GTFM - USER DOCUMENTATION

Functional Description, Theoretical Development,
and Software Architecture.

FINAL Report

Version 1.0

Prepared by

INTERA Consultants

96-218 22 August 1996

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Graph Theoretic Field Model (GTFM) is a computer program designed to assist in the interpretation and analysis of borehole hydraulic and gas tests conducted in low-permeability media. GTFM has evolved over a period of 13 years (1983 to 1996) from a relatively simple batch input program which ran on Hewlett-Packard 9816 series computers to a complex code with sophisticated graphical pre- and post-processing capabilities which runs under MS-DOS on 32-bit PC compatibles.

Although GTFM's capabilities have increased considerably, the technical documentation to describe the theoretical underpinnings of the various enhancements has not kept pace. This document is an attempt to redress this situation. The document offers a complete description of the functionality, theoretical basis and software architecture of GTFM version 6.00, the most recent release. The following gives a brief overview of each area:

Section 2 - Functional Description This section provides an overview of GTFM's capabilities. It includes a summary of the types of hydrogeologic systems considered, the processes and boundary conditions which can be simulated, the data pre- and post-processing capabilities, and the parameter optimization and sampling capabilities.

Section 3 - Theoretical Development The mathematical bases of GTFM's functionality are developed in this section. The section includes: a description of the graph theoretical approach to developing field models, the equations describing system geometry, the equations for fluid and gas flow processes, the approaches used for solution of linear and non-linear matrix equations, data processing procedures, a description of the implementation of the optimization and error analysis capabilities, and an overview of the sampling functionality.

Section 4 - Software Architecture This section documents: the computer languages used in the GTFM source code, the specific compilers, linkers and other tools required to build an executable from source code, and the structure of the GTFM program and its supporting libraries. Appendix A contains a listing of all GTFM source files with a brief description of the purpose and architectural classification of each file.

Section 5 - Input/Output This section documents the files used by and/or created by GTFM.

2 FUNCTIONAL DESCRIPTION

This section provides an overview of GTFM's capabilities and limitations.

2.1 Assumptions

GTFM was designed as a general purpose numeric tool for simulating single-phase borehole hydraulic (liquid) and gas tests in a variety of media, particularly low-permeability media where conventional well-test analyses procedures are inappropriate. GTFM's formulation is based on the following assumptions:

- the formation to be analysed is a confined horizontal aquifer of finite radius (r_o);
- initially, the formation is at a constant pressure;
- the formation is completely saturated (liquid tests) or completely unsaturated (gas tests);
- formation fluid properties are constant throughout the formation, or can be defined as a function of pressure;
- the formation is bounded above and below by impermeable boundaries;
- there is no vertical flow in the formation;
- a single fully-penetrating well or borehole at the centre of the formation is the only external influence on formation response;
- a uniform external boundary condition is applied at a specified distance from the centre of the wellbore; and,
- the formation is at constant temperature throughout the duration of the simulation. Non-isothermal test-zone effects are incorporated for liquid phase tests only.

Some of these assumptions may limit the applicability of GTFM in specific situations. However, limitations due to these assumptions are generally not significant for the analysis of tests conducted in low-permeability media.

2.2 System Geometry

For most applications, GTFM is formulated assuming a one-dimensional radial geometry. In this case one-dimensional refers only to the fact that pressure is calculated at specific radial distances from the borehole, and that flow within the formation is either towards or away from the borehole (i.e. vertical flow within the formation is not considered). The actual physical geometry of the formation being simulated is of course three-dimensional. The "shape" of the formation is governed by its flow-dimension (Barker, 1988), n, where n = 1 corresponds to a linear system (flow system area is independent of distance from the borehole), n = 2 is a traditional radial flow system bounded above and below by parallel aquitards, and n =3 is a spherical system. The flow dimension of the system may also be fractional with a range from -50 (sub-linear) to 50 (super hyper-spherical).

An optional configuration allows for a second one dimensional flow system to be coupled with the radial system. This second system is intended to model flow through a sealing plug in the borehole. Figure 2.1 shows the available configurations.

2.3 Heterogeneity

GTFM treats the formation (see Figure 2.1) as a zone with either one or two sets of material properties. In the latter case, a "skin-zone" is placed between the test-zone and the remainder of the formation. The skin zone is of a specified thickness t_s as shown in Figure 2.2.

GTFM also allows for heterogenous properties through use of functional approximations for certain parameters. In these cases, parameter values may be represented as a function of distance from the centre of the test-zone.

2.4 Parameter Non-Linearity

Certain formation, fluid-property, and test-zone parameters can be defined as functions of pressure. This causes the overall system of equations to become non-linear, requiring solution with a non-linear matrix solution technique. Non-linear parameters cannot be used in dual-porosity (see Section 2.5) simulations.

2.5 Liquid System Flow Porosity

GTFM implements both single- and dual- porosity flow processes in liquid flow systems.

In the single-porosity system, there are two types of flow within the formation itself: advective flow, and flow due to storage. Advective flow is the physical movement of formation fluid either toward or away from the borehole, as dictated by pressure gradients. Flow due to storage involves fluid flow due to compressibility of the formation and the formation fluid itself as driven by pressure changes with time.

In dual-porosity systems, the flow domain is divided by a volumetric proportion constant into two connected systems: the fracture system, and the matrix system. Flow in the fracture system is analogous to flow in a single-porosity system with the addition of a third flow type, flow into/out of the adjacent matrix material. Flow within the matrix can be visualized as a number of individual single-porosity systems with zero-flow external boundaries. Each matrix system is connected only to the adjacent fracture material. The geometry of the fracture/matrix connection and the behaviour of the internal matrix flow systems are controlled by user-entered parameters.

Figure 2.1 GTFM geometric configurations

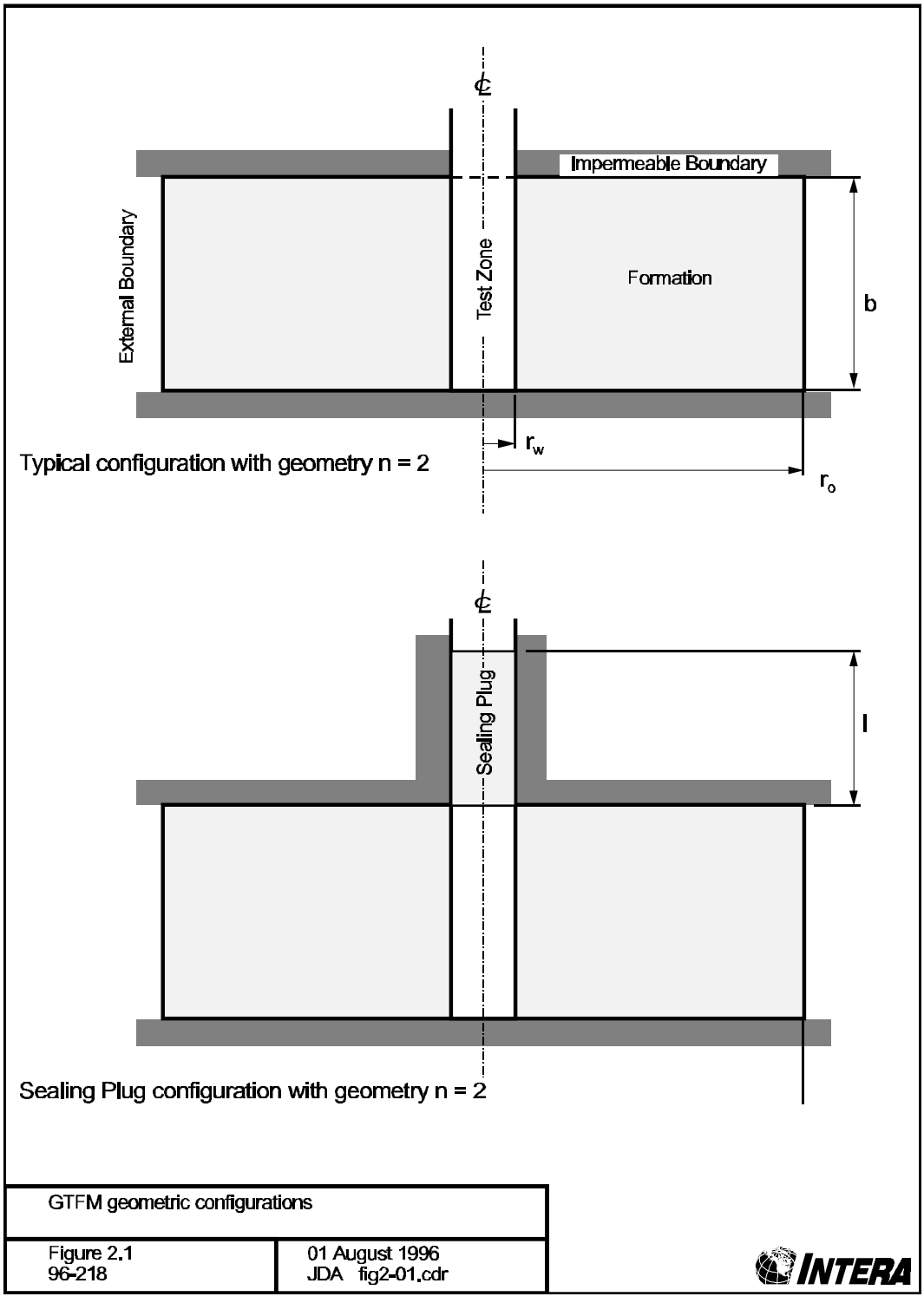
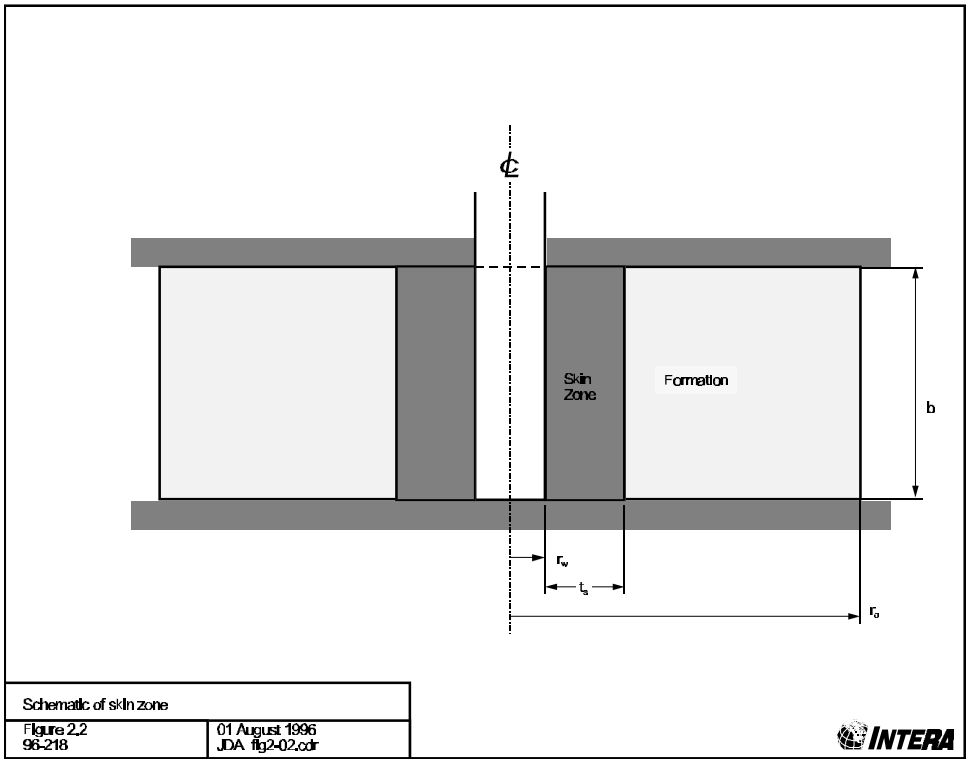


Figure 2.2 Schematic of skin zone



2.6 Gas System Flow Porosity

The gas flow processes implemented in GTFM are for single-porosity systems only. As stated in Section 2.4, the current implementation of the dual-porosity flow processes (see Section 3.1.5.2) is suitable for linear equations only. As presented in Section 3.1.4.2, the equations describing advective gas flow are intrinsically non-linear. Thus, the dual-porosity formulation cannot currently be used in gas test analyses.

2.7 Boundary Conditions

Boundary conditions are applied at: the test-zone, the external formation boundary, and, if a sealing plug is included, at the top of the plug. The following sub-sections describe available boundary conditions.

2.7.1 Test-Zone Boundary Conditions

Boundary conditions at the test-zone represent those commonly applied during a well- or borehole-test. GTFM allows different boundary conditions to be applied consecutively, so as to simulate actual test conditions. Available test-zone boundary conditions include:

1) Specified flow - a flow rate to be injected into (+ve) or withdrawn from (-ve) the test-zone is specified. The flow rate can be described as a constant, or as an arbitrary function of time (see Section 3.2). Well-bore storage may also be incorporated in the boundary condition. Two types of well-bore storage are available: open-hole (for liquid systems only), where a tubing string of constant diameter filled with liquid assumed to be connected to the test-zone, and isolated, where the test-zone is filled with a compressive liquid or a gas.

2) Specified pressure, or history - the pressure in the test-zone is specified as a constant or as a function of time.

3) Slug withdrawal/injection (liquid tests only) - The injection or withdrawal of liquid from an open borehole. The response of the formation will be indicated by the changes in liquid level subsequent to the injection/withdrawal as liquid flows into the formation (injection) or out of the formation (withdrawal). The pressure in the wellbore at the test horizon will vary according to the height and density of the column of liquid above the test zone.

4) Pulse injection/withdrawal - a pressure pulse or withdrawal is applied to a physically isolated test zone. Pressure will rise or decay in the test zone according to the formation parameters and the compressibility of the test-zone. For liquid phase tests, test-zone compressibility is usually higher than the fluid compressibility due to equipment compliance effects. For gas

phase tests, equipment compliance is assumed negligible relative to the compressibility of the gas.

2.7.2 External Boundary Conditions

GTFM provides zero-flow, fixed-pressure, and Carter-Tracey infinite acting boundary conditions to be applied at the external boundary. Carter-Tracey boundary conditions were not used in conducting WIPP PA analyses and are therefore not addressed in this manual.

The pressure value used for fixed-pressure boundary conditions is the user-entered value for static or initial (pre-test) formation pressure.

2.7.3 Sealing Plug Boundary Conditions

A fixed-pressure boundary condition is applied at the top of the sealing plug. For liquid systems, the pressure is set at 0.0 Pa. For gas systems, the pressure is set to the user-entered value for atmospheric pressure.

2.8 Data Pre- and Post-Processing

GTFM contains pre-processing facilities which support a number of input data conditioning and standard well-test interpretation diagnostics. Input data conditioning routines include manual and automatic data reduction, data smoothing, and low- and high-pass filtering. Diagnostic utilities include calculation of pressure derivatives and superposition times. Interpretation tools for graphical calculation of line slopes are also available.

Standard post-processing facilities include diagnostic plots that can be used to compare field data and simulation results. Other graphical facilities allow for 2D and 3D visualization of the simulated pressure versus distance into the formation and time.

2.9 Goodness-of-Fit Calculation

GTFM can also determine a measure of the quality of the match between field data and simulation results. Processed field data (i.e. pressure derivatives) can also be compared to processed simulation results and a fit metric determined. The standard fit metric is the Chi-squared value, or sum of the squared residuals. Other available metrics include sum of absolute values of residuals, or the average (normalized to number of points in the field data set) sum of squared or absolute values of residuals. Compound metrics can be calculated by adding two or more single fit calculations together. Components of compound metrics can be scaled individually to change weighting and contribution to overall fit.

Fit surfaces can be generated by calculating fit metrics while varying the values of two simulation parameters. These surfaces can be visualized as 3D surfaces, as contour plots, or as XY plots of fit value versus a single parameter value.

2.10 Optimization

GTFM's optimization facilities automatically determine the set of simulation parameters required to minimize a fit metric, or, in other words, to create the best match between field and simulation results. Optimizations require that two or more parameters be specified for optimization. In addition to individual parameters, the Y values of points defining parameters as a function of radius or pressure may also be optimized. Up to 10 parameters, or 20 individual Y components, or any combination of the two may be optimized.

Two minimization algorithms are available. The Simplex algorithm is robust and can be used to minimize any fit metric. The Levenberg-Marquardt algorithm is restricted to Chi-squared fit metrics but can converge on a solution quickly, particularly if the fit metric is well behaved.

Although any of the fit metrics described in section 2.9 can be minimized, the Chi-squared metric is of special significance as it allows for a number of error analyses routines. These include residual analysis, confidence intervals, covariance assessment and Jacobian evaluation to determine the significance of each parameter's contribution to the fit metric.

2.11 Sampling

One of GTFM's unique features is the optimization/sampling mode of operation which combines optimization with Latin Hypercube or Monte Carlo variable sampling routines. With sampling, uncertain variables (certain wellbore boundary conditions and model parameters) are assigned distributions, and numerous realizations of input data sets generated. An optimization is performed on each input data set. Optimized parameter results are stored in a data file. These can be post-processed to show ranges and distributions of optimized parameters, and to determine correlations between input parameter uncertainty and optimized parameter ranges.

3 THEORETICAL DEVELOPMENT

This section of the manual describes the theoretical and mathematical basis for the functionality available in GTFM. The section is subdivided to address the simulator itself, methods for approximating functional representations, pre-processing and analysis routines, and optimization, sampling and error analysis.

3.1 Simulator Theory

This subsection describes the graph theoretic approach as applied to field models, develops the equations describing system geometry and flow, formulates the algebraic system of equations, and finally, describes solution techniques used for linear and non-linear cases.

3.1.1 General Approach

GTFM is based on an application of graph theory as applied to field models. This approach, described in Savage and Kesavan (1976), applies a discretization procedure to a physics model to produce a system of algebraic equations which are amenable to computer solutions. In this sense, it differs from finite element and finite difference models which rely on an intermediary step whereby a continuos model formulation (usually a partial differential equation) is derived from the physics of the problem before discretization is performed.

