2I)
        S2I = S1I + (FNU+FN)*(RZR*C2I+RZI*C2R)
        S1R = C2R
        S1I = C2I
        C2R = S2R*C1R
        C2I = S2I*C1R
        YR(K) = C2R
        YI(K) = C2I
        K = K - 1
        FN = FN - 1.0D0
        IF (IFLAG.GE.3) GO TO 90
        STR = ABS(C2R)
        STI = ABS(C2I)
        C2M = MAX(STR,STI)
        IF (C2M.LE.ASCLE) GO TO 90
        IFLAG = IFLAG + 1
        ASCLE = BRY(IFLAG)
        S1R = S1R*C1R
        S1I = S1I*C1R
        S2R = C2R
        S2I = C2I
        S1R = S1R*CSSR(IFLAG)
        S1I = S1I*CSSR(IFLAG)
        S2R = S2R*CSSR(IFLAG)
        S2I = S2I*CSSR(IFLAG)
        C1R = CSRR(IFLAG)
   90 CONTINUE
  100 CONTINUE
      RETURN
C-----------------------------------------------------------------------
C     SET UNDERFLOW AND UPDATE PARAMETERS
C-----------------------------------------------------------------------
  110 CONTINUE
      IF (RS1.GT.0.0D0) GO TO 120
      YR(ND) = ZEROR
      YI(ND) = ZEROI
      NZ = NZ + 1
      ND = ND - 1
      IF (ND.EQ.0) GO TO 100
```

```
      CALL ZUOIK(ZR, ZI, FNU, KODE, 1, ND, YR, YI, NUF, TOL, ELIM, ALIM)
      IF (NUF.LT.0) GO TO 120
      ND = ND - NUF
      NZ = NZ + NUF
      IF (ND.EQ.0) GO TO 100
      FN = FNU + (ND-1)
      IF (FN.GE.FNUL) GO TO 30
      NLAST = ND
      RETURN
  120 CONTINUE
      NZ = -1
      RETURN
  130 CONTINUE
      IF (RS1.GT.0.0D0) GO TO 120
      NZ = N
      DO 140 I=1,N
        YR(I) = ZEROR
        YI(I) = ZEROI
  140 CONTINUE
      RETURN
      END
*DECK ZUNI2
      SUBROUTINE ZUNI2 (ZR, ZI, FNU, KODE, N, YR, YI, NZ, NLAST, FNUL,
     +   TOL, ELIM, ALIM)
C***BEGIN PROLOGUE  ZUNI2
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZBESI and ZBESK
C***LIBRARY   SLATEC
C***TYPE      ALL (CUNI2-A, ZUNI2-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     ZUNI2 COMPUTES I(FNU,Z) IN THE RIGHT HALF PLANE BY MEANS OF
C     UNIFORM ASYMPTOTIC EXPANSION FOR J(FNU,ZN) WHERE ZN IS Z*I
C     OR -Z*I AND ZN IS IN THE RIGHT HALF PLANE ALSO.
C
C     FNUL IS THE SMALLEST ORDER PERMITTED FOR THE ASYMPTOTIC
C     EXPANSION. NLAST=0 MEANS ALL OF THE Y VALUES WERE SET.
C     NLAST.NE.0 IS THE NUMBER LEFT TO BE COMPUTED BY ANOTHER
C     FORMULA FOR ORDERS FNU TO FNU+NLAST-1 BECAUSE FNU+NLAST-1.LT.FNUL.
C     Y(I)=CZERO FOR I=NLAST+1,N
C
C***SEE ALSO  ZBESI, ZBESK
C***ROUTINES CALLED  D1MACH, ZABS, ZAIRY, ZUCHK, ZUNHJ, ZUOIK
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C***END PROLOGUE  ZUNI2
C     COMPLEX AI,ARG,ASUM,BSUM,CFN,CI,CID,CIP,CONE,CRSC,CSCL,CSR,CSS,
C    *CZERO,C1,C2,DAI,PHI,RZ,S1,S2,Y,Z,ZB,ZETA1,ZETA2,ZN
      DOUBLE PRECISION AARG, AIC, AII, AIR, ALIM, ANG, APHI, ARGI,
     * ARGR, ASCLE, ASUMI, ASUMR, BRY, BSUMI, BSUMR, CIDI, CIPI, CIPR,
     * CONER, CRSC, CSCL, CSRR, CSSR, C1R, C2I, C2M, C2R, DAII,
     * DAIR, ELIM, FN, FNU, FNUL, HPI, PHII, PHIR, RAST, RAZ, RS1, RZI,
     * RZR, STI, STR, S1I, S1R, S2I, S2R, TOL, YI, YR, ZBI, ZBR, ZEROI,
     * ZEROR, ZETA1I, ZETA1R, ZETA2I, ZETA2R, ZI, ZNI, ZNR, ZR, CYR,
     * CYI, D1MACH, ZABS, CAR, SAR
      INTEGER I, IFLAG, IN, INU, J, K, KODE, N, NAI, ND, NDAI, NLAST,
```

```
     * NN, NUF, NW, NZ, IDUM
      DIMENSION BRY(3), YR(N), YI(N), CIPR(4), CIPI(4), CSSR(3),
     * CSRR(3), CYR(2), CYI(2)
      EXTERNAL ZABS
      DATA ZEROR,ZEROI,CONER / 0.0D0, 0.0D0, 1.0D0 /
      DATA CIPR(1),CIPI(1),CIPR(2),CIPI(2),CIPR(3),CIPI(3),CIPR(4),
     * CIPI(4)/ 1.0D0,0.0D0, 0.0D0,1.0D0, -1.0D0,0.0D0, 0.0D0,-1.0D0/
      DATA HPI, AIC  /
     1      1.5707963267948966D+00,     1.2655121234846454D+00/
C***FIRST EXECUTABLE STATEMENT  ZUNI2
      NZ = 0
      ND = N
      NLAST = 0
C-----------------------------------------------------------------------
C     COMPUTED VALUES WITH EXPONENTS BETWEEN ALIM AND ELIM IN MAG-
C     NITUDE ARE SCALED TO KEEP INTERMEDIATE ARITHMETIC ON SCALE,
C     EXP(ALIM)=EXP(ELIM)*TOL
C-----------------------------------------------------------------------
      CSCL = 1.0D0/TOL
      CRSC = TOL
      CSSR(1) = CSCL
      CSSR(2) = CONER
      CSSR(3) = CRSC
      CSRR(1) = CRSC
      CSRR(2) = CONER
      CSRR(3) = CSCL
      BRY(1) = 1.0D+3*D1MACH(1)/TOL
C-----------------------------------------------------------------------
C     ZN IS IN THE RIGHT HALF PLANE AFTER ROTATION BY CI OR -CI
C-----------------------------------------------------------------------
      ZNR = ZI
      ZNI = -ZR
      ZBR = ZR
      ZBI = ZI
      CIDI = -CONER
      INU = FNU
      ANG = HPI*(FNU-INU)
      C2R = COS(ANG)
      C2I = SIN(ANG)
      CAR = C2R
      SAR = C2I
      IN = INU + N - 1
      IN = MOD(IN,4) + 1
      STR = C2R*CIPR(IN) - C2I*CIPI(IN)
      C2I = C2R*CIPI(IN) + C2I*CIPR(IN)
      C2R = STR
      IF (ZI.GT.0.0D0) GO TO 10
      ZNR = -ZNR
      ZBI = -ZBI
      CIDI = -CIDI
      C2I = -C2I
   10 CONTINUE
C-----------------------------------------------------------------------
C     CHECK FOR UNDERFLOW AND OVERFLOW ON FIRST MEMBER
C-----------------------------------------------------------------------
      FN = MAX(FNU,1.0D0)
      CALL ZUNHJ(ZNR, ZNI, FN, 1, TOL, PHIR, PHII, ARGR, ARGI, ZETA1R,
     * ZETA1I, ZETA2R, ZETA2I, ASUMR, ASUMI, BSUMR, BSUMI)
```

```
      IF (KODE.EQ.1) GO TO 20
      STR = ZBR + ZETA2R
      STI = ZBI + ZETA2I
      RAST = FN/ZABS(STR,STI)
      STR = STR*RAST*RAST
      STI = -STI*RAST*RAST
      S1R = -ZETA1R + STR
      S1I = -ZETA1I + STI
      GO TO 30
   20 CONTINUE
      S1R = -ZETA1R + ZETA2R
      S1I = -ZETA1I + ZETA2I
   30 CONTINUE
      RS1 = S1R
      IF (ABS(RS1).GT.ELIM) GO TO 150
   40 CONTINUE
      NN = MIN(2,ND)
      DO 90 I=1,NN
        FN = FNU + (ND-I)
        CALL ZUNHJ(ZNR, ZNI, FN, 0, TOL, PHIR, PHII, ARGR, ARGI,
     *   ZETA1R, ZETA1I, ZETA2R, ZETA2I, ASUMR, ASUMI, BSUMR, BSUMI)
        IF (KODE.EQ.1) GO TO 50
        STR = ZBR + ZETA2R
        STI = ZBI + ZETA2I
        RAST = FN/ZABS(STR,STI)
        STR = STR*RAST*RAST
        STI = -STI*RAST*RAST
        S1R = -ZETA1R + STR
        S1I = -ZETA1I + STI + ABS(ZI)
        GO TO 60
   50   CONTINUE
        S1R = -ZETA1R + ZETA2R
        S1I = -ZETA1I + ZETA2I
   60   CONTINUE
C-----------------------------------------------------------------------
C     TEST FOR UNDERFLOW AND OVERFLOW
C-----------------------------------------------------------------------
        RS1 = S1R
        IF (ABS(RS1).GT.ELIM) GO TO 120
        IF (I.EQ.1) IFLAG = 2
        IF (ABS(RS1).LT.ALIM) GO TO 70
C-----------------------------------------------------------------------
C     REFINE  TEST AND SCALE
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
        APHI = ZABS(PHIR,PHII)
        AARG = ZABS(ARGR,ARGI)
        RS1 = RS1 + LOG(APHI) - 0.25D0*LOG(AARG) - AIC
        IF (ABS(RS1).GT.ELIM) GO TO 120
        IF (I.EQ.1) IFLAG = 1
        IF (RS1.LT.0.0D0) GO TO 70
        IF (I.EQ.1) IFLAG = 3
   70   CONTINUE
C-----------------------------------------------------------------------
C     SCALE S1 TO KEEP INTERMEDIATE ARITHMETIC ON SCALE NEAR
C     EXPONENT EXTREMES
C-----------------------------------------------------------------------
        CALL ZAIRY(ARGR, ARGI, 0, 2, AIR, AII, NAI, IDUM)
```

```
      CALL ZAIRY(ARGR, ARGI, 1, 2, DAIR, DAII, NDAI, IDUM)
      STR = DAIR*BSUMR - DAII*BSUMI
      STI = DAIR*BSUMI + DAII*BSUMR
      STR = STR + (AIR*ASUMR-AII*ASUMI)
      STI = STI + (AIR*ASUMI+AII*ASUMR)
      S2R = PHIR*STR - PHII*STI
      S2I = PHIR*STI + PHII*STR
      STR = EXP(S1R)*CSSR(IFLAG)
      S1R = STR*COS(S1I)
      S1I = STR*SIN(S1I)
      STR = S2R*S1R - S2I*S1I
      S2I = S2R*S1I + S2I*S1R
      S2R = STR
      IF (IFLAG.NE.1) GO TO 80
      CALL ZUCHK(S2R, S2I, NW, BRY(1), TOL)
      IF (NW.NE.0) GO TO 120
   80 CONTINUE
      IF (ZI.LE.0.0D0) S2I = -S2I
      STR = S2R*C2R - S2I*C2I
      S2I = S2R*C2I + S2I*C2R
      S2R = STR
      CYR(I) = S2R
      CYI(I) = S2I
      J = ND - I + 1
      YR(J) = S2R*CSRR(IFLAG)
      YI(J) = S2I*CSRR(IFLAG)
      STR = -C2I*CIDI
      C2I = C2R*CIDI
      C2R = STR
   90 CONTINUE
      IF (ND.LE.2) GO TO 110
      RAZ = 1.0D0/ZABS(ZR,ZI)
      STR = ZR*RAZ
      STI = -ZI*RAZ
      RZR = (STR+STR)*RAZ
      RZI = (STI+STI)*RAZ
      BRY(2) = 1.0D0/BRY(1)
      BRY(3) = D1MACH(2)
      S1R = CYR(1)
      S1I = CYI(1)
      S2R = CYR(2)
      S2I = CYI(2)
      C1R = CSRR(IFLAG)
      ASCLE = BRY(IFLAG)
      K = ND - 2
      FN = K
      DO 100 I=3,ND
        C2R = S2R
        C2I = S2I
        S2R = S1R + (FNU+FN)*(RZR*C2R-RZI*C2I)
        S2I = S1I + (FNU+FN)*(RZR*C2I+RZI*C2R)
        S1R = C2R
        S1I = C2I
        C2R = S2R*C1R
        C2I = S2I*C1R
        YR(K) = C2R
        YI(K) = C2I
        K = K - 1
```

```
        FN = FN - 1.0D0
        IF (IFLAG.GE.3) GO TO 100
        STR = ABS(C2R)
        STI = ABS(C2I)
        C2M = MAX(STR,STI)
        IF (C2M.LE.ASCLE) GO TO 100
        IFLAG = IFLAG + 1
        ASCLE = BRY(IFLAG)
        S1R = S1R*C1R
        S1I = S1I*C1R
        S2R = C2R
        S2I = C2I
        S1R = S1R*CSSR(IFLAG)
        S1I = S1I*CSSR(IFLAG)
        S2R = S2R*CSSR(IFLAG)
        S2I = S2I*CSSR(IFLAG)
        C1R = CSRR(IFLAG)
  100 CONTINUE
  110 CONTINUE
      RETURN
  120 CONTINUE
      IF (RS1.GT.0.0D0) GO TO 140
C-----------------------------------------------------------------------
C     SET UNDERFLOW AND UPDATE PARAMETERS
C-----------------------------------------------------------------------
      YR(ND) = ZEROR
      YI(ND) = ZEROI
      NZ = NZ + 1
      ND = ND - 1
      IF (ND.EQ.0) GO TO 110
      CALL ZUOIK(ZR, ZI, FNU, KODE, 1, ND, YR, YI, NUF, TOL, ELIM, ALIM)
      IF (NUF.LT.0) GO TO 140
      ND = ND - NUF
      NZ = NZ + NUF
      IF (ND.EQ.0) GO TO 110
      FN = FNU + (ND-1)
      IF (FN.LT.FNUL) GO TO 130
C      FN = CIDI
C      J = NUF + 1
C      K = MOD(J,4) + 1
C      S1R = CIPR(K)
C      S1I = CIPI(K)
C      IF (FN.LT.0.0D0) S1I = -S1I
C      STR = C2R*S1R - C2I*S1I
C      C2I = C2R*S1I + C2I*S1R
C      C2R = STR
      IN = INU + ND - 1
      IN = MOD(IN,4) + 1
      C2R = CAR*CIPR(IN) - SAR*CIPI(IN)
      C2I = CAR*CIPI(IN) + SAR*CIPR(IN)
      IF (ZI.LE.0.0D0) C2I = -C2I
      GO TO 40
  130 CONTINUE
      NLAST = ND
      RETURN
  140 CONTINUE
      NZ = -1
      RETURN
```

```
  150 CONTINUE
      IF (RS1.GT.0.0D0) GO TO 140
      NZ = N
      DO 160 I=1,N
        YR(I) = ZEROR
        YI(I) = ZEROI
  160 CONTINUE
      RETURN
      END
*DECK ZUNIK
      SUBROUTINE ZUNIK (ZRR, ZRI, FNU, IKFLG, IPMTR, TOL, INIT, PHIR,
     +   PHII, ZETA1R, ZETA1I, ZETA2R, ZETA2I, SUMR, SUMI, CWRKR, CWRKI)
C***BEGIN PROLOGUE  ZUNIK
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZBESI and ZBESK
C***LIBRARY   SLATEC
C***TYPE      ALL (CUNIK-A, ZUNIK-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C        ZUNIK COMPUTES PARAMETERS FOR THE UNIFORM ASYMPTOTIC
C        EXPANSIONS OF THE I AND K FUNCTIONS ON IKFLG= 1 OR 2
C        RESPECTIVELY BY
C
C        W(FNU,ZR) = PHI*EXP(ZETA)*SUM
C
C        WHERE       ZETA=-ZETA1 + ZETA2       OR
C                         ZETA1 - ZETA2
C
C        THE FIRST CALL MUST HAVE INIT=0. SUBSEQUENT CALLS WITH THE
C        SAME ZR AND FNU WILL RETURN THE I OR K FUNCTION ON IKFLG=
C        1 OR 2 WITH NO CHANGE IN INIT. CWRK IS A COMPLEX WORK
C        ARRAY. IPMTR=0 COMPUTES ALL PARAMETERS. IPMTR=1 COMPUTES PHI,
C        ZETA1,ZETA2.
C
C***SEE ALSO  ZBESI, ZBESK
C***ROUTINES CALLED  D1MACH, ZDIV, ZLOG, ZSQRT
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C   930122  Added EXTERNAL statement with ZLOG and ZSQRT.  (RWC)
C***END PROLOGUE  ZUNIK
C     COMPLEX CFN,CON,CONE,CRFN,CWRK,CZERO,PHI,S,SR,SUM,T,T2,ZETA1,
C    *ZETA2,ZN,ZR
      DOUBLE PRECISION AC, C, CON, CONEI, CONER, CRFNI, CRFNR, CWRKI,
     * CWRKR, FNU, PHII, PHIR, RFN, SI, SR, SRI, SRR, STI, STR, SUMI,
     * SUMR, TEST, TI, TOL, TR, T2I, T2R, ZEROI, ZEROR, ZETA1I, ZETA1R,
     * ZETA2I, ZETA2R, ZNI, ZNR, ZRI, ZRR, D1MACH
      INTEGER I, IDUM, IKFLG, INIT, IPMTR, J, K, L
      DIMENSION C(120), CWRKR(16), CWRKI(16), CON(2)
      EXTERNAL ZLOG, ZSQRT
      DATA ZEROR,ZEROI,CONER,CONEI / 0.0D0, 0.0D0, 1.0D0, 0.0D0 /
      DATA CON(1), CON(2)  /
     1 3.9894228040143267D-01,  1.2533141373155003D+00 /
      DATA C(1), C(2), C(3), C(4), C(5), C(6), C(7), C(8), C(9), C(10),
     1     C(11), C(12), C(13), C(14), C(15), C(16), C(17), C(18),
     2     C(19), C(20), C(21), C(22), C(23), C(24)/
     3     1.0000000000000000D+00,    -2.0833333333333333D-01,
```

```
4     1.25000000000000000D-01,     3.34201388888888889D-01,
5    -4.01041666666666667D-01,     7.03125000000000000D-02,
6    -1.02581259645061728D+00,     1.84646267361111111D+00,
7    -8.91210937500000000D-01,     7.32421875000000000D-02,
8     4.66958442342624743D+00,    -1.12070026162229938D+01,
9     8.78912353515625000D+00,    -2.36408691406250000D+00,
A     1.12152099609375000D-01,    -2.82120725582002449D+01,
B     8.46362176746007346D+01,    -9.18182415432400174D+01,
C     4.25349987453884549D+01,    -7.36879435947963170D+00,
D     2.27108001708984375D-01,     2.12570130039217123D+02,
E    -7.65252468141181642D+02,     1.05999045252799988D+03/
 DATA C(25), C(26), C(27), C(28), C(29), C(30), C(31), C(32),
1     C(33), C(34), C(35), C(36), C(37), C(38), C(39), C(40),
2     C(41), C(42), C(43), C(44), C(45), C(46), C(47), C(48)/
3    -6.99579627376132541D+02,     2.18190511744211590D+02,
4    -2.64914304869515555D+01,     5.72501420974731445D-01,
5    -1.91945766231840700D+03,     8.06172218173730938D+03,
6    -1.35865500064341374D+04,     1.16553933368645332D+04,
7    -5.30564697861340311D+03,     1.20090291321635246D+03,
8    -1.08090919788394656D+02,     1.72772750258445740D+00,
9     2.02042913309661486D+04,    -9.69805983886375135D+04,
A     1.92547001232531532D+05,    -2.03400177280415534D+05,
B     1.22200464983017460D+05,    -4.11926549688975513D+04,
C     7.10951430248936372D+03,    -4.93915304773088012D+02,
D     6.07404200127348304D+00,    -2.42919187900551333D+05,
E     1.31176361466297720D+06,    -2.99801591853810675D+06/
 DATA C(49), C(50), C(51), C(52), C(53), C(54), C(55), C(56),
1     C(57), C(58), C(59), C(60), C(61), C(62), C(63), C(64),
2     C(65), C(66), C(67), C(68), C(69), C(70), C(71), C(72)/
3     3.76327129765640400D+06,    -2.81356322658653411D+06,
4     1.26836527332162478D+06,    -3.31645172484563578D+05,
5     4.52187689813627263D+04,    -2.49983048181120962D+03,
6     2.43805296995560639D+01,     3.28446985307203782D+06,
7    -1.97068191184322269D+07,     5.09526024926646422D+07,
8    -7.41051482115326577D+07,     6.63445122747290267D+07,
9    -3.75671766607633513D+07,     1.32887671664218183D+07,
A    -2.78561812808645469D+06,     3.08186404612662398D+05,
B    -1.38860897537170405D+04,     1.10017140269246738D+02,
C    -4.93292536645099620D+07,     3.25573074185765749D+08,
D    -9.39462359681578403D+08,     1.55359689957058006D+09,
E    -1.62108055210833708D+09,     1.10684281682301447D+09/
 DATA C(73), C(74), C(75), C(76), C(77), C(78), C(79), C(80),
1     C(81), C(82), C(83), C(84), C(85), C(86), C(87), C(88),
2     C(89), C(90), C(91), C(92), C(93), C(94), C(95), C(96)/
3    -4.95889784275030309D+08,     1.42062907797533095D+08,
4    -2.44740627257387285D+07,     2.24376817792244943D+06,
5    -8.40054336030240853D+04,     5.51335896122020586D+02,
6     8.14789096118312115D+08,    -5.86648149205184723D+09,
7     1.86882075092958249D+10,    -3.46320433881587779D+10,
8     4.12801855797539740D+10,    -3.30265997498007231D+10,
9     1.79542137311556001D+10,    -6.56329379261928433D+09,
A     1.55927986487925751D+09,    -2.25105661889415278D+08,
B     1.73951075539781645D+07,    -5.49842327572288687D+05,
C     3.03809051092238427D+03,    -1.46792612476956167D+10,
D     1.14498237732025810D+11,    -3.99096175224466498D+11,
E     8.19218669548577329D+11,    -1.09837515608122331D+12/
 DATA C(97), C(98), C(99), C(100), C(101), C(102), C(103), C(104),
1     C(105), C(106), C(107), C(108), C(109), C(110), C(111),
```

```
     2     C(112), C(113), C(114), C(115), C(116), C(117), C(118)/
     3     1.00815810686538209D+12,    -6.45364869245376503D+11,
     4     2.87900649906150589D+11,    -8.78670721780232657D+10,
     5     1.76347306068349694D+10,    -2.16716498322379509D+09,
     6     1.43157876718888981D+08,    -3.87183344257261262D+06,
     7     1.82577554742931747D+04,     2.86464035717679043D+11,
     8    -2.40629790002850396D+12,     9.10934118523989896D+12,
     9    -2.05168994109344374D+13,     3.05651255199353206D+13,
     A    -3.16670885847851584D+13,     2.33483640445818409D+13,
     B    -1.23204913055982872D+13,     4.61272578084913197D+12,
     C    -1.19655288019618160D+12,     2.05914503232410016D+11,
     D    -2.18229277575292237D+10,     1.24700929351271032D+09/
      DATA C(119), C(120)/
     1    -2.91883881222208134D+07,     1.18838426256783253D+05/
C***FIRST EXECUTABLE STATEMENT  ZUNIK
      IF (INIT.NE.0) GO TO 40
C-----------------------------------------------------------------------
C     INITIALIZE ALL VARIABLES
C-----------------------------------------------------------------------
      RFN = 1.0D0/FNU
C-----------------------------------------------------------------------
C     OVERFLOW TEST (ZR/FNU TOO SMALL)
C-----------------------------------------------------------------------
      TEST = D1MACH(1)*1.0D+3
      AC = FNU*TEST
      IF (ABS(ZRR).GT.AC .OR. ABS(ZRI).GT.AC) GO TO 15
      ZETA1R = 2.0D0*ABS(LOG(TEST))+FNU
      ZETA1I = 0.0D0
      ZETA2R = FNU
      ZETA2I = 0.0D0
      PHIR = 1.0D0
      PHII = 0.0D0
      RETURN
   15 CONTINUE
      TR = ZRR*RFN
      TI = ZRI*RFN
      SR = CONER + (TR*TR-TI*TI)
      SI = CONEI + (TR*TI+TI*TR)
      CALL ZSQRT(SR, SI, SRR, SRI)
      STR = CONER + SRR
      STI = CONEI + SRI
      CALL ZDIV(STR, STI, TR, TI, ZNR, ZNI)
      CALL ZLOG(ZNR, ZNI, STR, STI, IDUM)
      ZETA1R = FNU*STR
      ZETA1I = FNU*STI
      ZETA2R = FNU*SRR
      ZETA2I = FNU*SRI
      CALL ZDIV(CONER, CONEI, SRR, SRI, TR, TI)
      SRR = TR*RFN
      SRI = TI*RFN
      CALL ZSQRT(SRR, SRI, CWRKR(16), CWRKI(16))
      PHIR = CWRKR(16)*CON(IKFLG)
      PHII = CWRKI(16)*CON(IKFLG)
      IF (IPMTR.NE.0) RETURN
      CALL ZDIV(CONER, CONEI, SR, SI, T2R, T2I)
      CWRKR(1) = CONER
      CWRKI(1) = CONEI
      CRFNR = CONER
```

