529633

Analysis Report
Task 3 of AP-088
Conditioning of Base T Fields to Steady-State Heads

(AP-088: Analysis Plan for Evaluation of the Effects of Head Changes on Calibration of Culebra Transmissivity Fields)

Task Number 1.3.5.~~1.2.1~~ 3.1.2 MC 5/29/03

Report Date: May 13, 2003

Authors:	[illegible] for Sean A. McKenna PMTS, Geohydrology Department (6115)	Date: 5/28/03
	[illegible] for David Hart Student Intern, Geohydrology Department (6115)	Date: 5/28/03
Technical Review:	[illegible] for Scott James SMTS, Geohydrology Department (6115)	Date: 5/28/03
QA Review:	[illegible] Mario J. Chavez Carlsbad Programs Group (6820)	Date: 5/28/03
Management Review	[illegible] Paul Shoemaker Manager, Carlsbad Programs Group (6820)	Date: 05/28/03

WIPP:1.3.5.3.1.2:TD:QA-L; 522085

1 of 112

1 of 226

Table of Contents

Table of Figures

Table of Tables

Introduction

This analysis report describes the activities of Task 3 of AP-088, "Analysis Plan for Evaluation of the Effects of Head Changes on Calibration of Culebra Transmissivity Fields" (Beauheim, 2002b). The purpose of this task is calibrate a set of mean transmissivity fields created in Task 2 (see Holt and Yarbrough, 2002) to fit observed steady-state, or equilibrium, heads measured at three different time periods as well as to the steady-state heads used for CCA model calibration.

Task 3 of Analysis Plan 088 is divided into four subtasks and the work done on these four subtasks makes up the following sections of this report:

1) *Analysis of Pilot Point Geometry*
   Developments in the field of stochastic inverse modeling in the past six to eight years have caused some fundamental changes in the way that stochastic inverse models are created. During the CCA calculations, pilot points were located sequentially during the inversion process at locations that would have the maximum impact on the ability of the inverse model to fit the observed data (see Lavenue, 1996). In the CCA calculations, the location of each pilot point was determined, this point was used to calibrate the model, and then the location of the next point was determined and so on until the maximum number of pilot points was reached.

   Current approaches to stochastic inverse modeling with pilot points (e.g., Gomez-Hernandez et al., 1997; Capilla et al., 1998) locate all pilot points at the start of the modeling and then simultaneously adjust all of them to match the observed heads better. Subtask 1 describes how to locate these pilot points at the start of the modeling procedure.

2) *Estimation of Boundary Conditions and Construction of Seed Realizations*
   The boundary conditions used in this model are either fixed-head or no-flow boundaries. The no-flow boundary is along Nash Draw and the fixed-head boundaries are estimated on the rest of the model domain boundary with kriging. Kriging is based on the head measurements within the model domain. For each of 1980, 1990, 2000, and the CCA, a unique head data set is available. These data display a relatively strong trend for each time period and must be detrended before kriging. The detrending is done by fitting a bivariate normal distribution to the data from each time period. The residuals between these bivariate normal distributions and the observed data are used to build variograms. The residual data and the variograms provide the basis for kriging the residual values across the domain. When these kriged residuals are added back to the bivariate trend model, the results are the initial heads. The initial heads estimated at the constant-head boundary locations are held fixed throughout the groundwater flow modeling.

The base transmissivity (T) fields created in Task 2 of this analysis (see Holt and Yarbrough, 2002) are based on multiple regression and therefore only fit the transmissivity measurements in the mean sense. It is necessary to update these base T fields to match the T at the measurement locations. This updating is done through simulation of residuals between the base T field and the measured T data. The updated fields are known as "seed" T fields.

3) *Forward Modeling*
   To test all of the techniques and the software prior to the actual inverse modeling and to develop a baseline set of travel times, the seed fields are used as input to the groundwater flow and particle-tracking routines for forward (uncalibrated) calculations.
4) *Steady-State Inverse Modeling*
   The last step in this analysis is to use PEST (Doherty, 1998) and MODFLOW (Harbaugh et al., 2000) to calibrate the seed fields independently to four different sets of "steady–state" heads. These heads include measurements made in 1980, 1990, and 2000, and the heads used for CCA model calibration

Descriptions of the activities associated with each of these subtasks make up the major sections of this report. Prior to these descriptions, a number of data feeds to this analysis are discussed along with a description of the computational hardware used for this analysis. The source of each type of data used in the analysis is documented. The final sections of this report compare the calibration and travel time results for the four sets of forward and inverse models.

Available Data

Calibration of the Culebra transmissivity fields to four sets of steady-state heads requires a variety of input data as well as some modeling decisions (e.g., size and discretization of the model domain) that must be made. Each modeling decision and each type of input data, the original source of this data, and any modifications to the original data are documented in this section.

Table 1. Data feed descriptions for the Culebra T-field analysis.

	Description	ERMS #	Reference	File Name
Head Data	Equilibrated (steady-state) heads measured at the wells for the 1980, 1990, and 2000 time periods and estimated for the CCA (fluid densities are also included in this data set, but are not used in this analysis)	522580	Beauheim, 2002	TFieldHeads.xls
Transmissivity Data	Well locations, transmissivity and residuals between measured transmissivity and base transmissivity at well locations	523889	Holt and Yarbrough, 2002	Residuals.dat
Base Transmissivity Fields	100 Realizations of the base transmissivity field created through multiple regression and indicator geostatistical simulation	523889	Holt and Yarbrough, 2002	newb01r.zip, newb02r.zip, newb03r.zip, newb04r.zip, newb05r.zip, newb06r.zip, newb07r.zip, newb08r.zip, newb09r.zip, newb10r.zip

Model Domain and Discretization

The north-south and east-west extent of the model domain were specified by Richard Beauheim, Robert Holt and Sean McKenna. This determination considered several factors including: 1) hydrogeological features in the vicinity of the WIPP site that could serve as groundwater flow boundaries (e.g. Nash Draw); 2) the areas to the north of the WIPP site that might create additional recharge to the Culebra due to water applied to potash tailings piles; and, 3) the limits imposed on the domain size by the available computational resources and the desired fine-scale discretization of the domain within the groundwater model. The final model domain is rectangular and aligned with the north-south and east-west directions. The coordinates of each corner of the domain are given in Table 2 in UTM coordinates.

Table 2. The coordinates of the corners of the numerical model domain.