With GTFM, the physical geometry of the aquifer being simulated is represented as a geometric continuum with a finite volume. This volume is discretized into smaller volumes, each represented by a single node. Adjacent nodes are connected by directed graph edges which represent flow from one volume to another. The direction of each graph edge is indicated by an arrow which represents the assumed direction of positive flow. Additional graph edges represent flow within each volume and flow into or out of the system at the boundaries. In a fluid flow system, these three types of graph edges represent Darcy flow, flow due to storage, and boundary flows respectively. Figure 3.1 illustrates these concepts schematically for a single porosity formation with no skin or sealing plug.

The formulation assumes that the pressure within each nodal volume is represented by a pressure value at each node. Pressures may change as a function of node number and time. The time continuum is also discretized into a series of time steps.

Figure 3.2 shows a segment of the directed graph within a flow system similar to that shown in figure 3.1. Nodes i, j, k are three consecutive nodes. Flow between nodes i and j is represented by the graph edge labelled d_i. Flow due to storage within the volume represented by node j is represented by the graph edge labelled s_j.

Figure 3.1 General graph theoretic approach

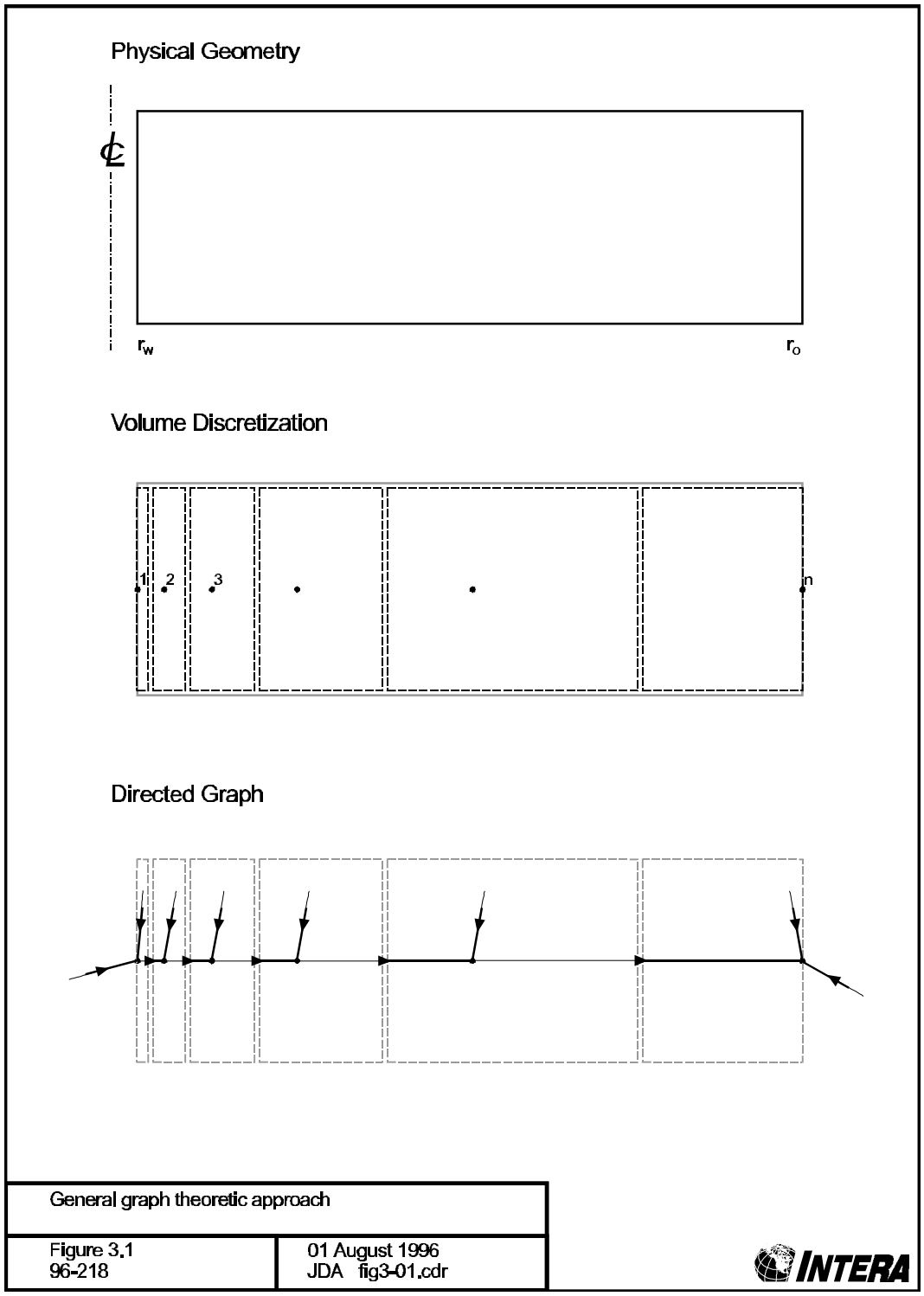
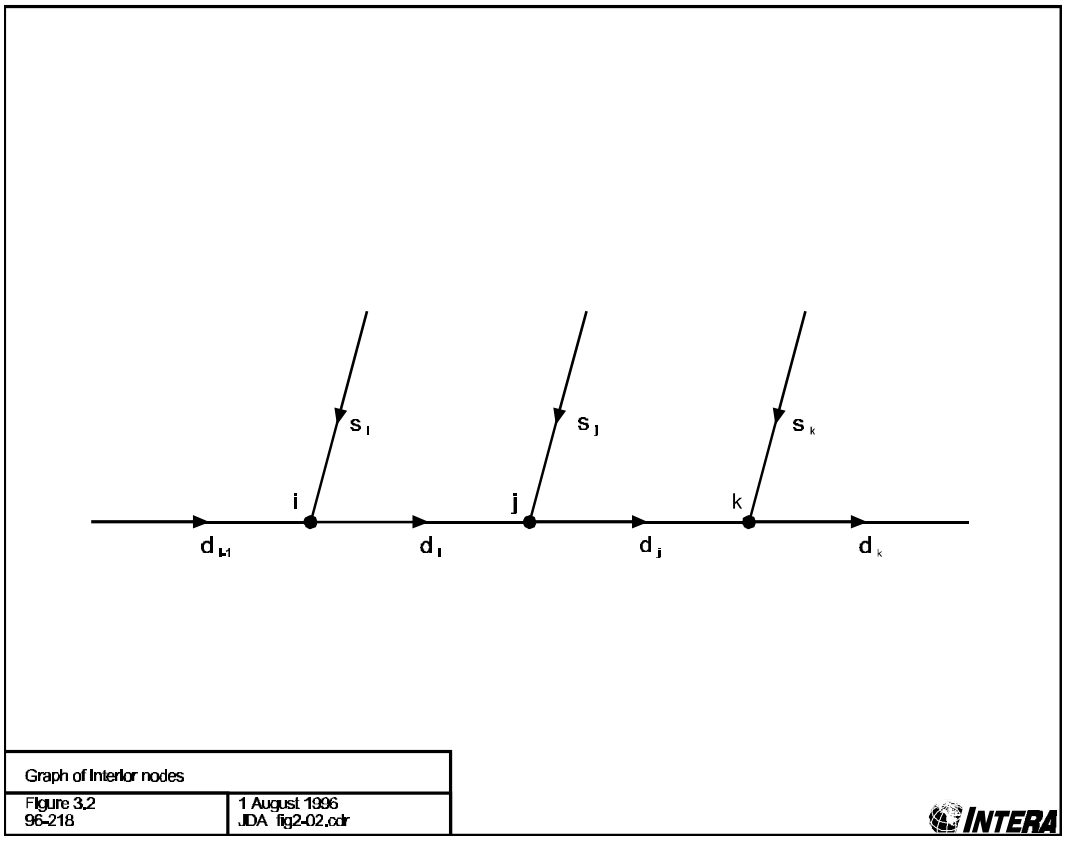


Figure 3.2 Graph of interior nodes



Constitutive relationships hold at each node. For example, at any given time t, at any interior node i:

$$d_{i-1} + s_i - d_i = 0 \qquad (3.1\text{-}1)$$

When coupled with geometric relationships and the flow equations developed in subsequent sub-sections, the constitutive relationships can be used to create a set of algebraic equations describing the system. Solving the equations yields nodal pressures and graph edge flow rates.

3.1.2 System Graphs

This subsection develops the system graphs used in different geometrical and system porosity configurations.

Figure 3.3 shows a basic system formulation - single porosity, no skin, and no sealing plug. The graph edges labelled b_w and b_o represent boundary flows: from the well into the formation, and from the formation out of the external boundary respectively. A logarithmic spacing is used between adjacent nodes. This serves to increase the node density close to the well, where pressure changes are likely to be most rapid. The dashed areas delimit the volume assigned to each node and indicate the radius at which the flow area between nodes is calculated.

Figure 3.4 depicts a sealing plug added to the basic system. Node 1 is located at the top of the plug, while node n_p is the last node within the plug above the well-bore. Pressure in the isolated test-zone is assumed to be uniform, and the pressure of the node at r_w is assumed also to be representative of the pressure in the test-zone at the bottom of the plug.

Figure 3.5 shows the effect of adding a skin to the basic system. Note that graph edges remain conceptually the same, however the spacing of nodes within the skin zone is obviously different from that in the formation.

Figure 3.6 is a dual porosity system with no skin or sealing plug. The total system is divided into two components: fracture and matrix. The relative volumes of the components are determined by a "matrix volume factor" where:

$$V_m = V_s\, M \qquad (3.1\text{-}2)$$
$$V_f = V_s\, (1.0 - M)$$

where:

V_m = volume of matrix, L^3
V_f = volume of fracture, L^3
V_s = volume of system, L^3
M = matrix volume factor

Figure 3.3 System graph for basic 11 node single-porosity system

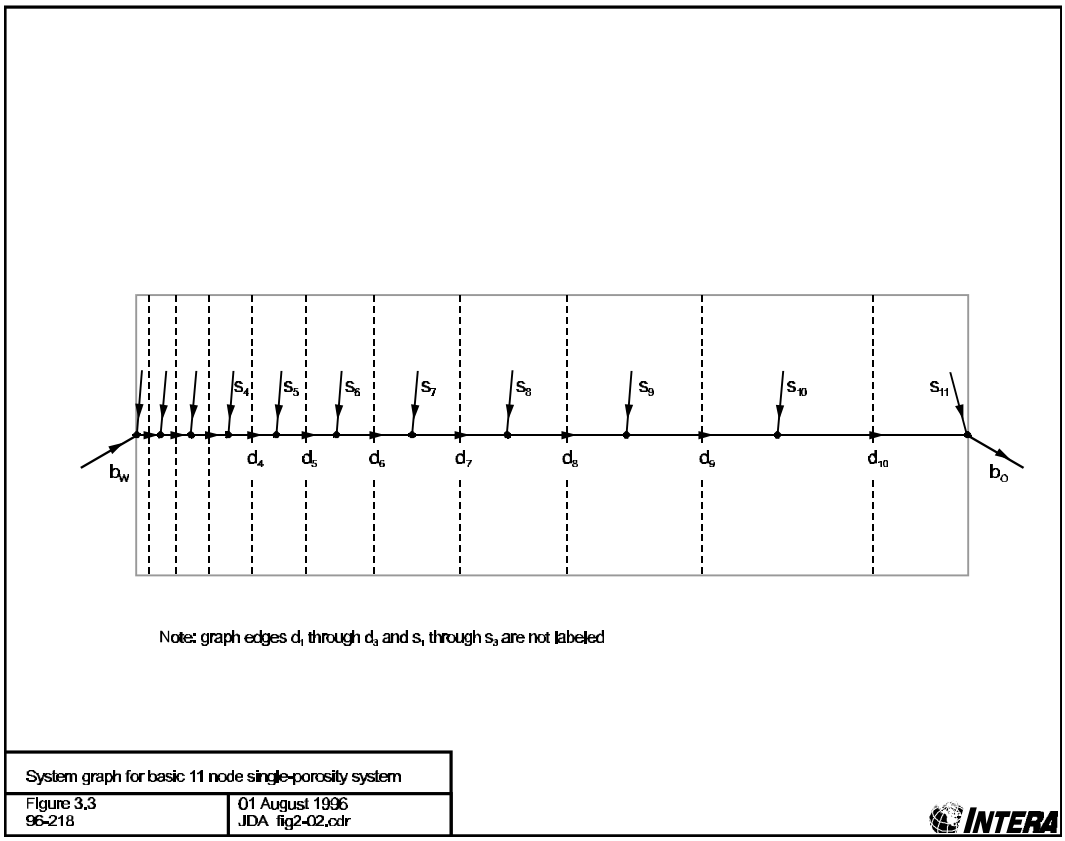


Figure 3.4 System graph for single-porosity system with 11 radial nodes and 5 plug nodes

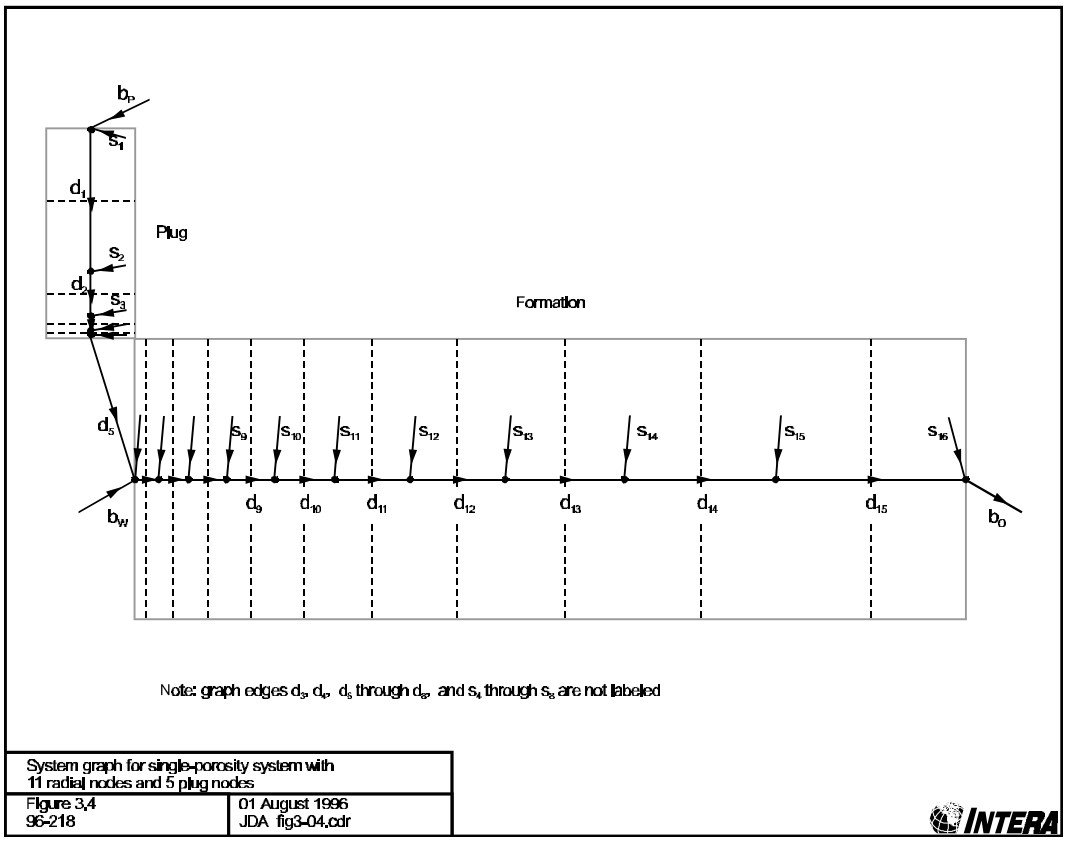


Figure 3.5 System graph for single-porosity system with 11 radial nodes including 5 skin nodes