```
      CRFNI = CONEI
      AC = 1.0D0
      L = 1
      DO 20 K=2,15
        SR = ZEROR
        SI = ZEROI
        DO 10 J=1,K
          L = L + 1
          STR = SR*T2R - SI*T2I + C(L)
          SI = SR*T2I + SI*T2R
          SR = STR
   10   CONTINUE
        STR = CRFNR*SRR - CRFNI*SRI
        CRFNI = CRFNR*SRI + CRFNI*SRR
        CRFNR = STR
        CWRKR(K) = CRFNR*SR - CRFNI*SI
        CWRKI(K) = CRFNR*SI + CRFNI*SR
        AC = AC*RFN
        TEST = ABS(CWRKR(K)) + ABS(CWRKI(K))
        IF (AC.LT.TOL .AND. TEST.LT.TOL) GO TO 30
   20 CONTINUE
      K = 15
   30 CONTINUE
      INIT = K
   40 CONTINUE
      IF (IKFLG.EQ.2) GO TO 60
C-----------------------------------------------------------------------
C     COMPUTE SUM FOR THE I FUNCTION
C-----------------------------------------------------------------------
      SR = ZEROR
      SI = ZEROI
      DO 50 I=1,INIT
        SR = SR + CWRKR(I)
        SI = SI + CWRKI(I)
   50 CONTINUE
      SUMR = SR
      SUMI = SI
      PHIR = CWRKR(16)*CON(1)
      PHII = CWRKI(16)*CON(1)
      RETURN
   60 CONTINUE
C-----------------------------------------------------------------------
C     COMPUTE SUM FOR THE K FUNCTION
C-----------------------------------------------------------------------
      SR = ZEROR
      SI = ZEROI
      TR = CONER
      DO 70 I=1,INIT
        SR = SR + TR*CWRKR(I)
        SI = SI + TR*CWRKI(I)
        TR = -TR
   70 CONTINUE
      SUMR = SR
      SUMI = SI
      PHIR = CWRKR(16)*CON(2)
      PHII = CWRKI(16)*CON(2)
      RETURN
      END
```

```
*DECK ZUOIK
      SUBROUTINE ZUOIK (ZR, ZI, FNU, KODE, IKFLG, N, YR, YI, NUF, TOL,
     +   ELIM, ALIM)
C***BEGIN PROLOGUE  ZUOIK
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZBESH, ZBESI and ZBESK
C***LIBRARY   SLATEC
C***TYPE      ALL (CUOIK-A, ZUOIK-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     ZUOIK COMPUTES THE LEADING TERMS OF THE UNIFORM ASYMPTOTIC
C     EXPANSIONS FOR THE I AND K FUNCTIONS AND COMPARES THEM
C     (IN LOGARITHMIC FORM) TO ALIM AND ELIM FOR OVER AND UNDERFLOW
C     WHERE ALIM.LT.ELIM. IF THE MAGNITUDE, BASED ON THE LEADING
C     EXPONENTIAL, IS LESS THAN ALIM OR GREATER THAN -ALIM, THEN
C     THE RESULT IS ON SCALE. IF NOT, THEN A REFINED TEST USING OTHER
C     MULTIPLIERS (IN LOGARITHMIC FORM) IS MADE BASED ON ELIM. HERE
C     EXP(-ELIM)=SMALLEST MACHINE NUMBER*1.0E+3 AND EXP(-ALIM)=
C     EXP(-ELIM)/TOL
C
C     IKFLG=1 MEANS THE I SEQUENCE IS TESTED
C          =2 MEANS THE K SEQUENCE IS TESTED
C     NUF = 0 MEANS THE LAST MEMBER OF THE SEQUENCE IS ON SCALE
C         =-1 MEANS AN OVERFLOW WOULD OCCUR
C     IKFLG=1 AND NUF.GT.0 MEANS THE LAST NUF Y VALUES WERE SET TO ZERO
C             THE FIRST N-NUF VALUES MUST BE SET BY ANOTHER ROUTINE
C     IKFLG=2 AND NUF.EQ.N MEANS ALL Y VALUES WERE SET TO ZERO
C     IKFLG=2 AND 0.LT.NUF.LT.N NOT CONSIDERED. Y MUST BE SET BY
C             ANOTHER ROUTINE
C
C***SEE ALSO  ZBESH, ZBESI, ZBESK
C***ROUTINES CALLED  D1MACH, ZABS, ZLOG, ZUCHK, ZUNHJ, ZUNIK
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C   930122  Added ZLOG to EXTERNAL statement.  (RWC)
C***END PROLOGUE  ZUOIK
C     COMPLEX ARG,ASUM,BSUM,CWRK,CZ,CZERO,PHI,SUM,Y,Z,ZB,ZETA1,ZETA2,ZN,
C    *ZR
      DOUBLE PRECISION AARG, AIC, ALIM, APHI, ARGI, ARGR, ASUMI, ASUMR,
     * ASCLE, AX, AY, BSUMI, BSUMR, CWRKI, CWRKR, CZI, CZR, ELIM, FNN,
     * FNU, GNN, GNU, PHII, PHIR, RCZ, STR, STI, SUMI, SUMR, TOL, YI,
     * YR, ZBI, ZBR, ZEROI, ZEROR, ZETA1I, ZETA1R, ZETA2I, ZETA2R, ZI,
     * ZNI, ZNR, ZR, ZRI, ZRR, D1MACH, ZABS
      INTEGER I, IDUM, IFORM, IKFLG, INIT, KODE, N, NN, NUF, NW
      DIMENSION YR(N), YI(N), CWRKR(16), CWRKI(16)
      EXTERNAL ZABS, ZLOG
      DATA ZEROR,ZEROI / 0.0D0, 0.0D0 /
      DATA AIC / 1.265512123484645396D+00 /
C***FIRST EXECUTABLE STATEMENT  ZUOIK
      NUF = 0
      NN = N
      ZRR = ZR
      ZRI = ZI
      IF (ZR.GE.0.0D0) GO TO 10
      ZRR = -ZR
      ZRI = -ZI
```

```
   10 CONTINUE
      ZBR = ZRR
      ZBI = ZRI
      AX = ABS(ZR)*1.7321D0
      AY = ABS(ZI)
      IFORM = 1
      IF (AY.GT.AX) IFORM = 2
      GNU = MAX(FNU,1.0D0)
      IF (IKFLG.EQ.1) GO TO 20
      FNN = NN
      GNN = FNU + FNN - 1.0D0
      GNU = MAX(GNN,FNN)
   20 CONTINUE
C-----------------------------------------------------------------------
C     ONLY THE MAGNITUDE OF ARG AND PHI ARE NEEDED ALONG WITH THE
C     REAL PARTS OF ZETA1, ZETA2 AND ZB. NO ATTEMPT IS MADE TO GET
C     THE SIGN OF THE IMAGINARY PART CORRECT.
C-----------------------------------------------------------------------
      IF (IFORM.EQ.2) GO TO 30
      INIT = 0
      CALL ZUNIK(ZRR, ZRI, GNU, IKFLG, 1, TOL, INIT, PHIR, PHII,
     * ZETA1R, ZETA1I, ZETA2R, ZETA2I, SUMR, SUMI, CWRKR, CWRKI)
      CZR = -ZETA1R + ZETA2R
      CZI = -ZETA1I + ZETA2I
      GO TO 50
   30 CONTINUE
      ZNR = ZRI
      ZNI = -ZRR
      IF (ZI.GT.0.0D0) GO TO 40
      ZNR = -ZNR
   40 CONTINUE
      CALL ZUNHJ(ZNR, ZNI, GNU, 1, TOL, PHIR, PHII, ARGR, ARGI, ZETA1R,
     * ZETA1I, ZETA2R, ZETA2I, ASUMR, ASUMI, BSUMR, BSUMI)
      CZR = -ZETA1R + ZETA2R
      CZI = -ZETA1I + ZETA2I
      AARG = ZABS(ARGR,ARGI)
   50 CONTINUE
      IF (KODE.EQ.1) GO TO 60
      CZR = CZR - ZBR
      CZI = CZI - ZBI
   60 CONTINUE
      IF (IKFLG.EQ.1) GO TO 70
      CZR = -CZR
      CZI = -CZI
   70 CONTINUE
      APHI = ZABS(PHIR,PHII)
      RCZ = CZR
C-----------------------------------------------------------------------
C     OVERFLOW TEST
C-----------------------------------------------------------------------
      IF (RCZ.GT.ELIM) GO TO 210
      IF (RCZ.LT.ALIM) GO TO 80
      RCZ = RCZ + LOG(APHI)
      IF (IFORM.EQ.2) RCZ = RCZ - 0.25D0*LOG(AARG) - AIC
      IF (RCZ.GT.ELIM) GO TO 210
      GO TO 130
   80 CONTINUE
C-----------------------------------------------------------------------
```

```
C     UNDERFLOW TEST
C-----------------------------------------------------------------------
      IF (RCZ.LT.(-ELIM)) GO TO 90
      IF (RCZ.GT.(-ALIM)) GO TO 130
      RCZ = RCZ + LOG(APHI)
      IF (IFORM.EQ.2) RCZ = RCZ - 0.25D0*LOG(AARG) - AIC
      IF (RCZ.GT.(-ELIM)) GO TO 110
   90 CONTINUE
      DO 100 I=1,NN
        YR(I) = ZEROR
        YI(I) = ZEROI
  100 CONTINUE
      NUF = NN
      RETURN
  110 CONTINUE
      ASCLE = 1.0D+3*D1MACH(1)/TOL
      CALL ZLOG(PHIR, PHII, STR, STI, IDUM)
      CZR = CZR + STR
      CZI = CZI + STI
      IF (IFORM.EQ.1) GO TO 120
      CALL ZLOG(ARGR, ARGI, STR, STI, IDUM)
      CZR = CZR - 0.25D0*STR - AIC
      CZI = CZI - 0.25D0*STI
  120 CONTINUE
      AX = EXP(RCZ)/TOL
      AY = CZI
      CZR = AX*COS(AY)
      CZI = AX*SIN(AY)
      CALL ZUCHK(CZR, CZI, NW, ASCLE, TOL)
      IF (NW.NE.0) GO TO 90
  130 CONTINUE
      IF (IKFLG.EQ.2) RETURN
      IF (N.EQ.1) RETURN
C-----------------------------------------------------------------------
C     SET UNDERFLOWS ON I SEQUENCE
C-----------------------------------------------------------------------
  140 CONTINUE
      GNU = FNU + (NN-1)
      IF (IFORM.EQ.2) GO TO 150
      INIT = 0
      CALL ZUNIK(ZRR, ZRI, GNU, IKFLG, 1, TOL, INIT, PHIR, PHII,
     * ZETA1R, ZETA1I, ZETA2R, ZETA2I, SUMR, SUMI, CWRKR, CWRKI)
      CZR = -ZETA1R + ZETA2R
      CZI = -ZETA1I + ZETA2I
      GO TO 160
  150 CONTINUE
      CALL ZUNHJ(ZNR, ZNI, GNU, 1, TOL, PHIR, PHII, ARGR, ARGI, ZETA1R,
     * ZETA1I, ZETA2R, ZETA2I, ASUMR, ASUMI, BSUMR, BSUMI)
      CZR = -ZETA1R + ZETA2R
      CZI = -ZETA1I + ZETA2I
      AARG = ZABS(ARGR,ARGI)
  160 CONTINUE
      IF (KODE.EQ.1) GO TO 170
      CZR = CZR - ZBR
      CZI = CZI - ZBI
  170 CONTINUE
      APHI = ZABS(PHIR,PHII)
      RCZ = CZR
```

```
      IF (RCZ.LT.(-ELIM)) GO TO 180
      IF (RCZ.GT.(-ALIM)) RETURN
      RCZ = RCZ + LOG(APHI)
      IF (IFORM.EQ.2) RCZ = RCZ - 0.25D0*LOG(AARG) - AIC
      IF (RCZ.GT.(-ELIM)) GO TO 190
  180 CONTINUE
      YR(NN) = ZEROR
      YI(NN) = ZEROI
      NN = NN - 1
      NUF = NUF + 1
      IF (NN.EQ.0) RETURN
      GO TO 140
  190 CONTINUE
      ASCLE = 1.0D+3*D1MACH(1)/TOL
      CALL ZLOG(PHIR, PHII, STR, STI, IDUM)
      CZR = CZR + STR
      CZI = CZI + STI
      IF (IFORM.EQ.1) GO TO 200
      CALL ZLOG(ARGR, ARGI, STR, STI, IDUM)
      CZR = CZR - 0.25D0*STR - AIC
      CZI = CZI - 0.25D0*STI
  200 CONTINUE
      AX = EXP(RCZ)/TOL
      AY = CZI
      CZR = AX*COS(AY)
      CZI = AX*SIN(AY)
      CALL ZUCHK(CZR, CZI, NW, ASCLE, TOL)
      IF (NW.NE.0) GO TO 180
      RETURN
  210 CONTINUE
      NUF = -1
      RETURN
      END
*DECK ZWRSK
      SUBROUTINE ZWRSK (ZRR, ZRI, FNU, KODE, N, YR, YI, NZ, CWR, CWI,
     +   TOL, ELIM, ALIM)
C***BEGIN PROLOGUE  ZWRSK
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZBESI and ZBESK
C***LIBRARY   SLATEC
C***TYPE      ALL (CWRSK-A, ZWRSK-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     ZWRSK COMPUTES THE I BESSEL FUNCTION FOR RE(Z).GE.0.0 BY
C     NORMALIZING THE I FUNCTION RATIOS FROM ZRATI BY THE WRONSKIAN
C
C***SEE ALSO  ZBESI, ZBESK
C***ROUTINES CALLED  D1MACH, ZABS, ZBKNU, ZRATI
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C***END PROLOGUE  ZWRSK
C     COMPLEX CINU,CSCL,CT,CW,C1,C2,RCT,ST,Y,ZR
      DOUBLE PRECISION ACT, ACW, ALIM, ASCLE, CINUI, CINUR, CSCLR, CTI,
     * CTR, CWI, CWR, C1I, C1R, C2I, C2R, ELIM, FNU, PTI, PTR, RACT,
     * STI, STR, TOL, YI, YR, ZRI, ZRR, ZABS, D1MACH
      INTEGER I, KODE, N, NW, NZ
```

```
      DIMENSION YR(N), YI(N), CWR(2), CWI(2)
      EXTERNAL ZABS
C***FIRST EXECUTABLE STATEMENT  ZWRSK
C-----------------------------------------------------------------------
C     I(FNU+I-1,Z) BY BACKWARD RECURRENCE FOR RATIOS
C     Y(I)=I(FNU+I,Z)/I(FNU+I-1,Z) FROM CRATI NORMALIZED BY THE
C     WRONSKIAN WITH K(FNU,Z) AND K(FNU+1,Z) FROM CBKNU.
C-----------------------------------------------------------------------
C
      NZ = 0
      CALL ZBKNU(ZRR, ZRI, FNU, KODE, 2, CWR, CWI, NW, TOL, ELIM, ALIM)
      IF (NW.NE.0) GO TO 50
      CALL ZRATI(ZRR, ZRI, FNU, N, YR, YI, TOL)
C-----------------------------------------------------------------------
C     RECUR FORWARD ON I(FNU+1,Z) = R(FNU,Z)*I(FNU,Z),
C     R(FNU+J-1,Z)=Y(J),  J=1,...,N
C-----------------------------------------------------------------------
      CINUR = 1.0D0
      CINUI = 0.0D0
      IF (KODE.EQ.1) GO TO 10
      CINUR = COS(ZRI)
      CINUI = SIN(ZRI)
   10 CONTINUE
C-----------------------------------------------------------------------
C     ON LOW EXPONENT MACHINES THE K FUNCTIONS CAN BE CLOSE TO BOTH
C     THE UNDER AND OVERFLOW LIMITS AND THE NORMALIZATION MUST BE
C     SCALED TO PREVENT OVER OR UNDERFLOW. CUOIK HAS DETERMINED THAT
C     THE RESULT IS ON SCALE.
C-----------------------------------------------------------------------
      ACW = ZABS(CWR(2),CWI(2))
      ASCLE = 1.0D+3*D1MACH(1)/TOL
      CSCLR = 1.0D0
      IF (ACW.GT.ASCLE) GO TO 20
      CSCLR = 1.0D0/TOL
      GO TO 30
   20 CONTINUE
      ASCLE = 1.0D0/ASCLE
      IF (ACW.LT.ASCLE) GO TO 30
      CSCLR = TOL
   30 CONTINUE
      C1R = CWR(1)*CSCLR
      C1I = CWI(1)*CSCLR
      C2R = CWR(2)*CSCLR
      C2I = CWI(2)*CSCLR
      STR = YR(1)
      STI = YI(1)
C-----------------------------------------------------------------------
C     CINU=CINU*(CONJG(CT)/ABS(CT))*(1.0D0/ABS(CT) PREVENTS
C     UNDER- OR OVERFLOW PREMATURELY BY SQUARING ABS(CT)
C-----------------------------------------------------------------------
      PTR = STR*C1R - STI*C1I
      PTI = STR*C1I + STI*C1R
      PTR = PTR + C2R
      PTI = PTI + C2I
      CTR = ZRR*PTR - ZRI*PTI
      CTI = ZRR*PTI + ZRI*PTR
      ACT = ZABS(CTR,CTI)
      RACT = 1.0D0/ACT
```

```
      CTR = CTR*RACT
      CTI = -CTI*RACT
      PTR = CINUR*RACT
      PTI = CINUI*RACT
      CINUR = PTR*CTR - PTI*CTI
      CINUI = PTR*CTI + PTI*CTR
      YR(1) = CINUR*CSCLR
      YI(1) = CINUI*CSCLR
      IF (N.EQ.1) RETURN
      DO 40 I=2,N
        PTR = STR*CINUR - STI*CINUI
        CINUI = STR*CINUI + STI*CINUR
        CINUR = PTR
        STR = YR(I)
        STI = YI(I)
        YR(I) = CINUR*CSCLR
        YI(I) = CINUI*CSCLR
   40 CONTINUE
      RETURN
   50 CONTINUE
      NZ = -1
      IF(NW.EQ.(-2)) NZ=-2
      RETURN
      END
*DECK DGAMLN
      DOUBLE PRECISION FUNCTION DGAMLN (Z, IERR)
C***BEGIN PROLOGUE  DGAMLN
C***SUBSIDIARY
C***PURPOSE  Compute the logarithm of the Gamma function
C***LIBRARY   SLATEC
C***CATEGORY  C7A
C***TYPE      DOUBLE PRECISION (GAMLN-S, DGAMLN-D)
C***KEYWORDS  LOGARITHM OF GAMMA FUNCTION
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C              **** A DOUBLE PRECISION ROUTINE ****
C         DGAMLN COMPUTES THE NATURAL LOG OF THE GAMMA FUNCTION FOR
C         Z.GT.0.  THE ASYMPTOTIC EXPANSION IS USED TO GENERATE VALUES
C         GREATER THAN ZMIN WHICH ARE ADJUSTED BY THE RECURSION
C         G(Z+1)=Z*G(Z) FOR Z.LE.ZMIN.  THE FUNCTION WAS MADE AS
C         PORTABLE AS POSSIBLE BY COMPUTING ZMIN FROM THE NUMBER OF BASE
C         10 DIGITS IN A WORD, RLN=MAX(-ALOG10(R1MACH(4)),0.5E-18)
C         LIMITED TO 18 DIGITS OF (RELATIVE) ACCURACY.
C
C         SINCE INTEGER ARGUMENTS ARE COMMON, A TABLE LOOK UP ON 100
C         VALUES IS USED FOR SPEED OF EXECUTION.
C
C     DESCRIPTION OF ARGUMENTS
C
C         INPUT      Z IS D0UBLE PRECISION
C           Z      - ARGUMENT, Z.GT.0.0D0
C
C         OUTPUT      DGAMLN IS DOUBLE PRECISION
C           DGAMLN  - NATURAL LOG OF THE GAMMA FUNCTION AT Z.NE.0.0D0
C           IERR    - ERROR FLAG
C                     IERR=0, NORMAL RETURN, COMPUTATION COMPLETED
C                     IERR=1, Z.LE.0.0D0,    NO COMPUTATION
```

```
C
C
C***REFERENCES  COMPUTATION OF BESSEL FUNCTIONS OF COMPLEX ARGUMENT
C                 BY D. E. AMOS, SAND83-0083, MAY, 1983.
C***ROUTINES CALLED  D1MACH, I1MACH
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   830501  REVISION DATE from Version 3.2
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C   920128  Category corrected.  (WRB)
C   921215  DGAMLN defined for Z negative.  (WRB)
C***END PROLOGUE  DGAMLN
      DOUBLE PRECISION CF, CON, FLN, FZ, GLN, RLN, S, TLG, TRM, TST,
     * T1, WDTOL, Z, ZDMY, ZINC, ZM, ZMIN, ZP, ZSQ, D1MACH
      INTEGER I, IERR, I1M, K, MZ, NZ, I1MACH
      DIMENSION CF(22), GLN(100)
C           LNGAMMA(N), N=1,100
      DATA GLN(1), GLN(2), GLN(3), GLN(4), GLN(5), GLN(6), GLN(7),
     1     GLN(8), GLN(9), GLN(10), GLN(11), GLN(12), GLN(13), GLN(14),
     2     GLN(15), GLN(16), GLN(17), GLN(18), GLN(19), GLN(20),
     3     GLN(21), GLN(22)/
     4     0.0000000000000000D+00,     0.0000000000000000D+00,
     5     6.9314718055994309D-01,     1.7917594692280550D+00,
     6     3.1780538303479456D+00,     4.7874917427820459D+00,
     7     6.5792512120101001D+00,     8.5251613610654143D+00,
     8     1.0604602902745251D+01,     1.2801827480081469D+01,
     9     1.5104412573075515D+01,     1.7502307845873885D+01,
     A     1.9987214495661886D+01,     2.2552163853123423D+01,
     B     2.5191221182738681D+01,     2.7899271383840891D+01,
     C     3.0671860106080672D+01,     3.3505073450136889D+01,
     D     3.6395445208033054D+01,     3.9339884187199494D+01,
     E     4.2335616460753485D+01,     4.5380138898476908D+01/
      DATA GLN(23), GLN(24), GLN(25), GLN(26), GLN(27), GLN(28),
     1     GLN(29), GLN(30), GLN(31), GLN(32), GLN(33), GLN(34),
     2     GLN(35), GLN(36), GLN(37), GLN(38), GLN(39), GLN(40),
     3     GLN(41), GLN(42), GLN(43), GLN(44)/
     4     4.8471181351835224D+01,     5.1606675567764374D+01,
     5     5.4784729398112319D+01,     5.8003605222980520D+01,
     6     6.1261701761002002D+01,     6.4557538627006338D+01,
     7     6.7889743137181535D+01,     7.1257038967168009D+01,
     8     7.4658236348830164D+01,     7.8092223553315311D+01,
     9     8.1557959456115037D+01,     8.5054467017581518D+01,
     A     8.8580827542197679D+01,     9.2136175603687093D+01,
     B     9.5719694542143203D+01,     9.9330612454787427D+01,
     C     1.0296819861451381D+02,     1.0663176026064346D+02,
     D     1.1032063971475740D+02,     1.1403421178146170D+02,
     E     1.1777188139974507D+02,     1.2153308151543863D+02/
      DATA GLN(45), GLN(46), GLN(47), GLN(48), GLN(49), GLN(50),
     1     GLN(51), GLN(52), GLN(53), GLN(54), GLN(55), GLN(56),
     2     GLN(57), GLN(58), GLN(59), GLN(60), GLN(61), GLN(62),
     3     GLN(63), GLN(64), GLN(65), GLN(66)/
     4     1.2531727114935690D+02,     1.2912393363912722D+02,
     5     1.3295257503561631D+02,     1.3680272263732637D+02,
     6     1.4067392364823426D+02,     1.4456574394634489D+02,
     7     1.4847776695177303D+02,     1.5240959258449736D+02,
     8     1.5636083630307879D+02,     1.6033112821663091D+02,
     9     1.6432011226319518D+02,     1.6832744544842765D+02,
     A     1.7235279713916280D+02,     1.7639584840699735D+02,
```

```
     B     1.80456291417543771D+02,     1.84533828861449491D+02,
     C     1.88628173423671591D+02,     1.92739047287844902D+02,
     D     1.96866181672889994D+02,     2.01009316399281527D+02,
     E     2.05168199482641199D+02,     2.09342586752536836D+02/
      DATA GLN(67), GLN(68), GLN(69), GLN(70), GLN(71), GLN(72),
     1     GLN(73), GLN(74), GLN(75), GLN(76), GLN(77), GLN(78),
     2     GLN(79), GLN(80), GLN(81), GLN(82), GLN(83), GLN(84),
     3     GLN(85), GLN(86), GLN(87), GLN(88)/
     4     2.13532241494563261D+02,     2.17736934113954227D+02,
     5     2.21956441819130334D+02,     2.26190548323727593D+02,
     6     2.30439043565776952D+02,     2.34701723442818268D+02,
     7     2.38978389561834323D+02,     2.43268849002982714D+02,
     8     2.47572914096186884D+02,     2.51890402209723194D+02,
     9     2.56221135550009525D+02,     2.60564940971863209D+02,
     A     2.64921649798552801D+02,     2.69291097651019823D+02,
     B     2.73673124285693704D+02,     2.78067573440366143D+02,
     C     2.82474292687630396D+02,     2.86893133295426994D+02,
     D     2.91323950094270308D+02,     2.95766601350760624D+02,
     E     3.00220948647014132D+02,     3.04686856765668715D+02/
      DATA GLN(89), GLN(90), GLN(91), GLN(92), GLN(93), GLN(94),
     1     GLN(95), GLN(96), GLN(97), GLN(98), GLN(99), GLN(100)/
     2     3.09164193580146922D+02,     3.13652829949879062D+02,
     3     3.18152639620209327D+02,     3.22663499126726177D+02,
     4     3.27185287703775217D+02,     3.31717887196928473D+02,
     5     3.36261181979198477D+02,     3.40815058870799018D+02,
     6     3.45379407062266854D+02,     3.49954118040770237D+02,
     7     3.54539085519440809D+02,     3.59134205369575399D+02/
C            COEFFICIENTS OF ASYMPTOTIC EXPANSION
      DATA CF(1), CF(2), CF(3), CF(4), CF(5), CF(6), CF(7), CF(8),
     1     CF(9), CF(10), CF(11), CF(12), CF(13), CF(14), CF(15),
     2     CF(16), CF(17), CF(18), CF(19), CF(20), CF(21), CF(22)/
     3     8.33333333333333333D-02,    -2.77777777777777778D-03,
     4     7.93650793650793651D-04,    -5.95238095238095238D-04,
     5     8.41750841750841751D-04,    -1.91752691752691753D-03,
     6     6.41025641025641026D-03,    -2.95506535947712418D-02,
     7     1.79644372368830573D-01,    -1.39243221690590112D+00,
     8     1.34028640441683920D+01,    -1.56848284626002017D+02,
     9     2.19310333333333333D+03,    -3.61087712537249894D+04,
     A     6.91472268851313067D+05,    -1.52382215394074162D+07,
     B     3.82900751391414141D+08,    -1.08822660357843911D+10,
     C     3.47320283765002252D+11,    -1.23696021422692745D+13,
     D     4.88788064793079335D+14,    -2.13203339609193739D+16/
C
C            LN(2*PI)
      DATA CON                   /     1.83787706640934548D+00/
C
C***FIRST EXECUTABLE STATEMENT  DGAMLN
      IERR=0
      IF (Z.LE.0.0D0) GO TO 70
      IF (Z.GT.101.0D0) GO TO 10
      NZ = Z
      FZ = Z - NZ
      IF (FZ.GT.0.0D0) GO TO 10
      IF (NZ.GT.100) GO TO 10
      DGAMLN = GLN(NZ)
      RETURN
   10 CONTINUE
      WDTOL = D1MACH(4)
```

