nSIGHTS Version 2.41a

Design Document

WIPP PA

Document Version 2.41a

ERMS #555650

TABLE OF CONTENTS

List of Tables

1.0 INTRODUCTION

The n-dimensional Statistical Graphical Hydraulic Test Simulator (nSIGHTS) is an analysis code used to interpret hydrogeologic well tests. The code was designed and implemented for the windows platform (Windows NT, 2000, XP, 7) and contains the functionality of the DOS based well-test simulator graph theoretic field model (GTFM) Version 6.20. References to earlier GTFM QA documents are included in Section 7.0.

1.1 Purpose

This Design Document (DD) describes the design constraints, attributes, and theoretical development of nSights. In essence, it summarizes the functionality of nSights 2.41a and all enhancements from nSIGHTS version 1.0. The code's design is in conformance with the recommendations of the Waste Isolation Pilot Plant (WIPP) Nuclear Waste Management (NWMP) Quality Assurance (QA) procedure (NP) 19-1 Revision 12, *Software Requirements.*

1.2 Scope

This DD is for the development of nSIGHTS, Version 2.41a. An nSIGHTS analysis will provide parameter estimates to support WIPP Performance Assessment (PA). As such, the code must be documented to fulfill requirements of the Nuclear Waste Management Program Quality Assurance Procedure NP 19-1 Revision 12.

nSIGHTS has statistical routines that allow the uncertainty in the estimates of fitting parameters to be quantified. The goal for using nSIGHTS is to quantify the uncertainty in well test data before passing the values to PA for calculating transmissivity fields.

1.3 WIPP QA Software Identification

nSIGHTS version 2.41a / WIPP Code Prefix: nSIGHTS.

1.4 Points of Contact

Lead: Kevin Barnhart
Sandia National Laboratories
575/234-0006
ksbarnh@sandia.gov

Development Team: John Avis
Intera
613/232-2525
javis@intera.com

Technical Reviewer: Dale Bowman
Sandia National Laboratories
575/234-0055
dachace@sandia.gov

1.5 Acronyms

Acronyms of terms that relate to nSIGHTS are defined the first time they are used in the text, as is customary.

1.6 References

Section 7 contains a list of references.

2.0 GENERAL DESCRIPTION

This document describes the design and architecture of nSIGHTS V2.41a (**n** dimensional **S**tatistical **I**nverse **G**raphical **H**ydraulic **T**est **S**imulator). nSIGHTS will be implemented under the governance of NP19-1 Software Quality Assurance procedure. In essence, nSIGHTS contains the functionality of the DOS based well-test simulator GTFM Version 6.20, re-designed and re-implemented for the Windows platform (Windows 95, 98, NT, 2000, XP, and 7).

Functional requirements for nSIGHTS V2.41a are described herein. The theoretical and the embodied mathematical basis for the code is described in Appendix A of this document "NSIGHTS Version 2.41a – Functional Description and Theoretical Development". Appendix A is provided as a separate file because of concerns with managing a single electronic document containing a large number of specially formatted equations.

Sections of this document are:
Section 1 – Introduction to nSIGHTS and layout of the design document.
Section 2 –General Description of nSIGHTS.
Section 3 Basic Architecture subdivision of functionality into major components, basic design of each component.
Section 4 Data and Code Structures definition of major data structures and classes to be used.
Section 5 Menu Specification design of menus for nSIGHTS pre-processor component.
Section 6 Parameter Limits numeric ranges for parameter values
Section 7 References
Appendix A nSIGHTS Theory

The following table maps sections in this document to Design Document Criteria checklist items (NP 19-1, Rev 12, page 22):

Table 1.1 Document/Criteria Mapping

NP 19-1-4 Checklist	Section in this Document
Major Software Components	Section 3 Architecture
Functional Design	Companion document, Appendix A; NSIGHTS version 2.41a Functional Description and Theoretical Development
Theoretical Development	Companion document, Appendix A; NSIGHTS version 2.41a Functional Description and Theoretical Development
Major Control Flow	Section 3 Architecture
Control Logic	Section 3 Architecture
Data Structures	Section 4 Data Structures
Allowed or Prescribed Ranges	Section 6 Parameter Limits
Verifiability	See NSIGHTS version 2.41a – Validation Document.

3.0 BASIC ARCHITECTURE

nSIGHTS consists of two logically distinct entities:

1) The pre-processor (nPre) is used to enter parameters, to prepare for, and to perform simulations.

2) The post-processor (nPost) performs graphical and arithmetic post-processing of nPre output files, and performs special pre-processing of external data files for use by nPre.

In general, an object-oriented event driven data-flow architecture controls execution and movement of data between computational entities used in nSIGHTS. Each entity performs a specific data processing or visualization task. For the remainder of this document, the term “functional object” refers to these individual entities. In implementation terms, the objects are C++ classes derived from the base class FuncObjC (see section 4.2.3).

Functional objects are used as wrappers for most data classes. OpenGL 2D and 3D graphics code forms the basis of all data visualization capabilities. nSIGHTS is written in C++. Microsoft Visual C++, Version 7 is the development environment. Also, nSIGHTS relies extensively on the data structures and algorithms developed for GTFM 6.20. These were ported from their implementation language (Pascal) to C++.

Throughout the remainder of this document, reference will be made to “platform independent” and “platform dependent” code. In this document, these terms refer to individual source code components, rather than a complete application.

In this context, “platform independent” refers to code that conforms to the C++ standard and does not use any operating system dependent features or libraries. Platform independent code is thus code that would compile on multiple hardware platforms using any C++ standard compliant compiler. For example, some nSights source code was originally developed on Unix workstations (Alpha, HP, SGI, and Sun) using a variety of compilers. This same code compiles unchanged on Windows platforms using the Microsoft Visual C++ compiler. None of this code would require changes if a Unix version of nSights were to be developed.

In contrast, “platform dependent” refers to source code that uses operating system dependent features. For example, the nSights user-interface components use the Microsoft Foundation Classes (MFC) for Windows. A Unix based implementation of nSights would require converting this code to an equivalent X Windows library, or a platform neutral UI library such as Qt.

To as great an extent as possible, the nSights design has been partitioned to maximize the use of platform independent code.

3.1 nPre Design

nPre is a single-document Windows application. This is significant from a design point of view as it allowed for more straightforward porting of existing GTFM data structures and global variables.

3.1.1 User Interface

nSIGHTS works with a wide variety of input data. Examples include single-valued parameters, functional (i.e f(P), f(t), f(r)) parameters, control flags (e.g. single or dual-porosity, gas or liquid flow), well bore boundary condition specification, numeric parameters, sampling distributions, optimization ranges, etc. As a result, the user-interface for nPre exhibits a significant level of complexity.

The main user interface in the document area has nested tabbed controls. Each tab corresponds to a single category of input. Tabs may contain a nested tab control for sub-categories of input. Tabs or nested tabs may contain grid, tree, or dialog type menus for the entry of data.

nPre does not follow standard Window UI conventions in that nPre supports multiple top-level windows for graphics and text/HTML output.

The user-interface code is separated from the data structures containing the simulator data.

3.1.2 Simulator

Classes containing simulator input data were translated from the GTFM Pascal source.

The optimizer/sampler code was substantially redesigned and separated logically from the actual simulator. A C++ base class "C_Optimizable" (see section 4.4.1) was designed as the target for the optimizer. The actual nSights simulator is packaged in a class derived from C_Optimizable.

The simulator was developed in a "thread-safe" manner to support implementation on multi-processor architectures. Multiple instances of the simulator may run simultaneously.

3.2 nPost Design

nPost provides a basic UI which the user may populate with a variety of functional objects to support data processing and visualization. The nPost user-interface uses a "Windows Explorer" type approach. A tree control is used with first-level nodes (leaves) representing data pages (for data-processing objects) and 2D and 3D plots. Second-level nodes represent specific data or plot objects. The area to the right of the tree structure contains the UI for the currently selected second level node.

3.3 nPreX Design

The basic architecture of nPreX is the same as nPre, without user interface components. The intent of nPreX is to run a simulation based on a configuration file generated with nPre. There is a small amount of new code to call the existing routines that read the configuration file and execute its contents. This code uses standard MPI (Message Passing Interface) to allow the simulations to be spawned to multiple processors. The user's machine, or a cluster of machines, must have mpi libraries installed to execute nPreX. nPreX was developed and tested using mpich.

nPreX uses the same source code base as nPre, minus those components which are strictly UI or plotting related (e.g. genPPD, genPFO, genPPO). A compile time flag, ADCONSOLEAPP, is defined for nPreX compilation. C++ preprocessor directives (#ifdef, #endif, #elseif) are used to exclude UI code, or (rarely) to include code specific to nPreX.