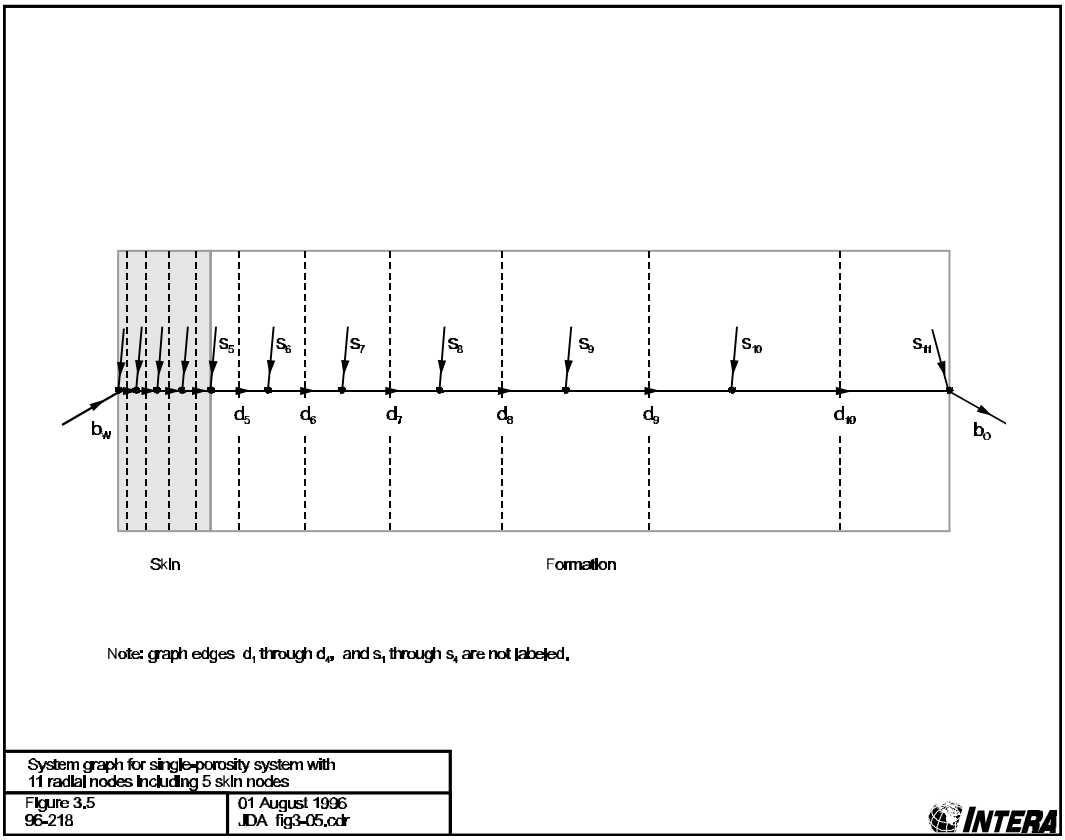
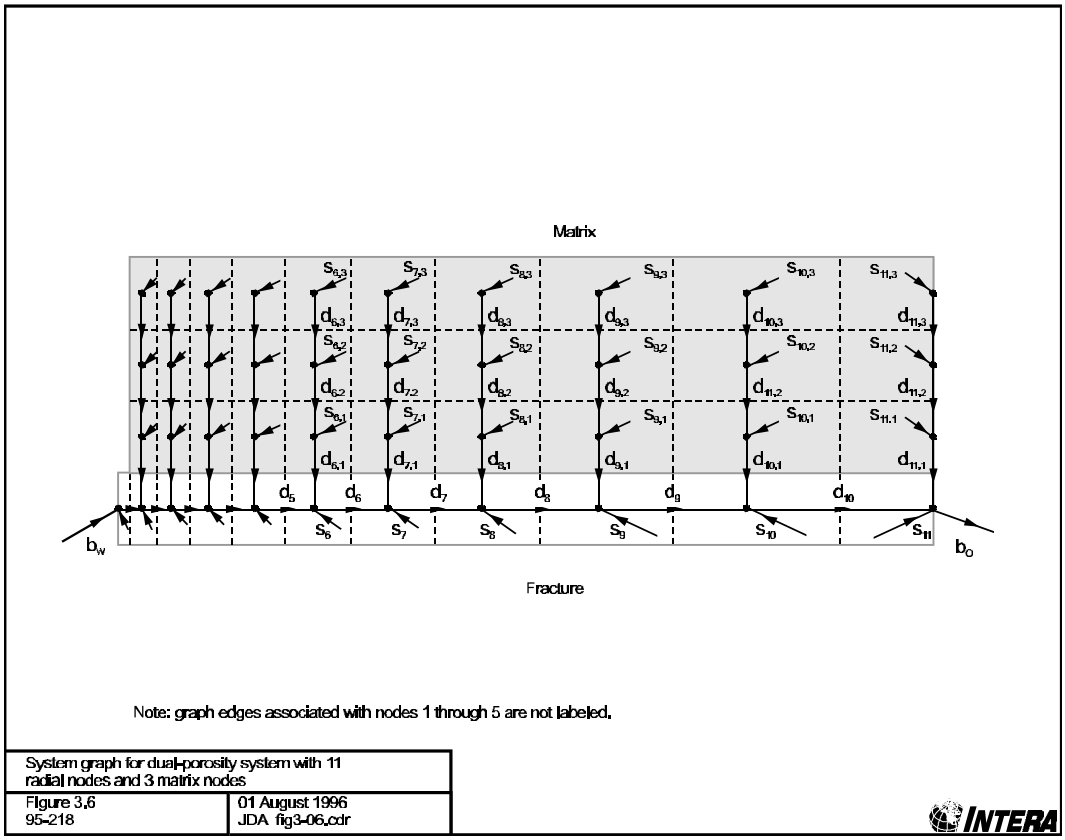


Figure 3.6 System graph for dual-porosity system with 11 radial nodes and 3 matrix nodes



Radial flow between nodes and flow in the boundary edges occurs only in the fracture. The matrix flow system can be viewed as a number of 1-D systems with zero flow external boundaries coupling into the fracture system. The number of nodes in the matrix system controls the degree of "transientness" of the double porosity response. A single node corresponds to a pseudo-steady-state (PSS) response, while more than ten nodes yields a "transient" response. PSS and "transient" are terms from the oilfield literature which describe different dual-porosity reservoir responses.

Figure 3.7 shows a system with dual-porosity and skin. The figure indicates that the dual-porosity systems are not incorporated in the skin zone, and that the full volume of the system is assigned to the fracture system within the skin-zone.

3.1.3 System Geometry

There are two elements to specifying system geometry: the spacing of the nodes in the system, and the volumes and areas associated with each node.

3.1.3.1 Node Spacing

For a system without a skin a simple logarithmic spacing is used for all nodes within the formation:

$$r_i = r_{i-1}\, e^{\Delta a} \tag{3.1-3}$$

where:

r_i = radius of node i, L
Δa = logarithmic grid increment

where:

$$\Delta a = \frac{\ln(r_o) - \ln(r_w)}{n - 1} \tag{3.1-4}$$

where:

r_o = outside radius of grid, L
r_w = radius of borehole or well, L
n = number of nodes in grid

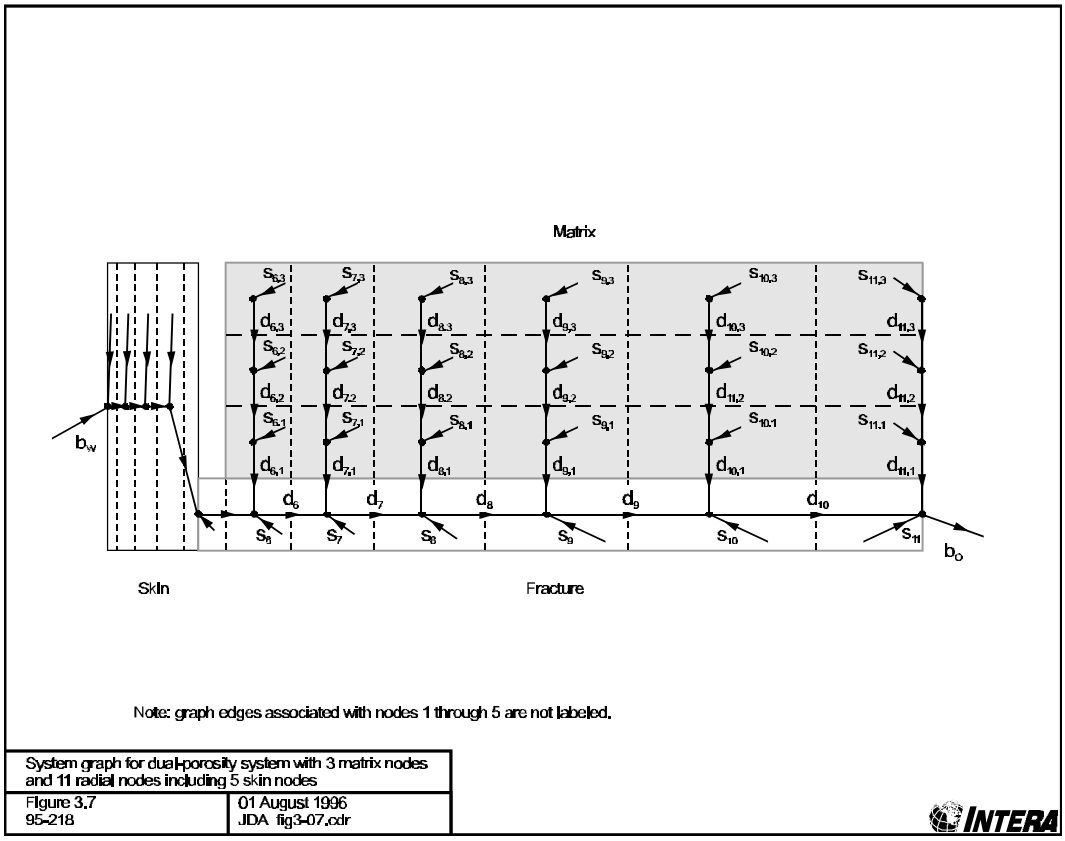
For systems with a skin, two grid increments are used. Within the skin (i.e. for nodes 1 through n_s), equation 3.1-5 applies:

$$\Delta a_s = \frac{\ln(r_w + t_s) - \ln(r_w)}{n_s - 1} \tag{3.1-5}$$

where:

t_s = thickness of skin, L
n_s = number of nodes in skin

Figure 3.7 System graph for dual-porosity system with 3 matrix nodes and 11 radial nodes including 5 skin nodes



Outside the skin (nodes n_{s+1} to node n), the grid increment described by equation 3.1-6 is used:

$$\Delta a = \frac{\ln(r_o) - \ln(r_w + t_s)}{n - n_s - 1} \quad (3.1\text{-}6)$$

If a sealing plug is included, nodes within the sealing plug are also geometrically spaced. However, in this case a "plug spacing factor" is entered, from which the grid increment is derived:

$$\Delta a_p = \frac{\ln(X_{plug})}{n_p - 1} \quad (3.1\text{-}7)$$

where:
- X_{plug} = plug spacing factor
- n_p = number of nodes in plug

Location of nodes within the plug is then described by:

$$l_i = L_{plug}\left[1 - e^{-\Delta a(i-1)}\right] \quad (3.1\text{-}8)$$

where:
- l_i = distance of node i from top of plug, L
- L_{plug} = length of plug, L

3.1.3.2 Flow Areas

The flow area is defined as the area over which advective, or Darcy, flow between adjacent nodes occurs. In a single porosity radial system of flow dimension equal to 2, the flow area is simply:

$$A_i = b\, 2\pi\, r_{a_i} \quad (3.1\text{-}9)$$

where:
- A_i = flow area between nodes i and i + 1, L^2
- b = thickness of formation, L
- r_{a_i} = average radius between node i and i+1, L
  $= \frac{r_i + r_{i+1}}{2}$

In a generalized system of flow dimension n, the equation from Barker (1988) describes the flow area of the nodes in the fracture zone (i.e. non-matrix, non-plug nodes}:

$$\alpha_i = \frac{2\pi^{n/2}}{\Gamma(n/2)} \quad (3.1\text{-}10)$$
$$A_i = b^{3-n}\,\alpha_i\, r_{a_i}^{n-1}$$

where:
- n = system flow dimension
- Γ = gamma function

Equations 3.1-9 and 3.1-10 are identical for $n = 2$. However, when simulating actual tests, where the flow area at the well is as specified by equation 3.1-9 with $r_{ai} = r_{well}$, using equation 3.1-10 leads to incorrect flow areas being used. Consequently, GTFM allows for compensation of the area. Compensation is used whenever the flow dimension is specified as a function of radius, or if the well-bore area must match a specified borehole area. In these cases, the area is calculated from the area of the previous node, and the flow dimension at the node :

$$A_i = \frac{A_{i-1}}{r_{i-1}^{n_i - 1}} \, r_{a_i}^{n_i - 1} \qquad (3.1\text{-}11)$$

where:

n_i = flow dimension at node i

Within the plug portion of a system the flow area is simply the cross-sectional area of the borehole:

$$A_p = \pi \, r_w^2 \qquad (3.1\text{-}12)$$

where:

A_p = flow area of all plug nodes, L^2

Dual-porosity systems always assume a flow dimension of 2 in the fracture. The flow area in the matrix nodes associated with each fracture node is the area of the fracture system in contact with the matrix:

$$A_{i_m} = \frac{\pi}{4} \left[(r_i + r_{i+1})^2 - (r_{i-1} + r_i)^2 \right] \qquad (3.1\text{-}13)$$

where:

A_{i_m} = flow area of all matrix nodes associated with fracture node i, L^2

3.1.3.3 Node Volumes

The volumes assigned to each node are used in formulating the equations describing the graph edges representing the contribution of storage flow to the system.

For nodes in the fracture zone the volume assigned to node *i* is calculated as follows:

$$V_i = V_{A_i} - V_{A_{i-1}} \qquad (3.1\text{-}14)$$

where:

V_{A_i} = volume enclosed within flow area i

V_A are determined by integrating equation 3.1-10 or 3.1-11 with respect to *r:*

$$V_{A_i} = \int A_i \, dr_{a_i} \qquad (3.1\text{-}15)$$

or

$$V_{A_i} = \frac{A_i}{n_i} r_{a_i}$$

Plug node volumes are:

$$V_{i_p} = \frac{A_p}{2} \left[l_{i+1} - l_{i-1} \right] \qquad (3.1\text{-}16)$$

Matrix node volumes are described as:

$$V_{m_i} = \frac{A_{m_i} \, b \, M}{n_m} \qquad (3.1\text{-}17)$$

where:

- b = formation thickness (assumed constant as f(r)), L
- M = matrix volume factor
- n_m = number of matrix nodes

3.1.4 Edge Equations

Edge equations describe the flow in the graph edges. They are categorized by graph edge type (Darcy, storage, or boundary) and system type (gas or liquid).

3.1.4.1 Liquid Flow

Liquid flow equations describe the edge flow in volumetric flow terms ($L^3 \, t^{-1}$). Fluid density is assumed to be invariant as a function of pressure so as to ensure that constitutive relationships hold.

Flow in Darcy edges is described as:

$$d_i = \frac{k}{\mu} A_i \frac{\Delta P}{\Delta l} \quad (3.1\text{-}18)$$

or

$$d_i = \frac{K}{\rho g} A_i \frac{\Delta P}{\Delta l}$$

where:

- k = permeability, L^2
- u = viscosity, $ML^{-1}t^{-1}$
- K = hydraulic conductivity, Lt^{-1}
- ρ = formation fluid density, ML^{-3}
- g = gravitational constant, Lt^{-2}
- ΔP = change in pressure between nodes i and i+1, $ML^{-1}t^{-2}$
- Δl = distance between nodes i and i+1, L

Permeability and/or viscosity or hydraulic conductivity may be described as functions of pressure. Alternatively, for non-plug, non-matrix nodes, permeability and hydraulic conductivity may also be described as functions of radius.

For matrix nodes, the Darcy flow equation is slightly modified to account for different matrix geometries:

$$d_{i,j} = \alpha \frac{k}{\mu} A_{m_i} n_m t \Delta P \quad (3.1\text{-}19)$$

or

$$d_{i,j} = \alpha \frac{K}{\rho g} A_{m_i} n_m t \Delta P$$

where:

- α = interporosity flow parameter (shape factor), L^2
- t = unit thickness, L

Flow due to storage is described as:

$$s_i = (C_r + \theta C_w) V_i \frac{\Delta P}{\Delta t} \quad (3.1\text{-}20)$$

or

$$s_i = \frac{S}{\rho g} V_i \frac{\Delta P}{\Delta t}$$

where:

- C_r = rock compressibility, $M^{-1}Lt^2$
- θ = rock porosity
- C_w = formation fluid compressibility, $M^{-1}Lt^2$
- S = formation storativity, L^{-1}
- ΔP = change in pressure between time steps i and i+1, $ML^{-1}t^{-2}$
- Δt = size of time step, t

Wellbore boundary flows (b_w edges) vary according to the wellbore boundary condition assigned.

For specified pressure:

$$b_w = \text{unknown} \tag{3.1-21}$$

For iso-thermal pulse sequences:

$$b_w = V_w C_t \frac{\Delta P_w}{\Delta t} \tag{3.1-22}$$

where:

V_w = volume of test-zone, L^3

C_t = test-zone compressibility, $M^{-1}Lt^2$

C_t may be a constant, a function of time, or a function of pressure. V_w may be a constant or a function of time.

For non iso-thermal pulse sequences:

$$b_w = \frac{V_w}{\Delta t} \left(C_t \Delta P_w - C_T \Delta T_w\right) \tag{3.1-23}$$

where:

C_T = test-zone fluid thermal expansion coefficient, T^{-1}

ΔT_w = change in test-zone temperature over a single time step, T

T_w is defined as a function of time.

For slug sequences:

$$b_w = \frac{\pi r_c^2}{\rho g} \frac{\Delta P_w}{\Delta t} \tag{3.1-24}$$

where:

r_c = radius of casing where liquid level changes are occurring, L

For specified flow with no wellbore storage:

$$b_w = Q_w \tag{3.1-25}$$

where:

Q_w = specified flow , L^3t^{-1}

Q_w may be a constant or a function of time.

For specified flow with "isolated" wellbore storage:

$$b_w = Q_w - V_w C_t \frac{\Delta P_w}{\Delta t} \quad (3.1\text{-}26)$$

For specified flow with "open hole" wellbore storage:

$$b_w = Q_w - \frac{\pi r_c^2}{\rho g} \frac{\Delta P_w}{\Delta t} \quad (3.1\text{-}27)$$

Flows associated with external boundary conditions, $b_{o,}$ are dependent on the assigned boundary condition:

For specified pressure:

$$b_o = \text{unknown} \quad (3.1\text{-}28)$$

For zero-flow:

$$b_o = 0 \quad (3.1\text{-}29)$$

Flows associated with fixed-pressure plug external boundary conditions, $b_{p,}$ are:

$$b_p = \text{unknown} \quad (3.1\text{-}30)$$

3.1.4.2 Gas Flow

Gas flow equations describe the edge flow in mass flow terms (Mt^{-1}) or volumetric flow terms ($L^3 t^{-1}$) at standard temperature and pressure (STP). The two are related by:

$$q_{STP} = q_M \frac{R T_s}{m_w P_s} \quad (3.1\text{-}31)$$

where:

- q_{STP} = flow rate at standard temperature and pressure, $L^3 t^{-1}$
- q_M = mass flow rate , $M t^{-1}$
- R = universal gas constant
- m_w = molecular weight of gas, $M\,mole^{-1}$
- T_s = standard temperature, T
- P_s = standard pressure, $M L^{-1} t^{-2}$

Flow in Darcy edges is described as:

$$d_{i_M} = \frac{m_w}{RT}\,\frac{k}{\mu}\,A_i\,P\,\frac{\Delta P}{\Delta l} \qquad (3.1\text{-}32)$$

or

$$d_{i_{STP}} = \frac{T_s}{P_s T}\,\frac{k}{\mu}\,A_i\,P\,\frac{\Delta P}{\Delta l}$$

where:

P = average pressure between nodes i and i+1, $M\,L^{-1}\,t^{-2}$

$$= \frac{P_i + P_{i+1}}{2}$$

Note that the *P* term in equation 3.1-32 makes the flow equation intrinsically non-linear. Permeability and/or viscosity can be described as functions of pressure. For non-plug nodes, permeability may be described functions of radius.