Domain Corner	**X Coordinate (meters)**	**Y Coordinate (meters)**
Northeast	624,025.0	3,597,125.0
Northwest	601,675.0	3,597,125.0
Southeast	624,025.0	3,566,475.0
Southwest	601,675.0	3,566,475.0

A no-flow boundary corresponding roughly to the center of Nash Draw is shown in Figure 1 as a purple line extending from the northern to southern boundaries in the western one-third of the model domain. Model cells falling to the west of this boundary are considered to be inactive in the groundwater flow calculations. Within **MODFLOW 2000 (MF2K)**, the status of all the cells in the model (active, inactive, or constant head) is controlled by an array of integers – the IBND array. The no-flow boundary was provided by Holt and Yarbrough (2002) and the IBND array was specified by setting array entries corresponding to cells west of the no-flow boundary equal to "0", which defines them as inactive. The cells on the domain boundary to the east of the no-flow boundary are fixed-head cells with corresponding values of "-1" in the IBND array.

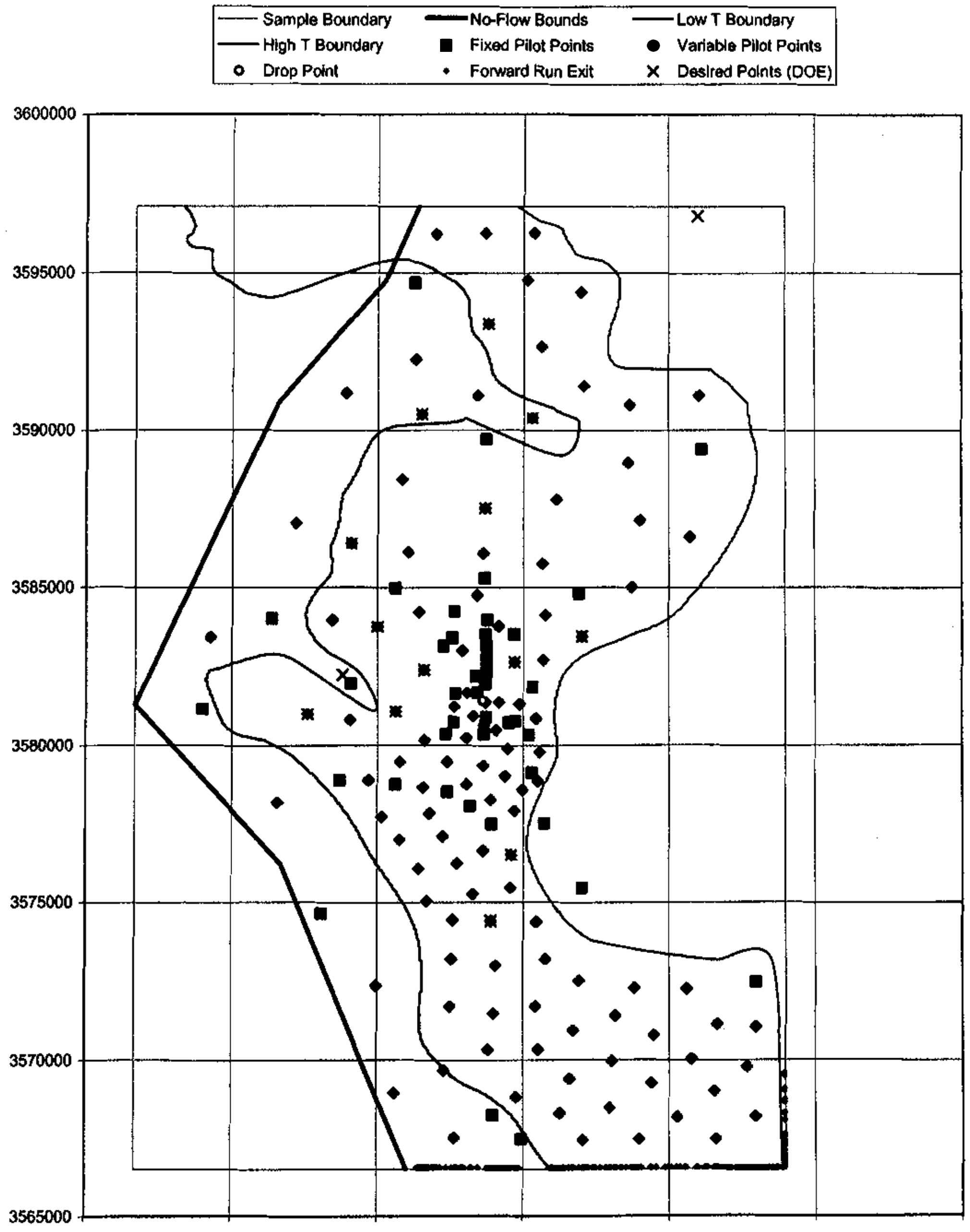


Figure 1. Locations of the pilot points and other features within the Culebra flow model domain.

The flow model is discretized into 274,011 regular, orthogonal cells each of which is 50x50 meters. Details of the grid generation can be found in Appendix C of Holt and Yarbrough (2002). A constant Culebra thickness of 7.75 meters is used (U.S. DOE, 1996, Appendix TFIELD.4.1.1). The 50-meter grid discretization was selected to make the finite-difference grid cell sizes considerably finer than those used in the CCA calculations, but still computationally tractable. In the CCA calculations, a telescoping finite-difference grid was used with the smallest cell being approximately 100x100 meters near the center of the domain. The largest cells in the CCA flow model grid were approximately 800x800 meters near the edges of the domain (Lavenue, 1996).

The elevation of the top of the Culebra was specified in an ascii text file, *culebra_top.txt*, generated by Lance Yarbrough (University of Mississippi). The calculations done for the top of the Culebra elevation surface are discussed in Appendix D of Holt and Yarbrough (2002).

The discretization of the flow model domain into 50x50 meter cells leads to a total of 274,011 cells (447 east-west by 613 north-south). However, 62,118 of these cells lie to the west of the no-flow boundary, so the total number of active cells in the model is 211,893. This number is more than a factor of 20 larger than the 10,800 (108x100) cells used in the CCA calculations.

PARTICLE TRACKING

For the particle-tracking calculations, a single particle is tracked from a starting location of X = 613,602 meters, Y = 3,581,425 meters until it exits the WIPP site boundary. The starting location corresponds to the center of the repository footprint, the "drop point" in Figure 1, and is the same location used to start particles in the CCA calculations. The coordinates of the corner points defining the WIPP site boundary are given in Table 3. The particle-tracking calculations use a constant advective porosity of 0.16 - the same value used in the CCA calculations.

Table 3. UTM Coordinates of the WIPP Site boundary.

Domain Corner	X Coordinate (meters)	Y Coordinate (meters)
Northeast	616,941	3,585,109
Northwest	610,495	3,585,068
Southeast	617,015	3,578,681
Southwest	610,567	3,578,623

Computing Platform

For these analyses, a parallel computing platform was constructed by creating a cluster of linked PC's. This platform consists of 16 PC's that serve as computational nodes and two servers. The configuration and the files on the two servers are identical and they serve as redundant backups of each other. All of the computational nodes contain a 1.9GHz equivalent AMD microprocessor with 1GB of RAM and a 40GB hard drive. The computational nodes are connected to each other and to the servers with 100Mb/sec ethernet switches. The linux, version 7.2 operating system is installed on all machines facilitating the use of the entire cluster as a single parallel computer. A picture of the computing platform, known as the "6115 linux cluster" is shown in Figure 2.