4.0 DATA AND CODE STRUCTURES

4.1 Data and Code Structures

nSights has been developed as two main application programs (nPre and nPost) that are both dependent upon a series of structured libraries. Library names and descriptions are:

1) **GenLib** This library contains a mixture of platform independent and platform dependent code that provides generic support for graphics intensive applications.

2) **ObjLib** This library adds defining classes and functional objects for processing and plotting curve, grid, cube, time series (XY), and tabular data.

3) **OsLib** This library adds classes and functional objects to support optimisation and sampling in scientific applications.

4) **NsLib** This library adds nSights specific classes and functional objects primarily related to well-test analysis.

The following subsections address each library in more detail, followed by descriptions of the nPre and nPost application structure.

4.2 GenLib Structure

GenLib provides framework support and components that are independent of any specific application. To as great an extent as possible, GenLib source is also platform independent, in the sense that the code will compile unchanged on a variety of operating systems and hardware. GenLib source code is structured in subdirectories that reflect the differentiation of functionality. Each GenLib subdirectory is addressed below.

4.2.1 genAlloc

Allocator objects to implement functional object factory code. If linked into an application the application can create the associated objects. Code in this directory is not physically included in the library itself but is conceptually part of the library. Each object allocator is derived from a template defined in C_AllocObj in genApp.

4.2.2 genApp

Code which defines and implements the basic framework user-interface (UI) and provides OS platform specific implementation of common facilities such as copy/paste and configuration file I/O. Major classes and functionality are as follows:

AppConfigFile	basic support for object serialization to/from text files or clipboard. Provides object header support and general data type I/O.
AppCopyPaste	uses object serialization to/from the Windows clipboard to provide copy, paste, and duplicate support for individual or groups of objects and pages.
C_AllocObj	object factory for creating DPO_XXX, LPO_XXX, PPO_XXX, and UPO_XXX functional objects given an object identifier.

C_AllocPage	object factory for creating new menu pages
C_AppMenu	support for Object and Page pulldown and popup menus in applications.
C_CfgFileObj	abstract base class for all serializable objects. Used by multiple inheritance.
C_ExposedObj	base class for exposing selected properties in a modeless dialogs.
C_MenuBase	base class for all functional object user-interfaces (DPO_XXXPS, LPO_XXXPS, PPO_XXXPS, UPO_XXXPS). Handles menu creation, UI tasks common to all FuncObjC derived objects (object ID, object buttons), and object calculation initiation. Defines virtual functions for button handling.
C_MenuObj	base class for all functional objects with associated UI menus (DPO_XXX, LPO_XXX, PPO_XXX, UPO_XXX). Derived from TreeNode and CfgFileObj.
C_MenuPage	collects MenuObjC objects on a page in the object tree UI. Handles child pages for drag/drop and object re-arrangement. Derived from TreeNode and CfgFileObj.
C_DataPage	page type for data objects and child data pages
C_PlotPage	page type for plot objects and child plot folder pages
C_CompositePage	page type for composite plots
C_ListPage	for list objects
C_PlotFolderPages	page type which is child of plot page and contains plot objects only.
C_MenuRoot	collects MenuPageC objects on a single tree. Derived from CfgFileObj.
C_ObjHelp	maps help ID numbers to object names for use by Help system.
C_TreeBase	provides UI support for object tree operations. Includes drag/drop for nested pages and for object re-arrangement.
C_TreeNode	provides UI support for a single node (MenuObj or MenuPage) on a tree. Displays icon and object description.

ExposedList	implements exposed object support for list selection type data. Derived from ExposedObjC.
ExposedReal	implements exposed object support for real valued data. Derived from ExposedObjC.
G_Version	defines version globals for application (ID, date, major/minor version number). Actual implementation is in application.
ProjectFrame	basic MFC application class
SingleFrame	ProjectFrame derived class for applications with a single menu tree (like nPost)
ProjectUtil	sets/gets registry data describing application settings and properties.
U_Menu	basic UI support classes used by MenuBaseC derived functional objects.
U_MenuDPO	basic UI support classes used primarily by DPO_XXXPS functional objects.
U_MenuPPO	basic UI support classes used primarily by PPO_XXXPS functional objects.
WindowSelector	implements WindowSelector UI control.

4.2.3 genClass

Platform independent and application independent base classes and support classes. Major classes and categories include:

C_FuncObjC	the fundamental framework class. It is the base class for all processing/plotting/listing functional objects. FuncObjC manages all inter-object communication and creates the object execution tree. Virtual functions defined in FuncObjC are implemented in derived classes to provide actual object functionality.
C_GlobalFunc	FuncObjC derived class for functional objects implemented at the application level with no user interface or file I/O.
C_InteractiveObj	FuncObjC derivation adds virtual method for callback processing.
DC_XXX	data classes that define the basic types of data used within the framework.

DC_ColorMap array of colors

DC_ContourSpec contour values, colors, and line type

DC_DataLimit specification for mapping real values to integer intervals

DC_PenSet defines pen set consisting of 24 separate colors

DC_TableData tablular data.

DC_XYData time series (XY) data

C_DataObj base class for wrapper objects used to communicate data classes between FuncObjC derived objects.

DO_XXX lightweight classes derived from DataObjC to wrap data classes DC_XXX.

IO_XXX reads/writes data class from/to files.

IO_XYData reads/writes DC_XYData in a variety of formats.

SC_Array base class for all array classes.

T_SC_Array<type> derived from SC_Array. Provides a template for implementing a lightweight vector class. Examples:

SC_IntArray derived from T_SC_Array<int>

SC_DoubleArray derived from T_SC_Array<double>

SC_DoubleMatrix derived from T_SC_Array<SC_DoubleArray>

T_SC_BasicPtrArray<type> derived from T_SC_Array to provide basic facilities for operating on arrays of pointers to type. Does not handle type construction/destruction.

T_SC_AllocPtrArray<type> derived from T_SC_Array to provide basic facilities for operating on arrays of allocated pointers to type. Does handle type construction/destruction.

SC_XXX other generic support classes. Examples include:

SC_ColorSpec defines color by RGB or HSV components

SC_RealConversion provides support for converting real numbers to string representations.

SC_SetupErr common error detection and processing support

SC_Statistics	performs univariate statistics calculations.
SC_Triangulation	triangulates scattered or gridded XY data for contouring.

U_XXX	utility procedures. No class specifications.
U_Msg	generic messages to the UI. Definition only. Implementation is in genApp/U_MsgPS
U_Real	real number utilities (matrix solvers, statistical calculations, etc).
U_String	string processing utilities.
GenLib	also provides error trapped file I/O routines:
C_BufFile	reads/writes binary formatted files.
C_Text	base class for reading/writing text formatted data (file or clipboard).
C_TxtFile	derived from C_Text for file I/O.
C_ConvFile	special support for reading and converting ASCII files produced by numeric models.

Collections of small classes:

C_Common	contains basic 2D (Point2D) and 3D (Coord3D) classes used for plotting.
C_Graphics	defines line types and symbol specifications.

4.2.4 genClassPS

Platform specific but application independent code.

PS_ArgList	encapsulates arguments to UI creation routines primarily used for defining UI component layout. Based on original mView Unix implementation.
PS_UIBase	individual UI elements. MFC implementation of original Unix X wrappers. Defines BaseWidget and derived classes for all common UI controls. All layout automatically managed by X Form like attachments.
PS_UIClass	adds labels or frames to individual PS_UIBase controls.
PS_OGLBase	base class for OpenGL visual. Enumerates available visual and selects. Provides information about current visual properties.

PS_Color	maps SC_ColorSpec to Windows API COLORREF
PS_DialogShell	base class for all modeless dialog windows.
PS_FormContainer	basic abstraction of Unix X Form widget used for MFC implementation.
PS_Import	base class for reading data from the Windows clipboard.
PS_MainWindow	keeps track of all top-level windows for use by window selector.
PS_TxtClipboard	derived from TextC. Encapsulates/implements clipboard I/O.
PS_UIGrid	implements grid as a BaseWidget derived object.
PS_WriteXXX	writes plot bitmaps in JPG, TGA, or BMP format.

4.2.5 genCtrl

Windows specific code for implementing basic controls with callbacks on resource defined dialog forms.

UpdateableCtrl	defines callback processing for dialog
CtrlUpdate	abstract class to define callback mechanism for controls
ButtonCtrl	UI button
CheckBoxCtrl	UI check box
ComboBoolCtrl	dropdown for two (true/false) entries
ComboIntCtrl	dropdown with multiple entries mapped to integers
ComboFOCtrl	dropdown with list of defined FuncObjC for object connection
FileCtrl	text field and button for browse selection
IntCtrl	enter integer value
RealCtrl	enter real numeric value
TextCtrl	enter text string
TextDialogCtrl	displays text string, calls up dialog when selected

4.2.6 genDataClass

Defines common components used in FuncObjC derived functional objects.