An alternate formulation is available for viscosity:

$$\mu = \mu_0 + b_\mu P \qquad (3.1\text{-}33)$$

where:

μ_0 = viscosity at $P = 0$, $M\,L^{-1}\,t^{-1}$

b_μ = linear viscosity coefficient, t^{-1}

An alternate formulation is also available for permeability:

$$k = k_l\left(1 + \frac{b_k}{P}\right) \qquad (3.1\text{-}34)$$

where:

k_l = permeability to liquid, L^2

b_k = Klinkenburg coefficient, $M^{-1}\,t$

Flow due to storage is described as:

$$s_{i_M} = \frac{m_w}{R\,T}\,\theta\,V_i\,\frac{\Delta P}{\Delta t} \qquad (3.1\text{-}35)$$

or

$$s_{i_{STP}} = \frac{T_s}{P_s\,T}\,\theta\,V_i\,\frac{\Delta P}{\Delta t}$$

Boundary condition edge flow equations for pulse tests are:

$$b_{w_M} = \frac{m_w}{R\,T} V_w \frac{\Delta P_w}{\Delta t} \quad (3.1\text{-}36)$$

or

$$b_{w_{STP}} = \frac{T_s}{P_S\,T} V_w \frac{\Delta P_w}{\Delta t}$$

Other boundary condition equations are analogous to those developed for liquid systems. Slug wellbore boundary conditions are not available for gas systems.

3.1.5 Matrix Equations

3.1.5.1 Single Porosity

The basic edge equations can be reduced to the following form:

$$\begin{aligned} d_i &= D_i\,(P_i - P_{i+1}) \quad (3.1\text{-}37)\\ s_i &= S_i\,(P_i - P_{i_{t-\Delta t}})\\ b_w &= Q_w + B_w\,(P_w - P_{w_{t-\Delta t}})\\ b_o &= Q_o \end{aligned}$$

where D_i, S_i, and B_w may be constants or may be functions of pressure (D_i for all gas, otherwise if $f(P)$ parameters). B_w may also be a function of time. Qw, and Qo may be constants or may be functions of time. A fully implicit (backwards in time) time discretization is used in the s_i and b_w equations. Applying constitutive relationships (equation 3.1-1) at node i yields:

$$D_{i-1}\,(P_{i-1} - P_i) + S_i\,(P_i - P_{i_{t-\Delta t}}) - D_i\,(P_i - P_{i+1}) = 0 \quad (3.1\text{-}38)$$

or:

$$D_{i-1}\,P_{i-1} + (-D_{i-1} - D_i)\,P_i + D_i\,P_{i+1} + S_i\,(P_i - P_{i_{t-\Delta t}}) = 0$$

For a non-plug single porosity system, applying 3.1-38 to the entire system graph leads to the following:

$$\begin{bmatrix} -D_1 & D_1 & & & & & \\ D_1 & -D_1-D_2 & D_2 & & & & \\ & D_2 & -D_2-D_3 & D_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & D_{n-2} & -D_{n-2}-D_{n-1} & D_{n-1} & \\ & & & & D_{n-1} & -D_{n-1} & \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ \vdots \\ P_{n-1} \\ P_n \end{bmatrix} + \begin{bmatrix} S_1 + B_w & & & & & \\ & S_2 & & & & \\ & & S_3 & & & \\ & & & \ddots & & \\ & & & & S_{n-1} & \\ & & & & & S_n \end{bmatrix} \left(\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ \vdots \\ P_{n-1} \\ P_n \end{bmatrix}_t - \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ \vdots \\ P_{n-1} \\ P_n \end{bmatrix}_{t-\Delta t} \right) = \begin{bmatrix} -Q_w \\ 0 \\ 0 \\ \vdots \\ 0 \\ Q_o \end{bmatrix} \quad (3.1\text{-}39)$$

Re-arranging yields:

$$\begin{bmatrix} -D_1+S_1+B_w & D_1 & & & & & \\ D_1 & -D_1-D_2+S_2 & D_2 & & & & \\ & D_2 & -D_2-D_3+S_3 & D_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & D_{n-2} & -D_{n-2}-D_{n-1}+S_{n-1} & D_{n-1} & \\ & & & & D_{n-1} & -D_{n-1}+S_n \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ \vdots \\ P_{n-1} \\ P_n \end{bmatrix} = \begin{bmatrix} -Q_w+(B_w+S_1)P_{1_{t-\Delta t}} \\ S_2 P_{2_{t-\Delta t}} \\ S_3 P_{3_{t-\Delta t}} \\ \vdots \\ S_{n-1} P_{n-1_{t-\Delta t}} \\ Q_o+S_n P_{n_{t-\Delta t}} \end{bmatrix} \quad (3.1\text{-}40)$$

If a plug is included, additional nodes are placed before the well bore node (node 1 in a non-plug system) and a set of equations analogous to 3.1-40 are developed, with node 1 corresponding to the fixed pressure boundary at the top of the plug.

All fixed pressure boundaries (top of plug for plug systems, well bore for history sequences, and external boundary node if not set to zero flow) are implemented by partitioning the matrix and re-formulating for unknown pressures only. For example, a fixed pressure external boundary results in:

$$\begin{bmatrix} -D_1+S_1+B_w & D_1 & & & & \\ D_1 & -D_1-D_2+S_2 & D_2 & & & \\ & D_2 & -D_2-D_3+S_3 & D_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & D_{n-2} & -D_{n-2}-D_{n-1}+S_{n-1} \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ \vdots \\ P_{n-1} \end{bmatrix} = \begin{bmatrix} -Q_w+(B_w+S_1)P_{1_{t-\Delta t}} \\ S_2 P_{2_{t-\Delta t}} \\ S_3 P_{3_{t-\Delta t}} \\ \vdots \\ S_{n-1} P_{n-1_{t-\Delta t}} - D_{n-1} P_n \end{bmatrix} \quad (3.1\text{-}41)$$

This ensures that all terms in the vector on the right hand side of the equation are known, and all on the left are unknown. Matrices are re-partitioned at the start of each sequence, depending upon the wellbore boundary condition.

3.1.5.2 Dual-Porosity

Dual porosity adds another term to the constitutive equations for a fracture node *i*. Equation 3.1-1 is recast as:

$$d_{i-1} + s_i + d_{i,1} - d_i = 0 \quad (3.1\text{-}42)$$

where $d_{i,1}$ is the flow from the matrix into the fracture.

The matrix system associated with any one fracture node can be considered as a separate and independent one-dimensional system which can be represented as a set of matrix equations analogous to equation 3.1-39. If the matrix properties are linear and constant, then, due to the geometry of the matrix, the equations representing each matrix system are identical, except for a scaling factor determined by the area of matrix in contact with the fracture system. Consequently the matrix

equations can be written once for a unit area system, and then scaled by the contact area. For a matrix system with three nodes the following system of equations results:

$$\begin{bmatrix} -D_{i,1} & D_{i,1} & & \\ D_{i,1} & -D_{i,1}-D_{i,2}+S_{i,1} & D_{i,2} & \\ & -D_{i,2} & -D_{i,2}-D_{i,3}+S_{i,2} & D_{i,3} \\ & & D_{i,3} & -D_{i,3}+S_{i,3} \end{bmatrix} \begin{bmatrix} P_i \\ P_{i,1} \\ P_{i,2} \\ P_{i,3} \end{bmatrix} = \begin{bmatrix} d_{i,1} \\ S_{i,1}P_{i,1_{t-\Delta t}} \\ S_{i,2}P_{i,3_{t-\Delta t}} \\ S_{i,3}P_{i,3_{t-\Delta t}} \end{bmatrix} \tag{3.1-43}$$

Gaussian elimination is used to transform the matrix to lower triangular form, allowing $d_{i,1}$ to be expressed as:

$$d_{i,1} = A_{m_i}\left(X_0 D_{i,1} P_i + X_1 S_{i,1} P_{i,1_{t-\Delta t}} + X_2 S_{i,2} P_{i,2_{t-\Delta t}} + \cdots + X_{m-1} S_{i,m-1} P_{i,m-1_{t-\Delta t}} + X_m S_{i,m} P_{i,m_{t-\Delta t}}\right) \tag{3.1-44}$$

or:

$$d_{i,1} = D_{m_i} P_i + C_{m_i}$$

where:

X_j = coefficients representing results of Gaussian elimination

Substituting 3.1-44 into 3.1-42 yields the following matrix equations for double porosity systems:

$$\begin{bmatrix} -D_1+S_1+B_w & D_1 & & & & & \\ D_1 & -D_1-D_2+S_2+D_{m_2} & D_2 & & & & \\ & D_2 & -D_2-D_3+S_3+D_{m_3} & D_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & D_{n-2} & -D_{n-2}-D_{n-1}+S_{n-1}+D_{m_{n-1}} & D_{n-1} & \\ & & & & D_{n-1} & -D_{n-1}+D_{m_n}+S_n \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ \vdots \\ P_{n-1} \\ P_n \end{bmatrix} = \begin{bmatrix} -Q_w+(B_w+S_1)P_{1_{t-\Delta t}} \\ S_2 P_{2_{t-\Delta t}} - C_{m_2} \\ S_3 P_{3_{t-\Delta t}} - C_{m_3} \\ \vdots \\ S_{n-1} P_{n-1_{t-\Delta t}} - C_{m_{n-1}} \\ Q_o + S_n P_{n_{t-\Delta t}} - C_{m_n} \end{bmatrix} \tag{3.1-45}$$

3.1.6 Matrix Solution

Equations 3.1-41 and 3.1-45 can be reduced to:

$$\begin{bmatrix} A_{1,1} & A_{1,2} & & & & \\ A_{2,1} & A_{2,2} & A_{2,3} & & & \\ & A_{3,2} & A_{3,3} & A_{3,4} & & \\ & & \ddots & \ddots & \ddots & \\ & & & A_{n-2,n-3} & A_{n-2,n-2} & A_{n-1,n} \\ & & & & A_{n,n-1} & A_{n,n} \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ \vdots \\ P_{n-1} \\ P_n \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ \vdots \\ C_{n-1} \\ C_n \end{bmatrix} \tag{3.1-46}$$

where:

A = coefficient matrix
P = unknown pressure vector
C = right-hand side vector of constants

If all elements of ***A*** are constants, 3.1-46 is amenable to direct solution. This will be the case for liquid simulations with linear formation properties and linear boundary conditions. In this case GTFM uses an algorithm which takes advantage of the symmetric tri-diagonal form of the matrices. For dual porosity simulations, matrix system pressures are calculated after solution of the fracture system by back substituting calculated pressures into the diagonalized matrix remaining after Gaussian elimination.

For non-linear systems of equations, which include all gas systems and any liquid system with one or more formation parameters or wellbore boundary condition parameters expressed as a function of pressure, a Newton-Raphson solution technique is used. This is an iterative approach which successively improves the estimate of the solution variable (**P**) with each iteration:

$$\mathbf{P}_{new} = \mathbf{P}_{old} + \delta\mathbf{P} \qquad (3.1\text{-}47)$$

where:

P_{new} = new estimate of solution vector
P_{old} = previous iteration estimate of solution vector
$\delta\mathbf{P}$ = change vector

The technique requires calculation of the partial derivatives of each nodal equation with respect to nodal pressure. The following system of equations is then solved for $\delta\boldsymbol{P}$:

$$\begin{bmatrix} \frac{\partial A_{1,1}}{\partial P_1} & \frac{\partial A_{1,2}}{\partial P_2} & & & & & \\ \frac{\partial A_{2,1}}{\partial P_1} & \frac{\partial A_{2,2}}{\partial P_2} & \frac{\partial A_{2,3}}{\partial P_3} & & & & \\ & \frac{\partial A_{3,2}}{\partial P_2} & \frac{\partial A_{3,3}}{\partial P_3} & \frac{\partial A_{3,4}}{\partial P_4} & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \frac{\partial A_{n-1,n-2}}{\partial P_{n-2}} & \frac{\partial A_{n-1,n-1}}{\partial P_{n-1}} & \frac{\partial A_{n-1,n}}{\partial P_n} & \\ & & & & \frac{\partial A_{n,n-1}}{\partial P_{n-1}} & \frac{\partial A_{n,n}}{\partial P_n} & \end{bmatrix} \begin{bmatrix} \delta P_1 \\ \delta P_2 \\ \delta P_3 \\ \vdots \\ \delta P_{n-1} \\ \delta P_n \end{bmatrix} = \begin{bmatrix} C_1 - A_{1,1}P_{old_1} - A_{1,2}P_{old_2} \\ C_2 - A_{2,1}P_{old_1} - A_{2,2}P_{old_2} - A_{2,3}P_{old_3} \\ C_3 - A_{3,2}P_{old_2} - A_{3,3}P_{old_3} - A_{3,4}P_{old_4} \\ \vdots \\ C_{n-1} - A_{n-1,n-2}P_{old_{n-2}} - A_{n-1,n-1}P_{old_{n-1}} - A_{n-1,n}P_{old_n} \\ C_n - A_{n,n-1}P_{old_{n-1}} - A_{n,n}P_{old_n} \end{bmatrix} \qquad (3.1\text{-}48)$$

A two tolerance approach is used to control the number of iterations required before a solution is reached. The first tolerance is the sum of the nodal pressure change vector $\delta\boldsymbol{P}$ for the iteration. The second is the sum of the right hand side vector. Both of these tend towards zero as the solution improves. If either meets a user specified criteria, convergence is assumed:

$$TOL_P > \sum \delta P_i \qquad (3.1\text{-}49)$$

or:

$$TOL_Q > \sum (C_i - \mathbf{A}\,P_{old})$$

Partial derivatives are calculated using both analytic and numeric approaches. Analytic approaches are used when the matrix term is linear or is a simple polynomial. Numeric approaches are used for all other cases.

3.2 Functional Approximations

GTFM uses functional approximations to represent:

1) well-bore boundary conditions varying as a function of time (e.g., variable pumping rates),

2) radial heterogeneity due to formation properties varying as a function of distance from the well-bore, and

3) pressure-dependent non-linear parameters where parameter values vary as a function of pressure.

GTFM uses several methods for expressing these functional approximations. Most methods require that XY data representing the desired function be available. X values are also usually required to be in ascending order. Available methods are:

1) polynomial of order 1 to 10 - a polynomial of specified order is calculated using a linear regression on the entered XY data. The polynomial will not represent the input data exactly unless the order of the polynomial is one less than the number of points in the input data.

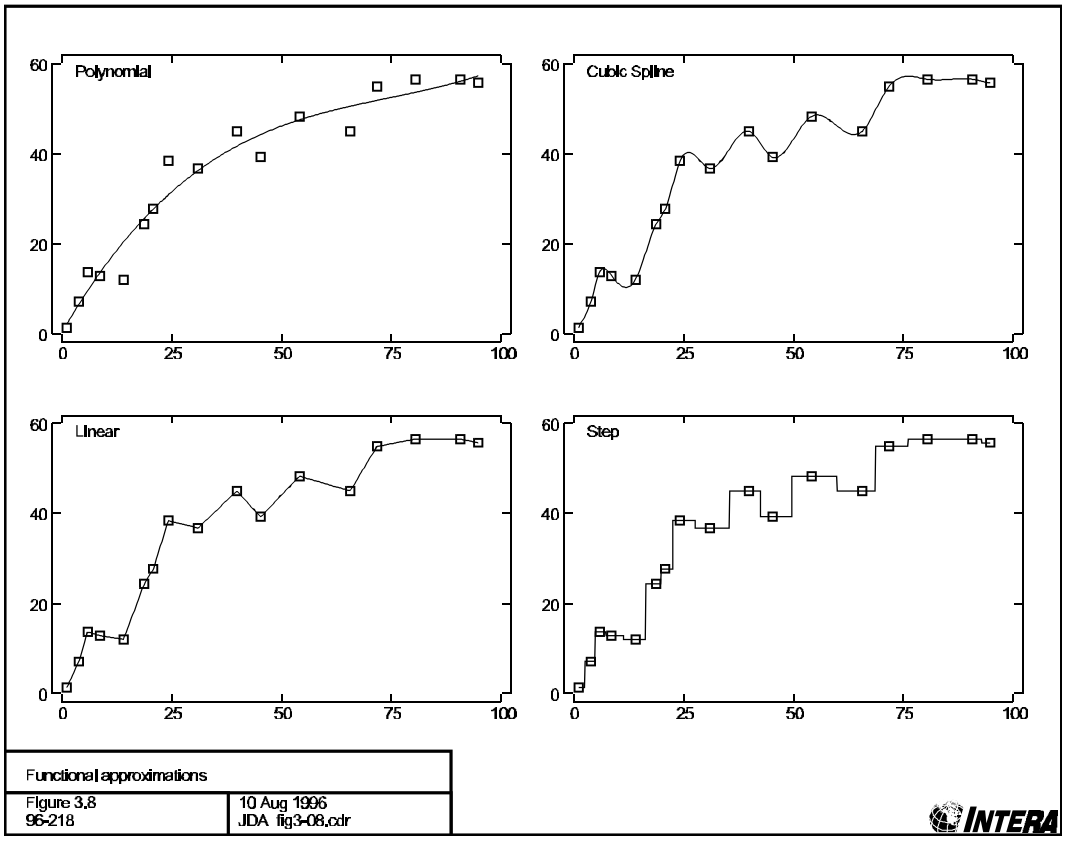
2) cubic spline - a piece-wise polynomial approximation of the XY data is calculated, continuos in first and second derivatives. Function slopes at the extremes must be specified. A tension factor can be used to modify the shape of the function. The cubic spline will represent each point in the input data exactly.

3) linear - the functional representation is a series of straight lines joining consecutive XY data points. Derivatives are not continuos.

4) step - function values are constant at XY data point values. Values change at the linear mid-point between adjacent data points.

Figure 3.8 illustrates the functional approximations for a single XY data set.