Figure 2. David Hart and Lane Yarrington with the 6115 linux cluster used to generate and optimize the Culebra T-fields.

The computing platform was designed for parallel processing. However, for this work, it turned out that the speed of the forward MF2K runs was so fast (10-15 seconds each) that the wait cycle programmed into the parallel version of PEST was not able to keep up efficiently with 16 jobs running simultaneously and the nearly constant communications across the ethernet connections. The approach used to solve this problem dedicated each processor in the parallel cluster to a single realization and all calculations for that realization were completed on the single processor. When those calculations were complete, a new realization was assigned to a waiting processor until all 100 realizations were complete. The time to do one set of 100 steady-state inverse calibrations with all 16 computational nodes running simultaneously was approximately 80 hours (3+ days).

Subtask 1: Analysis of Pilot Point geometry

A major development in the field of stochastic inverse modeling that has occurred since the T fields were constructed for the CCA in 1996 is that inverse techniques are now capable of simultaneously determining optimal T values at a large number of pilot points. In the T fields constructed for the CCA, pilot points were added one at a time and each point was calibrated prior to the addition of the next pilot point. Furthermore, the total number of pilot points was limited to less than or equal to the total number of observations to avoid numerical instabilities in the solution of the inverse problem. With the techniques now available and implemented in PEST, it is possible to use many more pilot points than there are head observations and to calibrate these pilot points simultaneously. However, locating the pilot points still requires technical judgement.

The original plan for this subtask (AP-088) was to conduct a series of calculations on a hypothetical site to determinine the optimal locations for pilot points. Results of these calculations would be used to develop a heuristic algorithm that could be applied to locating pilot points within the Culebra. However, after several discussions with John Doherty (the author of PEST), it was determined that locating pilot points is a problem and goal specific activity and cannot be easily coded into an algorithm. A strong element of expert judgment goes into determining pilot point locations. The deliverable for this subtask was changed from a memo documenting the calculations done to determine the best pilot point locations to a more general memo documenting issues that need to be considered when locating pilot points. This memo was written by John Doherty under contract to Sandia and is attached to this status report as Appendix 1.

Delivery of the memo in Appendix 1 of this status report fulfills the deliverable for subtask 3.1. Pilot points were located according to the guidance put forth in the Appendix 1 memo and are shown in Figure 1. Pilot points located at the transmissivity measurement locations are held as fixed values during the optimization (fixed pilot points shown in magenta in Figure 1). The variable pilot points (dark blue in Figure 1) are those where the transmissivity value is adjusted during the calibration procedure. A total of 43 fixed and 115 variable pilot points was used in the steady-state Culebra calibration process.

Subtask 2: Estimation of Boundary Conditions and Construction of Seed Realizations

Within the indicator zonation provided by Robert Holt in Task 2 (see Analysis Plan 088, Beauheim, 2002) and using the T values estimated through multiple regression as secondary information, a series of T fields is generated with geostatistical simulation. These are the seed T fields that will be used as input to the stochastic inverse modeling. Prior to the stochastic inverse modeling, these seed T-fields are not conditioned to the head measurements. During the stochastic inverse modeling, each of these seed fields will be conditioned to available T measurements and the best estimate of the T value provided by the multiple regression.

Fixed-Head Boundary Conditions

For each of the sets of measured or estimated heads (1980, 1990, CCA and 2000), a set of initial heads values is estimated across the flow model domain. The head values estimated for the fixed-head cells along the north, east and south boundaries of the model domain remain constant for the groundwater flow calculation. The head values estimated at the cells in the interior of the domain are used as initial values of the heads and are subsequently updated by the groundwater flow model until the final solution is achieved. The estimation of the initial and boundary heads is done by kriging. Observed heads both within and outside of the flow model domain (Figure 3) are used in the kriging process.

Kriging is a geostatistical estimation technique that uses a variogram model to estimate values of a sampled property at unsampled locations. Kriging is designed for the estimation of stationary fields (see Goovaerts, 1997); however, the available head data for all four sets show significant trends (non-stationary behavior) from high head in the northern part of the domain to low head in the southern part of the domain. This behavior is typical of groundwater head values measured across a large area with a head gradient. To use kriging with this type of non-stationary data, a polynomial function is fit to the data, and the differences between the polynomial and the measured data, the "residuals," are calculated and a variogram of the residuals is constructed. This variogram and a kriging algorithm are then used to estimate the value of the residual at all locations within a domain. The final step in the process is to add the trend from the previously defined polynomial to the estimated residuals to get the final head estimates. This head estimation process is similar to that used in the Culebra calculations done for the CCA (Lavenue, 1996). However, here, there are four different sets of heads and therefore four different functions fit to the data and finally four sets of initial and boundary heads.

The available head data for each of the four sets are shown in Figure 3. There are 16, 28, 34 and 37 head measurements for the 1980, 1990, CCA and 2000 sets, respectively. In general, these head measurements show a trend from high head in the north to lower head in the south. For each set of heads, the trend is modeled with a bivariate Gaussian function. The use of this Gaussian function with five estimated parameters allows considerable flexibility in the shape of the trend that can be fit through the observed data. The value of the Gaussian function, *Z*, is:

$$Z = a\exp\left[-\frac{1}{2}\left(\left(\frac{X-X_0}{b}\right)^2+\left(\frac{Y-Y_0}{c}\right)^2\right)\right]$$

where X_0 and Y_0 are the coordinates of the center of the function and b and c are the standard deviations of the function in the X (east-west) and Y (north-south) directions, respectively. The parameter, a, controls the height of the function. The Gaussian function is fit to each set of measurements using the regression wizard tool in the SigmaPlot 2001 graphing software. The parameters estimated for the Gaussian function for each set of head measurements are presented in Table 4. Detailed results and diagnostics of fitting the Gaussian trend surface to the 1980 data are provided in Appendix 2.

Table 4. Parameters for the Gaussian trend surface model fit to the four sets of heads.