```
      WDTOL = MAX(WDTOL,0.5D-18)
      I1M = I1MACH(14)
      RLN = D1MACH(5)*I1M
      FLN = MIN(RLN,20.0D0)
      FLN = MAX(FLN,3.0D0)
      FLN = FLN - 3.0D0
      ZM = 1.8000D0 + 0.3875D0*FLN
      MZ = ZM + 1
      ZMIN = MZ
      ZDMY = Z
      ZINC = 0.0D0
      IF (Z.GE.ZMIN) GO TO 20
      ZINC = ZMIN - NZ
      ZDMY = Z + ZINC
   20 CONTINUE
      ZP = 1.0D0/ZDMY
      T1 = CF(1)*ZP
      S = T1
      IF (ZP.LT.WDTOL) GO TO 40
      ZSQ = ZP*ZP
      TST = T1*WDTOL
      DO 30 K=2,22
        ZP = ZP*ZSQ
        TRM = CF(K)*ZP
        IF (ABS(TRM).LT.TST) GO TO 40
        S = S + TRM
   30 CONTINUE
   40 CONTINUE
      IF (ZINC.NE.0.0D0) GO TO 50
      TLG = LOG(Z)
      DGAMLN = Z*(TLG-1.0D0) + 0.5D0*(CON-TLG) + S
      RETURN
   50 CONTINUE
      ZP = 1.0D0
      NZ = ZINC
      DO 60 I=1,NZ
        ZP = ZP*(Z+(I-1))
   60 CONTINUE
      TLG = LOG(ZDMY)
      DGAMLN = ZDMY*(TLG-1.0D0) - LOG(ZP) + 0.5D0*(CON-TLG) + S
      RETURN
C
C
   70 CONTINUE
      DGAMLN = D1MACH(2)
      IERR=1
      RETURN
      END
*DECK FDUMP
      SUBROUTINE FDUMP
C***BEGIN PROLOGUE  FDUMP
C***PURPOSE  Symbolic dump (should be locally written).
C***LIBRARY   SLATEC (XERROR)
C***CATEGORY  R3
C***TYPE      ALL (FDUMP-A)
C***KEYWORDS  ERROR, XERMSG
C***AUTHOR  Jones, R. E., (SNLA)
C***DESCRIPTION
```

```
C
C        ***Note*** Machine Dependent Routine
C        FDUMP is intended to be replaced by a locally written
C        version which produces a symbolic dump.  Failing this,
C        it should be replaced by a version which prints the
C        subprogram nesting list.  Note that this dump must be
C        printed on each of up to five files, as indicated by the
C        XGETUA routine.  See XSETUA and XGETUA for details.
C
C     Written by Ron Jones, with SLATEC Common Math Library Subcommittee
C
C***REFERENCES  (NONE)
C***ROUTINES CALLED  (NONE)
C***REVISION HISTORY  (YYMMDD)
C   790801  DATE WRITTEN
C   861211  REVISION DATE from Version 3.2
C   891214  Prologue converted to Version 4.0 format.  (BAB)
C***END PROLOGUE  FDUMP
C***FIRST EXECUTABLE STATEMENT  FDUMP
      RETURN
      END
*DECK J4SAVE
      FUNCTION J4SAVE (IWHICH, IVALUE, ISET)
C***BEGIN PROLOGUE  J4SAVE
C***SUBSIDIARY
C***PURPOSE  Save or recall global variables needed by error
C            handling routines.
C***LIBRARY   SLATEC (XERROR)
C***TYPE      INTEGER (J4SAVE-I)
C***KEYWORDS  ERROR MESSAGES, ERROR NUMBER, RECALL, SAVE, XERROR
C***AUTHOR  Jones, R. E., (SNLA)
C***DESCRIPTION
C
C     Abstract
C        J4SAVE saves and recalls several global variables needed
C        by the library error handling routines.
C
C     Description of Parameters
C      --Input--
C        IWHICH - Index of item desired.
C                = 1 Refers to current error number.
C                = 2 Refers to current error control flag.
C                = 3 Refers to current unit number to which error
C                    messages are to be sent.  (0 means use standard.)
C                = 4 Refers to the maximum number of times any
C                     message is to be printed (as set by XERMAX).
C                = 5 Refers to the total number of units to which
C                     each error message is to be written.
C                = 6 Refers to the 2nd unit for error messages
C                = 7 Refers to the 3rd unit for error messages
C                = 8 Refers to the 4th unit for error messages
C                = 9 Refers to the 5th unit for error messages
C        IVALUE - The value to be set for the IWHICH-th parameter,
C                 if ISET is .TRUE. .
C        ISET   - If ISET=.TRUE., the IWHICH-th parameter will BE
C                 given the value, IVALUE.  If ISET=.FALSE., the
C                 IWHICH-th parameter will be unchanged, and IVALUE
C                 is a dummy parameter.
```

```
C      --Output--
C        The (old) value of the IWHICH-th parameter will be returned
C        in the function value, J4SAVE.
C
C***SEE ALSO  XERMSG
C***REFERENCES  R. E. Jones and D. K. Kahaner, XERROR, the SLATEC
C                Error-handling Package, SAND82-0800, Sandia
C                Laboratories, 1982.
C***ROUTINES CALLED  (NONE)
C***REVISION HISTORY  (YYMMDD)
C   790801  DATE WRITTEN
C   891214  Prologue converted to Version 4.0 format.  (BAB)
C   900205  Minor modifications to prologue.  (WRB)
C   900402  Added TYPE section.  (WRB)
C   910411  Added KEYWORDS section.  (WRB)
C   920501  Reformatted the REFERENCES section.  (WRB)
C***END PROLOGUE  J4SAVE
      LOGICAL ISET
      INTEGER IPARAM(9)
      SAVE IPARAM
      DATA IPARAM(1),IPARAM(2),IPARAM(3),IPARAM(4)/0,2,0,10/
      DATA IPARAM(5)/1/
      DATA IPARAM(6),IPARAM(7),IPARAM(8),IPARAM(9)/0,0,0,0/
C***FIRST EXECUTABLE STATEMENT  J4SAVE
      J4SAVE = IPARAM(IWHICH)
      IF (ISET) IPARAM(IWHICH) = IVALUE
      RETURN
      END
*DECK XERCNT
      SUBROUTINE XERCNT (LIBRAR, SUBROU, MESSG, NERR, LEVEL, KONTRL)
C***BEGIN PROLOGUE  XERCNT
C***SUBSIDIARY
C***PURPOSE  Allow user control over handling of errors.
C***LIBRARY   SLATEC (XERROR)
C***CATEGORY  R3C
C***TYPE      ALL (XERCNT-A)
C***KEYWORDS  ERROR, XERROR
C***AUTHOR  Jones, R. E., (SNLA)
C***DESCRIPTION
C
C     Abstract
C        Allows user control over handling of individual errors.
C        Just after each message is recorded, but before it is
C        processed any further (i.e., before it is printed or
C        a decision to abort is made), a call is made to XERCNT.
C        If the user has provided his own version of XERCNT, he
C        can then override the value of KONTROL used in processing
C        this message by redefining its value.
C        KONTRL may be set to any value from -2 to 2.
C        The meanings for KONTRL are the same as in XSETF, except
C        that the value of KONTRL changes only for this message.
C        If KONTRL is set to a value outside the range from -2 to 2,
C        it will be moved back into that range.
C
C     Description of Parameters
C
C      --Input--
C        LIBRAR - the library that the routine is in.
```

```
C        SUBROU - the subroutine that XERMSG is being called from
C        MESSG  - the first 20 characters of the error message.
C        NERR   - same as in the call to XERMSG.
C        LEVEL  - same as in the call to XERMSG.
C        KONTRL - the current value of the control flag as set
C                 by a call to XSETF.
C
C      --Output--
C        KONTRL - the new value of KONTRL.  If KONTRL is not
C                 defined, it will remain at its original value.
C                 This changed value of control affects only
C                 the current occurrence of the current message.
C
C***REFERENCES  R. E. Jones and D. K. Kahaner, XERROR, the SLATEC
C                 Error-handling Package, SAND82-0800, Sandia
C                 Laboratories, 1982.
C***ROUTINES CALLED  (NONE)
C***REVISION HISTORY  (YYMMDD)
C   790801  DATE WRITTEN
C   861211  REVISION DATE from Version 3.2
C   891214  Prologue converted to Version 4.0 format.  (BAB)
C   900206  Routine changed from user-callable to subsidiary.  (WRB)
C   900510  Changed calling sequence to include LIBRARY and SUBROUTINE
C           names, changed routine name from XERCTL to XERCNT.  (RWC)
C   920501  Reformatted the REFERENCES section.  (WRB)
C***END PROLOGUE  XERCNT
      CHARACTER*(*) LIBRAR, SUBROU, MESSG
C***FIRST EXECUTABLE STATEMENT  XERCNT
      RETURN
      END
*DECK XERHLT
      SUBROUTINE XERHLT (MESSG)
C***BEGIN PROLOGUE  XERHLT
C***SUBSIDIARY
C***PURPOSE  Abort program execution and print error message.
C***LIBRARY   SLATEC (XERROR)
C***CATEGORY  R3C
C***TYPE      ALL (XERHLT-A)
C***KEYWORDS  ABORT PROGRAM EXECUTION, ERROR, XERROR
C***AUTHOR  Jones, R. E., (SNLA)
C***DESCRIPTION
C
C     Abstract
C        ***Note*** machine dependent routine
C        XERHLT aborts the execution of the program.
C        The error message causing the abort is given in the calling
C        sequence, in case one needs it for printing on a dayfile,
C        for example.
C
C     Description of Parameters
C        MESSG is as in XERMSG.
C
C***REFERENCES  R. E. Jones and D. K. Kahaner, XERROR, the SLATEC
C                 Error-handling Package, SAND82-0800, Sandia
C                 Laboratories, 1982.
C***ROUTINES CALLED  (NONE)
C***REVISION HISTORY  (YYMMDD)
C   790801  DATE WRITTEN
```

```
C   861211  REVISION DATE from Version 3.2
C   891214  Prologue converted to Version 4.0 format.  (BAB)
C   900206  Routine changed from user-callable to subsidiary.  (WRB)
C   900510  Changed calling sequence to delete length of character
C           and changed routine name from XERABT to XERHLT.  (RWC)
C   920501  Reformatted the REFERENCES section.  (WRB)
C***END PROLOGUE  XERHLT
      CHARACTER*(*) MESSG
C***FIRST EXECUTABLE STATEMENT  XERHLT
      STOP
      END
*DECK ZAIRY
      SUBROUTINE ZAIRY (ZR, ZI, ID, KODE, AIR, AII, NZ, IERR)
C***BEGIN PROLOGUE  ZAIRY
C***PURPOSE  Compute the Airy function Ai(z) or its derivative dAi/dz
C            for complex argument z.  A scaling option is available
C            to help avoid underflow and overflow.
C***LIBRARY   SLATEC
C***CATEGORY  C10D
C***TYPE      COMPLEX (CAIRY-C, ZAIRY-C)
C***KEYWORDS  AIRY FUNCTION, BESSEL FUNCTION OF ORDER ONE THIRD,
C             BESSEL FUNCTION OF ORDER TWO THIRDS
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C                      ***A DOUBLE PRECISION ROUTINE***
C         On KODE=1, ZAIRY computes the complex Airy function Ai(z)
C         or its derivative dAi/dz on ID=0 or ID=1 respectively. On
C         KODE=2, a scaling option exp(zeta)*Ai(z) or exp(zeta)*dAi/dz
C         is provided to remove the exponential decay in -pi/3<arg(z)
C         <pi/3 and the exponential growth in pi/3<abs(arg(z))<pi where
C         zeta=(2/3)*z**(3/2).
C
C         While the Airy functions Ai(z) and dAi/dz are analytic in
C         the whole z-plane, the corresponding scaled functions defined
C         for KODE=2 have a cut along the negative real axis.
C
C         Input
C           ZR     - DOUBLE PRECISION real part of argument Z
C           ZI     - DOUBLE PRECISION imag part of argument Z
C           ID     - Order of derivative, ID=0 or ID=1
C           KODE   - A parameter to indicate the scaling option
C                    KODE=1  returns
C                            AI=Ai(z)  on ID=0
C                            AI=dAi/dz on ID=1
C                            at z=Z
C                        =2  returns
C                            AI=exp(zeta)*Ai(z)  on ID=0
C                            AI=exp(zeta)*dAi/dz on ID=1
C                            at z=Z where zeta=(2/3)*z**(3/2)
C
C         Output
C           AIR    - DOUBLE PRECISION real part of result
C           AII    - DOUBLE PRECISION imag part of result
C           NZ     - Underflow indicator
C                    NZ=0    Normal return
C                    NZ=1    AI=0 due to underflow in
C                            -pi/3<arg(Z)<pi/3 on KODE=1
```

```
C           IERR   - Error flag
C                    IERR=0  Normal return     - COMPUTATION COMPLETED
C                    IERR=1  Input error       - NO COMPUTATION
C                    IERR=2  Overflow          - NO COMPUTATION
C                            (Re(Z) too large with KODE=1)
C                    IERR=3  Precision warning - COMPUTATION COMPLETED
C                            (Result has less than half precision)
C                    IERR=4  Precision error   - NO COMPUTATION
C                            (Result has no precision)
C                    IERR=5  Algorithmic error - NO COMPUTATION
C                            (Termination condition not met)
C
C *Long Description:
C
C         Ai(z) and dAi/dz are computed from K Bessel functions by
C
C                Ai(z) =  c*sqrt(z)*K(1/3,zeta)
C               dAi/dz = -c*   z   *K(2/3,zeta)
C                    c =  1/(pi*sqrt(3))
C                 zeta =  (2/3)*z**(3/2)
C
C         when abs(z)>1 and from power series when abs(z)<=1.
C
C         In most complex variable computation, one must evaluate ele-
C         mentary functions.  When the magnitude of Z is large, losses
C         of significance by argument reduction occur.  Consequently, if
C         the magnitude of ZETA=(2/3)*Z**(3/2) exceeds U1=SQRT(0.5/UR),
C         then losses exceeding half precision are likely and an error
C         flag IERR=3 is triggered where UR=MAX(D1MACH(4),1.0D-18) is
C         double precision unit roundoff limited to 18 digits precision.
C         Also, if the magnitude of ZETA is larger than U2=0.5/UR, then
C         all significance is lost and IERR=4.  In order to use the INT
C         function, ZETA must be further restricted not to exceed
C         U3=I1MACH(9)=LARGEST INTEGER.  Thus, the magnitude of ZETA
C         must be restricted by MIN(U2,U3).  In IEEE arithmetic, U1,U2,
C         and U3 are approximately 2.0E+3, 4.2E+6, 2.1E+9 in single
C         precision and 4.7E+7, 2.3E+15, 2.1E+9 in double precision.
C         This makes U2 limiting is single precision and U3 limiting
C         in double precision.  This means that the magnitude of Z
C         cannot exceed approximately 3.4E+4 in single precision and
C         2.1E+6 in double precision.  This also means that one can
C         expect to retain, in the worst cases on 32-bit machines,
C         no digits in single precision and only 6 digits in double
C         precision.
C
C         The approximate relative error in the magnitude of a complex
C         Bessel function can be expressed as P*10**S where P=MAX(UNIT
C         ROUNDOFF,1.0E-18) is the nominal precision and 10**S repre-
C         sents the increase in error due to argument reduction in the
C         elementary functions.  Here, S=MAX(1,ABS(LOG10(ABS(Z))),
C         ABS(LOG10(FNU))) approximately (i.e., S=MAX(1,ABS(EXPONENT OF
C         ABS(Z),ABS(EXPONENT OF FNU)) ).  However, the phase angle may
C         have only absolute accuracy.  This is most likely to occur
C         when one component (in magnitude) is larger than the other by
C         several orders of magnitude.  If one component is 10**K larger
C         than the other, then one can expect only MAX(ABS(LOG10(P))-K,
C         0) significant digits; or, stated another way, when K exceeds
C         the exponent of P, no significant digits remain in the smaller
```

```
C         component.  However, the phase angle retains absolute accuracy
C         because, in complex arithmetic with precision P, the smaller
C         component will not (as a rule) decrease below P times the
C         magnitude of the larger component. In these extreme cases,
C         the principal phase angle is on the order of +P, -P, PI/2-P,
C         or -PI/2+P.
C
C***REFERENCES  1. M. Abramowitz and I. A. Stegun, Handbook of Mathe-
C                  matical Functions, National Bureau of Standards
C                  Applied Mathematics Series 55, U. S. Department
C                  of Commerce, Tenth Printing (1972) or later.
C               2. D. E. Amos, Computation of Bessel Functions of
C                  Complex Argument and Large Order, Report SAND83-0643,
C                  Sandia National Laboratories, Albuquerque, NM, May
C                  1983.
C               3. D. E. Amos, A Subroutine Package for Bessel Functions
C                  of a Complex Argument and Nonnegative Order, Report
C                  SAND85-1018, Sandia National Laboratory, Albuquerque,
C                  NM, May 1985.
C               4. D. E. Amos, A portable package for Bessel functions
C                  of a complex argument and nonnegative order, ACM
C                  Transactions on Mathematical Software, 12 (September
C                  1986), pp. 265-273.
C
C***ROUTINES CALLED  D1MACH, I1MACH, ZABS, ZACAI, ZBKNU, ZEXP, ZSQRT
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   890801  REVISION DATE from Version 3.2
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C   920128  Category corrected.  (WRB)
C   920811  Prologue revised.  (DWL)
C   930122  Added ZEXP and ZSQRT to EXTERNAL statement.  (RWC)
C***END PROLOGUE  ZAIRY
C     COMPLEX AI,CONE,CSQ,CY,S1,S2,TRM1,TRM2,Z,ZTA,Z3
      DOUBLE PRECISION AA, AD, AII, AIR, AK, ALIM, ATRM, AZ, AZ3, BK,
     * CC, CK, COEF, CONEI, CONER, CSQI, CSQR, CYI, CYR, C1, C2, DIG,
     * DK, D1, D2, ELIM, FID, FNU, PTR, RL, R1M5, SFAC, STI, STR,
     * S1I, S1R, S2I, S2R, TOL, TRM1I, TRM1R, TRM2I, TRM2R, TTH, ZEROI,
     * ZEROR, ZI, ZR, ZTAI, ZTAR, Z3I, Z3R, D1MACH, ZABS, ALAZ, BB
      INTEGER ID, IERR, IFLAG, K, KODE, K1, K2, MR, NN, NZ, I1MACH
      DIMENSION CYR(1), CYI(1)
      EXTERNAL ZABS, ZEXP, ZSQRT
      DATA TTH, C1, C2, COEF /6.6666666666666667D-01,
     * 3.55028053887817240D-01,2.5881940379280679 9D-01,
     * 1.83776298473930683D-01/
      DATA ZEROR, ZEROI, CONER, CONEI /0.0D0,0.0D0,1.0D0,0.0D0/
C***FIRST EXECUTABLE STATEMENT  ZAIRY
      IERR = 0
      NZ=0
      IF (ID.LT.0 .OR. ID.GT.1) IERR=1
      IF (KODE.LT.1 .OR. KODE.GT.2) IERR=1
      IF (IERR.NE.0) RETURN
      AZ = ZABS(ZR,ZI)
      TOL = MAX(D1MACH(4),1.0D-18)
      FID = ID
      IF (AZ.GT.1.0D0) GO TO 70
C-----------------------------------------------------------------------
C     POWER SERIES FOR ABS(Z).LE.1.
```

```
C-----------------------------------------------------------------------
      S1R = CONER
      S1I = CONEI
      S2R = CONER
      S2I = CONEI
      IF (AZ.LT.TOL) GO TO 170
      AA = AZ*AZ
      IF (AA.LT.TOL/AZ) GO TO 40
      TRM1R = CONER
      TRM1I = CONEI
      TRM2R = CONER
      TRM2I = CONEI
      ATRM = 1.0D0
      STR = ZR*ZR - ZI*ZI
      STI = ZR*ZI + ZI*ZR
      Z3R = STR*ZR - STI*ZI
      Z3I = STR*ZI + STI*ZR
      AZ3 = AZ*AA
      AK = 2.0D0 + FID
      BK = 3.0D0 - FID - FID
      CK = 4.0D0 - FID
      DK = 3.0D0 + FID + FID
      D1 = AK*DK
      D2 = BK*CK
      AD = MIN(D1,D2)
      AK = 24.0D0 + 9.0D0*FID
      BK = 30.0D0 - 9.0D0*FID
      DO 30 K=1,25
        STR = (TRM1R*Z3R-TRM1I*Z3I)/D1
        TRM1I = (TRM1R*Z3I+TRM1I*Z3R)/D1
        TRM1R = STR
        S1R = S1R + TRM1R
        S1I = S1I + TRM1I
        STR = (TRM2R*Z3R-TRM2I*Z3I)/D2
        TRM2I = (TRM2R*Z3I+TRM2I*Z3R)/D2
        TRM2R = STR
        S2R = S2R + TRM2R
        S2I = S2I + TRM2I
        ATRM = ATRM*AZ3/AD
        D1 = D1 + AK
        D2 = D2 + BK
        AD = MIN(D1,D2)
        IF (ATRM.LT.TOL*AD) GO TO 40
        AK = AK + 18.0D0
        BK = BK + 18.0D0
   30 CONTINUE
   40 CONTINUE
      IF (ID.EQ.1) GO TO 50
      AIR = S1R*C1 - C2*(ZR*S2R-ZI*S2I)
      AII = S1I*C1 - C2*(ZR*S2I+ZI*S2R)
      IF (KODE.EQ.1) RETURN
      CALL ZSQRT(ZR, ZI, STR, STI)
      ZTAR = TTH*(ZR*STR-ZI*STI)
      ZTAI = TTH*(ZR*STI+ZI*STR)
      CALL ZEXP(ZTAR, ZTAI, STR, STI)
      PTR = AIR*STR - AII*STI
      AII = AIR*STI + AII*STR
      AIR = PTR
```

```
      RETURN
   50 CONTINUE
      AIR = -S2R*C2
      AII = -S2I*C2
      IF (AZ.LE.TOL) GO TO 60
      STR = ZR*S1R - ZI*S1I
      STI = ZR*S1I + ZI*S1R
      CC = C1/(1.0D0+FID)
      AIR = AIR + CC*(STR*ZR-STI*ZI)
      AII = AII + CC*(STR*ZI+STI*ZR)
   60 CONTINUE
      IF (KODE.EQ.1) RETURN
      CALL ZSQRT(ZR, ZI, STR, STI)
      ZTAR = TTH*(ZR*STR-ZI*STI)
      ZTAI = TTH*(ZR*STI+ZI*STR)
      CALL ZEXP(ZTAR, ZTAI, STR, STI)
      PTR = STR*AIR - STI*AII
      AII = STR*AII + STI*AIR
      AIR = PTR
      RETURN
C-----------------------------------------------------------------------
C     CASE FOR ABS(Z).GT.1.0
C-----------------------------------------------------------------------
   70 CONTINUE
      FNU = (1.0D0+FID)/3.0D0
C-----------------------------------------------------------------------
C     SET PARAMETERS RELATED TO MACHINE CONSTANTS.
C     TOL IS THE APPROXIMATE UNIT ROUNDOFF LIMITED TO 1.0D-18.
C     ELIM IS THE APPROXIMATE EXPONENTIAL OVER- AND UNDERFLOW LIMIT.
C     EXP(-ELIM).LT.EXP(-ALIM)=EXP(-ELIM)/TOL    AND
C     EXP(ELIM).GT.EXP(ALIM)=EXP(ELIM)*TOL       ARE INTERVALS NEAR
C     UNDERFLOW AND OVERFLOW LIMITS WHERE SCALED ARITHMETIC IS DONE.
C     RL IS THE LOWER BOUNDARY OF THE ASYMPTOTIC EXPANSION FOR LARGE Z.
C     DIG = NUMBER OF BASE 10 DIGITS IN TOL = 10**(-DIG).
C-----------------------------------------------------------------------
      K1 = I1MACH(15)
      K2 = I1MACH(16)
      R1M5 = D1MACH(5)
      K = MIN(ABS(K1),ABS(K2))
      ELIM = 2.303D0*(K*R1M5-3.0D0)
      K1 = I1MACH(14) - 1
      AA = R1M5*K1
      DIG = MIN(AA,18.0D0)
      AA = AA*2.303D0
      ALIM = ELIM + MAX(-AA,-41.45D0)
      RL = 1.2D0*DIG + 3.0D0
      ALAZ = LOG(AZ)
C-----------------------------------------------------------------------
C     TEST FOR PROPER RANGE
C-----------------------------------------------------------------------
      AA=0.5D0/TOL
      BB=I1MACH(9)*0.5D0
      AA=MIN(AA,BB)
      AA=AA**TTH
      IF (AZ.GT.AA) GO TO 260
      AA=SQRT(AA)
      IF (AZ.GT.AA) IERR=3
      CALL ZSQRT(ZR, ZI, CSQR, CSQI)
```