DSC_IndexMSSpecBase	master/slave connected to list selection
DSC_RealMSSpecBase	master/slave connected to real value entry
DSC_ScaleTransform	common arithmetic and transform operations on real values
DSC_Threshold	thresholding values
DSC_TimeBase	common time unit conversion

4.2.7 genGrid

Windows specific code for implementing basic spreadsheet type data entry/display.

GridCtrlBase	overall definition/control of sheet
GridCell	base class for individual cells
CheckEditWnd	edit control for check box
CheckEditCell	grid cell containing CheckEditWnd
ComboEditWnd	edit control for combo box
ComboEditCell	grid cell containing ComboEditWnd
TextEditWnd	edit control for text entry
TextEditCell	grid cell containing TextEditWnd

4.2.8 genHelp

Help system mappings for genLib functional objects.

4.2.9 genInterface

Windows specific code for implementing Outlook style application with dialogs and object trees.

OutlookFrame	ProjectFrame derived class for applications with a multiple menu trees (like nPre)

4.2.10 genListClass

Platform independent base classes to allow creation of listings in HTML browser windows.

C_ListDef	basic definition of a listing window. Maintains lists of connected listing functional objects.
C_ListObj	base class for objects that list data to ListDefC functional objects.
C_ErrorListObj	performs listing of error conditions.
C_ListFactory	abstract class for creating modal listings.
LFO_ObjectListing	creates listing of defined FuncObjC connections.

4.2.11 genListClassPS

Platform specific implementation of genListClass codes adds Windows specific code for window creation and use of IE HTML renderer.

C_ListDefPS	creates top-level window with IE browser in client area.

4.2.12 genPlotClass

Platform independent base classes for plot definitions and plot functional objects.

C_PlotDef	basic definition of a plot window. Maintains lists of connected plot and annotation objects. Controls overall plot display. Handles default mouse right click menus.
C_PlotObjBase	InteractiveObjC derived class which links an object to be plotted to the plot it is defined on. Explicitly connects object to base plot and pen set.
C_PlotObj	derived from PlotObjBaseC. Adds plot layering and reporting support for 2D objects and polygon offset support for 3D objects.
C_ActiveObj	PlotObjC derived object that responds to mouse input.
C_AnnoObj	PlotObjC derived object that is plotted in 2D annotation coordinate system.
PC_2DPlotAnno	defines axes/increment layout and annotation for 2D plots
PC_3DAxesFormat	defines axes/increment layout for 3D plots

PC_3DAxesLayout	defines axes/increment annotation for 3D plots
PC_Axes	basic axes definition and derived class for Lin/Log axes.
PC_CallbackMenu	for defining/processing right click menus in ActiveObjC derived objects.
PC_Lighting	defines OpenGL lights for 3D plots.
PC_Platform	defines platform specific cababilities.
PC_Report	adds class to support cursor reporting on 2D plot objects.
PC_View	defines axes limits
PD_XX	derived from PlotDefC – defines plot types
PD_2D	defines basic 2D plot
PD_2DXY	derived from PD_2D – defines 2D plot with lin/log axes
PD_3D	defines basic 3D plot
PD_3DXYZ	derived from PD_3D – defines 3D plot with lin/log axes
PD_Composite	defines basic composite plot – includes list of sub-plots

PSC_XXX support classes for defining common plot elements.

PSC_AnnoSpec	elements common to most annotation objects (legend frame, pen, font)
PSC_ArrowSpec	arrow and arrow head
PSC_ColorBase	maps data to color ranges
PSC_ContourSpec	define contouring of data
PSC_ColorLegendSpec	how data is mapped to colors
PSC_ExtrusionSpec	how extrusion is plotted on 3D plot
PSC_Font	defines font characteristics
PSC_GridLine	single grid line and label
PSC_LabelSpec	basic definition for label

4.2.13 genPlotClassPS

Platform specific implementation of genPlotClass codes adds Windows and OpenGL specific code. Includes operating system independent implementation of 2D and 3D OpenGL functionality (platform specific in this case refers to the OpenGL renderer).

OGL_Base basic OpenGL plotting space

OGL_2DBase plotting space for 2D plots

OGL_2DXY – adds coordinate transforms for log axes.
OGL_3DBase plotting space for 3D plots
OGL_3DXYZ – adds coordinate transforms for log axes.

OGL_Font support for bitmapped and polygon fonts using wgl API routines

OGL_StringProcessing does text processing (e.g. for subscripts/superscripts) and plots results

PC_PlatformPS – Windows specific implementation of PC_Platform. Creates top level window, manages mouse, menu bar, status bar and tool bar.

PC_Platform2DPS adds capabilities for 2D plotting cursor reporting area.

PC_Platform3DPS adds capabilities for 3D plotting only view control sliders and view animation dialog.

PC_ReportPS implements reporting window management

PD_2DXYPS merges OGL_2DXY with PD_2DXY. Implements axes drawing routines.

PD_3DXYZPS merges OGL_3DXYZ with PD_3DXYZ. Implements axes drawing routines.

PD_CompositePS merges OGL_Base with PD_Composite

PS_ObjectDialog implements Object Control dialog.

PS_PSObjectSetup implements PostScript Output dialog.

PS_TGAObjectSetup implements Bitmap Output dialog.

PS_ViewAnimationDialog implements 3D View Animation dialog.

PSC_FontPS maps PSC_Font structure to MFC CFont

4.2.14 genUIExt

Windows specific code for MFC user-interface extension or enhancement classes.

4.2.15 genUnits

Platform independent and Windows specific (derived from genCtrl and genGrid) classes for performing unit conversions and for entering/displaying dimensioned numeric data in dialogs and grids.

C_GenUnits	unit conversion and basic units. Additional units can be added by application.
C_UnitListObj	adds unit conversion to ListObjC
ExposedUnitReal	adds unit conversion to ExposedRealC
U_Units	unit controls (unit selection combined real/units) for use in form managed UI
UnitConfigFile	unit value I/O
UnitGridCell	unit control for grid
UnitIndexCtrl	unit selection for resource based dialogs
UnitRealCtrl	combined value entry/unit selection for resource based dialogs

4.2.16 genDFO

Basic functional objects (all derived from FuncObjC) used by all applications, primarily concerned with defining/operating on pen sets and colour maps:

DFO_PenSet	defines colors for up to 24 pens used by a plot.
DFO_LinColorMap	defines a basic color map.
DFO_BlendColorMap	combines two color maps as a linear combination.
DFO_MergeColorMap	combines two color maps sequentially.
DFO_ReadColorMap	reads a color map from a file.
DFO_EnterColorMap	define individual colors in a color map.

Also includes base classes for common operations with different application specific implementations:

DFO_Histogram	calculates a histogram for an input vector.
DFO_InterpVal	interpolates a Y value from XY input given an X value. Various options are available (linear, log, nearest, etc).
DFO_MathBase	for common mathematical operations (+-/*) on two input vectors.
DFO_RangeBase	for extracting ranges of values from an input vector.

DFO_ScaleTransform	performs basic scaling/transform operations on an input vector.
DFO_Statistics	performs univariate statistics on an input vector.

4.2.17 genDPO

File I/O and user interface for all genDFO functional objects. All code is platform independent but it relies upon platform dependent code in genApp and genClassPS.

DPO_XXXX	is file/IO for DFO_XXXX objects
DPO_XXXXPS	is the user interface for DFO_XXXX

4.2.18 genPFO

Basic plotting objects used by all applications, primarily concerned with plot annotation:

PFO_ColorLegend	legend bar for objects plotted according to a color map.
PFO_DataLabels	labels that are linked to values produced by other objects.
PFO_GridLine	adds a labeled grid line at a specific location on an XY plot.
PFO_SeriesLegend	creates legend box showing line types and symbols with legend.
PFO_UserLabels	basic user entered text for labels.

Also defines a base class used by all objects that map data values to colors in a color map.

4.2.19 genPFOGL

Implements renderer dependent code to actually draw genPFO functional objects on a plot. Is platform independent but relies on platform dependent code in genPlotClassPS.

PFO_XXXXGL	the drawing code for PFO_XXXX.

Also includes support classes for OpenGL drawing of common objects such as lines, symbols, polygons, arrows and 3D extrusions. These are renderer dependent.

OGLBaseObj	included multiple inheritance on all PFO_XXXGL objects to link object to renderer and perform common tasks.
OGLObj	classes for issuing OpenGL renderer commands for different drawing operations.

GL_Arrow draws arrow defined by PSC_ArrowSpec.

GL_ColorLegendSpec draws color legend box defined by PSC_ColorLegendSpec.

PC_GridLine support class to draw grid lines.

PC_Label support class to draw a 2D label.