Trend Surface Parameters	1980	1990	CCA	2000
X_0	626195.36	615691.51	497048.41	611011.89
Y_0	4149817.94	3927177.23	3712731.95	3780891.50
a	1323.29	1155.98	1024.16	1134.61
b	163929.49	124127.33	4378431.60	73559.35
c	674926.86	517624.71	287104.56	313474.40

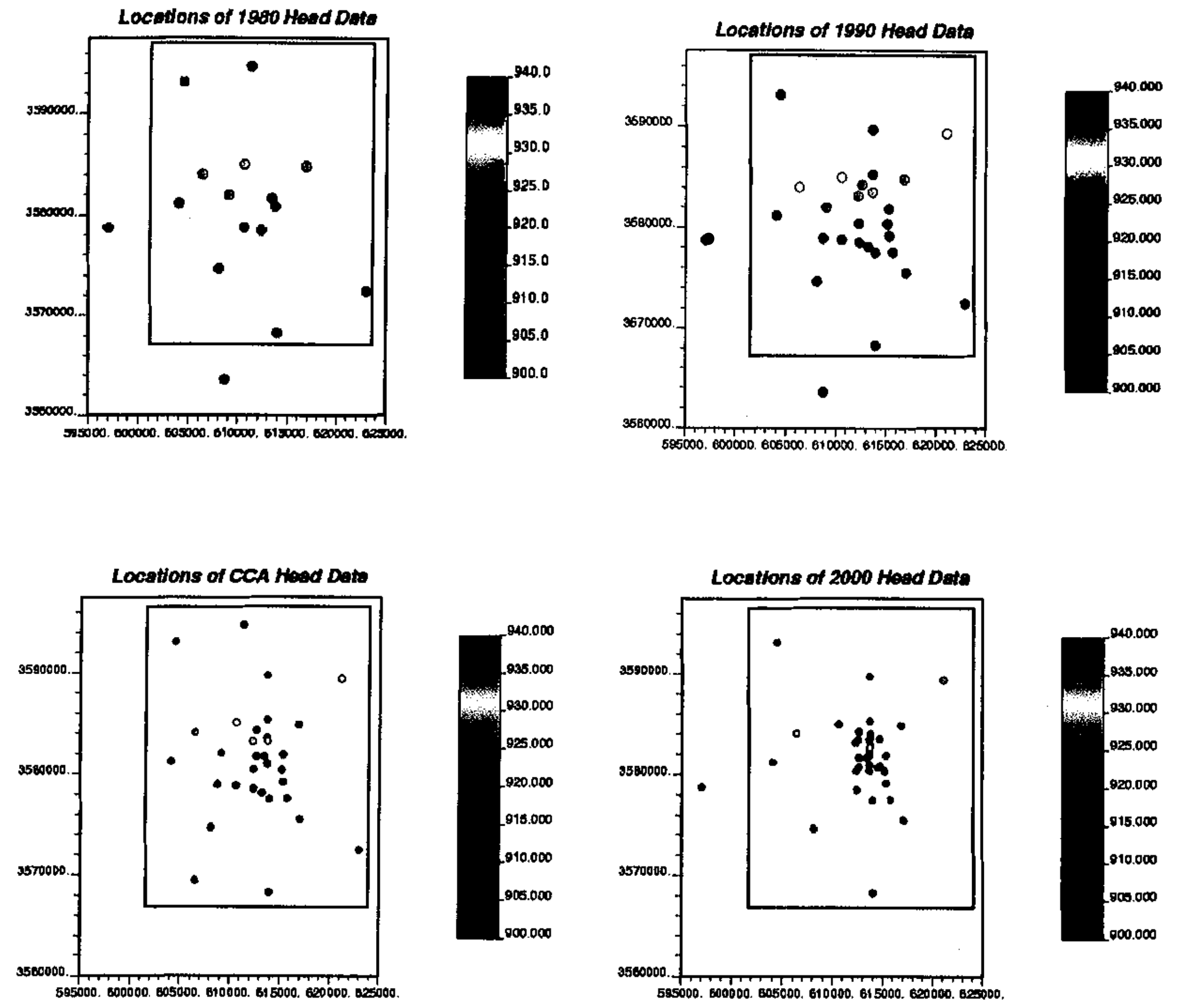


Figure 3. Locations and values of the head measurements for each of the four sets of heads considered in the steady-state calibrations. The approximate extent of the numerical model domain is shown by the black rectangle in each image.

The fit of the Gaussian trend surface to each set of heads is shown in Figure 4. From Figure 4, the fits to the different data sets are all similar with the exception of the CCA head data where the Gaussian trend surface resembles a planar function.