```
      ZTAR = TTH*(ZR*CSQR-ZI*CSQI)
      ZTAI = TTH*(ZR*CSQI+ZI*CSQR)
C-----------------------------------------------------------------------
C     RE(ZTA).LE.0 WHEN RE(Z).LT.0, ESPECIALLY WHEN IM(Z) IS SMALL
C-----------------------------------------------------------------------
      IFLAG = 0
      SFAC = 1.0D0
      AK = ZTAI
      IF (ZR.GE.0.0D0) GO TO 80
      BK = ZTAR
      CK = -ABS(BK)
      ZTAR = CK
      ZTAI = AK
   80 CONTINUE
      IF (ZI.NE.0.0D0) GO TO 90
      IF (ZR.GT.0.0D0) GO TO 90
      ZTAR = 0.0D0
      ZTAI = AK
   90 CONTINUE
      AA = ZTAR
      IF (AA.GE.0.0D0 .AND. ZR.GT.0.0D0) GO TO 110
      IF (KODE.EQ.2) GO TO 100
C-----------------------------------------------------------------------
C     OVERFLOW TEST
C-----------------------------------------------------------------------
      IF (AA.GT.(-ALIM)) GO TO 100
      AA = -AA + 0.25D0*ALAZ
      IFLAG = 1
      SFAC = TOL
      IF (AA.GT.ELIM) GO TO 270
  100 CONTINUE
C-----------------------------------------------------------------------
C     CBKNU AND CACON RETURN EXP(ZTA)*K(FNU,ZTA) ON KODE=2
C-----------------------------------------------------------------------
      MR = 1
      IF (ZI.LT.0.0D0) MR = -1
      CALL ZACAI(ZTAR, ZTAI, FNU, KODE, MR, 1, CYR, CYI, NN, RL, TOL,
     * ELIM, ALIM)
      IF (NN.LT.0) GO TO 280
      NZ = NZ + NN
      GO TO 130
  110 CONTINUE
      IF (KODE.EQ.2) GO TO 120
C-----------------------------------------------------------------------
C     UNDERFLOW TEST
C-----------------------------------------------------------------------
      IF (AA.LT.ALIM) GO TO 120
      AA = -AA - 0.25D0*ALAZ
      IFLAG = 2
      SFAC = 1.0D0/TOL
      IF (AA.LT.(-ELIM)) GO TO 210
  120 CONTINUE
      CALL ZBKNU(ZTAR, ZTAI, FNU, KODE, 1, CYR, CYI, NZ, TOL, ELIM,
     * ALIM)
  130 CONTINUE
      S1R = CYR(1)*COEF
      S1I = CYI(1)*COEF
      IF (IFLAG.NE.0) GO TO 150
```

```
      IF (ID.EQ.1) GO TO 140
      AIR = CSQR*S1R - CSQI*S1I
      AII = CSQR*S1I + CSQI*S1R
      RETURN
  140 CONTINUE
      AIR = -(ZR*S1R-ZI*S1I)
      AII = -(ZR*S1I+ZI*S1R)
      RETURN
  150 CONTINUE
      S1R = S1R*SFAC
      S1I = S1I*SFAC
      IF (ID.EQ.1) GO TO 160
      STR = S1R*CSQR - S1I*CSQI
      S1I = S1R*CSQI + S1I*CSQR
      S1R = STR
      AIR = S1R/SFAC
      AII = S1I/SFAC
      RETURN
  160 CONTINUE
      STR = -(S1R*ZR-S1I*ZI)
      S1I = -(S1R*ZI+S1I*ZR)
      S1R = STR
      AIR = S1R/SFAC
      AII = S1I/SFAC
      RETURN
  170 CONTINUE
      AA = 1.0D+3*D1MACH(1)
      S1R = ZEROR
      S1I = ZEROI
      IF (ID.EQ.1) GO TO 190
      IF (AZ.LE.AA) GO TO 180
      S1R = C2*ZR
      S1I = C2*ZI
  180 CONTINUE
      AIR = C1 - S1R
      AII = -S1I
      RETURN
  190 CONTINUE
      AIR = -C2
      AII = 0.0D0
      AA = SQRT(AA)
      IF (AZ.LE.AA) GO TO 200
      S1R = 0.5D0*(ZR*ZR-ZI*ZI)
      S1I = ZR*ZI
  200 CONTINUE
      AIR = AIR + C1*S1R
      AII = AII + C1*S1I
      RETURN
  210 CONTINUE
      NZ = 1
      AIR = ZEROR
      AII = ZEROI
      RETURN
  270 CONTINUE
      NZ = 0
      IERR=2
      RETURN
  280 CONTINUE
```

```
      IF(NN.EQ.(-1)) GO TO 270
      NZ=0
      IERR=5
      RETURN
  260 CONTINUE
      IERR=4
      NZ=0
      RETURN
      END
*DECK ZASYI
      SUBROUTINE ZASYI (ZR, ZI, FNU, KODE, N, YR, YI, NZ, RL, TOL, ELIM,
     +   ALIM)
C***BEGIN PROLOGUE  ZASYI
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZBESI and ZBESK
C***LIBRARY   SLATEC
C***TYPE      ALL (CASYI-A, ZASYI-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     ZASYI COMPUTES THE I BESSEL FUNCTION FOR REAL(Z).GE.0.0 BY
C     MEANS OF THE ASYMPTOTIC EXPANSION FOR LARGE ABS(Z) IN THE
C     REGION ABS(Z).GT.MAX(RL,FNU*FNU/2). NZ=0 IS A NORMAL RETURN.
C     NZ.LT.0 INDICATES AN OVERFLOW ON KODE=1.
C
C***SEE ALSO  ZBESI, ZBESK
C***ROUTINES CALLED  D1MACH, ZABS, ZDIV, ZEXP, ZMLT, ZSQRT
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C   930122  Added ZEXP and ZSQRT to EXTERNAL statement.  (RWC)
C***END PROLOGUE  ZASYI
C     COMPLEX AK1,CK,CONE,CS1,CS2,CZ,CZERO,DK,EZ,P1,RZ,S2,Y,Z
      DOUBLE PRECISION AA, AEZ, AK, AK1I, AK1R, ALIM, ARG, ARM, ATOL,
     * AZ, BB, BK, CKI, CKR, CONEI, CONER, CS1I, CS1R, CS2I, CS2R, CZI,
     * CZR, DFNU, DKI, DKR, DNU2, ELIM, EZI, EZR, FDN, FNU, PI, P1I,
     * P1R, RAZ, RL, RTPI, RTR1, RZI, RZR, S, SGN, SQK, STI, STR, S2I,
     * S2R, TOL, TZI, TZR, YI, YR, ZEROI, ZEROR, ZI, ZR, D1MACH, ZABS
      INTEGER I, IB, IL, INU, J, JL, K, KODE, KODED, M, N, NN, NZ
      DIMENSION YR(N), YI(N)
      EXTERNAL ZABS, ZEXP, ZSQRT
      DATA PI, RTPI  /3.14159265358979324D0 , 0.15915494309189536D0 /
      DATA ZEROR,ZEROI,CONER,CONEI / 0.0D0, 0.0D0, 1.0D0, 0.0D0 /
C***FIRST EXECUTABLE STATEMENT  ZASYI
      NZ = 0
      AZ = ZABS(ZR,ZI)
      ARM = 1.0D+3*D1MACH(1)
      RTR1 = SQRT(ARM)
      IL = MIN(2,N)
      DFNU = FNU + (N-IL)
C-----------------------------------------------------------------------
C     OVERFLOW TEST
C-----------------------------------------------------------------------
      RAZ = 1.0D0/AZ
      STR = ZR*RAZ
      STI = -ZI*RAZ
      AK1R = RTPI*STR*RAZ
      AK1I = RTPI*STI*RAZ
```

```
      CALL ZSQRT(AK1R, AK1I, AK1R, AK1I)
      CZR = ZR
      CZI = ZI
      IF (KODE.NE.2) GO TO 10
      CZR = ZEROR
      CZI = ZI
   10 CONTINUE
      IF (ABS(CZR).GT.ELIM) GO TO 100
      DNU2 = DFNU + DFNU
      KODED = 1
      IF ((ABS(CZR).GT.ALIM) .AND. (N.GT.2)) GO TO 20
      KODED = 0
      CALL ZEXP(CZR, CZI, STR, STI)
      CALL ZMLT(AK1R, AK1I, STR, STI, AK1R, AK1I)
   20 CONTINUE
      FDN = 0.0D0
      IF (DNU2.GT.RTR1) FDN = DNU2*DNU2
      EZR = ZR*8.0D0
      EZI = ZI*8.0D0
C-----------------------------------------------------------------------
C     WHEN Z IS IMAGINARY, THE ERROR TEST MUST BE MADE RELATIVE TO THE
C     FIRST RECIPROCAL POWER SINCE THIS IS THE LEADING TERM OF THE
C     EXPANSION FOR THE IMAGINARY PART.
C-----------------------------------------------------------------------
      AEZ = 8.0D0*AZ
      S = TOL/AEZ
      JL = RL+RL + 2
      P1R = ZEROR
      P1I = ZEROI
      IF (ZI.EQ.0.0D0) GO TO 30
C-----------------------------------------------------------------------
C     CALCULATE EXP(PI*(0.5+FNU+N-IL)*I) TO MINIMIZE LOSSES OF
C     SIGNIFICANCE WHEN FNU OR N IS LARGE
C-----------------------------------------------------------------------
      INU = FNU
      ARG = (FNU-INU)*PI
      INU = INU + N - IL
      AK = -SIN(ARG)
      BK = COS(ARG)
      IF (ZI.LT.0.0D0) BK = -BK
      P1R = AK
      P1I = BK
      IF (MOD(INU,2).EQ.0) GO TO 30
      P1R = -P1R
      P1I = -P1I
   30 CONTINUE
      DO 70 K=1,IL
        SQK = FDN - 1.0D0
        ATOL = S*ABS(SQK)
        SGN = 1.0D0
        CS1R = CONER
        CS1I = CONEI
        CS2R = CONER
        CS2I = CONEI
        CKR = CONER
        CKI = CONEI
        AK = 0.0D0
        AA = 1.0D0
```

```
      BB = AEZ
      DKR = EZR
      DKI = EZI
      DO 40 J=1,JL
        CALL ZDIV(CKR, CKI, DKR, DKI, STR, STI)
        CKR = STR*SQK
        CKI = STI*SQK
        CS2R = CS2R + CKR
        CS2I = CS2I + CKI
        SGN = -SGN
        CS1R = CS1R + CKR*SGN
        CS1I = CS1I + CKI*SGN
        DKR = DKR + EZR
        DKI = DKI + EZI
        AA = AA*ABS(SQK)/BB
        BB = BB + AEZ
        AK = AK + 8.0D0
        SQK = SQK - AK
        IF (AA.LE.ATOL) GO TO 50
40    CONTINUE
      GO TO 110
50    CONTINUE
      S2R = CS1R
      S2I = CS1I
      IF (ZR+ZR.GE.ELIM) GO TO 60
      TZR = ZR + ZR
      TZI = ZI + ZI
      CALL ZEXP(-TZR, -TZI, STR, STI)
      CALL ZMLT(STR, STI, P1R, P1I, STR, STI)
      CALL ZMLT(STR, STI, CS2R, CS2I, STR, STI)
      S2R = S2R + STR
      S2I = S2I + STI
60    CONTINUE
      FDN = FDN + 8.0D0*DFNU + 4.0D0
      P1R = -P1R
      P1I = -P1I
      M = N - IL + K
      YR(M) = S2R*AK1R - S2I*AK1I
      YI(M) = S2R*AK1I + S2I*AK1R
70  CONTINUE
    IF (N.LE.2) RETURN
    NN = N
    K = NN - 2
    AK = K
    STR = ZR*RAZ
    STI = -ZI*RAZ
    RZR = (STR+STR)*RAZ
    RZI = (STI+STI)*RAZ
    IB = 3
    DO 80 I=IB,NN
      YR(K) = (AK+FNU)*(RZR*YR(K+1)-RZI*YI(K+1)) + YR(K+2)
      YI(K) = (AK+FNU)*(RZR*YI(K+1)+RZI*YR(K+1)) + YI(K+2)
      AK = AK - 1.0D0
      K = K - 1
80  CONTINUE
    IF (KODED.EQ.0) RETURN
    CALL ZEXP(CZR, CZI, CKR, CKI)
    DO 90 I=1,NN
```

```
        STR = YR(I)*CKR - YI(I)*CKI
        YI(I) = YR(I)*CKI + YI(I)*CKR
        YR(I) = STR
   90 CONTINUE
      RETURN
  100 CONTINUE
      NZ = -1
      RETURN
  110 CONTINUE
      NZ=-2
      RETURN
      END
*DECK ZBKNU
      SUBROUTINE ZBKNU (ZR, ZI, FNU, KODE, N, YR, YI, NZ, TOL, ELIM,
     +   ALIM)
C***BEGIN PROLOGUE  ZBKNU
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZAIRY, ZBESH, ZBESI and ZBESK
C***LIBRARY   SLATEC
C***TYPE      ALL (CBKNU-A, ZBKNU-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     ZBKNU COMPUTES THE K BESSEL FUNCTION IN THE RIGHT HALF Z PLANE.
C
C***SEE ALSO  ZAIRY, ZBESH, ZBESI, ZBESK
C***ROUTINES CALLED  D1MACH, DGAMLN, I1MACH, ZABS, ZDIV, ZEXP, ZKSCL,
C                    ZLOG, ZMLT, ZSHCH, ZSQRT, ZUCHK
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C   930122  Added ZEXP, ZLOG and ZSQRT to EXTERNAL statement.  (RWC)
C***END PROLOGUE  ZBKNU
C
      DOUBLE PRECISION AA, AK, ALIM, ASCLE, A1, A2, BB, BK, BRY, CAZ,
     * CBI, CBR, CC, CCHI, CCHR, CKI, CKR, COEFI, COEFR, CONEI, CONER,
     * CRSCR, CSCLR, CSHI, CSHR, CSI, CSR, CSRR, CSSR, CTWOR,
     * CZEROI, CZEROR, CZI, CZR, DNU, DNU2, DPI, ELIM, ETEST, FC, FHS,
     * FI, FK, FKS, FMUI, FMUR, FNU, FPI, FR, G1, G2, HPI, PI, PR, PTI,
     * PTR, P1I, P1R, P2I, P2M, P2R, QI, QR, RAK, RCAZ, RTHPI, RZI,
     * RZR, R1, S, SMUI, SMUR, SPI, STI, STR, S1I, S1R, S2I, S2R, TM,
     * TOL, TTH, T1, T2, YI, YR, ZI, ZR, DGAMLN, D1MACH, ZABS, ELM,
     * CELMR, ZDR, ZDI, AS, ALAS, HELIM, CYR, CYI
      INTEGER I, IFLAG, INU, K, KFLAG, KK, KMAX, KODE, KODED, N, NZ,
     * IDUM, I1MACH, J, IC, INUB, NW
      DIMENSION YR(N), YI(N), CC(8), CSSR(3), CSRR(3), BRY(3), CYR(2),
     * CYI(2)
      EXTERNAL ZABS, ZEXP, ZLOG, ZSQRT
C     COMPLEX Z,Y,A,B,RZ,SMU,FU,FMU,F,FLRZ,CZ,S1,S2,CSH,CCH
C     COMPLEX CK,P,Q,COEF,P1,P2,CBK,PT,CZERO,CONE,CTWO,ST,EZ,CS,DK
C
      DATA KMAX / 30 /
      DATA CZEROR,CZEROI,CONER,CONEI,CTWOR,R1/
     1  0.0D0 , 0.0D0 , 1.0D0 , 0.0D0 , 2.0D0 , 2.0D0 /
      DATA DPI, RTHPI, SPI ,HPI, FPI, TTH /
     1     3.14159265358979324D0,       1.2533141373155002 5D0,
     2     1.90985931710274403D0,       1.57079632679489662D0,
     3     1.89769999331517738D0,       6.66666666666666666D-01/
```

```
      DATA CC(1), CC(2), CC(3), CC(4), CC(5), CC(6), CC(7), CC(8)/
     1     5.77215664901532861D-01,    -4.20026350340952355D-02,
     2    -4.21977345555443367D-02,     7.21894324666309954D-03,
     3    -2.15241674114950973D-04,    -2.01348547807882387D-05,
     4     1.13302723198169588D-06,     6.11609510448141582D-09/
C***FIRST EXECUTABLE STATEMENT  ZBKNU
      CAZ = ZABS(ZR,ZI)
      CSCLR = 1.0D0/TOL
      CRSCR = TOL
      CSSR(1) = CSCLR
      CSSR(2) = 1.0D0
      CSSR(3) = CRSCR
      CSRR(1) = CRSCR
      CSRR(2) = 1.0D0
      CSRR(3) = CSCLR
      BRY(1) = 1.0D+3*D1MACH(1)/TOL
      BRY(2) = 1.0D0/BRY(1)
      BRY(3) = D1MACH(2)
      NZ = 0
      IFLAG = 0
      KODED = KODE
      RCAZ = 1.0D0/CAZ
      STR = ZR*RCAZ
      STI = -ZI*RCAZ
      RZR = (STR+STR)*RCAZ
      RZI = (STI+STI)*RCAZ
      INU = FNU+0.5D0
      DNU = FNU - INU
      IF (ABS(DNU).EQ.0.5D0) GO TO 110
      DNU2 = 0.0D0
      IF (ABS(DNU).GT.TOL) DNU2 = DNU*DNU
      IF (CAZ.GT.R1) GO TO 110
C-----------------------------------------------------------------------
C     SERIES FOR ABS(Z).LE.R1
C-----------------------------------------------------------------------
      FC = 1.0D0
      CALL ZLOG(RZR, RZI, SMUR, SMUI, IDUM)
      FMUR = SMUR*DNU
      FMUI = SMUI*DNU
      CALL ZSHCH(FMUR, FMUI, CSHR, CSHI, CCHR, CCHI)
      IF (DNU.EQ.0.0D0) GO TO 10
      FC = DNU*DPI
      FC = FC/SIN(FC)
      SMUR = CSHR/DNU
      SMUI = CSHI/DNU
   10 CONTINUE
      A2 = 1.0D0 + DNU
C-----------------------------------------------------------------------
C     GAM(1-Z)*GAM(1+Z)=PI*Z/SIN(PI*Z), T1=1/GAM(1-DNU), T2=1/GAM(1+DNU)
C-----------------------------------------------------------------------
      T2 = EXP(-DGAMLN(A2,IDUM))
      T1 = 1.0D0/(T2*FC)
      IF (ABS(DNU).GT.0.1D0) GO TO 40
C-----------------------------------------------------------------------
C     SERIES FOR F0 TO RESOLVE INDETERMINACY FOR SMALL ABS(DNU)
C-----------------------------------------------------------------------
      AK = 1.0D0
      S = CC(1)
```

```
      DO 20 K=2,8
        AK = AK*DNU2
        TM = CC(K)*AK
        S = S + TM
        IF (ABS(TM).LT.TOL) GO TO 30
   20 CONTINUE
   30 G1 = -S
      GO TO 50
   40 CONTINUE
      G1 = (T1-T2)/(DNU+DNU)
   50 CONTINUE
      G2 = (T1+T2)*0.5D0
      FR = FC*(CCHR*G1+SMUR*G2)
      FI = FC*(CCHI*G1+SMUI*G2)
      CALL ZEXP(FMUR, FMUI, STR, STI)
      PR = 0.5D0*STR/T2
      PI = 0.5D0*STI/T2
      CALL ZDIV(0.5D0, 0.0D0, STR, STI, PTR, PTI)
      QR = PTR/T1
      QI = PTI/T1
      S1R = FR
      S1I = FI
      S2R = PR
      S2I = PI
      AK = 1.0D0
      A1 = 1.0D0
      CKR = CONER
      CKI = CONEI
      BK = 1.0D0 - DNU2
      IF (INU.GT.0 .OR. N.GT.1) GO TO 80
C-----------------------------------------------------------------------
C     GENERATE K(FNU,Z), 0.0D0 .LE. FNU .LT. 0.5D0 AND N=1
C-----------------------------------------------------------------------
      IF (CAZ.LT.TOL) GO TO 70
      CALL ZMLT(ZR, ZI, ZR, ZI, CZR, CZI)
      CZR = 0.25D0*CZR
      CZI = 0.25D0*CZI
      T1 = 0.25D0*CAZ*CAZ
   60 CONTINUE
      FR = (FR*AK+PR+QR)/BK
      FI = (FI*AK+PI+QI)/BK
      STR = 1.0D0/(AK-DNU)
      PR = PR*STR
      PI = PI*STR
      STR = 1.0D0/(AK+DNU)
      QR = QR*STR
      QI = QI*STR
      STR = CKR*CZR - CKI*CZI
      RAK = 1.0D0/AK
      CKI = (CKR*CZI+CKI*CZR)*RAK
      CKR = STR*RAK
      S1R = CKR*FR - CKI*FI + S1R
      S1I = CKR*FI + CKI*FR + S1I
      A1 = A1*T1*RAK
      BK = BK + AK + AK + 1.0D0
      AK = AK + 1.0D0
      IF (A1.GT.TOL) GO TO 60
   70 CONTINUE
```

```
      YR(1) = S1R
      YI(1) = S1I
      IF (KODED.EQ.1) RETURN
      CALL ZEXP(ZR, ZI, STR, STI)
      CALL ZMLT(S1R, S1I, STR, STI, YR(1), YI(1))
      RETURN
C-----------------------------------------------------------------------
C     GENERATE K(DNU,Z) AND K(DNU+1,Z) FOR FORWARD RECURRENCE
C-----------------------------------------------------------------------
   80 CONTINUE
      IF (CAZ.LT.TOL) GO TO 100
      CALL ZMLT(ZR, ZI, ZR, ZI, CZR, CZI)
      CZR = 0.25D0*CZR
      CZI = 0.25D0*CZI
      T1 = 0.25D0*CAZ*CAZ
   90 CONTINUE
      FR = (FR*AK+PR+QR)/BK
      FI = (FI*AK+PI+QI)/BK
      STR = 1.0D0/(AK-DNU)
      PR = PR*STR
      PI = PI*STR
      STR = 1.0D0/(AK+DNU)
      QR = QR*STR
      QI = QI*STR
      STR = CKR*CZR - CKI*CZI
      RAK = 1.0D0/AK
      CKI = (CKR*CZI+CKI*CZR)*RAK
      CKR = STR*RAK
      S1R = CKR*FR - CKI*FI + S1R
      S1I = CKR*FI + CKI*FR + S1I
      STR = PR - FR*AK
      STI = PI - FI*AK
      S2R = CKR*STR - CKI*STI + S2R
      S2I = CKR*STI + CKI*STR + S2I
      A1 = A1*T1*RAK
      BK = BK + AK + AK + 1.0D0
      AK = AK + 1.0D0
      IF (A1.GT.TOL) GO TO 90
  100 CONTINUE
      KFLAG = 2
      A1 = FNU + 1.0D0
      AK = A1*ABS(SMUR)
      IF (AK.GT.ALIM) KFLAG = 3
      STR = CSSR(KFLAG)
      P2R = S2R*STR
      P2I = S2I*STR
      CALL ZMLT(P2R, P2I, RZR, RZI, S2R, S2I)
      S1R = S1R*STR
      S1I = S1I*STR
      IF (KODED.EQ.1) GO TO 210
      CALL ZEXP(ZR, ZI, FR, FI)
      CALL ZMLT(S1R, S1I, FR, FI, S1R, S1I)
      CALL ZMLT(S2R, S2I, FR, FI, S2R, S2I)
      GO TO 210
C-----------------------------------------------------------------------
C     IFLAG=0 MEANS NO UNDERFLOW OCCURRED
C     IFLAG=1 MEANS AN UNDERFLOW OCCURRED- COMPUTATION PROCEEDS WITH
C     KODED=2 AND A TEST FOR ON SCALE VALUES IS MADE DURING FORWARD
```

```
C     RECURSION
C-----------------------------------------------------------------------
  110 CONTINUE
      CALL ZSQRT(ZR, ZI, STR, STI)
      CALL ZDIV(RTHPI, CZEROI, STR, STI, COEFR, COEFI)
      KFLAG = 2
      IF (KODED.EQ.2) GO TO 120
      IF (ZR.GT.ALIM) GO TO 290
C     BLANK LINE
      STR = EXP(-ZR)*CSSR(KFLAG)
      STI = -STR*SIN(ZI)
      STR = STR*COS(ZI)
      CALL ZMLT(COEFR, COEFI, STR, STI, COEFR, COEFI)
  120 CONTINUE
      IF (ABS(DNU).EQ.0.5D0) GO TO 300
C-----------------------------------------------------------------------
C     MILLER ALGORITHM FOR ABS(Z).GT.R1
C-----------------------------------------------------------------------
      AK = COS(DPI*DNU)
      AK = ABS(AK)
      IF (AK.EQ.CZEROR) GO TO 300
      FHS = ABS(0.25D0-DNU2)
      IF (FHS.EQ.CZEROR) GO TO 300
C-----------------------------------------------------------------------
C     COMPUTE R2=F(E). IF ABS(Z).GE.R2, USE FORWARD RECURRENCE TO
C     DETERMINE THE BACKWARD INDEX K. R2=F(E) IS A STRAIGHT LINE ON
C     12.LE.E.LE.60. E IS COMPUTED FROM 2**(-E)=B**(1-I1MACH(14))=
C     TOL WHERE B IS THE BASE OF THE ARITHMETIC.
C-----------------------------------------------------------------------
      T1 = I1MACH(14)-1
      T1 = T1*D1MACH(5)*3.321928094D0
      T1 = MAX(T1,12.0D0)
      T1 = MIN(T1,60.0D0)
      T2 = TTH*T1 - 6.0D0
      IF (ZR.NE.0.0D0) GO TO 130
      T1 = HPI
      GO TO 140
  130 CONTINUE
      T1 = DATAN(ZI/ZR)
      T1 = ABS(T1)
  140 CONTINUE
      IF (T2.GT.CAZ) GO TO 170
C-----------------------------------------------------------------------
C     FORWARD RECURRENCE LOOP WHEN ABS(Z).GE.R2
C-----------------------------------------------------------------------
      ETEST = AK/(DPI*CAZ*TOL)
      FK = CONER
      IF (ETEST.LT.CONER) GO TO 180
      FKS = CTWOR
      CKR = CAZ + CAZ + CTWOR
      P1R = CZEROR
      P2R = CONER
      DO 150 I=1,KMAX
        AK = FHS/FKS
        CBR = CKR/(FK+CONER)
        PTR = P2R
        P2R = CBR*P2R - P1R*AK
        P1R = PTR
```