PC_Rectangle defines rectangle for legend box frames etc.

4.2.20 genPPO

File I/O and user interface for all genPFOGL functional objects.

PPO_XXX file/IO for PFO_XXXGL objects
PPO_XXXPS the UI for PFO_XXXGL

4.2.21 genPPD

File I/O and user-interface for basic plot definitions (i.e. plot size, axes types/limits, axes labels, increment labels, lighting for 3D plots)

PPD_Base components common to all plots (window size)

PPD_2D derived from PPD_Base. Adds components for all 2D plots

PPD_2DXY derived from PPD_2D. Adds components for 2DXY plots

PPD_3D derived from PPD_Base. Adds components for all 3D plots

PPD_3DXYZ derived from PPD_3D. Adds components for 3DXYZ plots

PPD_Composite adds components for composite plots.

PPD_CompositeLayout describes layout of sub-plots on composite plot.

PPD_LinLogAxis a single axis for an 2DXY or 3DXYZ plot

PPD_2DPlotAnno axes and increment annotation and formatting for a 2D plot

PPD_3DAxesLabel axes and increment labeling for 3D plot

PPD_3DAxesFormat increment/grid line formatting for 3D plots

PPD_3DLighting lighting specification for 3D plots

4.2.22 genUFO

Basic utility functional objects used by all applications:

UFO_WriteColorMap	writes a color map to a file

4.2.23 genUPO

File I/O and user-interface for genUFO

4.3 ObjLib

ObjLib provides functional objects and classes to support operations and plotting of common data types. ObjLib source code is structured in sub-directories similarly to GenLib.

4.3.1 objClass

Data classes, data object wrappers, and IO:

DC_Curve	cubic spline or polynomial fit to time series.
DC_GridData	data defined over regular 2D grid.
DC_CubeData	data defined over regular 3D grid.
DC_XYDataArray	array of time series.
DO_XXX	wrappers for above.
IO_GridData	I/O for DC_GridData.
IO_CubeData	I/O for DC_CubeData.

4.3.2 objDFO

Functional objects for operating on curves, cubes, grids, tables and XY (time-series) data.

Creation

DFO_CreateDataLabel	creates DO_Label classes for plotting with PFO_DataLabels.
DFO_CreateReal	creates real value for use by other functional objects.
DFO_CreateXYArray	combines XY data into DC_XYDataArray.

Cube Operations

DFO_CubeExtractGrid	extracts plane of data from cube.
DFO_CubeHistogram	histogram of data in cube.
DFO_CubeMath	add/subtract/multiply/divide contents of two cubes.
DFO_CubeNormalize	apply normalization operation to all data in cube.
DFO_CubeScaleTransform	scale/transform all data in a cube.
DFO_CubeStatistics	univariate statistics of data in cube.

Curve Operations

DFO_CreateCurve	creates a curve specification based on XY input data.
DFO_CurveInterp	interpolates data from curve.

Data Entry

DFO_EnterTable	user entry of table data in spreadsheet format. Also allows editing/viewing of existing table.
DFO_EnterXY	user entry of XY data in spreadsheet format. Also allows editing/viewing of existing data.

Grid Operations

DFO_GridExtractXY	extracts line of data from grid.
DFO_GridHistogram	histogram of data in grid.
DFO_GridMath	add/subtract/multiply/divide contents of two grids.
DFO_GridNormalize	apply normalization operation to all data in grid.
DFO_GridScaleTransform	scale/transform all data in a grid.
DFO_GridStatistics	univariate statistics of data in grid .

Read Input

DFO_ReadCubeData	reads cube data from ASCII file.
DFO_ReadCurveArray	reads curve data from ASCII file.
DFO_ReadGridData	reads grid data from ASCII file in several formats.
DFO_ReadLabelArray	reads label data for plotting.

DFO_ReadTable reads table data from ASCII file in several formats.

DFO_ReadXY reads XY data from ASCII file in several formats.

Table Operations

DFO_CombineTable creates a new table containing selected columns of input tables. All columns must be the same length.

DFO_TableAdd adds individual entries of one or more identically sized input tables to create an output table of the same size.

DFO_TableColScaleTransform creates an output table that is the same as the input table except a scale/transform operation has been applied to the data in a single column.

DFO_TableConcat Concatenates tables together.

DFO_TableHistogram creates a histogram based on a single input table column.

DFO_TableInterpVal interpolates Y value given X value and two selected table columns representing X and Y vectors.

DFO_TableRangeExtract extract rows from a table based upon values in a single column.

DFO_TableRowIndexLogic supports combining extraction operations from several tables.

DFO_TableRowStatistics calculates statistics for each row in a table. Produces a new table containing columns with each statistical measure selected.

DFO_TableStatistics calculates basic univariate statistics for a single column in a table.

DFO_TableToReal extracts a single value from a selected table column based upon a criteria (min, max, first, last, specific row).

DFO_TableToXY converts two columns in a table to an XY (time series) data type.

DFO_TableTranspose Switches a table's columns and rows.

XY (Time Series) Operations

DFO_XYAddNoise	adds normally or uniformly distributed noise to the Y values in an XY data set.
DFO_XYArrayScaleTransform	provides basic scale/transform operations for all data in an XY array.
DFO_XYDualScaleTransform and	provides basic scale/transform operations for both X Y components.
DFO_XYFourier	does inverse and forward FFTs on Y component of XY data set.
DFO_XYHistogram	creates a histogram based on X or Y values.
DFO_XYIntegrate	output XY is basic point by point integration of input XY.
DFO_XYMath	basic math (+-/*) on X or Y components of two input XY.
DFO_XYRangeExtract	reduces XY data based on range of values of either X or Y components.
DFO_XYReduction	options to reduce the X density of an input data set using a variety of methods.
DFO_XYRemoveDuplicate	removes duplicate X and Y values from a data set.
DFO_XYScaleTransform	provides extended scale/transform operations for either X or Y components.
DFO_XYSmoothFilter	applies smoothing and FFT based filtering operations to a data set.
DFO_XYStatistics	calculates univariate statistics for X or Y data.
DFO_XYToXYArray	adds successive XY data to an XY Array.
DFO_XYTranspose	swaps X and Y columns.

4.3.3 objDPO

I/O and user interface for all objDFO functional objects

4.3.4 objPFO

Plotting functional objects for cube, grid, tables and XY data.

PFO_2DAnalytic	plots a line that can be controlled by the mouse to measure slopes and distances.
PFO_2DMultTables	plots selected column pairs from one or more tables using symbols and/or lines on a 2D XY plot.
PFO_2DTableSeries	plots one or more selected columns against a single X column from a single table using symbols and/or lines on a 2D XY plot.
PFO_2DXYData	plots data from one or more XY data sets using symbols and/or lines on a 2D XY plot.
PFO_3DTableSeries	plots three selected columns from a single table using symbols and/or lines on a 3D XYZ plot.
PFO_3DXYZData	plots data from one or more XY data sets using symbols and/or lines on selected plane/value on a 3D XYZ plot.
PFO_CubeColorBlock	plots selected points in cubes as solid blocks on a 3D plot.
PFO_CubeColorPoints	plots selected points in cubes as symbols on a 3D plot.
PFO_EnterXYOnPlot	allows entry/editing of XY data with the mouse.
PFO_GridColorBlockFill	plots grids as solid colors on a 2D or 3D plot.
PFO_GridColorPointFill	plots grids as symbols on a 2D or 3D plot.
PFO_GridColorRangeFill	plots grids as varying colors on a 2D or 3D plot.
PFO_GridContour	plots contours of grid data on 2D or 3D plot.
PFO_GridFishnet	plots grid data as a fishnet on 3D plots.
PFO_NSXYLabels	plots data labels on a 2D plot.
PFO_NSXYZLabels	plots data labels on a 3D plot.
PFO_XYHorsetail	plots a horsetail plot based on data in an XY array.
PFO_TableHorsetail	plots a horsetail plot based on all columns in a table (a single column is selected for all X values).

4.3.5 objPFOGL

Implements code to draw objPFO functional objects on a plot.

4.3.6 objPPO

File I/O and user interface for all objPFOGL objects.

4.3.7 objUFO

Basic utility classes for writing data.

UFO_WriteCube	writes cube data to a file.
UFO_WriteGrid	writes grid data to a file.
UFO_WriteTable	writes a table to a file.
UFO_WriteXY	writes XY data to a file

4.3.8 objUPO

File I/O and user-interface for objUFO

4.4 OsLib

OsLib provides functional objects and classes to support sampling and optimization and processing/plotting of optimization output. OsLib source code is structured in sub-directories similarly to GenLib.