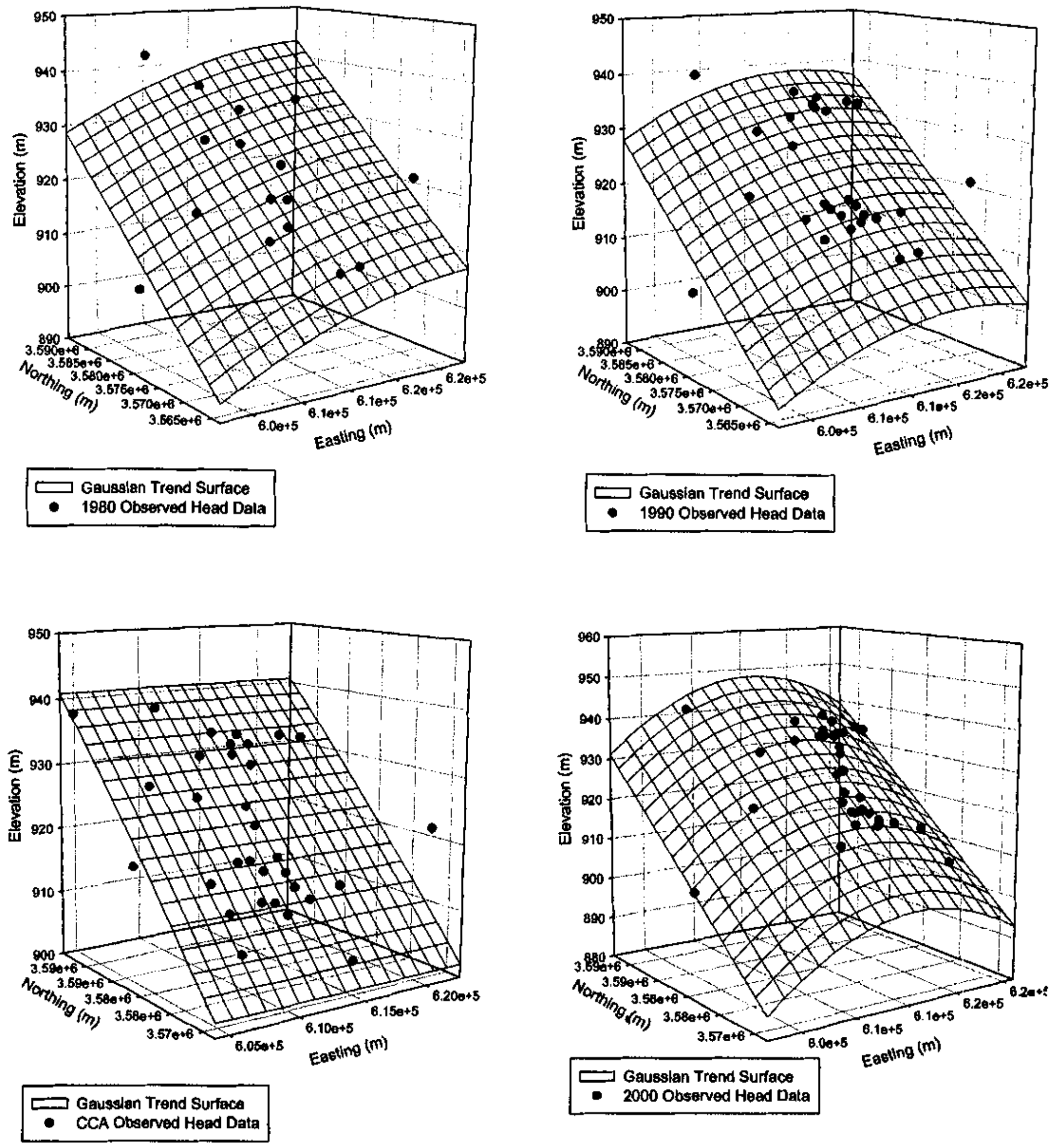


Figure 4. Gaussian trend surface fits to the observed data for the four different sets of heads.

The locations and values of the residuals (observed value – trend surface estimate) are shown in Figure 5.

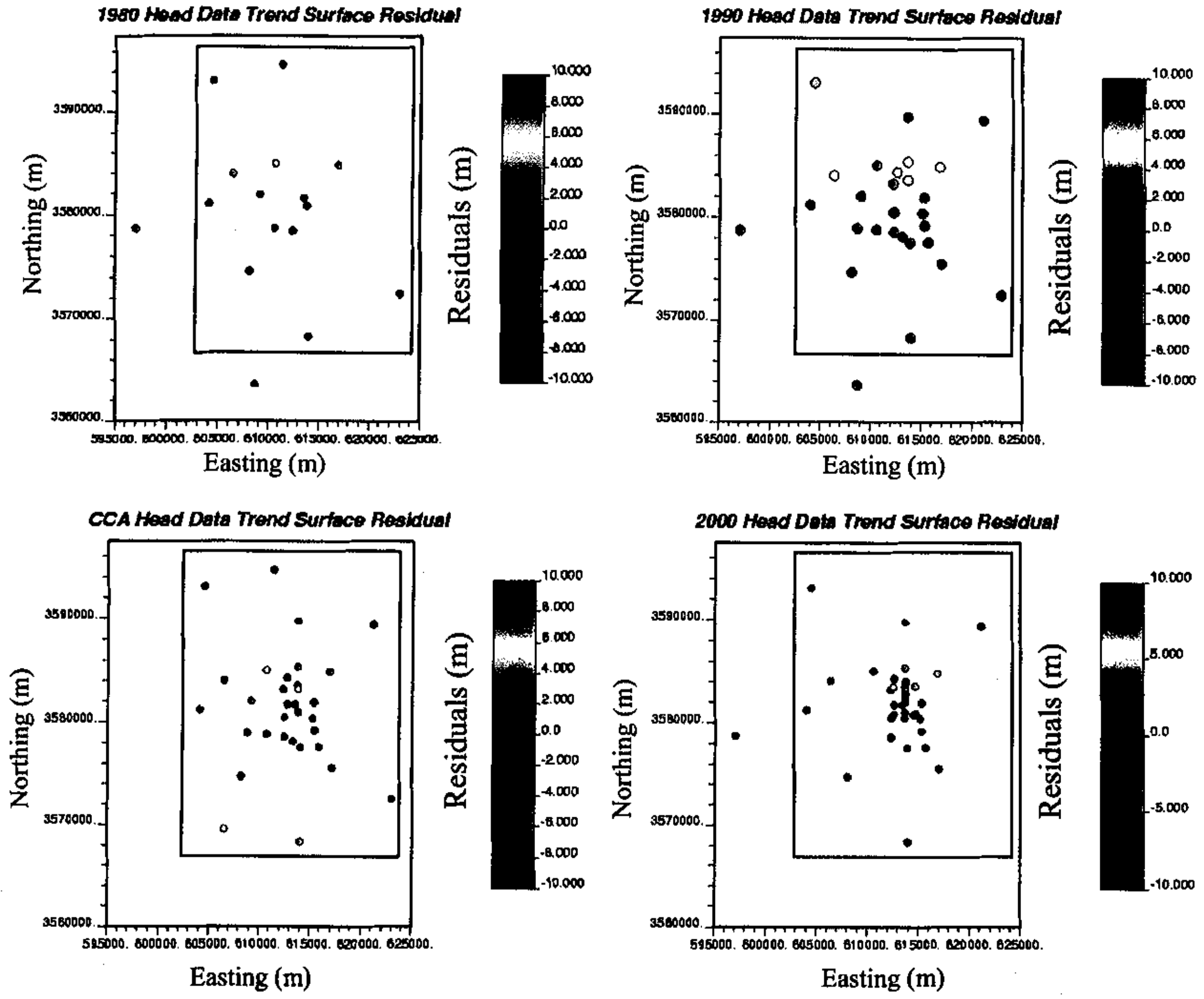


Figure 5. Locations and values of the residuals between the Gaussian trend surface model and the observed head data for each of the four sets of heads. The approximate boundary of the flow model is shown as a black rectangle in each image.

The next step in estimating the initial head values is to calculate an experimental variogram for each set of residuals and then fit a variogram model to each experimental variogram. Due to the rather limited number of data points, anisotropy in the spatial correlation of the residuals was not examined and an omnidirectional variogram was calculated for each set of residuals. These calculations were done using the VarioWin (version 2.21) software (Pannatier, 1996). To maintain consistency across the different time periods, a Gaussian variogram model was used to fit all of the experimental variograms. The Gaussian variogram model is:

$$\gamma(h) = C\left[1-\exp^{\left(-\frac{3h^2}{a^2}\right)}\right] \qquad \text{for } h > 0$$

where C is the sill of the variogram , h is the distance between any two samples, or the lag spacing, and a is the practical range of the variogram, or the distance at which the model reaches 95 percent of the value of C. In addition to the sill and range, the variogram model may also have a non-zero intercept with the gamma (Y) axis of the variogram plot known as the nugget. Due to numerical instabilities in the kriging process associated with the Gaussian model without a nugget value, a small nugget was used in fitting each of the variogram models. The model variograms were fit to the experimental data (Figure 6) and the parameters of these models are given in Table 5.

Table 5. Model variogram parameters for the head residuals.