Figure 3.8 Functional approximations



3.3 Data Analysis

Apart from it's simulation capabilities, GTFM contains facilities for performing various transformations and processing on user-supplied data and on simulation results. User-supplied data are usually well-test results acquired in the field. These data are typically available in the form of a pressure and/or flow-rate versus time record, describing the response of the formation to the testing procedures over the duration of the test. Ancillary data such as test-zone temperature versus time may also be available. GTFM produces simulation results in a similar form. The predicted pressure and/or flow rates in the test-zone (or pressure at specified radii from the test-zone) are produced for each time step in the simulation. The time record starts at the specified test-start time.

In many cases, the available form of the user-supplied and/or the GTFM predicted data is not appropriate for further analysis. GTFM provides the tools required to further process both user and simulation data. Additionally, GTFM provides a number of standard analytic tools used in well-test interpretation. These tools can be categorized as scaling/transformations, derivative calculations, and time/superposition functions.

3.3.1 Scaling /Transformations

User-supplied and GTFM predicted data contain X and Y components. Scaling/transformation procedures are applied to each component independently. Available scaling procedures include: multiplying or dividing data by constants, and/or adding or subtracting constants from the data. Scaling is useful for normalizing, converting units, and offsetting data. In certain cases, scaling constants can be replaced by variables associated with each simulation, such as static formation pressure.

Transformations can also be applied, either before or after scaling is applied. There are 11 available transformations: natural log, log base 10, square root, inverse, exponent, raise 10 to the power, square, absolute value, square root of the square root, inverse square root, and log base 10 of the absolute value.

3.3.2 Derivative Calculations

Pressure derivatives are frequently used in well-test interpretation to determine reservoir properties. GTFM can calculate the derivative of any XY data set with respect to several time functions. Available time functions include: normal, superposition, Horner, and Argawal time. These are described in further detail in the next sub-section. When computing the derivative, the GTFM derivative calculation procedures may also perform smoothing on the input data set. Smoothing is necessary to produce a useful derivative if the input data is noisy.

All derivative procedures select a subset of the available data on either side of a data point and use the subset as the basis for calculating the derivative value at the data point. There are six methods for selecting data subsets and calculating derivatives:

single point	derivative is calculated as slope of line joining adjacent data points. The X value for the derivative is set as the linear average of the two data values used.
2 slope average	the derivative at a point is calculated as the average of the slopes of the lines joining the point to adjacent points.
window	a specified number of data values on either side of the point are used in the calculation.
% lin span	all points that have X values within a specified distance of the point at which the derivative is being calculated are used in the calculation. The distance is expressed as a percentage of the linear range of the entire data set (X_{max} - X_{min}).
% log span	all points for which the log of the X value are within a specified distance of the log of the X value of the point at which the derivative is being calculated are used in the calculation. The distance is expressed as a percentage of the log range of the entire data set ($\log(X_{max})$ - $\log(X_{min})$).
full polynomial	the entire data set is used. A best-fit polynomial of order n is determined using a linear regression algorithm. Coefficients of the polynomial describing the derivative of the best-fit polynomial are determined and the derivative calculated for each X value in the input data set.

The “single point” and “two-slope average” methods are useful only for very smooth field data sets or for synthetic data generated by GTFM or by another method. The three windowed methods provide sufficient smoothing to render usable derivatives. The degree of smoothing is increased by increasing the number of points in the window, either directly for the “window” method, or by increasing the percentage of the data span. The full polynomial method, which yields very smooth derivatives, is unfortunately almost useless for normal test-interpretation as typical well-test responses are not well described by polynomials.

Further options for derivative calculation are available for the three windowed methods (described above as window, % lin span and % log span). These options determine how the derivative is calculated using the subset of the data which falls within the window:

1) regression - a best-fit straight-line for the windowed subset is determined using a linear regression algorithm. The slope of the line is the derivative at the selected point.

2) Clark - the derivative is calculated using an algorithm developed by Clark and von Golf-Racht, 1985:

$$\frac{dy_i}{dx_i} = \frac{\frac{y_i - y_s}{x_i - x_s}(x_e - x_i) + \frac{y_e - y_i}{x_e - x_i}(x_i - x_s)}{x_e - x_s} \qquad (3.3\text{-}1)$$

where:

$\frac{dy_i}{dx_i}$ = derivative at point i

x_s, y_s = X and Y value of point at start of window

x_e, y_e = X and Y value of point at end of window

3) Simmons - the algorithm developed by Simmons is used to calculate the derivative at the data point:

for i = 1

$$\frac{dy_1}{dx_1} = \frac{\left[\left(1 - \frac{(x_e - x_1)^2}{(x_m - x_1)^2}\right) y_1 + \frac{(x_e - x_1)^2}{(x_m - x_1)^2} y_m - y_e\right]}{\frac{(x_e - x_1)^2}{x_m - x_s} - (x_e - x_1)} \qquad (3.3\text{-}2)$$

for i = n

$$\frac{dy_n}{dx_n} = \frac{\left[\left(1 - \frac{(x_n - x_s)^2}{(x_n - x_m)^2}\right) y_n + \frac{(x_n - x_s)^2}{(x_n - x_m)^2} y_m - y_s\right]}{(x_n - x_s) - \frac{(x_n - x_s)^2}{x_n - x_m}}$$

otherwise:

$$\frac{dy_i}{dx_i} = \frac{(x_i - x_s)^2 y_e + ((x_e - x_i)^2 - (x_i - x_s)^2) y_i - (x_e - x_i)^2 y_s}{(x_i - x_s)^2 (x_e - x_i) + (x_e - x_i)^2 (x_i - x_s)}$$

where:

x_m = mid-point x value

y_m = mid-point y value

3.3.3 Time/Superposition Functions

Time functions are used to adjust for non-ideal initial conditions such as multi-rate flow periods and to perform specialized interpretations such as Horner plots. GTFM supports four time functions in addition to normal time.

Horner time (Horner, 1951) is described as:

$$t_h = \frac{t_p + \Delta t}{\Delta t} \quad (3.3\text{-}3)$$

where:
t_h = Horner time
t_p = length of flow period before shut-in
Δt = time since shut-in

Argawal equivalent time (Argawal, 1980) is:

$$t_e = \frac{t_p \, \Delta t}{t_p + \Delta t} \quad (3.3\text{-}4)$$

where:
t_e = Argawal equivalent time

Multiple rate Horner superposition time:

$$t_h = \frac{1}{q_n} \sum_{i=1}^{n} q_i \log\left(\frac{t_n - t_{i-1} + \Delta t}{t_n - t_i + \Delta t} \right) \quad (3.3\text{-}5)$$

where:
n = number of flow periods
t_i = end-time of flow period i
q_i = flow rate during period i
t_0 = 0.0

Multiple rate Bourdet superposition time:

$$t_s = \ln(\Delta t) + \frac{1}{q_n - q_{n-1}} \sum_{i=1}^{n-1} (q_i - q_{i-1}) \ln\left(\Delta t + \sum_{j=i}^{n-1} (t_{j+1} - t_j) \right) \quad (3.3\text{-}6)$$

where:
t_s = superposition time
q_0 = 0.0

3.4 Optimization

One of GTFM's most useful capabilities is the ability to perform optimizations or inverse modelling. These procedures search for the combination of parameter values which minimize the difference between two data sets: a field data set, and simulation results. Two optimization algorithms are available within GTFM: the downhill Simplex method and Levenberg-Marquardt. Both are described in some detail in Press et al (1992): Simplex in Section 10.4; Levenberg-Marquardt in Section 15.5. Both algorithms have the same goal: to minimize a function. The following sub-sections describe: the functions to be minimized; parameter normalisations used in optimizations; the optimization

procedures; the calculation of fit statistics; the calculation of covariance matrices and Jacobian data; and the determination of single- and joint-parameter confidence intervals.

3.4.1 Minimization Functions

In GTFM, the function to be minimized is the sum of one or more fit component functions:

$$f(x;\mathbf{a}) = \sum_{i=1}^{n_f} f_i(x;\mathbf{a}) \qquad (3.4\text{-}1)$$

where:

- $\mathbf{a}$ = vector of parameters to be optimized
- $f(x;\mathbf{a})$ = function to be minimized
- $f_i(x;\mathbf{a})$ = fit component function
- n_f = number of fit components

each of which represents some function comparing field or processed data and analogous simulation results. For example, one fit component may compare flow rates, another pressures in an observation well, while a third may compare pressure derivatives. Typically the X^2 (Chi-squared) function is used:

$$f_i(x;\mathbf{a}) = \frac{1}{\sigma_i^2} \sum_{j=1}^{N_i} \left[y_j - y(x_j;\mathbf{a})\right]^2 \qquad (3.4\text{-}2)$$

where:

- N_i = number of points to be fitted in data set i
- y_j = data to be fitted
- $y_j(x_j;\mathbf{a})$ = GTFM calculated result at time x_j
- σ_i = measurement error of data set

The data to be fitted, y_j, may be actual field data, processed field data (e.g. pressure derivatives), or an interpolated data set generated from actual/processed field data using one of the functional approximations described in Section 3.2. Interpolations can be used advantageously when the original field data are poorly spaced in time, however, the statistical validity of calculated confidence intervals may be questionable.

Equation 3.4-2 must be used when confidence limits are to be calculated or when the Levenberg-Marquardt algorithm is used. If more than one fit component is used, σ_i may be replaced with the normalized ratio of the means of the fit component data:

$$\sigma_i = \frac{\text{mean}_i}{\text{mean}_{min}} \qquad (3.4\text{-}3)$$

where:

- mean_i = mean of data set i
- mean_{min} = minimum mean of all data sets to be fitted

This approach should only be used when the actual measurement error is not known. Confidence intervals calculated using equation 3-4.3 will not be correct and should be used for qualitative comparisons only.

The following fit components can be used when the Simplex algorithm is used and confidence intervals or fit statistics are not required:

$$f_i(x;\mathbf{a}) = Sc\left(\sum_{j=1}^{N_i}\left[y_j - y(x_j;\mathbf{a})\right]^2\right) \quad (3.4\text{-}4)$$

and/or:

$$f_i(x;\mathbf{a}) = Sc\left(\sum_{j=1}^{N_i}\left|y_j - y(x_j;\mathbf{a})\right|\right)$$

where:

Sc(x) = scale and/or transform function (see Section 3.3.1)

3.4.2 Parameter Normalization

The parameter vector, **a**, used in the optimization consists of parameters selected by the user for optimization and normalized over a range 0 to 1. Both linear and logarithmic normalisations are available:

$$P_i = P_{i_{min}} + a_i\,(P_{i_{max}} - P_{i_{min}}) \quad (3.4\text{-}5)$$

or:

$$P_i = 10^{\log_{10}(P_{i_{min}}) + a_i\,(\log_{10}(P_{i_{max}}) - \log_{10}(P_{i_{min}}))}$$

where:

a_i = parameter value used within optimizer
P_i = parameter value used within simulator
$P_{i_{min}}$ = user specified minimum value for optimized parameter
$P_{i_{max}}$ = user specified minimum value for optimized parameter

3.4.3 Optimization Procedures

3.4.3.1 Simplex

This sub-section presents a brief overview of points relevant to using the Simplex procedure in GTFM. The Simplex procedure is described in detail in Section 10.4 of Press et al (1992). The algorithm requires $n_p + 1$ initial evaluations of the fit function to form the vertices of the "simplex",

a non-degenerate geometric structure in n_p dimensional parameter space:

$$S_o = f(x; \mathbf{a_0}) \quad (3.4\text{-}6)$$

and:

$$S_i = f(x; \mathbf{a_i})$$

where:

S_i = simplex vertices
$\mathbf{a_0}$ = initial user-entered estimate of parameter values
$\mathbf{a_i}$ = $\mathbf{a_0} + \lambda \mathbf{e_i}$
λ = vertex span
e_i = unit vector with element i = 1,all others = 0

The Simplex algorithm then examines the vertex values and successively adjusts parameter estimates $\boldsymbol{a}$ so that the values are reduced. The optimization is complete when the following criteria is reached:

$$TOL > 2 \frac{|S_{max} - S_{min}|}{|S_{max}| + |S_{min}|} \quad (3.4\text{-}7)$$

where:

S_{max} = $MAX(S_i)$, $i = 0..n_p$
S_{min} = $MIN(S_i)$, $i = 0..n_p$
TOL = user specified tolerance

3.4.3.2 Levenberg-Marquardt

The Levenberg-Marquardt (L-M) algorithm is described in Section 15.5 of Press et al (1992). The L-M method is a gradient method which requires evaluation of the Hessian matrix (partial derivative of the function with respect to its parameters) at each successful iteration. New estimates of the parameters are determined based on the gradient and a step parameter, λ. The smaller the value of λ, the greater the step change. If the new estimate improves the fit, the fit-point is updated and the gradient calculated again. Otherwise, λ is increased and a new estimate point recalculated. If a previous iteration improved the fit without increasing λ, the λ value is reduced by an adjustment factor to accelerate the convergence to a fit point. The fit-point is reached when either of the following conditions are met:

$$P_{TOL} > 2 \frac{|f(x; \mathbf{a_{min}}) - f(x; \mathbf{a_{last}})|}{f(x; \mathbf{a_{min}}) - f(x; \mathbf{a_{last}})} \quad (3.4\text{-}8)$$

or:

$$R_{TOL} > 1 - \frac{f(x; \mathbf{a_{last-5}})}{f(x; \mathbf{a_{last}})}$$

where:

a_{min} = parameters for previous minimum X^2 value
a_{last} = parameters for current iteration
P_{TOL} = user specified parameter tolerance
R_{TOL} = user specified relative change tolerance

3.4.4 Fit Statistics

After a Chi-squared optimization has reached a fit estimate the following fit statistics are calculated:

$$SSE_i = \sum_{j=1}^{N_i} [y_j - y(x_j; \mathbf{a_f})]^2 \qquad (3.4\text{-}9)$$

$$SSE = \sum_{i=1}^{n} SSE_i$$

$$N = \sum_{i=1}^{n} N_i$$

$$MSE = \frac{SSE}{N}$$

$$\text{est } \sigma_i = \frac{SSE_i}{N_i - n_p}$$

$$\text{est } \sigma = \frac{SSE}{N - n\, n_p}$$

where:

SSE_i	=	sum of squared errors for fit component i
SSE	=	sum of squared errors for full fit
N	=	number of points in full fit
MSE	=	mean squared errors for full fit
est σ_i	=	estimated variance for fit component i
est σ	=	estimated variance for full fit

3.4.5 Covariance Matrices

GTFM calculates several covariance matrices at the optimization solution point. $\boldsymbol{C_a}$, the "actual covariance matrix", is the estimated covariance matrix of the standard errors in the fitted normalized parameters **a**. It is calculated as:

$$\mathbf{C_a} = \left[\frac{1}{2} \frac{\partial^2 X^2}{\partial a_k \partial a_l} \right]^{-1} \qquad (3.4\text{-}10)$$

where:

C_a	=	covariance matrix
$\frac{\partial^2 X^2}{\partial a_k \partial a_l}$	=	Hessian matrix

Two formulations are used in calculating the Hessian:

Half (1st order): (3.4-11)

$$\frac{\partial^2 X^2}{\partial a_k \partial a_l} = 2\sum_{i=1}^{n_f}\frac{1}{\sigma_i^2}\sum_{j=1}^{N_i}\left[\frac{\partial y(x_j;\mathbf{a})}{\partial a_k}\frac{\partial y(x_j;\mathbf{a})}{\partial a_l}\right]$$

or, Full (2nd order):

$$\frac{\partial^2 X^2}{\partial a_k \partial a_l} = 2\sum_{i=1}^{n_f}\frac{1}{\sigma_i^2}\sum_{j=1}^{N_i}\left[\frac{\partial y(x_j;\mathbf{a})}{\partial a_k}\frac{\partial y(x_j;\mathbf{a})}{\partial a_l} - [y_j - y(x_j;\mathbf{a})]\frac{\partial^2 y(x_j;\mathbf{a})}{\partial a_l \partial a_k}\right]$$

The first (Half) calculation ignores the second derivative terms and has the advantage of requiring less simulations. It is also guaranteed to result in a positive definite Hessian matrix. The second (Full) equation is more strictly correct, but may be singular at the fit point if the fit is poor. Partial derivatives are calculated numerically:

Half: (3.4-12)

$$\frac{\partial y(x_j;\mathbf{a})}{\partial a_k} = \frac{y(x_j;\mathbf{a}_{+\Delta_k}) - y(x_j;\mathbf{a}_f)}{\Delta a}$$

Full:

$$\frac{\partial y(x_j;\mathbf{a})}{\partial a_k} = \frac{y(x_j;\mathbf{a}_{+\Delta_k}) - y(x_j;\mathbf{a}_{-\Delta_k})}{2\,\Delta a}$$

$$\frac{\partial^2 y(x_j;\mathbf{a})}{\partial a_k \partial a_k} = \frac{y(x_j;\mathbf{a}_{+\Delta_k}) - 2\,y(x_j;\mathbf{a}_f) - y(x_j;\mathbf{a}_{-\Delta_k})}{\Delta a^2}$$

$$\frac{\partial^2 y(x_j;\mathbf{a})}{\partial a_k \partial a_l} = \frac{y(x_j;\mathbf{a}_{+\Delta_{k,l}}) + y(x_j;\mathbf{a}_f) - y(x_j;\mathbf{a}_{+\Delta_k}) - y(x_j;\mathbf{a}_{+\Delta_l})}{\Delta a^2}$$

where:

- $\mathbf{a_f}$ = parameter values at fit point
- Δa = derivative span
- $\mathbf{a}_{+\Delta_k}$ = $\mathbf{a_f}$, except element $a_{\Delta_k} = a_{f_k} + \Delta a$
- $\mathbf{a}_{-\Delta_k}$ = $\mathbf{a_f}$, except element $a_{\Delta_k} = a_{f_k} - \Delta a$
- $\mathbf{a}_{+\Delta_{k,l}}$ = $\mathbf{a_f}$, except element $a_{\Delta_k} = a_{f_k} + \Delta a$ and element $a_{\Delta_l} = a_{f_l} + \Delta a$

The derivative span, Δa, is calculated using an iterative procedure:

$$Z_{new} = \frac{f(a+\Delta a) - f(a-\Delta a)}{1.5\Delta a} + \frac{f(a-2\Delta a) - f(a+2\Delta a)}{12\Delta a} \quad (3.4\text{-}13)$$

REPEAT

$Z_{old} = Z_{new}$

$\Delta a = \frac{\Delta a}{2}$

$Z_{new} = \frac{f(x+\Delta a) - f(x-\Delta a)}{1.5\Delta a} + \frac{f(x-2\Delta a) - f(x+2\Delta a)}{12\Delta a}$

criteria = MAX(TOL,TOL Z_{new})

UNTIL $\frac{|Z_{new} - Z_{old}|}{15}$ > criteria

where:

$f(a+\Delta a) = f(x; \mathbf{a}_{+\Delta_k})$, where k is parameter with largest Δf

TOL = user specified tolerance

$\boldsymbol{C_e}$, the "estimated covariance" is also calculated using equations 3.4-10 and 3.4-11 except that σi in 3.4-11 is replaced with *est* σ_i. Component estimated covariances $\boldsymbol{C_{ei}}$ are also calculated for each separate fit component.