4.4.1 osClass

Basic classes, data classes, data class wrappers, and IO:

C_Optimizable	all simulators with optimizable parameters must derive from this.
C_VarBase	base class for different variables types.
C_OptVar	variable to be optimized.
C_OptVarUnits	adds unit specification and conversion.
C_SampVar	variable to be sampled.
C_SampVarUnits	adds unit specification and conversion.
C_VaryVar	variable that can be multiply valued (suite or range).
C_VaryVarUnits	adds unit specification and conversion.
C_SimErr	error exception class for simulators.

DC_Covar	covariance matrix optimizer results and confidence limit processing.
DC_FitResults	optimizer results (fit value, residuals) for a single fit.
DC_Jacobian	optimizer results (Jacobian matrices) for a single fit.
DC_OptSimResults	class to package up all optimizer results for file I/O.
DC_RangeSimResults	class to package up optimizer/simulator results of multiple parameter value runs for file I/O.
DC_CovarArray	array of DC_Covar.
DC_CubeArray	array of DC_CubeData.
DC_GridArray	array of DC_GridData.
DO_XXX	wrappers for DC_XXX above.
IO_OptSimResults	binary file I/O for DC_OptSimResults.
IO_RangeSimResults	binary file I/O for DC_RangeSimResults.
U_Resid	basic residual utilities (standardization, normal distribution calculations).

4.4.2 osDataClass

Minor master/slave support classes for extracting individual results.

4.4.3 osMain

Main optimizer/sampler code and global variables and other support classes used by optimizer and application.

Main Code

C_Optimize	class implementing the optimizer.
C_Sampler	sampler implementation.

Variables & Support

G_Optimize	support for optimizer usage.
G_OptRange	support for combined multiple value/optimization runs.

G_Range	support for multiple value runs.
G_RealRange	support for extracting single values from multiple value runs.
G_Sample	support for sampler usage.
G_Vary	more support for multiple value runs.
C_OSListObj	base class for application modal listings .

4.4.4 osDFO

Functional objects for calculating fits, and for reading, selecting, and operating on optimizer results.

Fit_Calculation

DFO_SingleFit	calculates fit metric for simulated and field data.
DFO_CompositeFit	combines the results from multiple single fits (and/or other composite fits).
DSC_FandChiConfidence	Performs common confidence interval calculation for inherited use by DFO_ objects.
DFO_CalcConfidence	Calculates confidence of optimization results, using one result as the best case.
DFO_CalcConfidenceGrid	Calculates confidence of grid point results, where the grid value is the fit result, using one grid point as the best case.
DFO_CalcConfidenceTable	Calculates confidence of a table containing fit results, using one row as the best case.

Read Results

DFO_ReadOptSimResults	reads optimizer results from file.
DFO_ReadRangeSimResults	reads multiple parameter value results from file.

Selection

DFO_SelectOptCovar	selects one or more covariance results from optimizer results.

DFO_SelectOptJacob	selects one or more Jacobian results from optimizer results.
DFO_SelectOptResid	selects one or more residual results from optimizer results.
DFO_SelectRangeCube	extracts a single cube from three multiple parameter value results.
DFO_SelectRangeGrid	extracts a single grid from two multiple parameter value results.

Processing

DFO_BasicResidual	basic residual operations (standardize, sort).
DFO_JacobToTable	extracts data from Jacobian in DC_TableData form.
DFO_ResidualDiagnostic	quantile-normal and standard residual diagnostic calculations.

4.4.5 osDPO

I/O and user interface for all osDFO functional objects.

4.4.6 osLFO

Listing functional objects for optimizer results.

LFO_Covariance	list confidence limits.
LFO_Jacobian	list Jacobian results.
LFO_OptRun	lists summary data for an optimization run.

4.4.7 osLPO

File I/O and user interface for all osLFO functional objects.

4.4.8 osPFO

Plotting objects for covariance data.

PFO_CovarLimits	plots confidence limits of 2D or 3D plots.

4.4.9 osPFOGL

Implements code to draw osPFO functional objects on a plot.

4.4.10 osPPO

File I/O and user interface for all osPFOGL functional objects.

4.5 NsLib

NsLib provides functional objects and classes specific to nSights and well-test analysis.

4.5.1 nsClass

Basic classes, data classes, data class wrappers, and IO:

C_Derivative	definition of data required derivative calculation.
C_TimeProcess	definition of data required for Horner/Argawal/Bourdet time processing.
DC_Derivative	derived from C_Derivative and adds derivative calculation.
DC_ProfileSimResults	class for packaging P(r) data for file I/O.
DC_SequenceTimes	sequence names and start/end times.
DC_TimeProcess	derived from C_TimeProcess and adds calculation.
DC_XYResponseFunction	Class for packaging XY data and additional data required such as memory and time spacing.
DC_XYSimResults	class for packaging F(t) data for file I/O.
DO_XXX	wrappers for DC_XXX above.
IO_ProfileSimResults	file I/O for DC_ProfileSimResults.
IO_XYSimResults	file I/O for DC_XYSimResults .
nSightConst	gravitational constant.

4.5.2 nsDataClass

Minor master/slave support classes for extracting individual results.

4.5.3 nsDFO

Functional objects for calculating fits, and for reading, selecting, and operating on simulator results.

Fit Calculation

DFO_BasicSequenceFit calculates fit metric for one or more sequences.

DFO_BEETCompensation Corrects pressure data based on a response function, barometric pressure data and earth tide data.

DFO_CreateBEETResponseFunction Calculates a response function based on baseline pressure data, barometric pressure data and earth tide data

Read Results

DFO_ReadMiniTroll reads results from MiniTroll formatted file in raw text or post-processed CSV format. Calculates time as elapsed calendar time.

DFO_ReadProfileSimResults reads profile data from file.

DFO_ReadSequenceTimes reads sequence times from ASCII file.

DFO_ReadXYSimResults reads simulator XY results from file.

Selection

DFO_SelectProfile selects a single profile from profile results.

DFO_SelectXY selects one or more XY results from simulator results.

Processing

DFO_BasicTimeExtract extracts specified X interval from XY data and optionally interpolates to regular X interval.

DFO_Derivative calculates derivative of input data.

DFO_ExtractSequenceInterval extracts interval corresponding to single sequence from XY data.

DFO_TimeProcessing performs time processing (Horner/Argawal/Bourdet) on input data.

4.5.4 nsDPO

I/O and user interface for all nsDFO functional objects

4.5.5 nsPFO

PFO_SequenceGridLines plots grid lines and labels at sequence intervals

4.5.6 nsPFOGL

Implements code to draw nsPFO functional objects on a plot.

4.5.7 nsPPO

File I/O and user interface for all nsPFOGL functional objects.

4.6 nPre Application

nPre is organized in a main directory and nine subdirectories. Contents of each directory are described below.

4.6.1 Main

The nPre main directory contains the MFC files to create the application, and define its document and view classes.

MainFrm OutlookFrame derived class.

nPre CWinapp derived class to initialize application.

nPreView fills MFC requirement for CView derived class.

nPreDoc fills MFC requirement for CDocument derived class. Processes file new/save/open messages.

ProgressThreadDialog threading support for running simulation in background.

4.6.2 App

Subdirectory contains implementation of GenLib genApp derived code.

AppFuncObj performs application specific DO_XXX type conversions.

G_Version defines genLib G_Version variables.

nPreAppMenu defines structure and contents of Object/Page drop down menus.

nPreFile	writes/reads all nPre variables and functional objects from/to ASCII configuration file.
nPreRoot	defines MenuRootC defined nPre object trees.
nPreUtilities	loads/saves app settings in registry

4.6.3 AppAlloc

Contains allocators for all functional objects referred to in nPreAppMenu.

4.6.4 AppHelp

Maps functional object names to help system IDs.

4.6.5 Auto

Implements auto setup procedures.

4.6.6 Dlg

Implements all nPre dialogs.

4.6.7 Run

Contains a single file RunControl, which performs simulator execution according to settings. Also manages simulation progress dialog.

4.6.8 SimCore

Contains the actual well test simulator implementation and support classes to extract data from global variable data structures defined in subdirectory Var.

4.6.9 UI

Defines the dialog layout on tabbed menus.

4.6.10 Var

Defines classes for and variables containing all nPre input data.

C_CurveFile	class for error checking andd reading curve files into DC_CurveArray structures.
C_DataCapture	defines classes for specifying F(t) data to be extracted from the simulation.

C_FileOutput	base class for defining output files.
C_Parameter	class defines single simulator parameter.
C_Sequence	class defines single sequence for simulator.
C_TestZoneCurve	defines types of test zone boundary conditions for curve specified f(t).
C_Units	adds nSights specific units to UnitsBase.
E_Parameter	enumerated type with symbolic definition for each nPre parameter.
G_CalcParameter	calculates and lists parameter values.
G_Control	basic simulation configuration vars, file I/O, error checking and listing.
G_CurveFiles	variables for test zone BC, f(r) and f(P) curves.
G_DataCapture	data capture specification vars and associated file I/O, error checking and listing support.
G_Parameter	parameter specification and associated file I/O, error checking and listing support.
G_Sequence	sequence specification and associated file I/O, error checking and listing support.