Parameter	**1980**	**1990**	**CCA**	**2000**
Sill	26.0	38.0	29.0	22.0
Range (meters)	4800	5100.0	4100.0	3000.0
Nugget	2.0	1.0	1.5	4.5
Number of Data	16	28	34	37

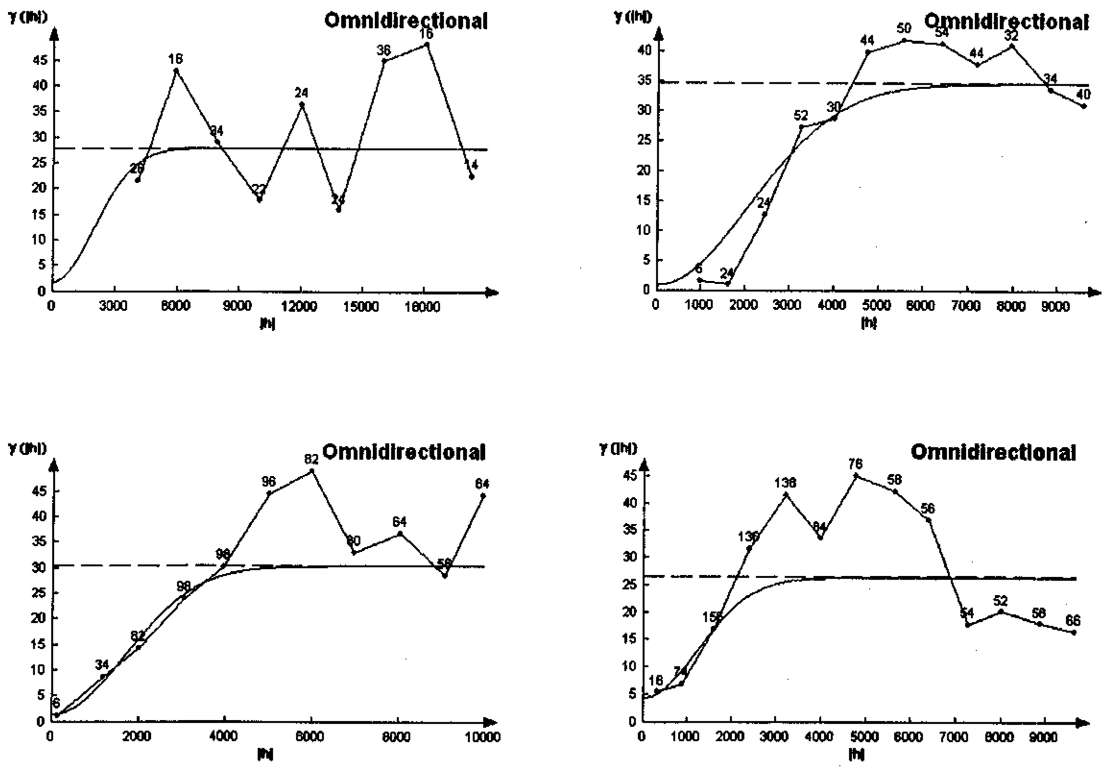


Figure 6. Omnidirectional experimental (straight-line segments) and model variograms of the head residuals (curves) for the four sets of heads: 1980 (upper left), 1990 (upper right), CCA (lower left) and 2000 (lower right). The numbers indicate the number of pairs of values that were used to calculate each point and the horizontal dashed line denotes the variance of each residual data set.

Figure 6 shows that the experimental variograms are well approximated by the Gaussian model for the 1990 and CCA data. The 1980 data set does not have enough data (16) to yield a good fit using any type of model. The experimental variogram calculated on the 2000 data shows a number of points between lags 2000 and 7000 meters that are above the variance of the data set (the horizontal dashed line). This behavior indicates that the Gaussian trend surface model used to calculate the residuals from the measured data did not remove the entire trend inherent in the observed data. A higher order trend surface model could be applied to these data to remove more of the trend, but we have chosen to keep the trend surface model consistent across all four data sets and feel that the Gaussian trend surface model provides a reasonable estimate of the trend in the data across all four sets.

The GSLIB kriging program **kt3d** is used to estimate the residual values at all points on the grid within the model domain. The results of this kriging program are then used as input to the code **add_trend**. The **add_trend** code adds the Gaussian trend surface to the estimated residual values to produce the final estimates of the initial head field. A slightly different version of the **add_trend** code was used for each time period, where the Gaussian trend surface parameters in Table 4 are hard-coded variables for each different time period. The **add_trend** source code for the 1980 calculations is included as Appendix 3.

FIXED-HEAD BOUNDARY AND INITIAL CONDITION RESULTS

The results of this kriging process with consideration of the trend are four sets of initial heads for use in MF2K. The values of the initial heads that correspond to the fixed boundary condition locations provide the boundary conditions for the calibration models. The initial (starting) head fields are shown in Figure 7 and the head values along each boundary of the model domain are shown in Figures 8 and 9. Note that these final head plots are for the model domain and do not represent heads along the no-flow boundary that is imposed on the problem later.