```
        CKR = CKR + CTWOR
        FKS = FKS + FK + FK + CTWOR
        FHS = FHS + FK + FK
        FK = FK + CONER
        STR = ABS(P2R)*FK
        IF (ETEST.LT.STR) GO TO 160
  150 CONTINUE
      GO TO 310
  160 CONTINUE
      FK = FK + SPI*T1*SQRT(T2/CAZ)
      FHS = ABS(0.25D0-DNU2)
      GO TO 180
  170 CONTINUE
C-----------------------------------------------------------------------
C     COMPUTE BACKWARD INDEX K FOR ABS(Z).LT.R2
C-----------------------------------------------------------------------
      A2 = SQRT(CAZ)
      AK = FPI*AK/(TOL*SQRT(A2))
      AA = 3.0D0*T1/(1.0D0+CAZ)
      BB = 14.7D0*T1/(28.0D0+CAZ)
      AK = (LOG(AK)+CAZ*COS(AA)/(1.0D0+0.008D0*CAZ))/COS(BB)
      FK = 0.12125D0*AK*AK/CAZ + 1.5D0
  180 CONTINUE
C-----------------------------------------------------------------------
C     BACKWARD RECURRENCE LOOP FOR MILLER ALGORITHM
C-----------------------------------------------------------------------
      K = FK
      FK = K
      FKS = FK*FK
      P1R = CZEROR
      P1I = CZEROI
      P2R = TOL
      P2I = CZEROI
      CSR = P2R
      CSI = P2I
      DO 190 I=1,K
        A1 = FKS - FK
        AK = (FKS+FK)/(A1+FHS)
        RAK = 2.0D0/(FK+CONER)
        CBR = (FK+ZR)*RAK
        CBI = ZI*RAK
        PTR = P2R
        PTI = P2I
        P2R = (PTR*CBR-PTI*CBI-P1R)*AK
        P2I = (PTI*CBR+PTR*CBI-P1I)*AK
        P1R = PTR
        P1I = PTI
        CSR = CSR + P2R
        CSI = CSI + P2I
        FKS = A1 - FK + CONER
        FK = FK - CONER
  190 CONTINUE
C-----------------------------------------------------------------------
C     COMPUTE (P2/CS)=(P2/ABS(CS))*(CONJG(CS)/ABS(CS)) FOR BETTER
C     SCALING
C-----------------------------------------------------------------------
      TM = ZABS(CSR,CSI)
      PTR = 1.0D0/TM
```

```
      S1R = P2R*PTR
      S1I = P2I*PTR
      CSR = CSR*PTR
      CSI = -CSI*PTR
      CALL ZMLT(COEFR, COEFI, S1R, S1I, STR, STI)
      CALL ZMLT(STR, STI, CSR, CSI, S1R, S1I)
      IF (INU.GT.0 .OR. N.GT.1) GO TO 200
      ZDR = ZR
      ZDI = ZI
      IF(IFLAG.EQ.1) GO TO 270
      GO TO 240
  200 CONTINUE
C-----------------------------------------------------------------------
C     COMPUTE P1/P2=(P1/ABS(P2)*CONJG(P2)/ABS(P2) FOR SCALING
C-----------------------------------------------------------------------
      TM = ZABS(P2R,P2I)
      PTR = 1.0D0/TM
      P1R = P1R*PTR
      P1I = P1I*PTR
      P2R = P2R*PTR
      P2I = -P2I*PTR
      CALL ZMLT(P1R, P1I, P2R, P2I, PTR, PTI)
      STR = DNU + 0.5D0 - PTR
      STI = -PTI
      CALL ZDIV(STR, STI, ZR, ZI, STR, STI)
      STR = STR + 1.0D0
      CALL ZMLT(STR, STI, S1R, S1I, S2R, S2I)
C-----------------------------------------------------------------------
C     FORWARD RECURSION ON THE THREE TERM RECURSION WITH RELATION WITH
C     SCALING NEAR EXPONENT EXTREMES ON KFLAG=1 OR KFLAG=3
C-----------------------------------------------------------------------
  210 CONTINUE
      STR = DNU + 1.0D0
      CKR = STR*RZR
      CKI = STR*RZI
      IF (N.EQ.1) INU = INU - 1
      IF (INU.GT.0) GO TO 220
      IF (N.GT.1) GO TO 215
      S1R = S2R
      S1I = S2I
  215 CONTINUE
      ZDR = ZR
      ZDI = ZI
      IF(IFLAG.EQ.1) GO TO 270
      GO TO 240
  220 CONTINUE
      INUB = 1
      IF(IFLAG.EQ.1) GO TO 261
  225 CONTINUE
      P1R = CSRR(KFLAG)
      ASCLE = BRY(KFLAG)
      DO 230 I=INUB,INU
        STR = S2R
        STI = S2I
        S2R = CKR*STR - CKI*STI + S1R
        S2I = CKR*STI + CKI*STR + S1I
        S1R = STR
        S1I = STI
```

```
        CKR = CKR + RZR
        CKI = CKI + RZI
        IF (KFLAG.GE.3) GO TO 230
        P2R = S2R*P1R
        P2I = S2I*P1R
        STR = ABS(P2R)
        STI = ABS(P2I)
        P2M = MAX(STR,STI)
        IF (P2M.LE.ASCLE) GO TO 230
        KFLAG = KFLAG + 1
        ASCLE = BRY(KFLAG)
        S1R = S1R*P1R
        S1I = S1I*P1R
        S2R = P2R
        S2I = P2I
        STR = CSSR(KFLAG)
        S1R = S1R*STR
        S1I = S1I*STR
        S2R = S2R*STR
        S2I = S2I*STR
        P1R = CSRR(KFLAG)
  230 CONTINUE
      IF (N.NE.1) GO TO 240
      S1R = S2R
      S1I = S2I
  240 CONTINUE
      STR = CSRR(KFLAG)
      YR(1) = S1R*STR
      YI(1) = S1I*STR
      IF (N.EQ.1) RETURN
      YR(2) = S2R*STR
      YI(2) = S2I*STR
      IF (N.EQ.2) RETURN
      KK = 2
  250 CONTINUE
      KK = KK + 1
      IF (KK.GT.N) RETURN
      P1R = CSRR(KFLAG)
      ASCLE = BRY(KFLAG)
      DO 260 I=KK,N
        P2R = S2R
        P2I = S2I
        S2R = CKR*P2R - CKI*P2I + S1R
        S2I = CKI*P2R + CKR*P2I + S1I
        S1R = P2R
        S1I = P2I
        CKR = CKR + RZR
        CKI = CKI + RZI
        P2R = S2R*P1R
        P2I = S2I*P1R
        YR(I) = P2R
        YI(I) = P2I
        IF (KFLAG.GE.3) GO TO 260
        STR = ABS(P2R)
        STI = ABS(P2I)
        P2M = MAX(STR,STI)
        IF (P2M.LE.ASCLE) GO TO 260
        KFLAG = KFLAG + 1
```

```
        ASCLE = BRY(KFLAG)
        S1R = S1R*P1R
        S1I = S1I*P1R
        S2R = P2R
        S2I = P2I
        STR = CSSR(KFLAG)
        S1R = S1R*STR
        S1I = S1I*STR
        S2R = S2R*STR
        S2I = S2I*STR
        P1R = CSRR(KFLAG)
  260 CONTINUE
      RETURN
C-----------------------------------------------------------------------
C     IFLAG=1 CASES, FORWARD RECURRENCE ON SCALED VALUES ON UNDERFLOW
C-----------------------------------------------------------------------
  261 CONTINUE
      HELIM = 0.5D0*ELIM
      ELM = EXP(-ELIM)
      CELMR = ELM
      ASCLE = BRY(1)
      ZDR = ZR
      ZDI = ZI
      IC = -1
      J = 2
      DO 262 I=1,INU
        STR = S2R
        STI = S2I
        S2R = STR*CKR-STI*CKI+S1R
        S2I = STI*CKR+STR*CKI+S1I
        S1R = STR
        S1I = STI
        CKR = CKR+RZR
        CKI = CKI+RZI
        AS = ZABS(S2R,S2I)
        ALAS = LOG(AS)
        P2R = -ZDR+ALAS
        IF(P2R.LT.(-ELIM)) GO TO 263
        CALL ZLOG(S2R,S2I,STR,STI,IDUM)
        P2R = -ZDR+STR
        P2I = -ZDI+STI
        P2M = EXP(P2R)/TOL
        P1R = P2M*COS(P2I)
        P1I = P2M*SIN(P2I)
        CALL ZUCHK(P1R,P1I,NW,ASCLE,TOL)
        IF(NW.NE.0) GO TO 263
        J = 3 - J
        CYR(J) = P1R
        CYI(J) = P1I
        IF(IC.EQ.(I-1)) GO TO 264
        IC = I
        GO TO 262
  263   CONTINUE
        IF(ALAS.LT.HELIM) GO TO 262
        ZDR = ZDR-ELIM
        S1R = S1R*CELMR
        S1I = S1I*CELMR
        S2R = S2R*CELMR
```

```
        S2I = S2I*CELMR
  262 CONTINUE
      IF(N.NE.1) GO TO 270
      S1R = S2R
      S1I = S2I
      GO TO 270
  264 CONTINUE
      KFLAG = 1
      INUB = I+1
      S2R = CYR(J)
      S2I = CYI(J)
      J = 3 - J
      S1R = CYR(J)
      S1I = CYI(J)
      IF(INUB.LE.INU) GO TO 225
      IF(N.NE.1) GO TO 240
      S1R = S2R
      S1I = S2I
      GO TO 240
  270 CONTINUE
      YR(1) = S1R
      YI(1) = S1I
      IF(N.EQ.1) GO TO 280
      YR(2) = S2R
      YI(2) = S2I
  280 CONTINUE
      ASCLE = BRY(1)
      CALL ZKSCL(ZDR,ZDI,FNU,N,YR,YI,NZ,RZR,RZI,ASCLE,TOL,ELIM)
      INU = N - NZ
      IF (INU.LE.0) RETURN
      KK = NZ + 1
      S1R = YR(KK)
      S1I = YI(KK)
      YR(KK) = S1R*CSRR(1)
      YI(KK) = S1I*CSRR(1)
      IF (INU.EQ.1) RETURN
      KK = NZ + 2
      S2R = YR(KK)
      S2I = YI(KK)
      YR(KK) = S2R*CSRR(1)
      YI(KK) = S2I*CSRR(1)
      IF (INU.EQ.2) RETURN
      T2 = FNU + (KK-1)
      CKR = T2*RZR
      CKI = T2*RZI
      KFLAG = 1
      GO TO 250
  290 CONTINUE
C-----------------------------------------------------------------------
C     SCALE BY EXP(Z), IFLAG = 1 CASES
C-----------------------------------------------------------------------
      KODED = 2
      IFLAG = 1
      KFLAG = 2
      GO TO 120
C-----------------------------------------------------------------------
C     FNU=HALF ODD INTEGER CASE, DNU=-0.5
C-----------------------------------------------------------------------
```

```
  300 CONTINUE
      S1R = COEFR
      S1I = COEFI
      S2R = COEFR
      S2I = COEFI
      GO TO 210
C
C
  310 CONTINUE
      NZ=-2
      RETURN
      END
*DECK ZEXP
      SUBROUTINE ZEXP (AR, AI, BR, BI)
C***BEGIN PROLOGUE  ZEXP
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZBESH, ZBESI, ZBESJ, ZBESK, ZBESY, ZAIRY and
C            ZBIRY
C***LIBRARY   SLATEC
C***TYPE      ALL (ZEXP-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     DOUBLE PRECISION COMPLEX EXPONENTIAL FUNCTION B=EXP(A)
C
C***SEE ALSO  ZAIRY, ZBESH, ZBESI, ZBESJ, ZBESK, ZBESY, ZBIRY
C***ROUTINES CALLED  (NONE)
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C***END PROLOGUE  ZEXP
      DOUBLE PRECISION AR, AI, BR, BI, ZM, CA, CB
C***FIRST EXECUTABLE STATEMENT  ZEXP
      ZM = EXP(AR)
      CA = ZM*COS(AI)
      CB = ZM*SIN(AI)
      BR = CA
      BI = CB
      RETURN
      END
*DECK ZKSCL
      SUBROUTINE ZKSCL (ZRR, ZRI, FNU, N, YR, YI, NZ, RZR, RZI, ASCLE,
     +   TOL, ELIM)
C***BEGIN PROLOGUE  ZKSCL
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZBESK
C***LIBRARY   SLATEC
C***TYPE      ALL (CKSCL-A, ZKSCL-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     SET K FUNCTIONS TO ZERO ON UNDERFLOW, CONTINUE RECURRENCE
C     ON SCALED FUNCTIONS UNTIL TWO MEMBERS COME ON SCALE, THEN
C     RETURN WITH MIN(NZ+2,N) VALUES SCALED BY 1/TOL.
C
C***SEE ALSO  ZBESK
C***ROUTINES CALLED  ZABS, ZLOG, ZUCHK
C***REVISION HISTORY  (YYMMDD)
```

```
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C   930122  Added ZLOG to EXTERNAL statement.  (RWC)
C***END PROLOGUE  ZKSCL
C     COMPLEX CK,CS,CY,CZERO,RZ,S1,S2,Y,ZR,ZD,CELM
      DOUBLE PRECISION ACS, AS, ASCLE, CKI, CKR, CSI, CSR, CYI,
     * CYR, ELIM, FN, FNU, RZI, RZR, STR, S1I, S1R, S2I,
     * S2R, TOL, YI, YR, ZEROI, ZEROR, ZRI, ZRR, ZABS,
     * ZDR, ZDI, CELMR, ELM, HELIM, ALAS
      INTEGER I, IC, IDUM, KK, N, NN, NW, NZ
      DIMENSION YR(N), YI(N), CYR(2), CYI(2)
      EXTERNAL ZABS, ZLOG
      DATA ZEROR,ZEROI / 0.0D0 , 0.0D0 /
C***FIRST EXECUTABLE STATEMENT  ZKSCL
      NZ = 0
      IC = 0
      NN = MIN(2,N)
      DO 10 I=1,NN
        S1R = YR(I)
        S1I = YI(I)
        CYR(I) = S1R
        CYI(I) = S1I
        AS = ZABS(S1R,S1I)
        ACS = -ZRR + LOG(AS)
        NZ = NZ + 1
        YR(I) = ZEROR
        YI(I) = ZEROI
        IF (ACS.LT.(-ELIM)) GO TO 10
        CALL ZLOG(S1R, S1I, CSR, CSI, IDUM)
        CSR = CSR - ZRR
        CSI = CSI - ZRI
        STR = EXP(CSR)/TOL
        CSR = STR*COS(CSI)
        CSI = STR*SIN(CSI)
        CALL ZUCHK(CSR, CSI, NW, ASCLE, TOL)
        IF (NW.NE.0) GO TO 10
        YR(I) = CSR
        YI(I) = CSI
        IC = I
        NZ = NZ - 1
   10 CONTINUE
      IF (N.EQ.1) RETURN
      IF (IC.GT.1) GO TO 20
      YR(1) = ZEROR
      YI(1) = ZEROI
      NZ = 2
   20 CONTINUE
      IF (N.EQ.2) RETURN
      IF (NZ.EQ.0) RETURN
      FN = FNU + 1.0D0
      CKR = FN*RZR
      CKI = FN*RZI
      S1R = CYR(1)
      S1I = CYI(1)
      S2R = CYR(2)
      S2I = CYI(2)
      HELIM = 0.5D0*ELIM
      ELM = EXP(-ELIM)
```

```
      CELMR = ELM
      ZDR = ZRR
      ZDI = ZRI
C
C     FIND TWO CONSECUTIVE Y VALUES ON SCALE. SCALE RECURRENCE IF
C     S2 GETS LARGER THAN EXP(ELIM/2)
C
      DO 30 I=3,N
        KK = I
        CSR = S2R
        CSI = S2I
        S2R = CKR*CSR - CKI*CSI + S1R
        S2I = CKI*CSR + CKR*CSI + S1I
        S1R = CSR
        S1I = CSI
        CKR = CKR + RZR
        CKI = CKI + RZI
        AS = ZABS(S2R,S2I)
        ALAS = LOG(AS)
        ACS = -ZDR + ALAS
        NZ = NZ + 1
        YR(I) = ZEROR
        YI(I) = ZEROI
        IF (ACS.LT.(-ELIM)) GO TO 25
        CALL ZLOG(S2R, S2I, CSR, CSI, IDUM)
        CSR = CSR - ZDR
        CSI = CSI - ZDI
        STR = EXP(CSR)/TOL
        CSR = STR*COS(CSI)
        CSI = STR*SIN(CSI)
        CALL ZUCHK(CSR, CSI, NW, ASCLE, TOL)
        IF (NW.NE.0) GO TO 25
        YR(I) = CSR
        YI(I) = CSI
        NZ = NZ - 1
        IF (IC.EQ.KK-1) GO TO 40
        IC = KK
        GO TO 30
   25   CONTINUE
        IF(ALAS.LT.HELIM) GO TO 30
        ZDR = ZDR - ELIM
        S1R = S1R*CELMR
        S1I = S1I*CELMR
        S2R = S2R*CELMR
        S2I = S2I*CELMR
   30 CONTINUE
      NZ = N
      IF(IC.EQ.N) NZ=N-1
      GO TO 45
   40 CONTINUE
      NZ = KK - 2
   45 CONTINUE
      DO 50 I=1,NZ
        YR(I) = ZEROR
        YI(I) = ZEROI
   50 CONTINUE
      RETURN
      END
```

```
*DECK ZLOG
      SUBROUTINE ZLOG (AR, AI, BR, BI, IERR)
C***BEGIN PROLOGUE  ZLOG
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZBESH, ZBESI, ZBESJ, ZBESK, ZBESY, ZAIRY and
C            ZBIRY
C***LIBRARY   SLATEC
C***TYPE      ALL (ZLOG-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     DOUBLE PRECISION COMPLEX LOGARITHM B=CLOG(A)
C     IERR=0,NORMAL RETURN      IERR=1, Z=CMPLX(0.0,0.0)
C***SEE ALSO  ZAIRY, ZBESH, ZBESI, ZBESJ, ZBESK, ZBESY, ZBIRY
C***ROUTINES CALLED  ZABS
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C***END PROLOGUE  ZLOG
      DOUBLE PRECISION AR, AI, BR, BI, ZM, DTHETA, DPI, DHPI
      DOUBLE PRECISION ZABS
      INTEGER IERR
      EXTERNAL ZABS
      DATA DPI , DHPI  / 3.14159265358979323846264338 3D+0,
     1                   1.57079632679489661923132169 6D+0/
C***FIRST EXECUTABLE STATEMENT  ZLOG
      IERR=0
      IF (AR.EQ.0.0D+0) GO TO 10
      IF (AI.EQ.0.0D+0) GO TO 20
      DTHETA = DATAN(AI/AR)
      IF (DTHETA.LE.0.0D+0) GO TO 40
      IF (AR.LT.0.0D+0) DTHETA = DTHETA - DPI
      GO TO 50
   10 IF (AI.EQ.0.0D+0) GO TO 60
      BI = DHPI
      BR = LOG(ABS(AI))
      IF (AI.LT.0.0D+0) BI = -BI
      RETURN
   20 IF (AR.GT.0.0D+0) GO TO 30
      BR = LOG(ABS(AR))
      BI = DPI
      RETURN
   30 BR = LOG(AR)
      BI = 0.0D+0
      RETURN
   40 IF (AR.LT.0.0D+0) DTHETA = DTHETA + DPI
   50 ZM = ZABS(AR,AI)
      BR = LOG(ZM)
      BI = DTHETA
      RETURN
   60 CONTINUE
      IERR=1
      RETURN
      END
*DECK ZRATI
      SUBROUTINE ZRATI (ZR, ZI, FNU, N, CYR, CYI, TOL)
C***BEGIN PROLOGUE  ZRATI
C***SUBSIDIARY
```

```
C***PURPOSE  Subsidiary to ZBESH, ZBESI and ZBESK
C***LIBRARY   SLATEC
C***TYPE      ALL (CRATI-A, ZRATI-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     ZRATI COMPUTES RATIOS OF I BESSEL FUNCTIONS BY BACKWARD
C     RECURRENCE.  THE STARTING INDEX IS DETERMINED BY FORWARD
C     RECURRENCE AS DESCRIBED IN J. RES. OF NAT. BUR. OF STANDARDS-B,
C     MATHEMATICAL SCIENCES, VOL 77B, P111-114, SEPTEMBER, 1973,
C     BESSEL FUNCTIONS I AND J OF COMPLEX ARGUMENT AND INTEGER ORDER,
C     BY D. J. SOOKNE.
C
C***SEE ALSO  ZBESH, ZBESI, ZBESK
C***ROUTINES CALLED  ZABS, ZDIV
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C***END PROLOGUE  ZRATI
      DOUBLE PRECISION AK, AMAGZ, AP1, AP2, ARG, AZ, CDFNUI, CDFNUR,
     * CONEI, CONER, CYI, CYR, CZEROI, CZEROR, DFNU, FDNU, FLAM, FNU,
     * FNUP, PTI, PTR, P1I, P1R, P2I, P2R, RAK, RAP1, RHO, RT2, RZI,
     * RZR, TEST, TEST1, TOL, TTI, TTR, T1I, T1R, ZI, ZR, ZABS
      INTEGER I, ID, IDNU, INU, ITIME, K, KK, MAGZ, N
      DIMENSION CYR(N), CYI(N)
      EXTERNAL ZABS
      DATA CZEROR,CZEROI,CONER,CONEI,RT2/
     1 0.0D0, 0.0D0, 1.0D0, 0.0D0, 1.4142135623730950 5D0 /
C***FIRST EXECUTABLE STATEMENT  ZRATI
      AZ = ZABS(ZR,ZI)
      INU = FNU
      IDNU = INU + N - 1
      MAGZ = AZ
      AMAGZ = MAGZ+1
      FDNU = IDNU
      FNUP = MAX(AMAGZ,FDNU)
      ID = IDNU - MAGZ - 1
      ITIME = 1
      K = 1
      PTR = 1.0D0/AZ
      RZR = PTR*(ZR+ZR)*PTR
      RZI = -PTR*(ZI+ZI)*PTR
      T1R = RZR*FNUP
      T1I = RZI*FNUP
      P2R = -T1R
      P2I = -T1I
      P1R = CONER
      P1I = CONEI
      T1R = T1R + RZR
      T1I = T1I + RZI
      IF (ID.GT.0) ID = 0
      AP2 = ZABS(P2R,P2I)
      AP1 = ZABS(P1R,P1I)
C---------------------------------------------------------------------
C     THE OVERFLOW TEST ON K(FNU+I-1,Z) BEFORE THE CALL TO CBKNU
C     GUARANTEES THAT P2 IS ON SCALE. SCALE TEST1 AND ALL SUBSEQUENT
C     P2 VALUES BY AP1 TO ENSURE THAT AN OVERFLOW DOES NOT OCCUR
C     PREMATURELY.
```

```
C-----------------------------------------------------------------------
      ARG = (AP2+AP2)/(AP1*TOL)
      TEST1 = SQRT(ARG)
      TEST = TEST1
      RAP1 = 1.0D0/AP1
      P1R = P1R*RAP1
      P1I = P1I*RAP1
      P2R = P2R*RAP1
      P2I = P2I*RAP1
      AP2 = AP2*RAP1
   10 CONTINUE
      K = K + 1
      AP1 = AP2
      PTR = P2R
      PTI = P2I
      P2R = P1R - (T1R*PTR-T1I*PTI)
      P2I = P1I - (T1R*PTI+T1I*PTR)
      P1R = PTR
      P1I = PTI
      T1R = T1R + RZR
      T1I = T1I + RZI
      AP2 = ZABS(P2R,P2I)
      IF (AP1.LE.TEST) GO TO 10
      IF (ITIME.EQ.2) GO TO 20
      AK = ZABS(T1R,T1I)*0.5D0
      FLAM = AK + SQRT(AK*AK-1.0D0)
      RHO = MIN(AP2/AP1,FLAM)
      TEST = TEST1*SQRT(RHO/(RHO*RHO-1.0D0))
      ITIME = 2
      GO TO 10
   20 CONTINUE
      KK = K + 1 - ID
      AK = KK
      T1R = AK
      T1I = CZEROI
      DFNU = FNU + (N-1)
      P1R = 1.0D0/AP2
      P1I = CZEROI
      P2R = CZEROR
      P2I = CZEROI
      DO 30 I=1,KK
        PTR = P1R
        PTI = P1I
        RAP1 = DFNU + T1R
        TTR = RZR*RAP1
        TTI = RZI*RAP1
        P1R = (PTR*TTR-PTI*TTI) + P2R
        P1I = (PTR*TTI+PTI*TTR) + P2I
        P2R = PTR
        P2I = PTI
        T1R = T1R - CONER
   30 CONTINUE
      IF (P1R.NE.CZEROR .OR. P1I.NE.CZEROI) GO TO 40
      P1R = TOL
      P1I = TOL
   40 CONTINUE
      CALL ZDIV(P2R, P2I, P1R, P1I, CYR(N), CYI(N))
      IF (N.EQ.1) RETURN
```

```
      K = N - 1
      AK = K
      T1R = AK
      T1I = CZEROI
      CDFNUR = FNU*RZR
      CDFNUI = FNU*RZI
      DO 60 I=2,N
        PTR = CDFNUR + (T1R*RZR-T1I*RZI) + CYR(K+1)
        PTI = CDFNUI + (T1R*RZI+T1I*RZR) + CYI(K+1)
        AK = ZABS(PTR,PTI)
        IF (AK.NE.CZEROR) GO TO 50
        PTR = TOL
        PTI = TOL
        AK = TOL*RT2
   50   CONTINUE
        RAK = CONER/AK
        CYR(K) = RAK*PTR*RAK
        CYI(K) = -RAK*PTI*RAK
        T1R = T1R - CONER
        K = K - 1
   60 CONTINUE
      RETURN
      END
*DECK ZSHCH
      SUBROUTINE ZSHCH (ZR, ZI, CSHR, CSHI, CCHR, CCHI)
C***BEGIN PROLOGUE  ZSHCH
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZBESH and ZBESK
C***LIBRARY   SLATEC
C***TYPE      ALL (CSHCH-A, ZSHCH-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     ZSHCH COMPUTES THE COMPLEX HYPERBOLIC FUNCTIONS CSH=SINH(X+I*Y)
C     AND CCH=COSH(X+I*Y), WHERE I**2=-1.
C
C***SEE ALSO  ZBESH, ZBESK
C***ROUTINES CALLED  (NONE)
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C***END PROLOGUE  ZSHCH
C
      DOUBLE PRECISION CCHI, CCHR, CH, CN, CSHI, CSHR, SH, SN, ZI, ZR
C***FIRST EXECUTABLE STATEMENT  ZSHCH
      SH = SINH(ZR)
      CH = COSH(ZR)
      SN = SIN(ZI)
      CN = COS(ZI)
      CSHR = SH*CN
      CSHI = CH*SN
      CCHR = CH*CN
      CCHI = SH*SN
      RETURN
      END
*DECK ZUNHJ
      SUBROUTINE ZUNHJ (ZR, ZI, FNU, IPMTR, TOL, PHIR, PHII, ARGR, ARGI,
     +   ZETA1R, ZETA1I, ZETA2R, ZETA2I, ASUMR, ASUMI, BSUMR, BSUMI)
```