4.7 nPost Application

nPost is organized in a main directory and two subdirectories. Contents of each directory are described below.

4.7.1 Main

The nPre main directory contains the MFC files to create the application, and define its document and view classes.

MainFrm	SingleFrame derived class.
nPost	CWinapp derived class to initialize application.
ProjectDoc	fills MFC requirement for CDocument derived class. Processes file new/save/open messages.

4.7.2 App

Subdirectory contains implementation of GenLib genApp derived code.

AppFuncObj	performs application specific DO_XXX type conversions.
G_Version	defines genLib G_Version variables.
nPostAppMenu	defines structure and contents of Object/Page drop down menus.
nPostFile	writes/reads nPost menu tree functional objects to/from ASCII configuration file.
nPostRoot	defines single MenuRootC derived object tree.

4.7.3 AppAlloc

Contains allocators for all functional objects referred to in nPostAppMenu.

4.7.4 AppHelp

Maps functional object names to help system IDs.

5.0 MENU STRUCTURE

5.1 nPre Menu Structure

This subsection describes the general UI layout for nPre. There are different menu structures/levels within nPre. For reference, these menus are defined as follows:

Task Bar
Navigation Pane
Primary Tabs
Nested sub-Tabs

The p*rimary tabs* and n*ested sub-tabs* often have dialog areas where the user must enter the requested information or choose between various options. Data and graphics are dealt with under three different options of the *Navigational Pane*: Field Data, Sequence, and Processing Setup. The capabilities used to define constraints for a simulation can be found using *primary tabs* and *nested sub-tabs*. For example, the *primary* tabs for the *Configuration* option on the Navigation Pane are from left to right: *Main, Curve Files, Liquid, Gas, Matrix, Default Units,* and *Test Description.* Dialog areas are found on each of these tabs, which are used to modify or define the constraints of the simulation. Nested sub-tabs are found under the Parameter *Navigation Pane* option. The *Primary tab* Parameters has *nested sub-tabs* for each group of parameters: formation, fluid, test-zone, and numeric.

The following sections will describe each Navigation Pane option and the associated tabs.

5.2 Menu Item Description

A description of task bar items is followed by a detailed description of menu pages for individual categories of input.

Task Bar

File		all operations on text configuration files
	New	
	Open	
	Save	
	Save As	
	Print Setup	
	Exit	
nPre		Lists the Navigation Pane options
	Configuration	
	Well ID and Output	
	Sequence	
	Parameter	
	f(p)/f(r) Points Parameter	
	Simulation Output	
	Fit Specification	
	Optimization	
	Sampling	
	Suite/Range	
	Output File Setup	
	Plot & Fit Setup	
List		creates text (or HTML) listings in other top-level windows.
	Current	listing of data/settings associated with currently selected top-level tab.
	Current Errors	listing of errors associated with currently selected top-level tab. Only sensitive if errors exist.
	Calculated Parameters	listing of calculated parameters
	All	invokes dialog for selection of categories, then listing of data/settings associated with selected categories.
	All Errors	listing of errors that require correction before running the simulation.
	Messages	
Auto Setup		for auto processing data and plots and populating type trees

Field data plots
Sequence plots
Basic fit plots

Object for populating type trees used in several categories
New list of available objects in context
Duplicate
Copy
Copy page
Paste
Delete

Apply
Connections
Create Plot Object Folder

Page for populating type trees used in several categories
New 2D xy plot
New 3D xyz plot
New data
New Composite Plot
Duplicate
Copy current
Copy all
Paste

Delete
Delete all pages
Bring page window to top
All connectionsCollapse Tree

Run error checks then runs simulation
Minimal
Verbose
Covariance only
Minimize main

View toolbar control and nested controls for viewing
Tool bar
Status bar
Control bar
Settings

Window
Window list F11control for popup screen
Minimize all windows

Help

Help Topics

About nPre — Mail complaints about program
Watch manufacturer web site

Configuration Navigation Pane option. The configuration menu contains seven primary tab controls.

Main — Basic configuration

Simulation type	Forward/Optimization	
Simulation sub-type	Single/ Sampled/Range	
Phase to Simulate	Liquid/Gas	
System porosity	Single/Double	*liquid only*
Leakage	None/Single/Dual	*liquid only*
Skin effects	yes/no	
External boundary	Fixed Pressure/Zero Flow	

Curve Files *.nCRV files containing functional approximation (f(P), f(r), f(t)) data

Wellbore boundary conditions	file name
[Browse]	
f(P) parameters	file name
[Browse]	
f(r) parameters	file name
[Browse]	
Reload curves	Button

Liquid liquid phase

Permeability/hydraulic conductivity	Permeability/Hydraulic Conductivity
Storage parameter	SpecificStorage/Porosity*Total Compressibility
Compensate flow dimension geometry	yes/no
Constant TZ surface area	no/yes
Test zone volume	Constant/Varying
Test zone compressibility	Constant /Varying
Test zone temperature	Constant/Varying
Default temperature	20 [deg C]
Solution variable	Head/Pressure
Default liquid density	1000 [kg/m^3] *head only*
Allow-ve pressure/head	yes/no

Gas gas only

Klinkenburg effects	yes/no
Viscosity as f(P)	yes/no
Gas flow solution variable	Mass Flow / Volume @ STP
STP temperature	20.0 [deg]

STP pressure 100.00 [kPa]

Matrix liquid & dual porosity only

Matrix geometry Prismatic/Spherical
Alpha Calculated/Entered

Default units default units set at this point and used throughout

Time
Distance
Volume
Mass
Volumetric flow rate
Mass flow rate
Pressure
Hydraulic conductivity
Permeability
Transmissivity
Storage
Compressibility
Wellbore storage
Density
Temperature
Thermal expansion
Matrix geometry
Inverse time
Ratio

Test Description Single text entry area for multiple lines of text. Default entries for version #, creation date.

Wells and Output Navigation Pane object with three Primary tabs.
Defines what data is extracted from simulations.

Main tab
Grid control with 4 columns and 23 rows.

Column	Contents	Description
1	ID	Data from pressure or flow
2	Type	Pressure/Flow/Production/Other
3	Sub-type	if Pressure type Test Zone/Observation Well/Superposition

		if Flow type Well/Formation/TZ/Storage if Production type Well/Formation/TZ/Seq.Change if Other TZ Temperature/TZ Compressibility/TZ Volume
4	Radius	if Pressure/Ob Well radius value if Pressure/Superposition superposition dialog
5	Radius Units	if Pressure/Ob Well radius units if Pressure/Superposition radius units
6	Output Units	[units]

Production Restart Tab
Grid control with one column and 8 rows. Enter up-to 8 ascending time values.

Superposition Tab
Grid control with 2 (forward, non sampling) or 3 (otherwise) columns.

Column	Contents	Description
1	Type	Constant/Sampling/Optimize
2	Radius	if Constant, value if Sampling sampling dialog (see Sampled Parameter Dialog) if Optimization optimization dialog (see Optimized Parameter Dialog)
3	Operation	add P/ subtract P/ add delta P/ sub delta P

Sample Parameter Dialog

Distribution type	Normal/Log-Normal/ Uniform/Log-Uniform/ Triangular/Log-Triangular	
Mean	_____	[units] *normal/log-normal*
Std. deviation.	_____	[units] *normal*
Log std. deviation.	_____	[units] *log-normal*
Peak	_____	[units] *non-normal*
Lower limit	_____	[units] *non-normal*
Upper limit	_____	[units] *non-normal*

Optimization Parameter Dialog

Minimum value	_____ [units]
Maximum value	_____ [units]
Best estimate value	_____ [units]

Stepping		Linear/logarithmic
Estimated std dev	_____	[units]

Sequences Navigation pane option.

The sequence contains primary tabs described as follows:

Time-Base tab

Sequence time entry method	Start-End/Duration		
Start time of first sequence	_____	[units]	
End time of last sequence	_____	[units]	*start-end only*

Sequences tab

Grid control with 7 columns and n (64) rows.