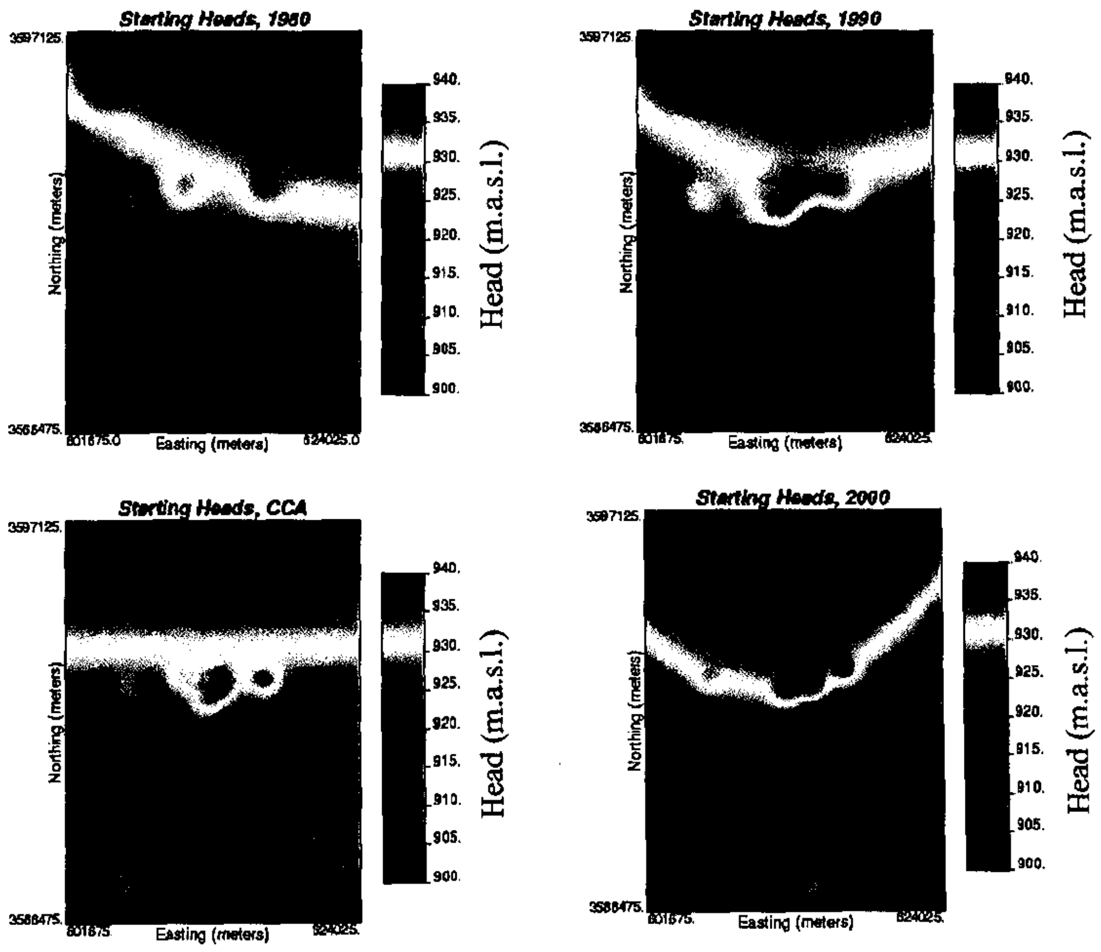
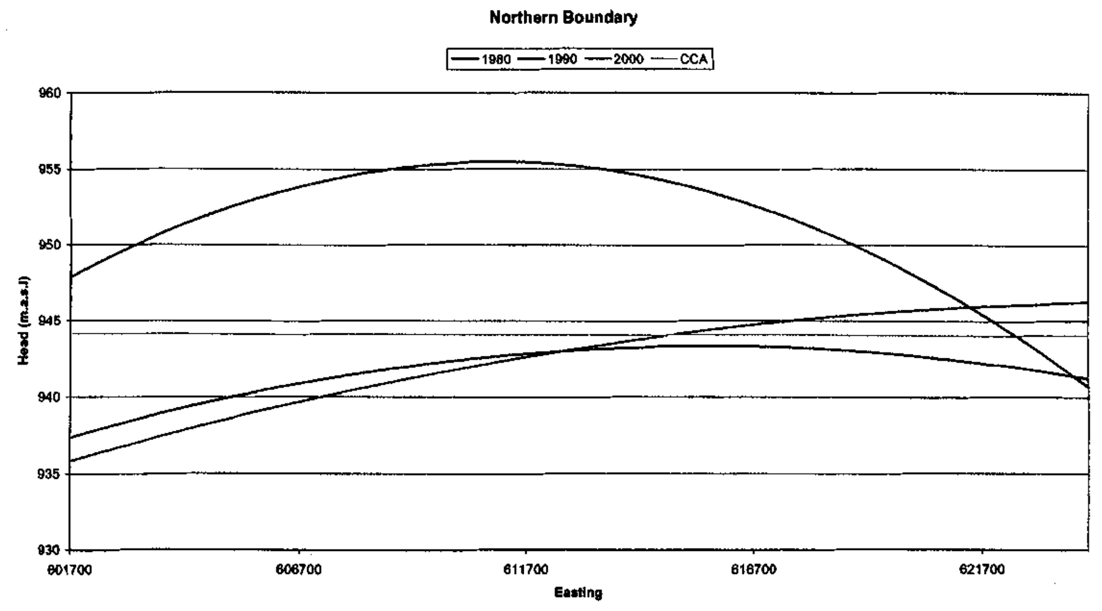


Figure 7. Initial heads for the four differentsets of heads. These four images show the extent of the model domain only.



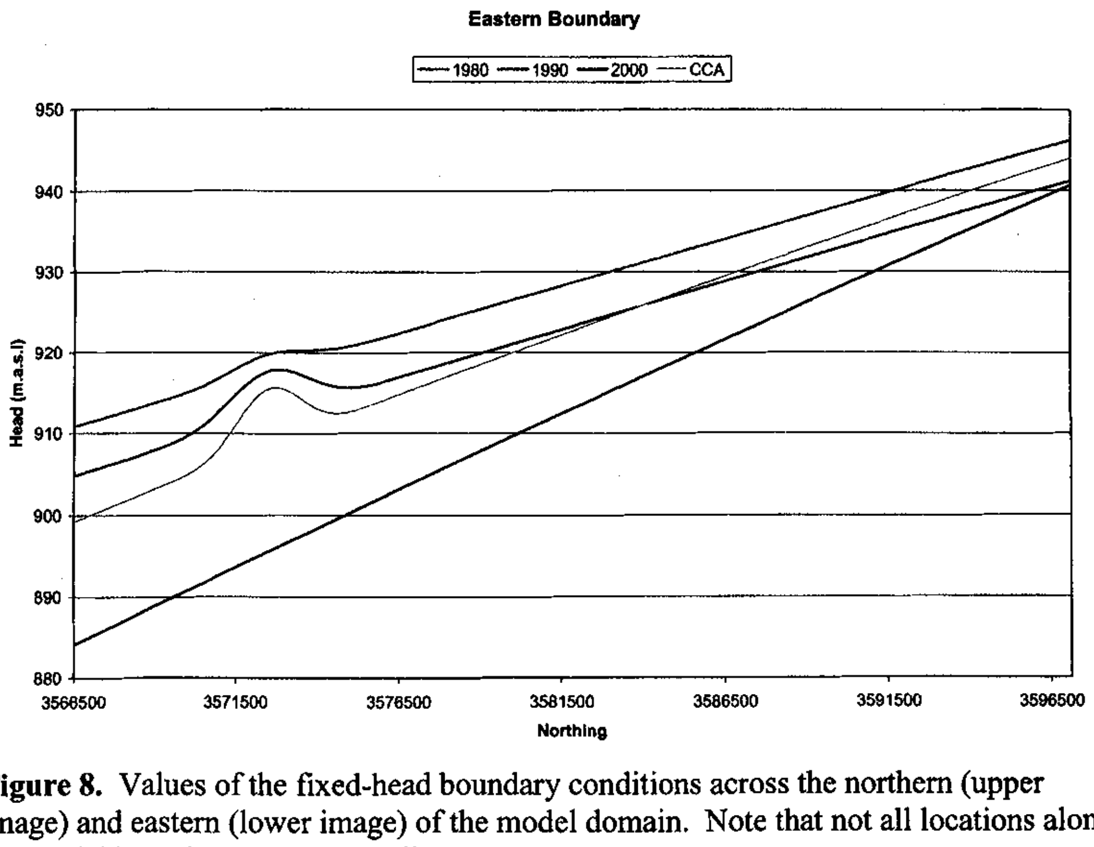
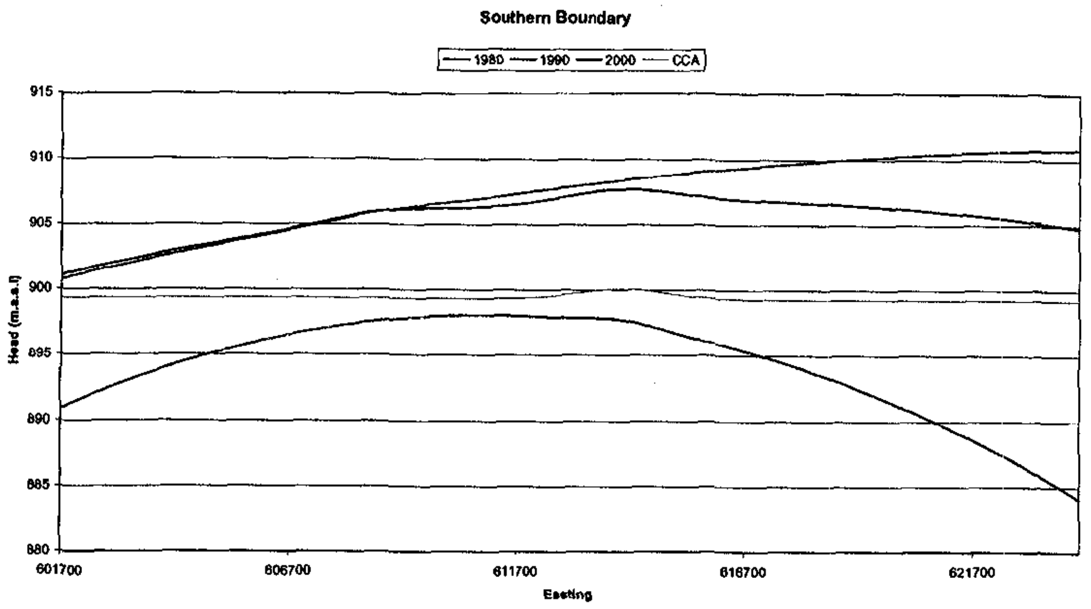