The covariance matrices $\boldsymbol{C_a}$, $\boldsymbol{C_e}$, $\boldsymbol{C_{ei}}$ are all with respect to the normalized parameters $\boldsymbol{a}$. De-normalization procedures can be applied so that covariance matrices reflect the ranges of the parameter values used in the simulator:

$$\mathbf{C_D} = \mathbf{T\ C\ T} \quad (3.4\text{-}14)$$

where:

$\mathbf{C_D}$ = denormalized covariance

$\mathbf{T}$ = $T_{i,i} = P_{i_{max}} - P_{i_{min}}$ linear normalization, or,

$T_{i,i} = \log(P_{i_{max}}) - \log(P_{i_{min}})$ logarithmic

$T_{i,j} = 0, \quad i \neq j$

Jacobian data are also calculated:

$$J_{k,j} = \frac{\sigma_{p_k}}{\Delta a\ \sigma_j^2} (y(x_j;\mathbf{a}_\Delta) - y(x_j;\mathbf{a}_f)) \quad (3.4\text{-}15)$$

where:

$J_{k,j}$ = Jacobian value for parameter k, point j

σ_{p_k} = std. deviation of parameter k

3.4.6 Confidence Limits

Single- and joint-parameter confidence limits can be calculated for any covariance matrix ***C***. The single parameter confidence limits are simply:

$$\delta a_k = \pm\sqrt{\Delta X_1^2}\,\sqrt{C_{kk}} \qquad (3.4\text{-}16)$$

where:

ΔX_1^2 = Chi-squared value for 1 degree of freedom corresponding to confidence limit

The joint confidence area for two parameters takes the form of an ellipse in parameter space centred on the fit point:

$$\frac{\delta a_1}{b_1^2} + \frac{\delta a_2}{b_2^2} = 1 \qquad (3.4\text{-}17)$$

where:

b_i = lengths of ellipse semi-major axes

The lengths of the semi-major axes are calculated using the eigenvalues of the inverse of the projected covariance matrix:

$$b_i = \sqrt{\frac{\Delta X_2^2}{D_i}} \qquad (3.4\text{-}18)$$

where:

D_i = eigenvalues of $\mathbf{C}_{\mathbf{proj}}^{-1}$

The rotation of the ellipse is defined by the eigenvectors of $\boldsymbol{C}_{proj}^{-1}$. Eigenvectors and eigenvalues are calculated using a Jacobi transformation routine presented in Press et. al. The values of the Chi-squared function corresponding to different confidence intervals and degrees of freedom are presented in the table below:

Table 3.4-1 Chi-squared values

confidence interval %	degree of freedom		
	1	2	3
68.3	1.00	2.30	3.53
90	2.71	4.61	6.25
95.4	4.00	6.17	8.02
99	6.63	9.21	11.3

Equations 3.4-13 and 3.4-14 extend naturally to three dimensions to provide confidence ellipsoids for three parameters.

3.5 Sampling

In sampling mode, GTFM combines optimizations of fitting-parameters with sampled values of non-fitting parameters and/or sequence boundary conditions to create a statistical description of the range of possible fitting parameter values.

GTFM uses sampling routines translated from the Fortran routines described in Iman and Shortencarier (1984). Two sampling modes are supported: Monte-Carlo and Latin-Hypercube (LHS). Up to 10 000 samples of a maximum of fifty variables can be generated. GTFM uses a seed-based random number generator. The seed is user input so that sample realizations are reproducible.

The LHS sampler allows correlations between variables to be specified or generated randomly. Figure 3.9 shows a plot of two strongly correlated (0.90) uniform distributions.

GTFM allows selected variables to be described by any one of six distributions: normal, log-normal, uniform, log-uniform, triangular, or log-triangular. Figure 3.10 shows histograms for example normal, triangular, and uniform distributions.

In post-processing, GTFM calculates various summary statistics for both sampled and optimized variables. Statistics include: minimum, maximum, mean, variance and standard deviation. Note that if the sampled variable or optimized variable is logarithmic (loguniform, lognormal, or logtriangular distributions for sampled variables, log optimization stepping for optimized variables) the mean, variance, and standard deviation are calculated using the $\log_{10}$ of the data. The mean is then raised to the power of 10 for display. Variance and standard deviation remain in log form.

Correlation matrices are also calculated for all sampled and optimized parameters. The linear correlation coefficient (Pearson's *r*) and a rank correlation coefficient (Spearman's r) are calculated for every combination of variable. For logarithmic data, the $\log_{10}$ of the data is taken before correlations are calculated. Pearson's *r* is given as:

$$r_p = \frac{\sum_i (x_i - \bar{x})\,(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2}\,\sqrt{\sum_i (y_i - \bar{y})^2}} \qquad (3.5\text{-}1)$$

Figure 3.9 Example correlation of two uniform distributions (1000 samples, correlation = 0.90)

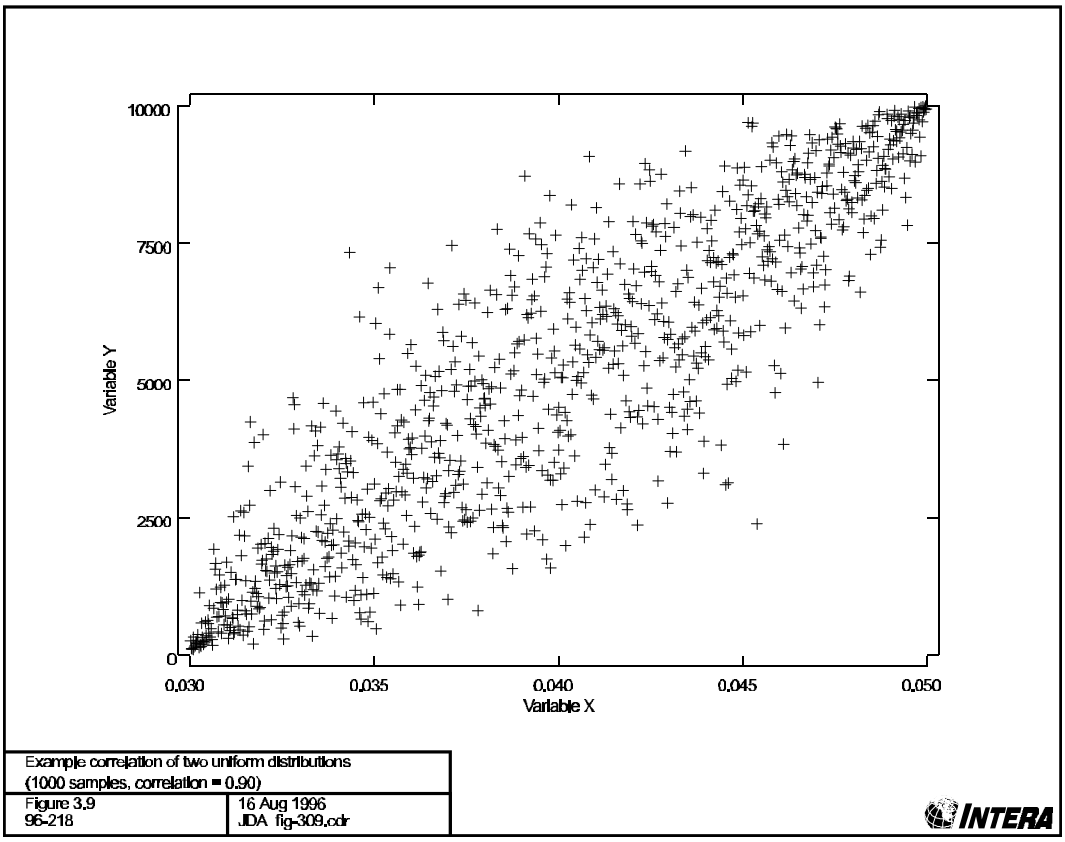
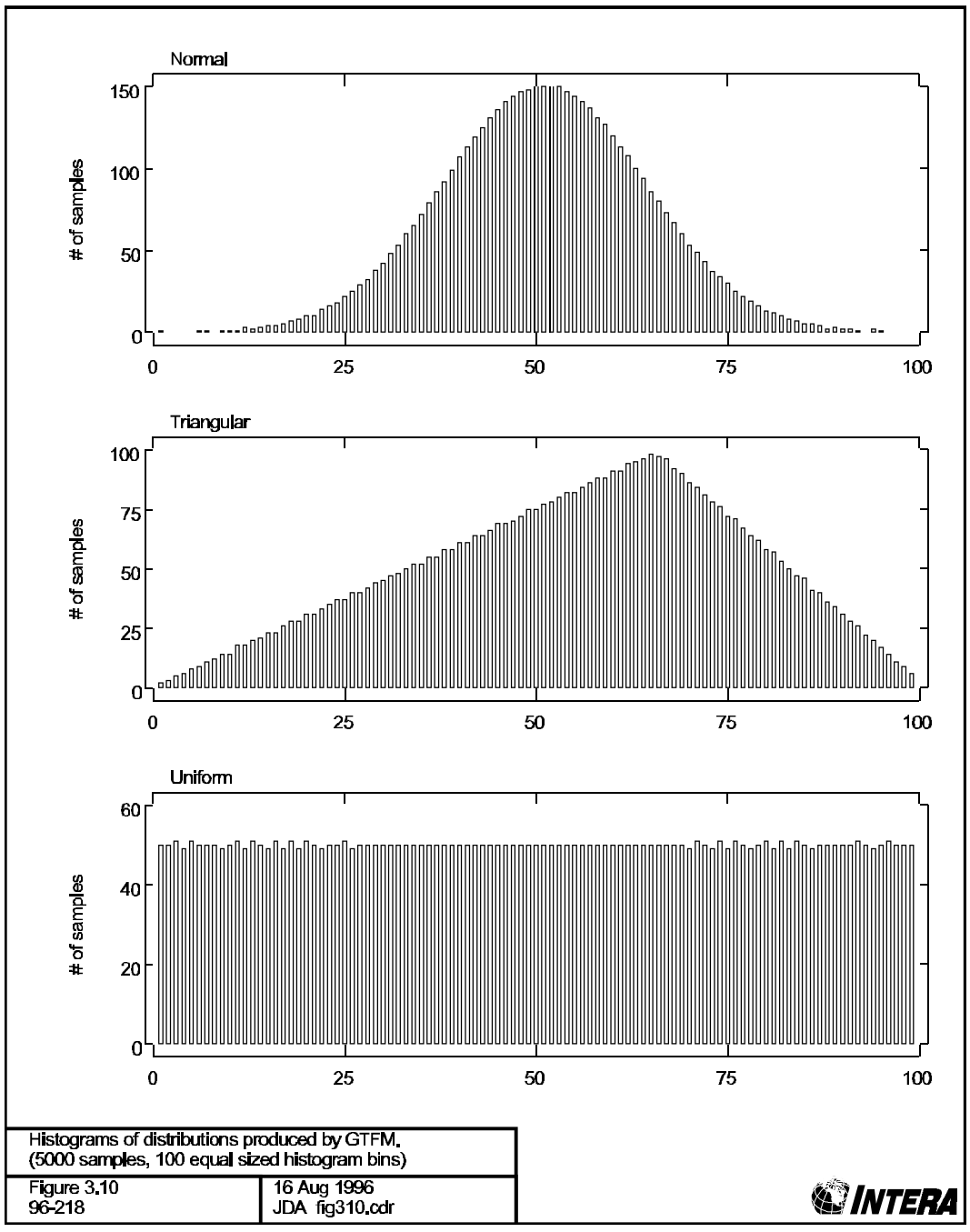


Figure 3.10 Histograms of distributions produced by GTFM (5000 samples, 100 equal sized histogram bins)



Spearman's r is:

$$r_s = \frac{\sum_i (R_i - \bar{R})\,(S_i - \bar{S})}{\sqrt{\sum_i (R_i - \bar{R})^2}\,\sqrt{\sum_i (S_i - \bar{S})^2}} \qquad (3.5\text{-}2)$$

where:

R_i = rank of x_i among all x
S_i = rank of y_i among all y

4 SOFTWARE ARCHITECTURE

This section documents: the computer languages used in the GTFM source code, the specific compilers, linkers and other tools required to build an executable from source code, and the structure of the GTFM program and its supporting libraries. Appendix A contains a listing of all GTFM source files with a brief description of the type of source (Pascal, assembler, or C), the purpose of the code, and the architectural classification of each file.

4.1 Language/Tools

GTFM is written primarily in Pascal with some additional code in C and 80386 assembler. The Pascal dialect used is Metaware Professional Pascal.

Pascal source code in GTFM is compiled with version 2.81 of the 32-bit Metaware compiler targeted for small model 32-bit Intel 80386/80486. Assembler code is translated with the Phar Lap Assembler (386ASM), version 4.0. C code is compiled with version 1.73 of the 32-bit Metaware High C compiler targeted for small model 32-bit Intel 80386/80486.

Object files and libraries are linked with version 4.0 of the Phar Lap linker (386LINK). Subsequently, the PharLap BIND386 utility is used to create a standalone 32-bit executable.

4.1.1 Compiler Configuration

The Metaware compiler uses a profile file, PP.PRO to control compiler toggles for code optimizations etc. The following settings are included in the PP.PRO file used for compiling all GTFM related Pascal source:

```
pragma Off(Check_stack);          -- no stack checks
pragma Off(Check_subscript);      -- no array subscript checks
pragma Off(Check_range);          -- no subrange assignment checks
pragma Off(Emit_line_table);      -- don't create debugging information

pragma On(Push_regsize);          -- optimizes parameter passing
pragma On(387);                   -- use hardware floating point
```

4.1.2 Linker Configuration

GTFM is linked by simply linking all object files and libraries described in Section 4.2. The linker switch governing stack size is set as **-stack 1024000**

4.2 Structure

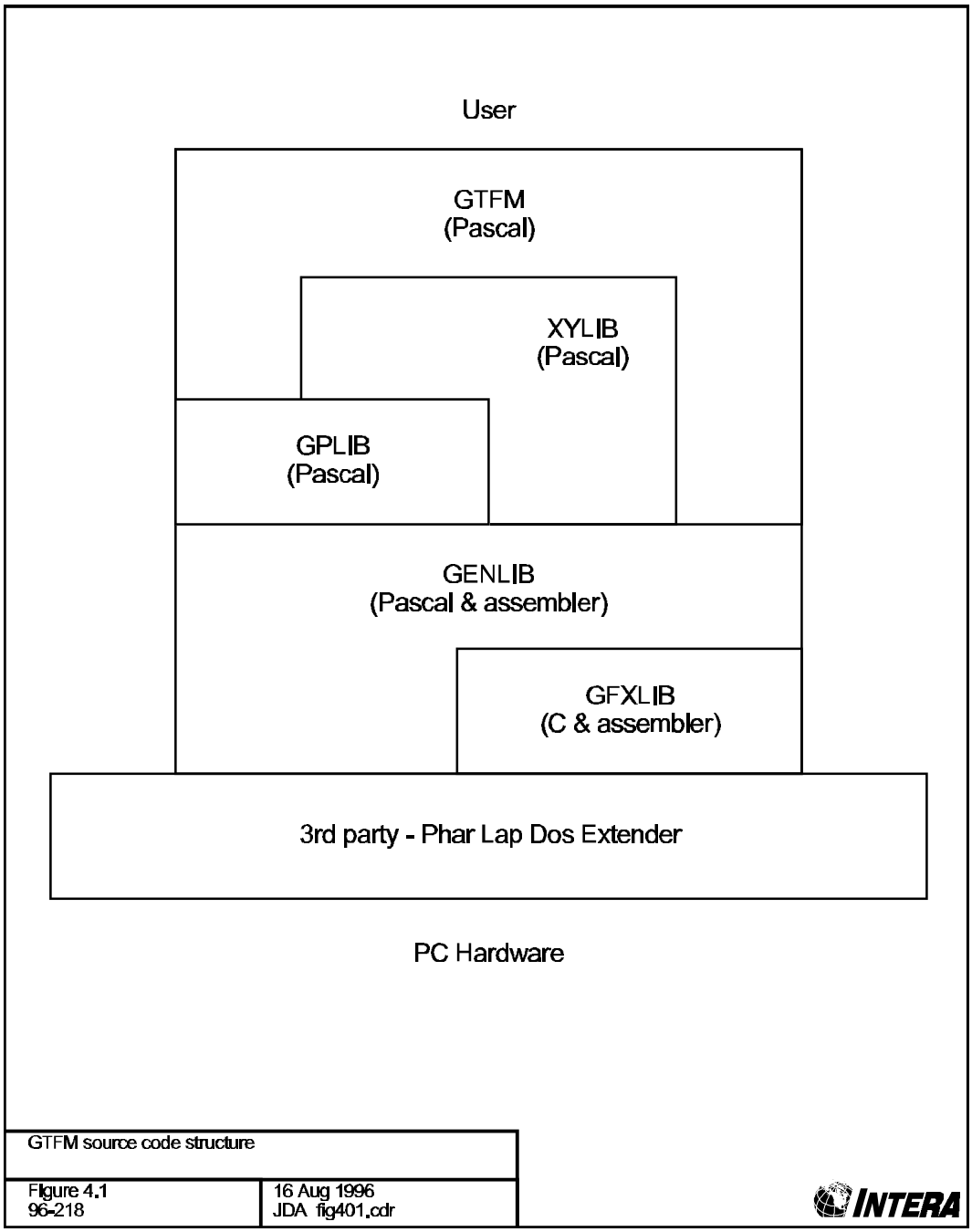
GTFM is a large interactive application, consisting of over 150 000 lines of code. It was designed in a modular fashion with segregation of source code modules based on functionality. GTFM uses several code libraries designed and developed by INTERA to support 32-bit DOS application programs. Code/library relationships are shown in Figure 4.1. The general functionality of each component is described in the table below:

Table 4.2-1 GTFM code components

Component	Function	# of source files
GTFM main	modules consisting of the source code unique to GTFM. Consists of all data input menus, simulator menus, and post-processing setup.	193 Pascal
XYLIB	implements entry/editing/plotting/printing and manipulation of XY data vectors.	14 Pascal
GPLIB	provides generic 2D and 3D plotting functionality, curve calculation and interpolation. Also includes standard menus used within GTFM to set up graphics related parameters such as axes formats.	51 Pascal
GENLIB	provide generic user interface support (menus, data entry), low-level device independent graphics, generic file-system support and hard copy output support. Also includes generic routines for system configuration (printer/plotter specification, screen graphics device selection, 256 colour palette entry/editing)	94 Pascal 11 Assembler
GFXLIB	3rd party graphics library which provides low-level device dependent screen graphics support. Supplied by C-Source Limited.	C and Assembler
PharLap	3rd party DOS extender which provides protected mode 32 bit capabilities under DOS.	n/a

Appendix A contains tables listing file names, source code categories, and brief functional descriptions for all INTERA produced source code (i.e. GTFM Main, XYLIB, GPLIB, and GENLIB).