Column	Contents	Description
1	Type	Flow/History/Pulse/Slug *slug n/a for gas*
2	Designation	user entered alpha numeric design
3	StartTime [sec]	start time or duration *duration has separate units*
4	Sequence data	
5	Duration [sec]	end time if duration method *display only* duration if start time method
6	Auto?	check box for auto set up

Flow Sequence Dialog

Time step type	Static/Log/Dynamic P/Dynamic Q		
Static time step size	_____	[units]	*static only*
Total # of time steps	100		*log only*
First log time step size	_____	[units]	*log only*
Minimum time step size	_____	[units]	*dynamic only*
Maximum time step size	_____	[units]	*dynamic only*
Flow type	Fixed/Curve/Sampled/Curve+Sampled		
Fixed flow	_____	[units]	*fixed only*
Sampling/vary data	sample dialog	[units]	*sampled/sample curve only*
Wellbore storage	None/Isolated/Open		*open n/a for gas*

History Sequence Dialog

Time stepping	*see flow sequence except no Dynamic P*		
Pressure type	Fixed/Curve/Sampled/Curve+Sampled		
Fixed pressure	_____	[units]	*fixed only*
Sampling /vary data	dialog	[units]	*sampled/sample curve only*
Wellbore storage	None/Isolated/Open		*open n/a for gas*

See Parameters below for description of sampled data dialog.

Pulse Sequence Dialog	
Time stepping	*see flow sequence*
Pulse type	Absolute/TS Relative/Sequence Relative
Pulse pressure	_____ [units]
TZ Thermal conditions	Isothermal/Non-Isothermal

Slug Sequence Dialog	
Time stepping	*see flow sequence*
Slug type	Absolute/TS Relative/Sequence Relative
Slug pressure	_____ [units]

TZ Curves tab

Grid control with 5 columns and n(64) rows.

Column	Contents	Description
1	Type	Pressure Flow Temperature Compressibility Volume change Volume
2	Curve ID	select from IDs of curves in test-zone BC curve file
3	Start Sequence	select from defined sequences
4	End Sequence	select from defined sequences
5	Curve Data	

Curve Data Dialog	
Time base	Test/Sequence
Y data units in curve	[units]
Y data is log 10	no/yes
Time data units	[units]
Time data is log 10	no/yes

Dynamic Time Step tab

Maximum pressure change	_____	[ratio] *decimal/%*
Minimum pressure change	_____	[ratio] *decimal/%*
Maximum flow rate change	_____	[ratio] *decimal/%*
Minimum flow rate change	_____	[ratio] *decimal/%*
Max # of TS in dynamic sequence	10000	

Partial Run tab

Simulation time extents	Full test/Partial test *all sequences/selected range*
Start sequence	select from defined sequences *selected only*
End sequence	select from defined sequences *selected only*

Parameters Navigation Pane object

The Parameter object contains primary tabs with nested sub-tabs for each group of parameters (see Table 6.1 for parameter groupings). The nested sub-tabs deal with calculated parameter values, units, and execution order of suite/range parameters. The Task Bar/ List option has an entry for list Calculated Parameters, which creates a text/html listing of specific derived parameter values.

Each parameter tab contains a grid control with 4 columns:

Column	Contents	Description	
1	Name	parameter name, fixed	
2	Type	Constant	*not all types valid for all parameters*
	Sample		
	Range		*max 3 range parameters*
	Suite		*max 3 suite parameters*
		Optimise	
		f(P) File	
		f(P) Points	
		f(r) File	
		f(r) Points	
3	Value	single value or type specific dialog	
4	Units	select from appropriate types	

Type specific dialogs are as follows:

Sample Parameter Dialog

Distribution type	Normal/Log-Normal/ Uniform/Log-Uniform/ Triangular/Log-Triangular	
Mean	_____	[units] *normal/log-normal*
Std. deviation.	_____	[units] *normal*
Log std. deviation.	_____	[units] *log-normal*
Peak	_____	[units] *non-normal*
Lower limit	_____	[units] *non-normal*
Upper limit	_____	[units] *non-normal*

Range Parameter Dialog

Minimum-value	_____ [units]
Maximum-value	_____ [units]
Stepping	Linear/Logarithmic
# of steps	single value or type specific dialog

Suite Parameter Dialog

Units	select from appropriate types

Single column grid with 10 rows

Optimization Parameter Dialog

Minimum value	_____ [units]
Maximum value	_____ [units]
Best estimate value	_____ [units]
Stepping	Linear/logarithmic
Estimated std dev	_____ [units]

f(r) File Dialog

Curve ID	*selected from desig in f(r) curve file*
Distance units in curve	[units]
Data units in curve	[units]
Distance is log10	yes/no
Data is log10	yes/no

f(P) File Dialog

Curve ID	*selected from desig in f(P) curve file*
Pressure units in curve	[units]
Data units in curve	[units]
Distance is log10	yes/no
Data is log10	yes/no

f(P) points and f(r) points are special cases with more complex dialogs implemented on separate main level tabs.

f(P)/f(r) Points Parameter Navigation Pane object

Primary tabs with nested secondary tabs. A Primary tab is defined for a valid f(P) point or f(r) point parameter. Each Primary tab contains 4 nested secondary tabs as described below.

Point Entry tab

Grid control with 32 rows and 6 columns in optimize mode, 4 columns in forward mode:

Column	Contents	Description
1	X Type	Constant Fixed/Suite/Optimize
2	X Opt ?	determined by parameter option

3	OptMin	if X Opt lower r for range
4	OptMax	if X Opt upper r for range
5	Y Type	Constant Fixed/Suite/Optimize
6	Y Opt ?	single value determined by parameter option

Interpolation tab Describes interpolation method for converting points to f(r).

Interpolation method	cubic spline/least squares/ linear / rings 1 / rings 2	
Cubic spline end slope	Natural/User-Set	*for cubic spline (CS)*
Start slope	0.0	*for user-set CS*
End slope	0.0	*for user-set CS*
Spline tension	10	*for CS*
Polynomial order	1	*for least-squares*

Units/Transform tab

Distance/Pressure units in curve	[units]
Data units in curve	[units]
Distance/Pressure is log10	yes/no
Data is log10	yes/no

Optimization tab *only sensitive if 1 or more Y values is to be optimized*

Minimum value	_____	[units]
Maximum value	_____	[units]
Stepping	Linear/Log	
Estimated std dev	_____	[units]

Fit Selection

Fit Selection Defined fits to use

Grid with 2 columns and 8 rows.

Optimization Navigation Pane option with 5 Primary tabs. The code uses non-linear regression algorithms to adjust the values of user-specified fitting parameters to obtain an optimal fit.

Main tab

Algorithm	Simplex/Lev-Mar	
Calculate confidence limits	yes/no	*simplex only*
Covariance matrix calculation	1st order/2nd order	*Lev-mar, or simplex with conf.*
Fixed deriv. span in covariance calc	yes/no	*Lev-mar, or simplex with conf.*
Fixed span	1E-06	*if fixed derivative span is yes.*
Multiple fit start point	last result/best estimate	
Update best estimat*es*	no/yes	

Tolerances tab

Parameter tolerance	1E-5	
Derivative adjustment tolerance	1E-5	
Maximum # of simulations	10000	

L-M Algorithm tab

Lambda factor multiplier	2	
Initial Lambda factor	1E-3	
Minimum lambda factor	1E-8	
Relative change tolerance	1E-5	
Maximum derivative span	1E-8	

Simplex Algorithm tab

Initial vertex span	0.1	
Initial derivative calc span	1E-02	*if confidence limits and not fixed*

Perturbation tab

# of perturbations	0	
Perturbation span	0.2	
Perturb from	start/last fit	
Random seed #	13597	

Sampling Navigation Pane option with four Primary tabs is used to configure sampling. This option is only sensitive in the sampling mode.

Main tab

Sampling procedure	LHS/Monte-Carlo	
Number of trials	100	
Random number seed	34969827	
User variable correlations	yes/no	*LHS only*
Force non-specified correlations to	no/yes	

Correlations tab Sensitive only if LHS and two or more variables selected from Sequence-Sequences. One secondary nested tab for each: Parameter, Sequence Q, Sequence P, Superposition R. The secondary nested tab is sensitive only if two or more from the Navigation Pane sequence/sequences are defined for sampling.

The Correlations tab is grid controlled with n rows and n columns, where n is the number of sampling variables in the category. Diagonal entries are 1.0 and are not sensitive. Off-diagonal entries are symmetric.

Samples tab Sensitive when sub-type mode selected for sampling in Optimization.

Graphics tab A tree control used with first-level nodes representing data pages (for data-processing objects) and 2D and 3D plots. Second-level nodes represent specific data or plot objects.

Suite/ Range Navigation Pane option. Sensitive when simulation type set to Optimization or simulation sub-type set to range.

Priority tab Sensitive only when at least one parameter set to suite/range. Sets the loop priority for defined range/suite parameters

Grid/Cube X (slowest grid/cube) parameter ID *select from defined suite/range*
Grid/Cube Y (fastest grid/middle) parameter ID *select from defined suite/range sensitive only if 2 or 3 defined*
Grid/Cube Z (fastest cube) parameter ID *select from defined suite/range sensitive only if 3 defined*

Output Files Navigation Pane object with four Primary tabs.
Specifies output files to create.