```
C***BEGIN PROLOGUE  ZUNHJ
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZBESI and ZBESK
C***LIBRARY   SLATEC
C***TYPE      ALL (CUNHJ-A, ZUNHJ-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     REFERENCES
C         HANDBOOK OF MATHEMATICAL FUNCTIONS BY M. ABRAMOWITZ AND I.A.
C         STEGUN, AMS55, NATIONAL BUREAU OF STANDARDS, 1965, CHAPTER 9.
C
C         ASYMPTOTICS AND SPECIAL FUNCTIONS BY F.W.J. OLVER, ACADEMIC
C         PRESS, N.Y., 1974, PAGE 420
C
C     ABSTRACT
C         ZUNHJ COMPUTES PARAMETERS FOR BESSEL FUNCTIONS C(FNU,Z) =
C         J(FNU,Z), Y(FNU,Z) OR H(I,FNU,Z) I=1,2 FOR LARGE ORDERS FNU
C         BY MEANS OF THE UNIFORM ASYMPTOTIC EXPANSION
C
C         C(FNU,Z)=C1*PHI*( ASUM*AIRY(ARG) + C2*BSUM*DAIRY(ARG) )
C
C         FOR PROPER CHOICES OF C1, C2, AIRY AND DAIRY WHERE AIRY IS
C         AN AIRY FUNCTION AND DAIRY IS ITS DERIVATIVE.
C
C               (2/3)*FNU*ZETA**1.5 = ZETA1-ZETA2,
C
C         ZETA1=0.5*FNU*CLOG((1+W)/(1-W)), ZETA2=FNU*W FOR SCALING
C         PURPOSES IN AIRY FUNCTIONS FROM CAIRY OR CBIRY.
C
C         MCONJ=SIGN OF AIMAG(Z), BUT IS AMBIGUOUS WHEN Z IS REAL AND
C         MUST BE SPECIFIED. IPMTR=0 RETURNS ALL PARAMETERS. IPMTR=
C         1 COMPUTES ALL EXCEPT ASUM AND BSUM.
C
C***SEE ALSO  ZBESI, ZBESK
C***ROUTINES CALLED  D1MACH, ZABS, ZDIV, ZLOG, ZSQRT
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C   930122  Added ZLOG and ZSQRT to EXTERNAL statement.  (RWC)
C***END PROLOGUE  ZUNHJ
C     COMPLEX ARG,ASUM,BSUM,CFNU,CONE,CR,CZERO,DR,P,PHI,PRZTH,PTFN,
C    *RFN13,RTZTA,RZTH,SUMA,SUMB,TFN,T2,UP,W,W2,Z,ZA,ZB,ZC,ZETA,ZETA1,
C    *ZETA2,ZTH
      DOUBLE PRECISION ALFA, ANG, AP, AR, ARGI, ARGR, ASUMI, ASUMR,
     * ATOL, AW2, AZTH, BETA, BR, BSUMI, BSUMR, BTOL, C, CONEI, CONER,
     * CRI, CRR, DRI, DRR, EX1, EX2, FNU, FN13, FN23, GAMA, GPI, HPI,
     * PHII, PHIR, PI, PP, PR, PRZTHI, PRZTHR, PTFNI, PTFNR, RAW, RAW2,
     * RAZTH, RFNU, RFNU2, RFN13, RTZTI, RTZTR, RZTHI, RZTHR, STI, STR,
     * SUMAI, SUMAR, SUMBI, SUMBR, TEST, TFNI, TFNR, THPI, TOL, TZAI,
     * TZAR, T2I, T2R, UPI, UPR, WI, WR, W2I, W2R, ZAI, ZAR, ZBI, ZBR,
     * ZCI, ZCR, ZEROI, ZEROR, ZETAI, ZETAR, ZETA1I, ZETA1R, ZETA2I,
     * ZETA2R, ZI, ZR, ZTHI, ZTHR, ZABS, AC, D1MACH
      INTEGER IAS, IBS, IPMTR, IS, J, JR, JU, K, KMAX, KP1, KS, L, LR,
     * LRP1, L1, L2, M, IDUM
      DIMENSION AR(14), BR(14), C(105), ALFA(180), BETA(210), GAMA(30),
     * AP(30), PR(30), PI(30), UPR(14), UPI(14), CRR(14), CRI(14),
     * DRR(14), DRI(14)
```

```
      EXTERNAL ZABS, ZLOG, ZSQRT
      DATA AR(1), AR(2), AR(3), AR(4), AR(5), AR(6), AR(7), AR(8),
     1     AR(9), AR(10), AR(11), AR(12), AR(13), AR(14)/
     2     1.00000000000000000D+00,     1.04166666666666667D-01,
     3     8.35503472222222222D-02,     1.28226574556327160D-01,
     4     2.91849026464140464D-01,     8.81627267443757652D-01,
     5     3.32140828186276754D+00,     1.49957629868625547D+01,
     6     7.89230130115865181D+01,     4.74451538868264323D+02,
     7     3.20749009089066193D+03,     2.40865496408740049D+04,
     8     1.98923119169509794D+05,     1.79190200777534383D+06/
      DATA BR(1), BR(2), BR(3), BR(4), BR(5), BR(6), BR(7), BR(8),
     1     BR(9), BR(10), BR(11), BR(12), BR(13), BR(14)/
     2     1.00000000000000000D+00,    -1.45833333333333333D-01,
     3    -9.87413194444444444D-02,    -1.43312053915895062D-01,
     4    -3.17227202678413548D-01,    -9.42429147957120249D-01,
     5    -3.51120304082635426D+00,    -1.57272636203680451D+01,
     6    -8.22814390971859444D+01,    -4.92355370523670524D+02,
     7    -3.31621856854797251D+03,    -2.48276742452085896D+04,
     8    -2.04526587315129788D+05,    -1.83844491706820990D+06/
      DATA C(1), C(2), C(3), C(4), C(5), C(6), C(7), C(8), C(9), C(10),
     1     C(11), C(12), C(13), C(14), C(15), C(16), C(17), C(18),
     2     C(19), C(20), C(21), C(22), C(23), C(24)/
     3     1.00000000000000000D+00,    -2.08333333333333333D-01,
     4     1.25000000000000000D-01,     3.34201388888888889D-01,
     5    -4.01041666666666667D-01,     7.03125000000000000D-02,
     6    -1.02581259645061728D+00,     1.84646267361111111D+00,
     7    -8.91210937500000000D-01,     7.32421875000000000D-02,
     8     4.66958442342624743D+00,    -1.12070026162229938D+01,
     9     8.78912353515625000D+00,    -2.36408691406250000D+00,
     A     1.12152099609375000D-01,    -2.82120725582002449D+01,
     B     8.46362176746007346D+01,    -9.18182415432400174D+01,
     C     4.25349987453884549D+01,    -7.36879435947963170D+00,
     D     2.27108001708984375D-01,     2.12570130039217123D+02,
     E    -7.65252468141181642D+02,     1.05999045252799988D+03/
      DATA C(25), C(26), C(27), C(28), C(29), C(30), C(31), C(32),
     1     C(33), C(34), C(35), C(36), C(37), C(38), C(39), C(40),
     2     C(41), C(42), C(43), C(44), C(45), C(46), C(47), C(48)/
     3    -6.99579627376132541D+02,     2.18190511744211590D+02,
     4    -2.64914304869515555D+01,     5.72501420974731445D-01,
     5    -1.91945766231840700D+03,     8.06172218173730938D+03,
     6    -1.35865500064341374D+04,     1.16553933368645332D+04,
     7    -5.30564697861340311D+03,     1.20090291321635246D+03,
     8    -1.08090919788394656D+02,     1.72772750258445740D+00,
     9     2.02042913309661486D+04,    -9.69805983886375135D+04,
     A     1.92547001232531532D+05,    -2.03400177280415534D+05,
     B     1.22200464983017460D+05,    -4.11926549688975513D+04,
     C     7.10951430248936372D+03,    -4.93915304773088012D+02,
     D     6.07404200127348304D+00,    -2.42919187900551333D+05,
     E     1.31176361466297720D+06,    -2.99801591853810675D+06/
      DATA C(49), C(50), C(51), C(52), C(53), C(54), C(55), C(56),
     1     C(57), C(58), C(59), C(60), C(61), C(62), C(63), C(64),
     2     C(65), C(66), C(67), C(68), C(69), C(70), C(71), C(72)/
     3     3.76327129765640400D+06,    -2.81356322658653411D+06,
     4     1.26836527332162478D+06,    -3.31645172484563578D+05,
     5     4.52187689813627263D+04,    -2.49983048181120962D+03,
     6     2.43805296995560639D+01,     3.28446985307203782D+06,
     7    -1.97068191184322269D+07,     5.09526024926646422D+07,
     8    -7.41051482115326577D+07,     6.63445122747290267D+07,
```

```
9     -3.7567176660763351 3D+07,     1.3288767166421818 3D+07,
A     -2.7856181280864546 9D+06,     3.0818640461266239 8D+05,
B     -1.3886089753717040 5D+04,     1.1001714026924673 8D+02,
C     -4.9329253664509962 0D+07,     3.2557307418576574 9D+08,
D     -9.3946235968157840 3D+08,     1.5535968995705800 6D+09,
E     -1.6210805521083370 8D+09,     1.1068428168230144 7D+09/
      DATA C(73), C(74), C(75), C(76), C(77), C(78), C(79), C(80),
1     C(81), C(82), C(83), C(84), C(85), C(86), C(87), C(88),
2     C(89), C(90), C(91), C(92), C(93), C(94), C(95), C(96)/
3     -4.9588978427503030 9D+08,     1.4206290779753309 5D+08,
4     -2.4474062725738728 5D+07,     2.2437681779224494 3D+06,
5     -8.4005433603024085 3D+04,     5.5133589612202058 6D+02,
6      8.1478909611831211 5D+08,    -5.8664814920518472 3D+09,
7      1.8688207509295824 9D+10,    -3.4632043388158777 9D+10,
8      4.1280185579753974 0D+10,    -3.3026599749800723 1D+10,
9      1.7954213731155600 1D+10,    -6.5632937926192843 3D+09,
A      1.5592798648792575 1D+09,    -2.2510566188941527 8D+08,
B      1.7395107553978164 5D+07,    -5.4984232757228868 7D+05,
C      3.0380905109223842 7D+03,    -1.4679261247695616 7D+10,
D      1.1449823773202581 0D+11,    -3.9909617522446649 8D+11,
E      8.1921866954857732 9D+11,    -1.0983751560812233 1D+12/
      DATA C(97), C(98), C(99), C(100), C(101), C(102), C(103), C(104),
1     C(105)/
2     1.0081581068653820 9D+12,    -6.4536486924537650 3D+11,
3     2.8790064990615058 9D+11,    -8.7867072178023265 7D+10,
4     1.7634730606834969 4D+10,    -2.1671649832237950 9D+09,
5     1.4315787671888898 1D+08,    -3.8718334425726126 2D+06,
6     1.8257755474293174 7D+04/
      DATA ALFA(1), ALFA(2), ALFA(3), ALFA(4), ALFA(5), ALFA(6),
1     ALFA(7), ALFA(8), ALFA(9), ALFA(10), ALFA(11), ALFA(12),
2     ALFA(13), ALFA(14), ALFA(15), ALFA(16), ALFA(17), ALFA(18),
3     ALFA(19), ALFA(20), ALFA(21), ALFA(22)/
4    -4.4444444444444444 D-03,    -9.2207792207792207 8D-04,
5    -8.8489288489288489 3D-05,     1.6592768783244973 7D-04,
6     2.4669137274179291 0D-04,     2.6599558934625478 0D-04,
7     2.6182429706150094 5D-04,     2.4873043734465560 9D-04,
8     2.3272104008323209 8D-04,     2.1636248571236508 2D-04,
9     2.0073885876275235 5D-04,     1.8626763663754517 2D-04,
A     1.7306077591787649 3D-04,     1.6109170592901575 2D-04,
B     1.5027477416090813 4D-04,     1.4050349739126979 4D-04,
C     1.3166881654592280 6D-04,     1.2366744559825326 1D-04,
D     1.1640527147473790 2D-04,     1.0979829837271336 9D-04,
E     1.0377241042299282 3D-04,     9.8262607836936344 8D-05/
      DATA ALFA(23), ALFA(24), ALFA(25), ALFA(26), ALFA(27), ALFA(28),
1     ALFA(29), ALFA(30), ALFA(31), ALFA(32), ALFA(33), ALFA(34),
2     ALFA(35), ALFA(36), ALFA(37), ALFA(38), ALFA(39), ALFA(40),
3     ALFA(41), ALFA(42), ALFA(43), ALFA(44)/
4     9.3212051724950325 6D-05,     8.8571085247871171 8D-05,
5     8.4296310571570022 3D-05,     8.0349754840779115 1D-05,
6     7.6698134535920738 8D-05,     7.3312215748177780 9D-05,
7     7.0166262516314133 3D-05,     6.7237563379016029 2D-05,
8     6.9373554135458897 4D-04,     2.3224174518292165 4D-04,
9    -1.4198627355669119 7D-05,    -1.1644493167204864 0D-04,
A    -1.5080355805304876 2D-04,    -1.5512192491809622 3D-04,
B    -1.4680975664646554 9D-04,    -1.3381550386749136 7D-04,
C    -1.1974497568425405 1D-04,    -1.0618431920797402 0D-04,
D    -9.3769954989119449 2D-05,    -8.2692304558819327 4D-05,
E    -7.2937434815522121 1D-05,    -6.4404235772101628 3D-05/
```

```
      DATA ALFA(45), ALFA(46), ALFA(47), ALFA(48), ALFA(49), ALFA(50),
     1     ALFA(51), ALFA(52), ALFA(53), ALFA(54), ALFA(55), ALFA(56),
     2     ALFA(57), ALFA(58), ALFA(59), ALFA(60), ALFA(61), ALFA(62),
     3     ALFA(63), ALFA(64), ALFA(65), ALFA(66)/
     4    -5.69611566009369048D-05,    -5.04731044303561628D-05,
     5    -4.48134868008882786D-05,    -3.98688727717598864D-05,
     6    -3.55400532972042498D-05,    -3.17414256609022480D-05,
     7    -2.83996793904174811D-05,    -2.54522720634870566D-05,
     8    -2.28459297164724555D-05,    -2.05352753106480604D-05,
     9    -1.84816217627666085D-05,    -1.66519330021393806D-05,
     A    -1.50179412980119482D-05,    -1.35554031379040526D-05,
     B    -1.22434746473858131D-05,    -1.10641884811308169D-05,
     C    -3.54211971457743841D-04,    -1.56161263945159416D-04,
     D     3.04465503594936410D-05,     1.30198655773242693D-04,
     E     1.67471106699712269D-04,     1.70222587683592569D-04/
      DATA ALFA(67), ALFA(68), ALFA(69), ALFA(70), ALFA(71), ALFA(72),
     1     ALFA(73), ALFA(74), ALFA(75), ALFA(76), ALFA(77), ALFA(78),
     2     ALFA(79), ALFA(80), ALFA(81), ALFA(82), ALFA(83), ALFA(84),
     3     ALFA(85), ALFA(86), ALFA(87), ALFA(88)/
     4     1.56501427608594704D-04,     1.36339170977445120D-04,
     5     1.14886692029825128D-04,     9.45869093034688111D-05,
     6     7.64498419250898258D-05,     6.07570334965197354D-05,
     7     4.74394299290508799D-05,     3.62757512005344297D-05,
     8     2.69939714979224901D-05,     1.93210938247939253D-05,
     9     1.30056674793963203D-05,     7.82620866744496661D-06,
     A     3.59257485819351583D-06,     1.44040049814251817D-07,
     B    -2.65396769697939116D-06,    -4.91346867098485910D-06,
     C    -6.72739296091248287D-06,    -8.17269379678657923D-06,
     D    -9.31304715093561232D-06,    -1.02011418798016441D-05,
     E    -1.08805962510592880D-05,    -1.13875481509603555D-05/
      DATA ALFA(89), ALFA(90), ALFA(91), ALFA(92), ALFA(93), ALFA(94),
     1     ALFA(95), ALFA(96), ALFA(97), ALFA(98), ALFA(99), ALFA(100),
     2     ALFA(101), ALFA(102), ALFA(103), ALFA(104), ALFA(105),
     3     ALFA(106), ALFA(107), ALFA(108), ALFA(109), ALFA(110)/
     4    -1.17519675674556414D-05,    -1.19987364870944141D-05,
     5     3.78194199201772914D-04,     2.02471952761816167D-04,
     6    -6.37938506318862408D-05,    -2.38598230603005903D-04,
     7    -3.10916256027361568D-04,    -3.13680115247576316D-04,
     8    -2.78950273791323387D-04,    -2.28564082619141374D-04,
     9    -1.75245280340846749D-04,    -1.25544063060690348D-04,
     A    -8.22982872820208365D-05,    -4.62860730588116458D-05,
     B    -1.72334302366962267D-05,     5.60690482304602267D-06,
     C     2.31395443148286800D-05,     3.62642745856793957D-05,
     D     4.58006124490188752D-05,     5.24595294959114050D-05,
     E     5.68396208545815266D-05,     5.94349820393104052D-05/
      DATA ALFA(111), ALFA(112), ALFA(113), ALFA(114), ALFA(115),
     1     ALFA(116), ALFA(117), ALFA(118), ALFA(119), ALFA(120),
     2     ALFA(121), ALFA(122), ALFA(123), ALFA(124), ALFA(125),
     3     ALFA(126), ALFA(127), ALFA(128), ALFA(129), ALFA(130)/
     4     6.06478527578421742D-05,     6.08023907788436497D-05,
     5     6.01577894539460388D-05,     5.89199657344698500D-05,
     6     5.72515823777593053D-05,     5.52804375585852577D-05,
     7     5.31063773802880170D-05,     5.08069302012325706D-05,
     8     4.84418647620094842D-05,     4.60568581607475370D-05,
     9    -6.91141397288294174D-04,    -4.29976633058871912D-04,
     A     1.83067735980039018D-04,     6.60088147542014144D-04,
     B     8.75964969951185931D-04,     8.77335235958235514D-04,
     C     7.49369585378990637D-04,     5.63832329756980918D-04,
```

```
D     3.68059319971443156D-04,     1.88464535514455599D-04/
 DATA ALFA(131), ALFA(132), ALFA(133), ALFA(134), ALFA(135),
1     ALFA(136), ALFA(137), ALFA(138), ALFA(139), ALFA(140),
2     ALFA(141), ALFA(142), ALFA(143), ALFA(144), ALFA(145),
3     ALFA(146), ALFA(147), ALFA(148), ALFA(149), ALFA(150)/
4     3.70663057664904149D-05,    -8.28520220232137023D-05,
5    -1.72751952869172998D-04,    -2.36314873605872983D-04,
6    -2.77966150694906658D-04,    -3.02079514155456919D-04,
7    -3.12594712643820127D-04,    -3.12872558758067163D-04,
8    -3.05678038466324377D-04,    -2.93226470614557331D-04,
9    -2.77255655582934777D-04,    -2.59103928467031709D-04,
A    -2.39784014396480342D-04,    -2.20048260045422848D-04,
B    -2.00443911094971498D-04,    -1.81358692210970687D-04,
C    -1.63057674478657464D-04,    -1.45712672175205844D-04,
D    -1.29425421983924587D-04,    -1.14245691942445952D-04/
 DATA ALFA(151), ALFA(152), ALFA(153), ALFA(154), ALFA(155),
1     ALFA(156), ALFA(157), ALFA(158), ALFA(159), ALFA(160),
2     ALFA(161), ALFA(162), ALFA(163), ALFA(164), ALFA(165),
3     ALFA(166), ALFA(167), ALFA(168), ALFA(169), ALFA(170)/
4     1.92821964248775885D-03,     1.35592576302022234D-03,
5    -7.17858090421302995D-04,    -2.58084802575270346D-03,
6    -3.49271130826168475D-03,    -3.46986299340960628D-03,
7    -2.82285233351310182D-03,    -1.88103076404891354D-03,
8    -8.89531718383947600D-04,     3.87912102631035228D-06,
9     7.28688540119691412D-04,     1.26566373053457758D-03,
A     1.62518158372674427D-03,     1.83203153216373172D-03,
B     1.91588388990527909D-03,     1.90588846755546138D-03,
C     1.82798982421825727D-03,     1.70389506421121530D-03,
D     1.55097127171097686D-03,     1.38261421852276159D-03/
 DATA ALFA(171), ALFA(172), ALFA(173), ALFA(174), ALFA(175),
1     ALFA(176), ALFA(177), ALFA(178), ALFA(179), ALFA(180)/
2     1.20881424230064774D-03,     1.03676532638344962D-03,
3     8.71437918068619115D-04,     7.16080155297701002D-04,
4     5.72637002558129372D-04,     4.42089819465802277D-04,
5     3.24724948503090564D-04,     2.20342042730246599D-04,
6     1.28412898401353882D-04,     4.82005924552095464D-05/
 DATA BETA(1), BETA(2), BETA(3), BETA(4), BETA(5), BETA(6),
1     BETA(7), BETA(8), BETA(9), BETA(10), BETA(11), BETA(12),
2     BETA(13), BETA(14), BETA(15), BETA(16), BETA(17), BETA(18),
3     BETA(19), BETA(20), BETA(21), BETA(22)/
4     1.79988721413553309D-02,     5.59964911064388073D-03,
5     2.88501402231132779D-03,     1.80096606761053941D-03,
6     1.24753110589199202D-03,     9.22878876572938311D-04,
7     7.14430421727287357D-04,     5.71787281789704872D-04,
8     4.69431007606481533D-04,     3.93232835462916638D-04,
9     3.34818889318297664D-04,     2.88952148495751517D-04,
A     2.52211615549573284D-04,     2.22280580798883327D-04,
B     1.97541838033062524D-04,     1.76836855019718004D-04,
C     1.59316899661821081D-04,     1.44347930197333986D-04,
D     1.31448068119965379D-04,     1.20245444949302884D-04,
E     1.10449144504599392D-04,     1.01828770740567258D-04/
 DATA BETA(23), BETA(24), BETA(25), BETA(26), BETA(27), BETA(28),
1     BETA(29), BETA(30), BETA(31), BETA(32), BETA(33), BETA(34),
2     BETA(35), BETA(36), BETA(37), BETA(38), BETA(39), BETA(40),
3     BETA(41), BETA(42), BETA(43), BETA(44)/
4     9.41998224204237509D-05,     8.74130545753834437D-05,
5     8.13466262162801467D-05,     7.59002269646219339D-05,
6     7.09906300634153481D-05,     6.65482874842468183D-05,
```

```
     6.2514695896927 5078D-05,     5.884033944262517 49D-05,
    -1.4928295321342 9172D-03,    -8.782047095463893 28D-04,
    -5.0291654957203 4614D-04,    -2.948221385127460 25D-04,
    -1.7546399697078 2828D-04,    -1.040085504608164 34D-04,
    -5.9614195304645 7895D-05,    -3.120389290760983 40D-05,
    -1.2608973598023 0047D-05,    -2.428926085757303 89D-07,
     8.0599616541427 3571D-06,     1.365070092621473 91D-05,
     1.7396412547292 6261D-05,     1.986727884213378 0D-05/
 DATA BETA(45), BETA(46), BETA(47), BETA(48), BETA(49), BETA(50),
     BETA(51), BETA(52), BETA(53), BETA(54), BETA(55), BETA(56),
     BETA(57), BETA(58), BETA(59), BETA(60), BETA(61), BETA(62),
     BETA(63), BETA(64), BETA(65), BETA(66)/
     2.1446326379082 2639D-05,     2.239546592324565 14D-05,
     2.2896778381471 2629D-05,     2.307853898111778 17D-05,
     2.3032197608090 9144D-05,     2.282360737203487 22D-05,
     2.2500588110529 2418D-05,     2.209810153619914 29D-05,
     2.1641842744810 3905D-05,     2.115076492562208 43D-05,
     2.0638874978217 0737D-05,     2.011652419970816 66D-05,
     1.9591345014117 9244D-05,     1.906893679104367 40D-05,
     1.8553371964163 6667D-05,     1.804757222596742 18D-05,
     5.5221307672129 2790D-04,     4.479325815523846 46D-04,
     2.7952065399202 0589D-04,     1.524681561984466 02D-04,
     6.9327110565704 3598D-05,     1.762586830699913 97D-05/
 DATA BETA(67), BETA(68), BETA(69), BETA(70), BETA(71), BETA(72),
     BETA(73), BETA(74), BETA(75), BETA(76), BETA(77), BETA(78),
     BETA(79), BETA(80), BETA(81), BETA(82), BETA(83), BETA(84),
     BETA(85), BETA(86), BETA(87), BETA(88)/
    -1.3574499634326 9136D-05,    -3.179724133504271 35D-05,
    -4.1886186169669 3365D-05,    -4.690048893791410 29D-05,
    -4.8766544741378 7352D-05,    -4.870100311867350 69D-05,
    -4.7475562089008 6638D-05,    -4.558130581386284 52D-05,
    -4.3330964451126 6036D-05,    -4.092301931577503 64D-05,
    -3.8482263860322 1274D-05,    -3.608571675354105 01D-05,
    -3.3779330612336 7417D-05,    -3.158885607721096 21D-05,
    -2.9526956175080 7315D-05,    -2.759789148283357 59D-05,
    -2.5800617466688 3713D-05,    -2.413083567612802 00D-05,
    -2.2582350951834 6033D-05,    -2.114796567689129 71D-05,
    -1.9820063888529 4927D-05,    -1.859098708010650 77D-05/
 DATA BETA(89), BETA(90), BETA(91), BETA(92), BETA(93), BETA(94),
     BETA(95), BETA(96), BETA(97), BETA(98), BETA(99), BETA(100),
     BETA(101), BETA(102), BETA(103), BETA(104), BETA(105),
     BETA(106), BETA(107), BETA(108), BETA(109), BETA(110)/
    -1.7453269984421 0224D-05,    -1.639978238544979 97D-05,
    -4.7461779655995 9808D-04,    -4.778645671473214 87D-04,
    -3.2039022806703 7603D-04,    -1.611050161199622 82D-04,
    -4.2577810128543 5204D-05,     3.445712942949675 03D-05,
     7.9709268407567 4924D-05,     1.031382367082722 00D-04,
     1.1246677526220 4158D-04,     1.131036421084813 89D-04,
     1.0865163484877 4268D-04,     1.014379515976619 73D-04,
     9.2929839659336 3896D-05,     8.402931330160899 78D-05,
     7.5272799134913 4062D-05,     6.696325219757308 72D-05,
     5.9256454732319 4704D-05,     5.221693088269755 67D-05,
     4.5853948516536 0646D-05,     4.014455138914868 08D-05/
 DATA BETA(111), BETA(112), BETA(113), BETA(114), BETA(115),
     BETA(116), BETA(117), BETA(118), BETA(119), BETA(120),
     BETA(121), BETA(122), BETA(123), BETA(124), BETA(125),
     BETA(126), BETA(127), BETA(128), BETA(129), BETA(130)/
     3.5048173003132 8081D-05,     3.051579950343466 59D-05,
```