Figure 4.1 GTFM source code structure



5 INPUT/OUTPUT FILES

An explanation of all input and output files used by/created by GTFM is given in the table below:

Table 5.0-1 GTFM File I/O

File Type	Default Extension	Description
Configuration	CNF	binary input/output - as an interactive program, most GTFM input is manually entered by the user. The CNF file is used to store all user-entered data so they may be archived or retrieved for modification or for subsequent repeat runs.
Multiple Curve	MCV	binary input - contains functional descriptions of curves (see Section 3.2) created by the program CURVE. Each MCV file contains one or more curve descriptions. GTFM can use 3 MCV files simultaneously: 1) for parameters defined as f(P) 2) for parameters defined as f(r) 3) for parameters and/or sequence boundary conditions described as f(t)
Curve	CRV	binary input - contains a single curve description created by program CURVE. Multiple CRV files can be used for parameters and/or sequence boundary conditions described as f(t). CRV files are available only for backward compatibility with versions of GTFM prior to 4.50. They have been supplanted by MCV files.
Restart	RST	binary input/output - contains nodal radii/pressures to be used as initial conditions on subsequent run.
Post-processing	PST	binary input/output - contains specified output data from one or more forward simulations and/or final best-fit runs from optimization and/or optimization sampling mode runs.
Profile post-processing	PRF	binary input/output - contains nodal pressures at specified times or time increments. Can be used in profile post-processing to produce 2-D and 3-D plots of pressure vs. time and/or radial distance.
Optimization post-processing	OPT	binary input/output - contains optimization result data from one or more optimization and/or optimization sampling mode runs. Read by post-processing routines to produce covariance, fit statistics, and correlation output.
Fit surface post-processing	GOF	binary input/output - contains fit surface values from one or more fit-surface runs
Plotter	PLT	text output - all vector screen graphics in GTFM can be re-directed to a HPGL device (plotter/printer) for hardcopy or the HPGL commands can be written to a text file to allow plots to be imported into other software.
Listing	PRN	text output - all listing data in GTFM can be re-directed to a printer for hardcopy or the text can be written to a file to allow data to be imported into other software.

Table 5.0-1 GTFM File I/O		
File Type	**Default Extension**	**Description**
Hardware Configuration	HCF	binary input/output - contains data describing the hardware on the computer GTFM is installed on. Includes graphics adapter specs, printer/plotter type and ports, and graphics and menu screen colours. Read by GTFM on startup. The file can be modified by the user with the **Hardware Setup Menu** within GTFM.

REFERENCES

Argawal, R.G. 1980. A new approach to account for producing time effects when drawdown curves are used to analyze pressure buildup and other test data. SPE 9289

Barker, J. A., 1988. A generalized radial flow model for hydraulic tests in fractured rock. Water Resources Research, Vol 24, No 10, pg 1796-1804

Clark, D.G. and T.D. van Golf-Racht, 1985. Pressure derivative approach to transient test analysis: A high-permeability North Sea reservoir example. J. Pet. Tech., Nov 1985.

Horner, D.R., 1951. Pressure build-up in wells. Proc. Third World Pet. Cong. The Hague II, pp 503 - 521

Iman, R.L. and Shortencarier M.J., A FORTRAN 77 Program and User's Guide for the Generation of Latin Hypercube and Random Samples for Use with Computer Models. SAND83-2365, NUREG/CR-3624, Sandia National Laboratories, Albuquerque, NM, 1984.

Press, W.H, S.A. Teukolsky, W.T.Vetterling, B.P.Flannery, Numerical Recipes, Second Edition, Cambridge University Press, 1992.

Appendix A - Source Code Files

Three different types of source file are described in this Appendices. They are differentiated by file extension:

.p Pascal Source main body. The vast majority of the files described in this appendix are .p files. All .p files have an associated .pf file.

.pf Pascal Source interface description. Most .pf files are associated with .p files and are not described separately in this document. Some .pf files have no associated .p file. Generally these files define types and constants only. They are described in the following tables.

.asm 80386 assembler. These files are associated with GENLIB only. Generally, they provide access to BIOS and DOS interrupts for low-level access to the hardware and operating system.

Table A.1 GENLIB Source Files

File Name	Description	Size Lines
Assembler		
zcrsasm.asm	access to BIOS interrupts to control cursor size and location	91
zdosasm.asm	access to command line and Phar Lap utilities for moving data between separate DOS applications	241
zenvasm.asm	get the DOS environment string to access PATH and GTFM SET data	94
zfilasm.asm	access to DOS interrupts to get information about files (exists, size, directory data)	1336
zgenasm.asm	controls speaker for beeps, get real-time-clock time and date	180
zjmpasm.asm	implements C language setjmp and longjump for error recovery	192
zkbdasm.asm	accesses BIOS keyboard interrupt, gets next keystroke, shift key status	122
zmouasm.asm	accesses the mouse interrupt to get mouse data	144
zparasm.asm	initialize, send bytes over parallel port	113
zscrasm.asm	writes characters and attributes on text screen, reads data from screen into memory	380
zserasm.asm	initialize, send/ recieve bytes over serial port	278
Pascal		
z256col.p	routines to display and set graphically the colour pallette for super VGA 256 colour graphics	606
zalthelp.p	implements response to Alt-Z	111
zbuffile.p	support for combined sequential/direct access binary file I/O	174
zbuftype.pf	types to support zbuffile.p	24
zcmdmenu.p	implements typical 1D Options or small selection list menu	302
zcolmenu.p	all other 1D (i.e. not tabular) menus	460
zcorel.p	writes current graphics setup (pen colours, widths) in format compatible with CorelDraw v5.0 CORELPLT.INI file. Simplifies importing GTFM graphics into Corel Draw	224

Table A.1 GENLIB Source Files		
File Name	**Description**	**Size Lines**
zdatscr.p	writes menu data (text, numbers, etc) to the screen	204
zdevglob.p	global variables describing printer, plotter, and digitizer device attributes and port connections	6
zdigcalc.p	digitizer transform calculations - NOT USED BY GTFM	367
zdigent.p	enter data from digitizer - NOT USED BY GTFM	140
zdigglob.p	global variables for digitier - NOT USED BY GTFM	7
zdigtype.pf	types describing digitizer data - NOT USED BY GTFM	14
zdigutil.p	utilities to communicate with digitizer - NOT USED BY GTFM	269
zdirfile.p	binary file utilities - open, close, seek, read, write	121
zeltime.p	elapsed time - set event start, calculate time since event start with real time clock	35
zerrutil.p	error utilities - writes message and stack dump for fatal errors or ASSERTs	29
zfildir.p	performs a directory search given a file spec, returns results in a vector	244
zfilent.p	file name data entry with directory picking	1537
zfilerr.p	file error-trapping utilities - checks and traps Pascal and DOS critical errors during access	137
zfiloglb.p	global variables for text file or plotter file output	7
zfilpath.p	checks directory names, makes full names out of partial file specs	105
zfilutil.p	general file utilities - check if file exists, find file, delete file, file name utilities	464
zfkutil.p	setup function keys for use in menus and data entry - name and key number for display/trapping	83
zfullscr.p	general screen writing, text, text & attributes, and attributes. set/get location of main data entry column	189
zgcursor.p	graphics cursor utilities - write/clear cursor on screen, get/set position, set cursor type	179
zgenent.p	generic data entry - utilities to support data specific data entry	502
zgentype.pf	standard string and menu type definitions, character constants	60
zgenutil.p	misc routines - logical XOR, initialize real	20
zglobal.p	global variable - for all menu and data entry - escape to main menu (ALT-F1 in GTFM)	9
zgpeglob.p	global variables - graphics pen descriptions - ID, colour, thickness, line type	54
zgrfcons.p	global variiables - start and end of user addresable area of virtual graphics screen - 0 to 1000.0 vertical, 0 to 1000 * aspect ratio in horizontal	15
zgrfmutl.p	graphics menu utilities - select pen, enter symbol data, enter line type, enter label origin	575
zgrfoglb.p	global variables - graphics output window and text size in HP plotter space	8
zgrftype.pf	types for text size, line type and symbol description	32
zgrfutil.p	low level graphics - open/close graphics mode - draw line, text, symbol, get current status - pen number, line type, text size	1513
zgscglob.p	global variables - current graphics mode, RGB values for standard (0 - 15)and 256 colour pallette (16 - 255) colours	54
zhelpset.p	sets menu index number for context sensitive help - FULL HELP IS NOT IMPLEMENTED IN GTFM	24
zhlpglob.p	global variables - currently active ALT key combinations	6

Table A.1 GENLIB Source Files		
File Name	**Description**	**Size Lines**
zhpglglb.p	global variables - offset and scaling factors in HPGL space	7
zhpglutl.p	plotter command equivalents for zgrfutil.p routines - used when vector graphics are being translated to HPGL and sent to file or plotter	576
zincmutl.p	converts between inches and cm for data entry and screen display - all values internally in cm - I/O dependent on zunitglb.pf flag	113
ziopstr.p	converts I/O port description to string for display	54
zioptype.pf	types to describe serial and parallel ports	33
zioputil.p	higher level routines for I/O port communication - call routines in zserasm.asm and zparasm.asm	267
zjmptype.pf	type describing context record (SP, BP, DS, ES regs) required to support longjump	21
zkbdglob.p	global variables	7
zkbdmisc.p	miscellaneous routines which require a single keystroke entry (CONFIRM, press any key, etc)	159
zkbdsupp.p	general F-key and A-key support for data entry and menu utilities	89
zkbdtype.pf	types for low-level keyboard routines	30
zmemutil.p	error checked memory allocation from heap, support for conformant array allocation	237
zmencons.p	constants used in many menus	14
zmenscrl.p	utilities to support scroll status data in multi page menus	438
zmensupp.p	F-key and A-key support for 1D and 2D menus	82
zmentype.pf	types and constants for variables used by menu routines	153
zmenutil.p	generic menu utilities - writes menu to screen with correct attribute	73
zmouse.p	low-level mouse routines - get/set coord, set sensitivity, set limits	164
zmsgutil.p	standard user error, status, and process progress messages	317
znumstr.p	converts numeric values to strings and vice-versa	275
zolhelp.p	main on-line/context sensitive help procedure - FULL HELP IS NOT IMPLEMENTED IN GTFM	863
zolhglob.p	global variables for on-line help	8
zolhtype.pf	types to support access to binary HLP file	33
zoutglob.p	global variables for output device selection - screen, printer, plotter, file	7
zoutmenu.p	menus to supports text and graphics output to hard copy devices or files	1322
zoututil.p	generic utilities to open/close and write to output device/file	381
zpltcons.p	constants describing plotter device characteristics (# of pens, paper size)	6
zpltglob.p	global variable containing description of current system plotter	7
zplttype.pf	types to describe plotter and status	22
zprnglob.p	global variable containing description of current system printer	7
zprntype.pf	types to describe printer and status	28
zptrutil.p	routines for initializing/allocating/de-allocating vectors of standard data types (words, reals, booleans)	192
zrandom.p	random number generator (translated from Sandia LHS package)	112
zrealfun.p	miscellaneous real functions and constants - e, power, mod, div, log base 10, gamma, erfc, inverse erfc, inverse normal	392
zrealtdu.p	real time date utilities - get time and date	43

Table A.1 GENLIB Source Files		
File Name	**Description**	**Size Lines**
zrfent.p	enter real format - general, fixed, or scientific, number of decimal places	76
zrfstr.p	converts integer representing real format to string for display	43
zscrglob.p	global variables - byte values for screen attributes associated with normal, inverse, dim, and blinking	8
zscrtype.pf	types for storing screen data	17
zsetstr.p	routines for entering and checking printer setup strings with non-ASCII valuesrror,	66
zstdent.p	standard data entry - enter text, numeric values	355
zstdscr.p	low level text screen utilities - clear screen, clear row, scroll rows	160
zstrcons.p	constant for “not set” menu items - ‘---’	8
zstrtend.p	program startup and shutdown utilities - initializes graphics, file system, heap tracking, reads hardware file, calls application startup/shutdown routine	130
zstrutil.p	string utilities - copy, trim blanks, parse, change case, justify	294
zsyscalt.p	sets up Alt keys to invoke system configuration menu	29
zsyscfil.p	reads and writes hardware configuration file (HCF)	405
zsyscglb.p	global variable contains name/path of current hardware file	6
zsyscmen.p	menus for setting/changing hardware setup including screen colours, graphics device, etc.	2865
zsysglob.p	global variables describing low-level system status - beep on errors, entering data, in help, etc	7
zsyskbd.p	low level routine for reading/translating keyboard/mouse input	549
zsysscr.p	low level routine for reading/writing text/attribute data to/from text screen	82
ztabmenu.p	2D row and column menu support	569
ztdcons.p	standard constants for time/date values and formats	6
ztdent.p	enter time/date	266
ztdform.p	enter time/date format	326
ztdscr.p	write time/date data to text screen	91
ztdstr.p	convert time/date data tostring	136
ztdtype.pf	types with records describing time and date data	79
ztdutil.p	utilities for adding, subtracting, comparing times and dates	439
ztrigfun.p	trigonometric functions and constants - pi, arc cos, arc sin, arc tan, radians to degrees, degreres to radians	33
ztxtfile.p	utilities to open, close, read, and write to/from DOS text files	159
ztxtoglb.p	global variables to control/format text/screen listing output	6
ztxtoutl.p	utilities for text/screen listing output	196
zunitchg.p	toggles unit setting between imperial and metric - affects in/cm display/conversion for graphics setup only - GTFM uses more extensive unit conversions - see unitconv.p in Table A.4	55
zunitglb.p	global variable with current setting	9
zwindow.p	writes text windows, stores/restores text settings in underlying screens	262
zwinmgr.p	places windows on screen, writes window related data	551

Table A.2 GPLIB Source Files		
File Name	**Description**	**Size Lines**
convutil.p	generic support for reading/writing text file data and converting to/from data, typically for importing/exporting data with include file support, error trapping, status reporting	337
curvgen.p	generates functional approximation from XY data	469
curvint.p	performs functional approximation - given X, returns Y	212
curvmutl.p	menu utilities to support data entry, screen writing for standard curve data	181
curvtype.pf	types of curves available and data structures for storing curve descriptive data	18
escalglb.p	global variables	6
gp2draw.p	for 2D plots - performs coordinate conversion (user to local and vice-versa), checks points for within axes range, draws lines clipped to axes boundary, draws histogram bars	201
gp3calc.p	for 3D plots - calculates 2D screen coordinate for XYZ coordinate in 3D local space, calculates distance fromXYZ point to viewpoint, determines whether current view is within range that axes and labels are plotted	299
gp3conv.p	for 3D plots - converts from user XYZ to local XYZ, clips lines, checks points for within axes	336
gp3fptr.p	for 3D plots - manages memory allocation/deallocation for 3D facet vectors	73
gp3ftype.pf	for 3D plots - 3D facet vectors for drawing arbitrary 3D oblects described by XYZ points of triangles/rectangle vertices, structured to minimze storage when vertices are common.	28
gp3label.p	for 3D plots - sets plane and orientation for XYX labels, draws XYZ labels and UV labels	98
gp3pglob.p	global variables - current eye position	7
gp3poly.p	for 3D plots - general 3D hidden line remover. initialize, add data, process all lines and polygons, cleanup when done. Also special routines to handle 3D mesh type data	2566
gp3ptype.pf	low-level 3D polygon type description	14
gp3type.pf	general 3D types - XYZ point, planes, oriented label planes	14
gpaoglob.p	global variables - keeps status of all/one plot feature	5
gpautosc.p	performs axes autoscaling	452
gpaxes.p	draws 2D and 3D axes and annotation blocks	1411
gpcons.p	common constants used within GPLIB - memory saving holdover from previous MS Pascal implementation	10
gpcont.p	contouring routine - given data matrix, calculates list of contour points, edges and topology and successively extracts continous line segments	547
gpctglob.p	global variables - contains currrent plot setup - pens, cursor data, screen allocation, etc. Most set with Plot Setup Menu (Alt-P)	7
gpctmenu.p	menus for common plot setup. Implements Plot Setup Menu (Alt-P)	578
gpcursor.p	sets up and moves 2D and 3D cursors	770
gpdlglob.p	global variables for plot data listing facility	22
gpdllist.p	writes captured plot data to file	234
gpdlmenu.p	sets up for capturing plot data, specifies format and output file, usually invoked by F2 - List in XY plot mode	395
gpdrape.p	drapes lines over 3D surfaces - given data matrix, calculates Z values at XY coordinates	533