XY Data tab Only mode not sensitive in is the range simulation sub-mode.

Write XY output	Check box for yes or no	*Listing output*
File name	_____ [browse]	
If file exists?	Overwrite/Append	
Run identifier	_____dialog	
Data to write	_____dialog	

Data To Write Dialog
Grid control with two columns, n rows, listing defined output, with check-box to select.

Profile tab sensitive only in forward/single mode

Write profile output	Check box for yes/no
File name	_____ [browse]
If file exists?	Append/Overwrite
Run identifier	_____ dialog
First sequence to write	select from sequences
Last sequence to write	select from sequences
Time step modulus	2
Node modulus	2

Range tab Sensitive only in range sub-mode, always written in that mode

Write range output	Check box for yes/no
File name	_____ [browse]
If file exists?	append/overwrite

Run identifier	_____	dialog

Optimization tab Only sensitive in optimization mode and sub-type normal or sampling.

Write optimization output	yes/no	
File name	_____[browse]	
If file exists?	add/overwrite	
Run identifier	_____	
Store residuals	yes/no	
Store sampled parameters	yes/no	only in sampling mode
Store residual Jacobian	yes/no	only in sampling mode

Plots & Data Processing

Navigation Pane option with 4 Primary tab options. A tree control is used with first-level nodes (leaves) representing data pages (for data-processing objects) and 2D and 3D plots. Second-level nodes represent specific data or plot objects.

Field Data Tab Navigation Pane option.

Field data contains a tree to be populated with second-level objects for reading XY data from text files, and providing quick plots of the data. A default tree can be constructed using Auto Setup/ Field data plots.

The default tree is defined as follows:

Field Data	top level collection object	*Input::Data*
Pressure Data	for reading P vs T data	
Flow Data	for reading Q vs T data	
Pressure Plot	basic XY plot connected to P (pressure) vs T(time) data	
Flow Plot	basic XY plot connected to Q (flow) vs T data	

Additional objects and plots can be defined. Available objects are oriented to reading and initial processing of standard data:
Read XY from text
Enter XY
Time Offset/Extract

Sequence Tab Navigation Pane option

Contains a tree to be populated with second level objects. Uses a data/graphics object tree structure to create data sets that use the simulated data generated from processing the corresponding field data constraints.

Fit Tab Navigation Pane option
Contains a tree control for paired regression analysis of simulated and field data.

Runtime Tab Navigation Pane option.
Contains a tree to be populated with second level objects used to create a wide variety of graphs to monitor the real-time progress of the optimization.

6.0 PARAMETER LIMITS

The following table lists all parameter names and IDs. Note that sensitivity of a specific parameter or group depends upon configuration settings.

Table 6.1 Parameter Groupings and Limits

Table 6.1	**Parameter Groupings & Limits** Note: all parameter limits in base SI values except pressures which are kPa			
Group	**Name**	**ID**	**Min**	**Max**
Formation	Formation thickness	t	1e-10	5000
	Flow dimension	n	-1e+04	1e+04
	Static formation pressure	P_f	0	1e+06
	External boundary radius	r_o	1.e-4	1e+06
	Formation Permeability	k_f	1e-30	1
	Formation Conductivity	K_f	1e-30	100
	Formation Specific Storage	s_f	1e-30	1000
	Formation Compressibility	C_f	1e-30	1e-01
	Formation Porosity	theta_f	0	1
	Klinkenburg factor	b_kf	0	1e+06
	Specific yield in formation	Sy_fm	1e-15	1e+3
	Formation vertical permeability	kv_fm	1e-30	1.e+0
	Formation vertical conductivity	Kv_fm	1e-30	1.e+2
Fracture	Permeability	k_fr	1e-30	1
	Conductivity	K_fr	1e-30	100
	Specific storage	s_fr	1e-30	1000
	Compressibility	C_fr	1e-30	1e-01
	Porosity within fracture	theta_fr	0	1
Matrix	Permeability	k_m	1e-30	1

Table 6.1 **Parameter Groupings & Limits**
Note: all parameter limits in base SI values except pressures which are kPa

Group	Name	ID	Min	Max
	Conductivity	K_m	1e-30	100
	Specific storage	Ss_m	1e-30	1000
	Compressibility	C_m	1e-30	1e-01
	Porosity	theta_m	0	1
	Volume factor	V_m	1e-05	0.999999
	Geometry factor Alpha	alpha	1e-10	1e+05
	Slab thickness	m_t	1e-10	1e+05
	Sphere diameter	m_d	1e-10	1e+05
Skin zone	Permeability	k_s	1e-30	1
	Conductivity	K_s	1e-30	100
	Specific storage	s_s	1e-30	1000
	Compressibility	C_s	1e-30	1e-01
	Porosity	theta_s	0	1
	Klinkenburg factor	b_ks	0	1e+06
	Radial thickness of Skin	t_s	1e-10	5e+04
	Skin zone specific yield	Sy_s	1e-15	1e+03
Fluid	Density	roe	100	2000
	Viscosity	mu	0.05	5
	Compressibility	C_w	1e-11	1e-09
	Thermal expansion coefficient	C_T	-1e-03	5e-03
Gas	Molecular weight	MW	1	1000
	Reference temperature	T	5	100
	Gas Viscosity	Mu_g	1e-06	1
	Gas Viscosity slope factor	mu_m	-1	1
	Atmospheric pressure [abs]	P_a	10	200
Test-zone	Well radius mgw	r_w	1e-04	1000

Table 6.1 **Parameter Groupings & Limits**
Note: all parameter limits in base SI values except pressures which are kPa

Group	Name	ID	Min	Max
	Tubing string radius	r_s	1e-04	100
	Volume change from normal	dV	-1e4	1e4
	Compressibility	C_TZ	1e-17	100
	Partial-penetration bottom offset	PO_tz	0.01	1000.0
	Partial-penetration screen length	PL_tz	0.01	1000.0
Leaky layer	Thickness	b_l	1e-10	5000
	Permeability	k_l	1e-30	1
	Conductivity	K_l	1e-30	100
	Specific storage	Ss_l	1e-30	1000
	Compressibility	C_l	1e-30	1e-01
	Porosity	theta_l	0	1
Upper leaky layer	Thickness	b_Ul	1e-10	5000
	Permeability	k_Ul	1e-30	1
	Conductivity	K_Ul	1e-30	100
	Specific storage	s_Ul	1e-30	1000
	Compressibility	C_Ul	1e-30	1e-01
	Porosity	theta_Ul	0	1
Lower leaky layer	Thickness	b_Ll	1e-10	5000
	Permeability	k_Ll	1e-30	1
	Conductivity	K_Ll	1e-30	100
	Specific storage	s_Ll	1e-30	1000
	Compressibility	C_Ll	1e-30	1e-01
	Porosity	theta_Ll	0	1
Numeric	# of radial nodes	n_r	10	1e+05
	# of matrix nodes	n_m	1	100
	# of skin nodes	n_s	5	9000
	# of leaky nodes	n_l	2	100

Table 6.1	Parameter Groupings & Limits Note: all parameter limits in base SI values except pressures which are kPa			
Group	**Name**	**ID**	**Min**	**Max**
	# of vertical well nodes	n_vw	2	100
	# of vertical nodes above well	n_vu	2	100
	# of vertical nodes below well	n_vb	2	100
	Pressure solution tolerance	tol_P	1e-20	1
	STP flow solution tolerance	tol_Qv	1e-30	1
	Mass flow solution tolerance	tol_Qm	1e-30	1
	Well conductance multiplier	kVMult	1.e+0	1.e+12

7.0 REFERENCES

1) Nuclear Waste Management Program. April 13, 2000. *GTFM 6.20 Requirements Document Version 1.1*. ERMS #504020. Albuquerque, NM: Sandia National Laboratories.

2) Nuclear Waste Management Program. August 15, 1996. *GTFM Functional Description, Theoretical Development, and Software Architecture*. ERMS #240244. Albuquerque, NM: Sandia National Laboratories.

3) Nuclear Waste Management Program. April 13, 2000. *GTFM 6.20 Validation Document, Version 1.10*. ERMS #504022 Albuquerque, NM: Sandia National Laboratories.

4) Nuclear Waste Management Program. November 2002. *nSIGHTS Version 1.0 Design Document,*. ERMS #522059 Albuquerque, NM: Sandia National Laboratories.

5) Nuclear Waste Management Program. June 2011. *nSIGHTS Version 2.41a Validation Document,*. ERMS # 555650, Albuquerque, NM: Sandia National Laboratories.

6) Nuclear Waste Management Program. 2006. *NWMP Software Requirements, Procedure NP 19-1 Revision 12*. ERMS #543743. Albuquerque, NM: Sandia National Laboratories.