```
5     2.64956119950516039D-05,     2.29363633690998152D-05,
6     1.97893056664021636D-05,     1.70091984636412623D-05,
7     1.45547428261524004D-05,     1.23886640995878413D-05,
8     1.04775876076583236D-05,     8.79179954978479373D-06,
9     7.36465810572578444D-04,     8.72790805146193976D-04,
A     6.22614862573135066D-04,     2.85998154194304147D-04,
B     3.84737672879366102D-06,    -1.87906003636971558D-04,
C    -2.97603646594554535D-04,    -3.45998126832656348D-04,
D    -3.53382470916037712D-04,    -3.35715635775048757D-04/
 DATA BETA(131), BETA(132), BETA(133), BETA(134), BETA(135),
1     BETA(136), BETA(137), BETA(138), BETA(139), BETA(140),
2     BETA(141), BETA(142), BETA(143), BETA(144), BETA(145),
3     BETA(146), BETA(147), BETA(148), BETA(149), BETA(150)/
4    -3.04321124789039809D-04,    -2.66722723047612821D-04,
5    -2.27654214122819527D-04,    -1.89922611854562356D-04,
6    -1.55058918599093870D-04,    -1.23778240761873630D-04,
7    -9.62926147717644187D-05,    -7.25178327714425337D-05,
8    -5.22070028895633801D-05,    -3.50347750511900522D-05,
9    -2.06489761035551757D-05,    -8.70106096849767054D-06,
A     1.13698686675100290D-06,     9.16426474122778849D-06,
B     1.56477785428872620D-05,     2.08223629482466847D-05,
C     2.48923381004595156D-05,     2.80340509574146325D-05,
D     3.03987774629861915D-05,     3.21156731406700616D-05/
 DATA BETA(151), BETA(152), BETA(153), BETA(154), BETA(155),
1     BETA(156), BETA(157), BETA(158), BETA(159), BETA(160),
2     BETA(161), BETA(162), BETA(163), BETA(164), BETA(165),
3     BETA(166), BETA(167), BETA(168), BETA(169), BETA(170)/
4    -1.80182191963885708D-03,    -2.43402962938042533D-03,
5    -1.83422663549856802D-03,    -7.62204596354009765D-04,
6     2.39079475256927218D-04,     9.49266117176881141D-04,
7     1.34467449701540359D-03,     1.48457495259449178D-03,
8     1.44732339830617591D-03,     1.30268261285657186D-03,
9     1.10351597375642682D-03,     8.86047440419791759D-04,
A     6.73073208165665473D-04,     4.77603872856582378D-04,
B     3.05991926358789362D-04,     1.60315694594721630D-04,
C     4.00749555270613286D-05,    -5.66607461635251611D-05,
D    -1.32506186772982638D-04,    -1.90296187989614057D-04/
 DATA BETA(171), BETA(172), BETA(173), BETA(174), BETA(175),
1     BETA(176), BETA(177), BETA(178), BETA(179), BETA(180),
2     BETA(181), BETA(182), BETA(183), BETA(184), BETA(185),
3     BETA(186), BETA(187), BETA(188), BETA(189), BETA(190)/
4    -2.32811450376937408D-04,    -2.62628811464668841D-04,
5    -2.82050469867598672D-04,    -2.93081563192861167D-04,
6    -2.97435962176316616D-04,    -2.96557334239348078D-04,
7    -2.91647363312090861D-04,    -2.83696203837734166D-04,
8    -2.73512317095673346D-04,    -2.61750155806768580D-04,
9     6.38585891212050914D-03,     9.62374215806377941D-03,
A     7.61878061207001043D-03,     2.83219055545628054D-03,
B    -2.09841352012720090D-03,    -5.73826764216626498D-03,
C    -7.70804244495414620D-03,    -8.21011692264844401D-03,
D    -7.65824520346905413D-03,    -6.47209729391045177D-03/
 DATA BETA(191), BETA(192), BETA(193), BETA(194), BETA(195),
1     BETA(196), BETA(197), BETA(198), BETA(199), BETA(200),
2     BETA(201), BETA(202), BETA(203), BETA(204), BETA(205),
3     BETA(206), BETA(207), BETA(208), BETA(209), BETA(210)/
4    -4.99132412004966473D-03,    -3.45612289713133280D-03,
5    -2.01785580014170775D-03,    -7.59430686781961401D-04,
6     2.84173631523859138D-04,     1.10891667586337403D-03,
```

```
     7     1.72901493872728771D-03,     2.16812590802684701D-03,
     8     2.45357710494539735D-03,     2.61281821058334862D-03,
     9     2.67141039656276912D-03,     2.65203073395980430D-03,
     A     2.57411652877287315D-03,     2.45389126236094427D-03,
     B     2.30460058071795494D-03,     2.13684837686712662D-03,
     C     1.95896528478870911D-03,     1.77737008679454412D-03,
     D     1.59690280765839059D-03,     1.42111975664438546D-03/
      DATA GAMA(1), GAMA(2), GAMA(3), GAMA(4), GAMA(5), GAMA(6),
     1     GAMA(7), GAMA(8), GAMA(9), GAMA(10), GAMA(11), GAMA(12),
     2     GAMA(13), GAMA(14), GAMA(15), GAMA(16), GAMA(17), GAMA(18),
     3     GAMA(19), GAMA(20), GAMA(21), GAMA(22)/
     4     6.29960524947436582D-01,     2.51984209978974633D-01,
     5     1.54790300415655846D-01,     1.10713062416159013D-01,
     6     8.57309395527394825D-02,     6.97161316958684292D-02,
     7     5.86085671893713576D-02,     5.04698873536310685D-02,
     8     4.42600580689154809D-02,     3.93720661543509966D-02,
     9     3.54283195924455368D-02,     3.21818857502098231D-02,
     A     2.94646240791157679D-02,     2.71581677112934479D-02,
     B     2.51768272973861779D-02,     2.34570755306078891D-02,
     C     2.19508390134907203D-02,     2.06210828235646240D-02,
     D     1.94388240897880846D-02,     1.83810633800683158D-02,
     E     1.74293213231963172D-02,     1.65685837786612353D-02/
      DATA GAMA(23), GAMA(24), GAMA(25), GAMA(26), GAMA(27), GAMA(28),
     1     GAMA(29), GAMA(30)/
     2     1.57865285987918445D-02,     1.50729501494095594D-02,
     3     1.44193250839954639D-02,     1.38184805735341786D-02,
     4     1.32643378994276568D-02,     1.27517121970498651D-02,
     5     1.22761545318762767D-02,     1.18338262398482403D-02/
      DATA EX1, EX2, HPI, GPI, THPI /
     1     3.33333333333333333D-01,     6.66666666666666667D-01,
     2     1.57079632679489662D+00,     3.14159265358979324D+00,
     3     4.71238898038468986D+00/
      DATA ZEROR,ZEROI,CONER,CONEI / 0.0D0, 0.0D0, 1.0D0, 0.0D0 /
C***FIRST EXECUTABLE STATEMENT  ZUNHJ
      RFNU = 1.0D0/FNU
C-----------------------------------------------------------------------
C     OVERFLOW TEST (Z/FNU TOO SMALL)
C-----------------------------------------------------------------------
      TEST = D1MACH(1)*1.0D+3
      AC = FNU*TEST
      IF (ABS(ZR).GT.AC .OR. ABS(ZI).GT.AC) GO TO 15
      ZETA1R = 2.0D0*ABS(LOG(TEST))+FNU
      ZETA1I = 0.0D0
      ZETA2R = FNU
      ZETA2I = 0.0D0
      PHIR = 1.0D0
      PHII = 0.0D0
      ARGR = 1.0D0
      ARGI = 0.0D0
      RETURN
   15 CONTINUE
      ZBR = ZR*RFNU
      ZBI = ZI*RFNU
      RFNU2 = RFNU*RFNU
C-----------------------------------------------------------------------
C     COMPUTE IN THE FOURTH QUADRANT
C-----------------------------------------------------------------------
      FN13 = FNU**EX1
```

```
      FN23 = FN13*FN13
      RFN13 = 1.0D0/FN13
      W2R = CONER - ZBR*ZBR + ZBI*ZBI
      W2I = CONEI - ZBR*ZBI - ZBR*ZBI
      AW2 = ZABS(W2R,W2I)
      IF (AW2.GT.0.25D0) GO TO 130
C-----------------------------------------------------------------------
C     POWER SERIES FOR ABS(W2).LE.0.25D0
C-----------------------------------------------------------------------
      K = 1
      PR(1) = CONER
      PI(1) = CONEI
      SUMAR = GAMA(1)
      SUMAI = ZEROI
      AP(1) = 1.0D0
      IF (AW2.LT.TOL) GO TO 20
      DO 10 K=2,30
        PR(K) = PR(K-1)*W2R - PI(K-1)*W2I
        PI(K) = PR(K-1)*W2I + PI(K-1)*W2R
        SUMAR = SUMAR + PR(K)*GAMA(K)
        SUMAI = SUMAI + PI(K)*GAMA(K)
        AP(K) = AP(K-1)*AW2
        IF (AP(K).LT.TOL) GO TO 20
   10 CONTINUE
      K = 30
   20 CONTINUE
      KMAX = K
      ZETAR = W2R*SUMAR - W2I*SUMAI
      ZETAI = W2R*SUMAI + W2I*SUMAR
      ARGR = ZETAR*FN23
      ARGI = ZETAI*FN23
      CALL ZSQRT(SUMAR, SUMAI, ZAR, ZAI)
      CALL ZSQRT(W2R, W2I, STR, STI)
      ZETA2R = STR*FNU
      ZETA2I = STI*FNU
      STR = CONER + EX2*(ZETAR*ZAR-ZETAI*ZAI)
      STI = CONEI + EX2*(ZETAR*ZAI+ZETAI*ZAR)
      ZETA1R = STR*ZETA2R - STI*ZETA2I
      ZETA1I = STR*ZETA2I + STI*ZETA2R
      ZAR = ZAR + ZAR
      ZAI = ZAI + ZAI
      CALL ZSQRT(ZAR, ZAI, STR, STI)
      PHIR = STR*RFN13
      PHII = STI*RFN13
      IF (IPMTR.EQ.1) GO TO 120
C-----------------------------------------------------------------------
C     SUM SERIES FOR ASUM AND BSUM
C-----------------------------------------------------------------------
      SUMBR = ZEROR
      SUMBI = ZEROI
      DO 30 K=1,KMAX
        SUMBR = SUMBR + PR(K)*BETA(K)
        SUMBI = SUMBI + PI(K)*BETA(K)
   30 CONTINUE
      ASUMR = ZEROR
      ASUMI = ZEROI
      BSUMR = SUMBR
      BSUMI = SUMBI
```

```
      L1 = 0
      L2 = 30
      BTOL = TOL*(ABS(BSUMR)+ABS(BSUMI))
      ATOL = TOL
      PP = 1.0D0
      IAS = 0
      IBS = 0
      IF (RFNU2.LT.TOL) GO TO 110
      DO 100 IS=2,7
        ATOL = ATOL/RFNU2
        PP = PP*RFNU2
        IF (IAS.EQ.1) GO TO 60
        SUMAR = ZEROR
        SUMAI = ZEROI
        DO 40 K=1,KMAX
          M = L1 + K
          SUMAR = SUMAR + PR(K)*ALFA(M)
          SUMAI = SUMAI + PI(K)*ALFA(M)
          IF (AP(K).LT.ATOL) GO TO 50
   40   CONTINUE
   50   CONTINUE
        ASUMR = ASUMR + SUMAR*PP
        ASUMI = ASUMI + SUMAI*PP
        IF (PP.LT.TOL) IAS = 1
   60   CONTINUE
        IF (IBS.EQ.1) GO TO 90
        SUMBR = ZEROR
        SUMBI = ZEROI
        DO 70 K=1,KMAX
          M = L2 + K
          SUMBR = SUMBR + PR(K)*BETA(M)
          SUMBI = SUMBI + PI(K)*BETA(M)
          IF (AP(K).LT.ATOL) GO TO 80
   70   CONTINUE
   80   CONTINUE
        BSUMR = BSUMR + SUMBR*PP
        BSUMI = BSUMI + SUMBI*PP
        IF (PP.LT.BTOL) IBS = 1
   90   CONTINUE
        IF (IAS.EQ.1 .AND. IBS.EQ.1) GO TO 110
        L1 = L1 + 30
        L2 = L2 + 30
  100 CONTINUE
  110 CONTINUE
      ASUMR = ASUMR + CONER
      PP = RFNU*RFN13
      BSUMR = BSUMR*PP
      BSUMI = BSUMI*PP
  120 CONTINUE
      RETURN
C-----------------------------------------------------------------------
C     ABS(W2).GT.0.25D0
C-----------------------------------------------------------------------
  130 CONTINUE
      CALL ZSQRT(W2R, W2I, WR, WI)
      IF (WR.LT.0.0D0) WR = 0.0D0
      IF (WI.LT.0.0D0) WI = 0.0D0
      STR = CONER + WR
```

```
      STI = WI
      CALL ZDIV(STR, STI, ZBR, ZBI, ZAR, ZAI)
      CALL ZLOG(ZAR, ZAI, ZCR, ZCI, IDUM)
      IF (ZCI.LT.0.0D0) ZCI = 0.0D0
      IF (ZCI.GT.HPI) ZCI = HPI
      IF (ZCR.LT.0.0D0) ZCR = 0.0D0
      ZTHR = (ZCR-WR)*1.5D0
      ZTHI = (ZCI-WI)*1.5D0
      ZETA1R = ZCR*FNU
      ZETA1I = ZCI*FNU
      ZETA2R = WR*FNU
      ZETA2I = WI*FNU
      AZTH = ZABS(ZTHR,ZTHI)
      ANG = THPI
      IF (ZTHR.GE.0.0D0 .AND. ZTHI.LT.0.0D0) GO TO 140
      ANG = HPI
      IF (ZTHR.EQ.0.0D0) GO TO 140
      ANG = DATAN(ZTHI/ZTHR)
      IF (ZTHR.LT.0.0D0) ANG = ANG + GPI
  140 CONTINUE
      PP = AZTH**EX2
      ANG = ANG*EX2
      ZETAR = PP*COS(ANG)
      ZETAI = PP*SIN(ANG)
      IF (ZETAI.LT.0.0D0) ZETAI = 0.0D0
      ARGR = ZETAR*FN23
      ARGI = ZETAI*FN23
      CALL ZDIV(ZTHR, ZTHI, ZETAR, ZETAI, RTZTR, RTZTI)
      CALL ZDIV(RTZTR, RTZTI, WR, WI, ZAR, ZAI)
      TZAR = ZAR + ZAR
      TZAI = ZAI + ZAI
      CALL ZSQRT(TZAR, TZAI, STR, STI)
      PHIR = STR*RFN13
      PHII = STI*RFN13
      IF (IPMTR.EQ.1) GO TO 120
      RAW = 1.0D0/SQRT(AW2)
      STR = WR*RAW
      STI = -WI*RAW
      TFNR = STR*RFNU*RAW
      TFNI = STI*RFNU*RAW
      RAZTH = 1.0D0/AZTH
      STR = ZTHR*RAZTH
      STI = -ZTHI*RAZTH
      RZTHR = STR*RAZTH*RFNU
      RZTHI = STI*RAZTH*RFNU
      ZCR = RZTHR*AR(2)
      ZCI = RZTHI*AR(2)
      RAW2 = 1.0D0/AW2
      STR = W2R*RAW2
      STI = -W2I*RAW2
      T2R = STR*RAW2
      T2I = STI*RAW2
      STR = T2R*C(2) + C(3)
      STI = T2I*C(2)
      UPR(2) = STR*TFNR - STI*TFNI
      UPI(2) = STR*TFNI + STI*TFNR
      BSUMR = UPR(2) + ZCR
      BSUMI = UPI(2) + ZCI
```

```
      ASUMR = ZEROR
      ASUMI = ZEROI
      IF (RFNU.LT.TOL) GO TO 220
      PRZTHR = RZTHR
      PRZTHI = RZTHI
      PTFNR = TFNR
      PTFNI = TFNI
      UPR(1) = CONER
      UPI(1) = CONEI
      PP = 1.0D0
      BTOL = TOL*(ABS(BSUMR)+ABS(BSUMI))
      KS = 0
      KP1 = 2
      L = 3
      IAS = 0
      IBS = 0
      DO 210 LR=2,12,2
        LRP1 = LR + 1
C-----------------------------------------------------------------------
C     COMPUTE TWO ADDITIONAL CR, DR, AND UP FOR TWO MORE TERMS IN
C     NEXT SUMA AND SUMB
C-----------------------------------------------------------------------
        DO 160 K=LR,LRP1
          KS = KS + 1
          KP1 = KP1 + 1
          L = L + 1
          ZAR = C(L)
          ZAI = ZEROI
          DO 150 J=2,KP1
            L = L + 1
            STR = ZAR*T2R - T2I*ZAI + C(L)
            ZAI = ZAR*T2I + ZAI*T2R
            ZAR = STR
  150     CONTINUE
          STR = PTFNR*TFNR - PTFNI*TFNI
          PTFNI = PTFNR*TFNI + PTFNI*TFNR
          PTFNR = STR
          UPR(KP1) = PTFNR*ZAR - PTFNI*ZAI
          UPI(KP1) = PTFNI*ZAR + PTFNR*ZAI
          CRR(KS) = PRZTHR*BR(KS+1)
          CRI(KS) = PRZTHI*BR(KS+1)
          STR = PRZTHR*RZTHR - PRZTHI*RZTHI
          PRZTHI = PRZTHR*RZTHI + PRZTHI*RZTHR
          PRZTHR = STR
          DRR(KS) = PRZTHR*AR(KS+2)
          DRI(KS) = PRZTHI*AR(KS+2)
  160   CONTINUE
        PP = PP*RFNU2
        IF (IAS.EQ.1) GO TO 180
        SUMAR = UPR(LRP1)
        SUMAI = UPI(LRP1)
        JU = LRP1
        DO 170 JR=1,LR
          JU = JU - 1
          SUMAR = SUMAR + CRR(JR)*UPR(JU) - CRI(JR)*UPI(JU)
          SUMAI = SUMAI + CRR(JR)*UPI(JU) + CRI(JR)*UPR(JU)
  170   CONTINUE
        ASUMR = ASUMR + SUMAR
```

```
        ASUMI = ASUMI + SUMAI
        TEST = ABS(SUMAR) + ABS(SUMAI)
        IF (PP.LT.TOL .AND. TEST.LT.TOL) IAS = 1
  180   CONTINUE
        IF (IBS.EQ.1) GO TO 200
        SUMBR = UPR(LR+2) + UPR(LRP1)*ZCR - UPI(LRP1)*ZCI
        SUMBI = UPI(LR+2) + UPR(LRP1)*ZCI + UPI(LRP1)*ZCR
        JU = LRP1
        DO 190 JR=1,LR
          JU = JU - 1
          SUMBR = SUMBR + DRR(JR)*UPR(JU) - DRI(JR)*UPI(JU)
          SUMBI = SUMBI + DRR(JR)*UPI(JU) + DRI(JR)*UPR(JU)
  190   CONTINUE
        BSUMR = BSUMR + SUMBR
        BSUMI = BSUMI + SUMBI
        TEST = ABS(SUMBR) + ABS(SUMBI)
        IF (PP.LT.BTOL .AND. TEST.LT.BTOL) IBS = 1
  200   CONTINUE
        IF (IAS.EQ.1 .AND. IBS.EQ.1) GO TO 220
  210 CONTINUE
  220 CONTINUE
      ASUMR = ASUMR + CONER
      STR = -BSUMR*RFN13
      STI = -BSUMI*RFN13
      CALL ZDIV(STR, STI, RTZTR, RTZTI, BSUMR, BSUMI)
      GO TO 120
      END
*DECK ZACAI
      SUBROUTINE ZACAI (ZR, ZI, FNU, KODE, MR, N, YR, YI, NZ, RL, TOL,
     +   ELIM, ALIM)
C***BEGIN PROLOGUE  ZACAI
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZAIRY
C***LIBRARY   SLATEC
C***TYPE      ALL (CACAI-A, ZACAI-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     ZACAI APPLIES THE ANALYTIC CONTINUATION FORMULA
C
C         K(FNU,ZN*EXP(MP))=K(FNU,ZN)*EXP(-MP*FNU) - MP*I(FNU,ZN)
C                 MP=PI*MR*CMPLX(0.0,1.0)
C
C     TO CONTINUE THE K FUNCTION FROM THE RIGHT HALF TO THE LEFT
C     HALF Z PLANE FOR USE WITH ZAIRY WHERE FNU=1/3 OR 2/3 AND N=1.
C     ZACAI IS THE SAME AS ZACON WITH THE PARTS FOR LARGER ORDERS AND
C     RECURRENCE REMOVED. A RECURSIVE CALL TO ZACON CAN RESULT IF ZACON
C     IS CALLED FROM ZAIRY.
C
C***SEE ALSO  ZAIRY
C***ROUTINES CALLED  D1MACH, ZABS, ZASYI, ZBKNU, ZMLRI, ZS1S2, ZSERI
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C***END PROLOGUE  ZACAI
C     COMPLEX CSGN,CSPN,C1,C2,Y,Z,ZN,CY
      DOUBLE PRECISION ALIM, ARG, ASCLE, AZ, CSGNR, CSGNI, CSPNR,
     * CSPNI, C1R, C1I, C2R, C2I, CYR, CYI, DFNU, ELIM, FMR, FNU, PI,
```

```
     * RL, SGN, TOL, YY, YR, YI, ZR, ZI, ZNR, ZNI, D1MACH, ZABS
      INTEGER INU, IUF, KODE, MR, N, NN, NW, NZ
      DIMENSION YR(N), YI(N), CYR(2), CYI(2)
      EXTERNAL ZABS
      DATA PI / 3.14159265358979324D0 /
C***FIRST EXECUTABLE STATEMENT  ZACAI
      NZ = 0
      ZNR = -ZR
      ZNI = -ZI
      AZ = ZABS(ZR,ZI)
      NN = N
      DFNU = FNU + (N-1)
      IF (AZ.LE.2.0D0) GO TO 10
      IF (AZ*AZ*0.25D0.GT.DFNU+1.0D0) GO TO 20
   10 CONTINUE
C-----------------------------------------------------------------------
C     POWER SERIES FOR THE I FUNCTION
C-----------------------------------------------------------------------
      CALL ZSERI(ZNR, ZNI, FNU, KODE, NN, YR, YI, NW, TOL, ELIM, ALIM)
      GO TO 40
   20 CONTINUE
      IF (AZ.LT.RL) GO TO 30
C-----------------------------------------------------------------------
C     ASYMPTOTIC EXPANSION FOR LARGE Z FOR THE I FUNCTION
C-----------------------------------------------------------------------
      CALL ZASYI(ZNR, ZNI, FNU, KODE, NN, YR, YI, NW, RL, TOL, ELIM,
     * ALIM)
      IF (NW.LT.0) GO TO 80
      GO TO 40
   30 CONTINUE
C-----------------------------------------------------------------------
C     MILLER ALGORITHM NORMALIZED BY THE SERIES FOR THE I FUNCTION
C-----------------------------------------------------------------------
      CALL ZMLRI(ZNR, ZNI, FNU, KODE, NN, YR, YI, NW, TOL)
      IF(NW.LT.0) GO TO 80
   40 CONTINUE
C-----------------------------------------------------------------------
C     ANALYTIC CONTINUATION TO THE LEFT HALF PLANE FOR THE K FUNCTION
C-----------------------------------------------------------------------
      CALL ZBKNU(ZNR, ZNI, FNU, KODE, 1, CYR, CYI, NW, TOL, ELIM, ALIM)
      IF (NW.NE.0) GO TO 80
      FMR = MR
      SGN = -DSIGN(PI,FMR)
      CSGNR = 0.0D0
      CSGNI = SGN
      IF (KODE.EQ.1) GO TO 50
      YY = -ZNI
      CSGNR = -CSGNI*SIN(YY)
      CSGNI = CSGNI*COS(YY)
   50 CONTINUE
C-----------------------------------------------------------------------
C     CALCULATE CSPN=EXP(FNU*PI*I) TO MINIMIZE LOSSES OF SIGNIFICANCE
C     WHEN FNU IS LARGE
C-----------------------------------------------------------------------
      INU = FNU
      ARG = (FNU-INU)*SGN
      CSPNR = COS(ARG)
      CSPNI = SIN(ARG)
```

```
      IF (MOD(INU,2).EQ.0) GO TO 60
      CSPNR = -CSPNR
      CSPNI = -CSPNI
   60 CONTINUE
      C1R = CYR(1)
      C1I = CYI(1)
      C2R = YR(1)
      C2I = YI(1)
      IF (KODE.EQ.1) GO TO 70
      IUF = 0
      ASCLE = 1.0D+3*D1MACH(1)/TOL
      CALL ZS1S2(ZNR, ZNI, C1R, C1I, C2R, C2I, NW, ASCLE, ALIM, IUF)
      NZ = NZ + NW
   70 CONTINUE
      YR(1) = CSPNR*C1R - CSPNI*C1I + CSGNR*C2R - CSGNI*C2I
      YI(1) = CSPNR*C1I + CSPNI*C1R + CSGNR*C2I + CSGNI*C2R
      RETURN
   80 CONTINUE
      NZ = -1
      IF(NW.EQ.(-2)) NZ=-2
      RETURN
      END
*DECK ZS1S2
      SUBROUTINE ZS1S2 (ZRR, ZRI, S1R, S1I, S2R, S2I, NZ, ASCLE, ALIM,
     +   IUF)
C***BEGIN PROLOGUE  ZS1S2
C***SUBSIDIARY
C***PURPOSE  Subsidiary to ZAIRY and ZBESK
C***LIBRARY   SLATEC
C***TYPE      ALL (CS1S2-A, ZS1S2-A)
C***AUTHOR  Amos, D. E., (SNL)
C***DESCRIPTION
C
C     ZS1S2 TESTS FOR A POSSIBLE UNDERFLOW RESULTING FROM THE
C     ADDITION OF THE I AND K FUNCTIONS IN THE ANALYTIC CON-
C     TINUATION FORMULA WHERE S1=K FUNCTION AND S2=I FUNCTION.
C     ON KODE=1 THE I AND K FUNCTIONS ARE DIFFERENT ORDERS OF
C     MAGNITUDE, BUT FOR KODE=2 THEY CAN BE OF THE SAME ORDER
C     OF MAGNITUDE AND THE MAXIMUM MUST BE AT LEAST ONE
C     PRECISION ABOVE THE UNDERFLOW LIMIT.
C
C***SEE ALSO  ZAIRY, ZBESK
C***ROUTINES CALLED  ZABS, ZEXP, ZLOG
C***REVISION HISTORY  (YYMMDD)
C   830501  DATE WRITTEN
C   910415  Prologue converted to Version 4.0 format.  (BAB)
C   930122  Added ZEXP and ZLOG to EXTERNAL statement.  (RWC)
C***END PROLOGUE  ZS1S2
C     COMPLEX CZERO,C1,S1,S1D,S2,ZR
      DOUBLE PRECISION AA, ALIM, ALN, ASCLE, AS1, AS2, C1I, C1R, S1DI,
     * S1DR, S1I, S1R, S2I, S2R, ZEROI, ZEROR, ZRI, ZRR, ZABS
      INTEGER IUF, IDUM, NZ
      EXTERNAL ZABS, ZEXP, ZLOG
      DATA ZEROR,ZEROI  / 0.0D0 , 0.0D0 /
C***FIRST EXECUTABLE STATEMENT  ZS1S2
      NZ = 0
      AS1 = ZABS(S1R,S1I)
      AS2 = ZABS(S2R,S2I)
```