Table A.2 GPLIB Source Files		
File Name	**Description**	**Size Lines**
gpgrmenu.p	implements most generic graphics menus - axes, plot format, annotation, viewport, contour/level slice data, etc.	2642
gpgrstr.p	converts axes/data plotting/anno settings to strings for displaying in menus	198
gpgrtype.pf	generic XY and XYX plotting types - axes, viewports, annotation, data display formats, contouring level/pen	195
gpgutil.p	misc common user messages and plotting functions for XY/XYZ graphics	50
gpidglob.p	global variables - pointer to common plot ID string - in GTFM set to test ID	9
gpincs.p	automatic axes increments for linear and log axes	160
gplglob.p	global variables - used internally by GPLIB to store data associated with each individual plot	24
gpltype.pf	type describing internal axes data	51
gplutil.p	converts local/plot coordinates, zooms/pans/unzooms axes	704
gpmain.p	opens/closes GPLIB graphics, sets up number/types of plots	2717
gpplot.p	high-level routines to: start/stop plot drawing, draw XY data vectors, draw 2D/3D histograms, contours level-slice colour maps, 3D meshes, 3D drapes	3046
gprutil.p	variable initialization and allocation/deallocation foraxes, contour levels, data display, annotation, etc.	234
gpsglob.p	global variables with sizes of text and axes tics	28
gpview.p	single procedure easy access to GP capabilities for straightforward (no special keyboard handling/interaction) plotting.	90
intrfile.p	reads single column of X values from file for interpolating curves at specific points.	115
intrmenu.p	curve interpolation setup menu	587
intrtype.pf	types to support curve interpolation	26
intrutil.p	puls out X data vector given interp spec, allocates/deallocates inter records	209
miscscr.p	writes std prompt for data tables	33
miscstr.p	converts menu index number to specially formatted string (‘# nn’)	29
numview.p	views tables of numeriic data in spreadsheet type display	349
realcons.p	common real constants used within GTFM andGPLIB - memory saving holdover from previous MS Pascal implementation	10
realptr.p	initialize, allocate, and deallocate: XY data vectors, real matrixes	173
realtype.pf	types defining XY data vectors, real matrixes	17
rfmenu.p	supports menu entry of real formats	118
scalmenu.p	menu for entry of common XY scaling routines	665
scalscr.p	writes scale settings on screen in menu	37
scalstr.p	converts scale to string for use in data listings	60
scaltype.pf	defines operations and data for scale/transform procedures	49
scalutil.p	performs scale/transform on data	203
stdalts.p	support for stanard alternate key setup and use - ALT-P for plot, ALT-O for oytput, ALT-S for system config	132

Table A.2 GPLIB Source Files		
File Name	**Description**	**Size Lines**
zerocons.p	real constant for data record with 0.0 value - memory saving holdover from previous MS Pascal implementation	12

Table A.3 XYLIB Source Files		
File Name	**Description**	**Size Lines**
xydtedit.p	on screen graphical data edit - pick points to delete	390
xydteglb.p	global variables with symbols and pens for editing	8
xydtemen.p	menu to setup editing symbols/pens	133
xydtfile.p	reads XY data from file according to format (table, X col#, Y col #, file spec)	250
xydtfmen.p	menu for set-up prior to file read	236
xydtmenu.p	main menu for entry/edit of XY data	1411
xydtnois.p	add Gaussian/uniform noise to XY data	506
xydtplot.p	plot XY data, on-screen graphical data entry	529
xydtredu.p	menus and procedures for data reduction - skip points, max change	825
xydtsmoo.p	menus and procedures for data smoothing/Fourier filtering	955
xydttype.pf	expanded XY data records which contain menu/file/plotting status data as well as data vectors	58
xydtutil.p	initialize/allocate/deallocate XY data records	103
xydtview.p	numeric listing of XY data	115
xypdmenu.p	setup display and plotting parameters for XY data	264

Table A.4 GTFM Source Files		
File Name	**Description**	**Size Lines**
appglob.p	passes default width of units on screen back to GENLIB zappglob.pf	6
astrtend.p	contains GTFM specific startup/shutdown routines - initialize/allocates all variables	18
calcpar.p	calculates values for K, T, S, other calculated parameter for suites and constants	1306
calcplis.p	lists calculated parameters	528
calcunit.p	units for calculated parameters	127
capdata.p	extracts simulation data during run, performs specified cap data normalization/reduction	283
capdglob.p	global variables which stores captured data during each run	59
capdtype.pf	types to support display of captured data	12
caphorn.p	extracts simulation data during run, performs specified horner/superposition time processing	322
cappdev.p	extracts simulation data during run, performs specified derivative processing	307
capprof.p	extracts pressure/distance profile from last time step	89
carttrac.p	constants defining Carter-Tracey boundary condition, routines to extract P_i and dP_i.	561
chkmenu.p	pre-run check menu - accesses chk routines for sub-categories	541
chkotype.pf	type describing check to perform	7
cnfdat10.p	reads data from pre version 4.10 configuration files	662
cnfgrf10.p	reads graphics setup data from pre version 4.10 configuration files	484

Table A.4 GTFM Source Files		
File Name	**Description**	**Size Lines**
confdcap.p	variables for transforming data capture specifications from version 5.00 and earlier	8
confglob.p	global variables with current configuration file name and settings	8
confil11.p	reads data from version 4.10 (last MS Pascal version) configuration files	917
confilxx.p	reads data from version 4.20 and later configuration files, writes data in version 6.00 configuration file format	3685
confmenu.p	menu to set up configuration file name, data directory and read/write options	231
confmsut.p	utilities for reading data from MS-Pascal created files	146
confproc.p	opens configuration file, read/writes header, calls appropriate read/write routine	510
confutil.p	file open/closeroutines used by GTFM and CURVE	493
crvdmenu.p	selects single curvefrom MCV file	148
crvdtype.pf	string vector type for input to crvdmenu.p	15
crvfchk.p	checks current curve file setup	431
crvfgglb.p	global variables to store configuration of curve file plotting setup	25
crvfglob.p	global variables to store curve file setup	23
crvflist.p	creates listing of curve file setup	139
crvfload.p	reads CRV or MCV file and loads data for a single curve	350
crvfmenu.p	enter curve file setup	794
crvfplot.p	plot curve file data over check data and sequence start/end times	652
crvftype.pf	type for loading curve data from files	17
crvfutil.p	initializes/allocates/deallocates curve file setup globals	70
ctdebug.p	checks for conditions that may produce errors in Carter-Tracey BC	42
dcapchk.p	checks data capture setup	200
dcapglob.p	global variable with data capture setup	151
dcaplist.p	list data capture setup	127
dcapmenu.p	menu for setting up data capture	1069
dcaputil.p	utilities for extracting/using/selecting data capture setup	472
descglob.p	global variable - test description	6
descmenu.p	enter test description	157
desmenu.p	enter designation (sequence, crvf, check data, other), select sequence designation	239
desutil.p	gets sequence index number given designation	24
ebndchk.p	check external boundary setup	110
ebndglob.p	global variable with Carter-Tracey external boundary data	33
ebndlist.p	list external boundary setup	92
ebndmenu.p	menu for setting up external boundary setup	727
ebndmglb.p	global variable with Carter-Tracey plot setup	54
ebndtype.pf	enumerated type - zero_flow, fixed_pressure, carter_tracey	7
flagglob.p	global variables - all system configuration toggles (gas/liquid, single/dual, skin/no skin, etc)	9
gasglob.p	global variable - gas phase simulation control globals - mass/vol STP, standard temp and press vals, Klinkenburg K	9
gmgautos.p	autoscales defined plots	201

Table A.4 GTFM Source Files		
File Name	**Description**	**Size Lines**
gmgcalc.p	allocates memory for plot data, determines which data must be recalculated after each simulation	428
gmgcglob.p	global variable - indexes of plots being displayed	8
gmgchk.p	checks plot/fit data and plot setup	953
gmgdglob.p	global variable - vectors of pointers to plot data	24
gmgdutil.p	initialize/allocate/deallocate all plot/fit data	325
gmgglob.p	global variable and types for plot/fit data and plot setup	111
gmghomen.p	check and simulation horner setup menus	556
gmgmscr.p	write check data name or sim data name on screen	56
gmgmutil.p	select check or sim data from list	176
gmgpdmen.p	check and simulation derivative setup menus	856
gmgplot.p	open close GTFM plotting, use GPLIB to plot data on screen	582
gmgpmenu.p	menu to define each plot	1022
gmgsdmen.p	menu for setting up simulation data for fit/plot	453
gmgsmenu.p	run-time graphics setup menu	364
gmgtmenu.p	common 2D menus for setting up check data, deriv, horner, etc	1975
gmgtype.pf	enumerated types - check data, deriv, horner, etc	12
gofcalc.p	initialize/allocate/deallocate storage for fits and Hessian data, calculate fit value after each simulation	634
gofcglob.p	global variable with fit calculation results	38
goffchk.p	checks fit file (.GOF) setup	136
goffglob.p	global variable - name of fit file and file settings	6
gofflist.p	list fit file setup	66
goffmenu.p	menu for fit file setup	206
gofgdglb.p	global variable with fit result processing data	87
gofgfile.p	opens GOF file, load selected fit surface	281
gofgglob.p	global variable - setup for plotting fit results	77
gofglist.p	list one fit - sorted by parameter or by fit value	422
gofgplot.p	plot fit surfaces, contours	757
gofopchk.p	check fit or optimization setup	515
gofopglb.p	global variables - fit and time sequence definitions	28
gofoptme.p	main setup menu for fit/optimization specification	323
gofpost.p	fit post-processing menus	2475
gofsmenu.p	fit and time sequence setup menus	1769
gofsutil.p	initialize/allocate/deallocate fit/time sequence setup menus	97
gtfmanno.p	supports automatic annotation of plots	111
gtfmescl.p	sets formation pressure variable for use in extended scaling	53
gtfmfile.p	makes file name out of test ID, current data directory, and extension - checks if file exists	64
gtfmglob.p	global variable with test ID	9
gtfmoutl.p	writes test ID and listing date at the top of all output listings	58
gtfmtype.pf	strings and vector types specific to but widely used by GTFM	33
gtfmutil.p	initialize/sort GTFM vector types	94
horncalc.p	calculate horner time/pressure	121

Table A.4 GTFM Source Files		
File Name	**Description**	**Size Lines**
lhscglob.p	global variable with LHS correlation matrices, seed value, and flag settings	22
lhschk.p	check LHS setup	292
lhscons.p	constants to initialize LHS specs in parameters and sequences	6
lhscutil.p	manages correlation matrices , update to reflect sequence insert/delete	212
lhsdglob.p	global variable - all sample realizations for each sampled variable	7
lhsgglob.p	global variable - correlation and histogram plots	49
lhsgmenu.p	menus to set up correlation and histogram plots	1357
lhslist.p	lists sample setup (type, seed, correlations, n_trials)	142
lhsmenu.p	main sampling setup menu	595
lhsmutil.p	menus to enter variable distributions and distribution parameters	290
lhsplot.p	plots histograms and correlations	455
lhssubs.p	performs sampling	838
lhstype.pf	types describing distributions and distribution parameters	28
listglob.p	global variable - controls what items are listed from main listing menu	7
listlist.p	performs main listing	196
listmenu.p	menu to specify data to list	232
mainmenu.p	GTFM main menu and intro screens	1338
mainutil.p	routines called on startup, at end, and when selecting test IDs	461
ofilchk.p	checks text output files	231
ofildesc.p	generates description to put in header of text output files	204
ofildlis.p	lists text output file header	168
ofilglob.p	global variable - contains setup for text output files (full, profile, derivative, and horner)	104
ofillist.p	lists output file setup	282
ofilmenu.p	menu to setup text output files	1205
optcglob.p	global variable - optimization algorithm selection/control variables	7
optfchk.p	checks optimization file setup	136
optfglob.p	global variable - optimization file name, and file control variables	6
optflist.p	lists optimization file setup	66
optfmenu.p	menu to enter optimization file setup	231
optgglob.p	global variable - plot setup for optimization post-processing graphics	99
optplot.p	menu to setup plots of optimization results	4744
optpost.p	menu to setup optimization post-processing	729
optsel.p	select optimization from those in post-processing file	343
optvfile.p	reads optimization post-processing file into memory	478
optvglob.p	global variable - contains optimization post-processing data read from file	6
optvlist.p	lists optimization results	2106
optvplot.p	plot optimization results	2841
parcglob.p	global variable with parameter curve file data	6
parchk.p	checks parameter setup	537
parcutil.p	loads parameter curve data from MCV file, generates curve	73
pargglob.p	global variable - geometry plotting graphics	63
parglob.p	global variable - all parameter data and parameter groupings	354

Table A.4 GTFM Source Files		
File Name	**Description**	**Size Lines**
parlist.p	list parameter setup	345
parmenu.p	enter/setup parameters	2273
parplot.p	plots geometry graphics	1485
parutil.p	utilities for checking/setting parameter status	453
pdevcalc.p	calculated pressure derivatives of check and captured data	598
pdevtype.pf	derivative types and records for derivative parameters	32
postdesc.p	generates descriptes for normal post-processing runs	126
postfile.p	reads data from normal post processing file	286
postglob.p	global variable - normal post-processing setup and control	70
postgraf.p	plots normal post-processing data	328
postmenu.p	menu to setup for normal post-processing	935
prfchk.p	checks profile post-processing file setup	152
prfdglob.p	global variable - profile post-processing setup/control	69
prffile.p	reads profile data from profile post-processing file	250
prfgglob.p	global variable - data describing profile plots	75
prfgplot.p	plot profile graphics	1079
prflist.p	list profile file setup	79
prfmenu.p	setup for writing profile file	274
prfmglob.p	global variable - profile file name and control	6
prfpost.p	profile post-processing setup menu	3008
pstchk.p	checks normal postprocessing setup	162
pstlist.p	list normal postprocessing setup	95
pstmenu.p	menu to setup for normal post-processing	421
pstmglob.p	global variable - file name and control variables for normal post-processing	53
restchk.p	checks restart file setup	262
restfile.p	load/store restart records	510
restglob.p	global variable - restart file names and control	9
restlist.p	list restart file setup	90
restmenu.p	menu to setup for restart runs	286
scallist.p	lists scale records	163
seqchk.p	checks sequence setup	325
seqgglob.p	global variable - sequence graphics setup	20
seqglob.p	global variable - sequence data	21
seqlist.p	lists sequence setup	280
seqmenu.p	menu to setup test sequences	1622
seqplot.p	plots sequence times over check data	339
sequtil.p	initialize/allocate/deallocate sequence records	139
simaglob.p	global variable - atmospheric pressure for gas simulations	9
simbglob.p	global variable - calculated boundary constants	8
simbound.p	initialize/allocate/deallocate boundary variables, sets up wellbore boundary conditions, extracts correct f(t) values for each time step	385
simcglob.p	global variable - control variables - seq number, simulation number, simulation flag	7

Table A.4 GTFM Source Files		
File Name	**Description**	**Size Lines**
simct.p	Carter-Tracey boundary implementation	104
simdata.p	initialize/allocate/deallocate captured data variables, calculates capturedd data values at each time step	412
simdglob.p	global variable - captured data vectors	9
simerr.p	writes error message, jumps to sim abort address	49
simfglob.p	global variable - variables for storing GOF run results	8
simfile.p	writes text output files	702
simfit.p	initialize/allocate/deallocate fit run results variables, stores fit run results	147
simgglob.p	global variable - runtime graphics status and control	8
simgoff.p	writes fit post-processing file	290
simgraf.p	plots runtime graphics - updates after each time step	217
simguts.p	main numeric core of simulator - sets up and solves matrices	961
simjglob.p	global variable - jump variables for interrupting simulation	9
simkbd.p	monitors key strokes for abort/graphics update	103
simlglob.p	global variable - current LHS sample num being simulated	9
simmain.p	controls simulation	700
simnglob.p	global variable - numbers of specific simulation nodes in cureent sequence	9
simogglb.p	global variable - optimization graphics control globals	7
simoglob.p	global variable - optimization results and control globals	7
simograf.p	plots optimization graphics	555
simopt.p	L-M and Simplex optimizer	1303
simoptf.p	writes optimization results to OPT post-processing file	510
simparam.p	extracts parameters, sets up geometry, calculates f(P) param values, calulates matrix s and d terms	1492
simpglob.p	global variable - variables to hold profile data	8
simpostf.p	writes normal run results to PST post-processing file	409
simprff.p	writes profile data to PRF post-processing file	336
simprof.p	captures profile data in variables	234
simrglob.p	global variable - runtime globals for matrices and control	8
simrset.p	initialize/allocate/deallocate simulation runtime variables	255
simscr.p	writes simulation data on text screen	941
simtglob.p	global variable - test zone variables - temperature, volume, etc	6
simtstep.p	sets up time step data, restarts after dynamic TS change	302
soptmenu.p	system configuration menu	952
statglob.p	global variable - status of various groups of variables	8
supemenu.p	menus for superposition time	102
supertim.p	calculates superposition time	258
supetype.pf	superposition time types	14
testent.p	enter/select test ID	631
tsumutil.p	utilities for listing data and variables	385
tsumview.p	driver for viewing all listings	246
unitconv.p	converts from user units to base units and vice-versa	42
unitent.p	enter real data with units	205

Table A.4 GTFM Source Files		
File Name	**Description**	**Size Lines**
unitglob.p	global variable - unit conversion factors, default formats, and text	219
unitmenu.p	menus to select units for values	112
unitscal.p	converts unit conversion to scale record	46
unitscr.p	writes real data on screen with correct units	95
unitstr.p	converts real data to string with correct units	71
unittype.pf	different types of unit dimensionality	18
verglob.pf	global variable - current version number and release data	7

Appendix B - Review Comments