```
      IF (S1R.EQ.0.0D0 .AND. S1I.EQ.0.0D0) GO TO 10
      IF (AS1.EQ.0.0D0) GO TO 10
      ALN = -ZRR - ZRR + LOG(AS1)
      S1DR = S1R
      S1DI = S1I
      S1R = ZEROR
      S1I = ZEROI
      AS1 = ZEROR
      IF (ALN.LT.(-ALIM)) GO TO 10
      CALL ZLOG(S1DR, S1DI, C1R, C1I, IDUM)
      C1R = C1R - ZRR - ZRR
      C1I = C1I - ZRI - ZRI
      CALL ZEXP(C1R, C1I, S1R, S1I)
      AS1 = ZABS(S1R,S1I)
      IUF = IUF + 1
   10 CONTINUE
      AA = MAX(AS1,AS2)
      IF (AA.GT.ASCLE) RETURN
      S1R = ZEROR
      S1I = ZEROI
      S2R = ZEROR
      S2I = ZEROI
      NZ = 1
      IUF = 0
      RETURN
      END



!*******************************************************************************
      subroutine mtdef(Btot,mustar,sig,poros)
      implicit none
      include "dimen.inc"
      integer i,j
      double precision alpha(nm),c1(nr),s1(nr,nm),bet(nm)
      double precision r0i,r0p,Rm,Qin,Qout,poros,b1,b2,concin,Timein,
     &  Btot,mustar,sig,rmax,Ro,ptot,Qins,Qouts,Timeins,rmaxs,disc,
     &  Tc0,Tc0s,TcE,TcEs,Trest,pmat,relerr
      character*40 distribution
      common/a/r0i,r0p,Qin,Qout,b1,b2,concin,Timein,
     &  Ro,disc,relerr,Tc0,TcE,Rm,Trest,pmat,rmax
      common/bet/bet
      common/g/alpha,c1,s1
      common/userdefine/distribution

c IF THE USER WISHES TO DEFINE THEIR OWN DISTRIBUTION OF MASS TRANSFER
C RATES, THEN THE PARAMETER idef MUST BE SET TO 1 IN THE INPUT FILE,
C AND THE ARRAYS beta AND alpha MUST BE GIVEN.  NORMALLY, THESE ARRAYS
C WILL BE GIVEN IN THE LAST FILE LISTED IN flname.dat.  THE USER
C THEREFORE MUST GENERALLY USE A PREPROCESSOR TO GENERATE THE NON-
C LOGNORMAL DISTRIBUTION.  OPTIONALLY, THE USER MAY EDIT THE CODE IN
C THIS SUBROUTINE TO "HARDWIRE" HIS OR HER OWN DISTRIBUTION, COMMENTING
C OUT THE READ STATEMENTS, ETC. GIVEN BELOW.

c Key parameter definitions are given here (i.e., parameters that might
c be used in defining alpha and beta).
```

```
c alpha = array of first-order mass transfer coefficients
c b1 = thickness of aquifer at injection well
c b2 = thickness of aquifer at pumping well
c bet = array of capacity coefficients associated with alphas
c Btot = total capacity coefficient, equal to sum of beta
c mustar = parameter to be used in defining distribution of alpha, beta
c pmat = total matrix porosity
c poros = total fracture porosity
c Qin = injection rate (2-well case)
c Qins = injection reate (single-well case)
c Qout = pumping rate (2-well case)
c Qouts = pumping rate (single-well case)
c Rm = retardation due to sorption along fracture wall (NOT in matrix)
c sig = parameter to be used in defining distribution of alpha, beta

c  EXAMPLE, setting all alpha = 1.d-3 and bet = equal fractions of
c  total capacity coefficient.  This is mathematically equivalent to
c  a single, first-order rate coefficient with beta = Btot.
c
c     do 10,i=1,nm
c      alpha(i) = 1.d-3
c      bet(i) = Btot/dble(nm)
c10   continue


c READING IN user-defined distribution of rate coefficients.  There must
c be nm values given in the input file for alpha and beta.  Comment out
c next 5 lines if you wish to hardwire a distribution right here:
       open (33,file=distribution)
       do 20,i=1,nm
         read(33,*) alpha(i),bet(i)
 20    continue
       close(33)


      return
      end
```

Appendix D

FORTRAN Code Manual

GRTCFTT_FD

Generalized Radial Transport Model for Interpreting Convergent Flow Tracer Tests – a Laplace Domain Finite Difference Solution

A FORTRAN code for modeling Convergent Flow Transport through an n-dimensional conduits composed of numerous sizes fracture-bounded Blocks

Version - 1.0

Md Lal Mamud
M.S. in Engineering Science (Hydrology)
Department of Geology and Geological Engineering
The University of Mississippi

Dr. Robert M. Holt
Professor
Department of Geology and Geological Engineering
The University of Mississippi

May 2019

1. Introduction

The program GRTCFTT_FD v1.0 was developed and written in FORTRAN 90 using Intel® Parallel Studio XE Composer Edition Fortran which includes the Intel® Fortran compiler and the Intel® Math Kernel Library and Windows 10 Operating System. The program was developed as sets of subroutines and functions. IMSL subroutines (International Mathematics and Statistics Libraries) have also been used to develop the code.

Subroutines and functions that are used in Injection and resting phase parts of the program were modified from STAMMT-R developed by Roy Haggerty and Sean W. Fleming in 1997 (Haggerty et al., 2000). Pumping phase part of the program was written by Robert M. Holt and Lal Mamud. Main subroutines are listed below with brief description.

The GRTCFTT_FD was developed for modeling convergent flow tracer test in fractured rock exhibiting multiple-rate of diffusion. The program can simulate different types of transport by changing some logical input (please comments in the params.dat file) in the **params.dat** input file. The program will run to estimate parameter if the **iest** value in the **params.dat** file is 0. It is recommended not to estimate parameters using this version 1.0 of the code.

The program can do followings forward simulations:

a. Combining Injection, Resting and Withdrawal Phase
    1. Transport without diffusion for pulse type source
    2. Transport without diffusion for continuous source.
    3. Transport with single-rate diffusion f for pulse type source
    4. Transport with single-rate diffusion for continuous source.
    5. Transport with multi-rate diffusion for pulse type source.
    6. Transport with multi-rate diffusion for continuous source.

b. Only Pumping Phase

    1. Transport without diffusion for pulse type source
    2. Transport without diffusion for continuous source.
    3. Transport with single-rate diffusion f for pulse type source
    4. Transport with single-rate diffusion for continuous source.
    5. Transport with multi-rate diffusion for pulse type source.
    6. Transport with multi-rate diffusion for continuous source.

In addition to using IMSL, three file are required to put in the directory to compile the GRTCFTT_FD program. The three file are **GRTCFTT_FD.f, dimen.inc** and **typscom.inc**. The **GRTCFTT_FD.f** is the main program and **dimen.inc** contains data related to array size. Any change to the **dimen.inc** file requires to recompilation of the program. The **typscom.inc** contains data and character specification, and common blocks.

2. Program GRTCFTT_FD

The GRTCFTT_FD is the main routine for following task.

1. Calling IMSL for calculation
2. Defining parameters and character types
3. Making common blocks
4. Reading File names from the **flname.dat** input file
5. Reading Parameter from the **params.dat** input file
6. Reading Tracer recovery data from the **trecove.dat** file
7. Calling the **FHeader.for** subroutine for writing file headers
8. Directing the code either for parameters estimation or forward simulation.

9. Calling the **inversion.for** to calculate inversion statistics in the case of estimation,
10. Calling either the **btc.for** subroutines to calculate the breakthrough curve
11. Calling the **distout.for** in the case of lognormal distributions of block radii and diffusion rate coefficients
12. Closing all open files

After successful compilation and run of the **GRTCFTT_FD** for forward simulation**,** the user should see the following command window showing messages and summary of simulation (Figure 1).

```
Non-dimensionalizing parameters for the Injecton Phase.
   > injecting for the CFTT simulation.
Done injection Phase...!!!.
    Doing Averaging and Coordinates Transformation
Done Averaging and Coordinates Transformation...!
Non-dimensionalizing parameters for the Pumping Phase.
   < Pumping for the CFTT simulation.
Congratulations...!!!  Done pumping for the  CFTT simulation.......!!!
CFTT Simulation Summary:
-------------------------------------------------
   n =   1.80000000000000
  Pe =   100.000000000000
  AP =  1.000000000000000E-002
  DP =  0.100000000000000
Btot =   10.0000000000000
  mu =  -6.90780000000000
Sig =   3.00000000000000
********************************************************************************
```

Figure 1. Sample Command window after successful compilation and running the GRTCFTT_FD program

3. Major Subroutines used in GRTCFTT_FD

Major subroutines and function that are used in the GRTCFTT_FD program are given below with short description.

3.1. Subroutine dunlsf (from IMSL)

This subroutine estimates the parameters using nonlinear least-squares problem using a modified Levenberg-Marquardt algorithm. The main program will call this subroutine if the **iest** value is 0 in the **params.dat** input file otherwise the code will do forward simulation. Flow dimension, geometric mean of the diffusion rate coefficients, standard deviation of the log-transformed diffusion rate coefficient, advective porosity and longitudinal dispersivity will be estimated.

3.2. Subroutine obj

The **dunlsf** subroutine will call the **obj** subroutine in the case of parameter estimation. This subroutine will call the **btc** subroutine and calculate optimum values of estimated parameter.

3.3. Subroutine btc

This subroutine will calculate breakthrough curves by calling of the others subroutines and functions for injection, resting and pumping phase. This subroutine also non-dimensionalizes parameters for injection and pumping phase separately. And, it also writes output files.

3.4. Subroutine dinlap (from IMSL)

This subroutine calculates the inverse Laplace transform of a Laplace domain into time domain using De Hoog algorithm. It is called in injection, resting and pumping phase for inversion

3.5. Subroutine dcsint (from IMSL)

This subroutine does the cubic spline interpolation. It is used, in conjunction with dcsval(3.6) in finding a continuous function running through a known number of points. Both of them are required at several points within the program.

3.6. Subroutine dcsval (from IMSL)

This subroutine computes an interpolated value between points known along a function, based on a cubic spline.

3.7. Subroutines dqdag, dq2ag (from IMSL)

These subroutines integrate a function using Gauss-Kronrod rules. These two functions are used within the program for doing azimuthal averaging and coordinate transformation.

3.8. Subroutine avint

When there is no continuous form of the function is available, but only a number of points along the function the the avint numerically integrates a function. This function is called for lognormal distributions of block radii and diffusion rate coefficients.

3.9. Function pushcL

This fucntion is called by **dinlap** and Laplace-domain-concentrations in the mobile zone for the injection phase. It calculates concentration as a function of the complex Laplace parameter p, and is called by the IMSL

3.10. Function pushsL

This function is called by **dinlap** and Laplace-domain-concentrations in the immobile zone for the injection phase. It calculates concentration as a function of the complex Laplace parameter p, and is called by the IMSL

3.11. Function restsL

This function is called by **dinlap** and Laplace-domain-concentrations in the mobile zone for the resting phase. It calculates concentration as a function of the complex Laplace parameter p, and is called by the IMSL

3.12. Function restsL

This function is called by **dinlap** and Laplace-domain-concentrations in the immobile zone for the resting phase. It calculates concentration as a function of the complex Laplace parameter p, and is called by the IMSL

3.13. Function crt

This function does azimuthally averaging the concentrations for the convergent flow tracer test after the injection and resting period. It calculates concentration as a function of the complex Laplace parameter p, and is called by the IMSL

3.14. Subroutine calc_mass

This subroutine calculates the total amount of mass contained in the aquifer by integrating under the current concentration profile.

3.15. Subroutine distout

This subroutine calculates and writes to **arcdf.out** file distributions of diffusion rate coefficients and block radii. This subroutine calls the subroutine **avint**

3.15. Subroutine mtdef

This subroutine read from the **distr.dat** file and uses a user-defined discrete distribution of first-order mass transfer rate coefficients, rather than the lognormal distribution of diffusion rate coefficients or single-rate diffusion coefficient which GRTCFTT_FD normally uses.

3.16. Subroutines zairy, zbiry

These subroutines calculate the value of the first and second Airy functions, Ai(x) and Bi(x), respectively. The first derivatives of these Airy functions, Ai'(x) and Bi'(x) are calculated by this subroutine. These subroutines contain machine constants for many different computers. These lines in the subroutines are all commented out, except for those lines which give the machine constants appropriate for the type of computer on which the code is compiled and run.

3.17. Function pullcLG

This function calculates the value of the Laplace-domain-concentrations in the for convergent flow in the for the finite difference solution. It calculates concentration as a function of the complex Laplace parameter p, and is called by the IMSL

4. Input and Output Files

The GRTCFTT_FD requires four input files and creates ten output files.

a. Input Files:

1. flname.dat: this file contains all of the file names to be read and create by the program.
2. params.dat: containing aquifer and tracer test parameters.
3. trecov.dat: containing tracer recovery data.
4. distr.dat: containing user-defined discrete distribution of first-order mass transfer rate coefficients.

b. Output Files:

1. cbtc.out: This file contains simulated breakthrough data time vs. concentration.
2. abboti.out: containing all alpha, Beta and Btot for the Injection Phase.
3. abbotp.out: containing all alpha, Beta and Btot for the pumping Phase.
4. cinjd.out: containing concentrations as a function of radial distance from the injection well at the end of injection.
5. cazav.out: containing Azimuthally Averaged concentrations as function of radial distance from the pumping well.
6. crest.out: containing concentrations at the end of the rest period for the CFTT run.
7. estcp.out: containing progress of estimated parameter.
8. estfp.out: containining current estimated values of parameters and breakthrough curve.
9. stats.out: containing inversion statistics if estimation is done.
10. arcdf.out: containing discrete values of the cumulative density functions describing the

diffusion rate coefficients and block radii.

5. Error Messages

GRTCFTT_FD might show following error messages on the command window. Brief description of error messages are given below:

(1) ERROR: TNM is greater than ntm
Modify TNM, or recompile (and re-QA) with larger ntm
Aborting STAMMT-R run

This error message indicates that the number of data given for the CFTT in File1 is greater than dimensions of code. The only solutions are to reduce the number of data or to redimension the code

(2) ERROR: first time in 2-well data is < Timein
Aborting run
Modify data and/or input parameter files

This error message indicates that the first time in CFTT data set is before injection ends at the injection well. Parameters must be modified to meet these conditions.

(3) ERROR: rmax for multiwell simulation is too large

This error message indicates that rmax is too large for the CFTTsimulation. Reduce the value of rmax.

6. Sample Simulation

It is expected that user can re-produce the following breakthrough curve (Figure 3) using the given param.dat input file (Figure 2) using the GRTCFTT_FD, if all of the things are in correct setting.

```
0                 skipm  !do CFTT simulation? 0 if yes, 1 if no
0.0000d0          Tc0    !start time of solute injection (can be 0), MW [T]
0.3000d0          TcE    !elapsed time from t=0 to end of solute inj., MW [T]
0.5000d0          Timein !elapsed time from t=0 to end of chaser inj., MW [T]
0.5000d0          Qin    !injection rate, MW [L^3/T]
1.000d0           Qout   !pumping rate, MW [L^3/T]
0.10D+00          alphLm !longitudinal dispersivity, MW [L]
3.0d0             rmax   !edge of grid for injection, MW [L]
10.0d0            Ro     !distance from injection to pumping wells, MW [L]
0.01000d0         r0i    !well radius (injection, MW) [L]
1                 MWtime !use (t,C) input file (set=0) or generate times (set=1), MW
1                 MWz    !if MWtime=1: constant time (=0) or constant ln(time) (=1) MW
960               MWpumpt!if MWtime=1: time from Timein to end of pumping [T], MW
100               TNM    !number of time vs concentration data points, MW
0                 Trest  ! Resting Time for the CFTT

0.00000d0        r0p    !radius of the pumping well  [L]
5.000d0          b1     !saturated thickness at injection well [L]
5.00d0           b2     !saturated thickness at pumping well [L]
1.d0             Cin    !injection conc. [M/L^3; if Cin=1, conc. dimensionless]
-6.9078D00       mus    !initial guess (parameter estimation) mus [ln(1/T)]
3.000d0          sig    !initial guess (parameter estimation) sig
1.0000D+00       Rm     !mobile zone retardation [-]
1.0000D+00       Rim    !immobile zone retardation [-]
0.20d0           ptot   !maximum permitted total porosity (inversion parameter) [-]
2.628d-6         Daq    !aqueous diffusion coefficient of solute [L^/T]
0.11             tort   !diffusive tortuosity [-]
0.01             dr     ! Spatial descritization
1.8              fdn    ! flow dimenison
0.01             poros  !advective porosity [-]
0.10             pmat   !diffusive porosity


T               cinject        ! pulse Source (T), else (F)
F               SingR          ! If Single-Rate (T), else (F)
T               MultR          ! If Multi-Rate (T), else (F)
F               FixOmBet       ! If Input Omega and Beta (T), else (F)
F               contsrc        ! Continuous source (T), else (F)
F               pumpSol        ! Only Pumping Phase Solution (T), else (F)
1.0d-2          tmin           ! Minimum Time [T]
1.0d+3          tmax           ! Minimum Time [T]
0.0             omegapf        ! Dimensionless Mass Transfer Rate Coeff.
0.0             betompf        ! Dimensionless Capacity Coeff.
0.01            alphar         ! Dimensional Mass Transfer Rate Coeff.


1               iest   !0=parameter estimation; 1=forward model only
0               idef   !0=Default distribution; 1 = User-defined distribution
0.0d0           disc   !
1000            kmax   !control permissible numerical error, etc.
1.d-5           relerr !
```

Figure 2. Sample **params.dat** input file

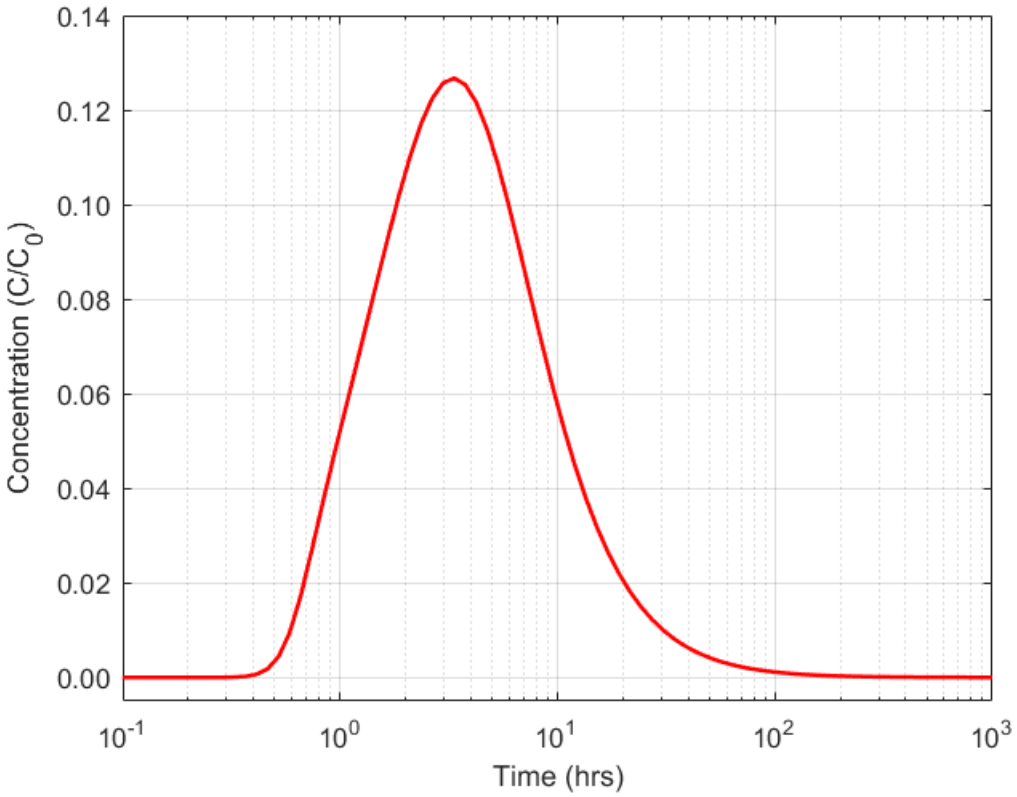

Figure 3. Sample breakthrough curve using the **params.dat** input file

VITA

Md Lal Mamud

EDUCATION

Ph.D. in Geological Engineering, University of Mississippi, University, Mississippi (Second Semester)
Dissertation: Topic not yet fixed.

M.S. in Hydrology, University of Mississippi, University, Mississippi (2018)
Thesis: Generalized Radial Transport Model for interpreting Convergent flow tracer test in Fractured Rocks

M.S. in Geophysics, University of Dhaka, Dhaka, Bangladesh (2014)
Thesis: Subsurface Structure Delineation and Prospect Re-evaluation of Fenchuganj Gas Field by Integrating Seismic and Wireline Log Data, Moulovibazar, Bangladesh

B.S. in Geology, University of Dhaka, Dhaka, Bangladesh (2011)

PROFESSINAL EMPLOMENT HISTORY

2016 - Present: Graduate Teaching Assistant, University of Mississippi, Department of and Geological Engineering

- Lab Courses Taught: Structural and Tectonic Geology (GEOL 303), Mineralogy & Petrology (GEOL 225), Geomorphology (GEOL 305) and Engineering Geology (340)
- Key Responsibilities: Answer Scripts Evaluation and Proctoring in Exam

2018 - Present: Graduate Research Assistant, National Center for Physical Acoustic. University, University of Mississippi, USA

- Key Responsibilities: Doing PhD Research and Conducting Geophysical Field Works,

2017: Research Assistant, Department of Geology and Geological Engineering, University of Mississippi, USA

- Project Title: Identifying Groundwater Recharge from an Oxbow Lake-Wetland System Based on Hydrologic Monitoring and Temperature Tracing
- Key Responsibilities: Downloading temperature and water level data and installing, adjusting and testing instruments.

2017: Research Assistant, National Center for Physical Acoustic. University, University of Mississippi, USA

- Project Title: Self-Potential Survey to Identify Leakage in Earthen Dams in Mississippi, USA
- Key Responsibilities: Building SP Cable for 80 Channels multimeter, Designing and conducting SP survey, and interpreting data.

2015: Research Assistant, University of Dhaka and Mukti (a local NGO), Bangladesh

- Project Title: Identification of Saline Water Intrusion Region in Shatkhira
- Key Responsibilities: Resistivity data acquisition, interpretation and report writing.

2015: Research Assistant, University of Dhaka and Annual Disaster Preparedness Centre (ADPC), Bangladesh

- Project Title: Single, Shallow and Array Microtremor Survey in Bogra
- Key Responsibilities: Shallow Seismic data acquisition, interpretation and report writing.

2014: Sedimentary Lab Analyst, University of Dhaka, Bangladesh Water Development Board (BWDB), Bangladesh

- Key Responsibilities: Analyzing sediments samples and report writing.

2012: Research Assistant, University of Dhaka, Department of Public Health and Engineering, Bangladesh

- Identification of saline water encroachment in coastal region, Cox`s Bazar, Bangladesh
- Key Responsibilities: Resistivity data acquisition, interpretation and report writing

2007 - 2011: Tutor, MABS, UCC, Dhaka Bangladesh

- Key responsibilities : Teaching Physics, Chemistry, Mathematics and English; and Evaluating answer scripts

PUBLICATIONS

Mamud, **M. L.,** A. S. M. Woobaidullah , S. Islam, M. Z. H. Sazal, M. A. Khan, M. Z. Alam, Subsurface structure delineation of Fenchugang Gas Field, Sylhet, Bangladesh;*International Journal of Emerging Technology and Advanced Engineering* Volume 6, Issue 3, March 2016.

T. Mohanta, S. Akter, C. Quamruzzaman, **M. L. Mamud**, M. Z. HossainSazal, K.M. I. Hossain, Case study on surrounding area of Barapukuria coal mine impeding soil fertility, Dinajpur, Bangladesh, International Journal of Scientific & Engineering Research, Volume 6, Issue 7, July 2015

CONFERENCE PRESENTATION

2017: Poster presentation, "Generalized Radial Transport Model for interpreting Convergent flow tracer test in Fractured Rocks."

- American Geophysical Union (AGU) – Fall Meeting 2017, New Orleans, LA

FIELD SUPERVISION EXPERIENCE

Title : Structural and Tectonic Geology Field Trip to Alabama
Position : Graduate Teaching Assistant
Duration : March 31 to April 1, 2017 and April (7-8), 2017
Institute : Department of Geology and Geological Engineering, University of Mississippi, USA

Title : Tishomingo Freshman Field Trip 2016, Mississippi
Position : Graduate Teaching Assistant
Duration : October 8, 2016
Institute : Department of Geology and Geological Engineering, University of Mississippi, USA

POSITIONS AND APPOINTMENTS

- Student Member of SWE (Society of Woman Engineers)
- Student Member of GSA (The Geological Society of America).
- Student Member of AGU (American Geophysical Union).
- Former Student Member of GCAGS (Gulf Coast Association of Geological Societies).
- Former Student Member of AAPG (Association of American Petroleum Geologist).
- Former Student Member of SEG (Society of Exploration Geophysicist).
- Former Student Member of SPE (Society of Petroleum Engineer).
- Former Member of Debating Club, Fazlul Huq Hall, University of Dhaka.
- Former Vice President of "Dhakastho Muktagacha Chatro Kollan Somiti" Dhaka, Bangladesh.