Figure 8. Values of the fixed-head boundary conditions across the northern (upper image) and eastern (lower image) of the model domain. Note that not all locations along the model boundary are active cells.



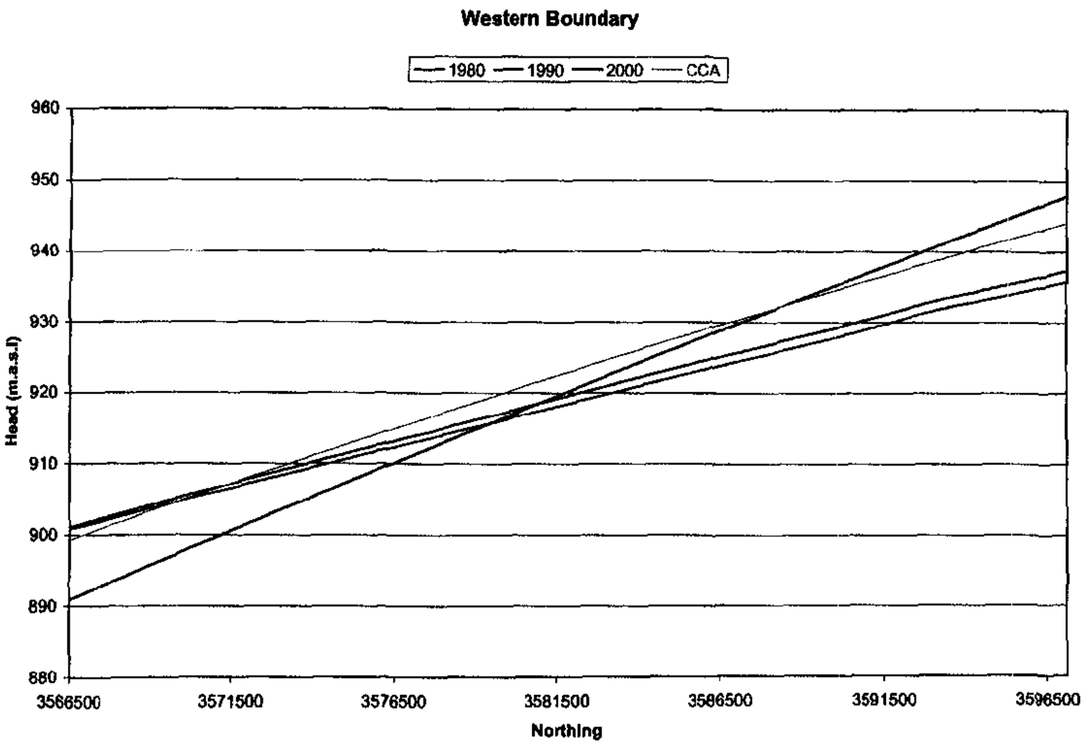


Figure 9. Values of the fixed-head boundary conditions across the southern (upper image) and western (lower image) of the model domain. Note that not all locations along the model boundary are active cells.

Creation of Seed Transmissivity Fields

The base transmissivity fields created in Task 2 (Holt and Yarbrough, 2002) rely on a regression model to estimate transmissivity at every location. By the nature of regression models, the estimated transmissivity values will not honor the measured transmissivity values at the measurement locations. Therefore, before using these base transmissivity fields in a flow model, they must be conditioned to the measured transmissivity values. This conditioning is performed with a Gaussian geostatistical simulation algorithm to generate a series of 100 spatially correlated residual fields where each field has a mean value of zero. These fields are conditional such that the residual value at each measurement location is the same in each realization and the residual value plus the base transmissivity field is equal to the known transmissivity value at all measurement locations. The result of adding the simulated residual field to the base transmissivity field is the "seed" realization.

This process is shown conceptually along a west to east cross section of the Culebra in Figure 10. The upper image shows the value of the residuals at five T measurement locations across the cross section. These residuals are calculated as the observed (measured) T value minus the base field T value at the same locations. Postive residuals are where the measured T value is greater than that of the base T field. To create a transmissivity field from these residuals, there needs to be a way to tie the base field to the measured transmissivity values. This tie is accomplished by creating a spatial simulation of the residual values, a "residual field". The middle image of Figure 10 is an example residual field as a red dashed line along the cross-section. This residual field is constructed through geostatistical simulation using a variogram model fit to the residual data. The residual field honors the measured residuals at their measurement locations and returns to a mean value of zero at distances far away from the measurement locations. Finally, this residual field is added to the base transmissivity field to create the seed transmissivity field. The base T field is represented by the solid blue line in the bottom image of Figure 10 and the seed T field is shown by the dotted line. The seed T field corresponds to the base T field except at those locations where it must deviate to match the measured T data. The large discontinuity shown in the base T field at the bottom of Figure 10 is due to the stochastic simulation of high-T zones within the Culebra (Holt and Yarbrough, 2002).

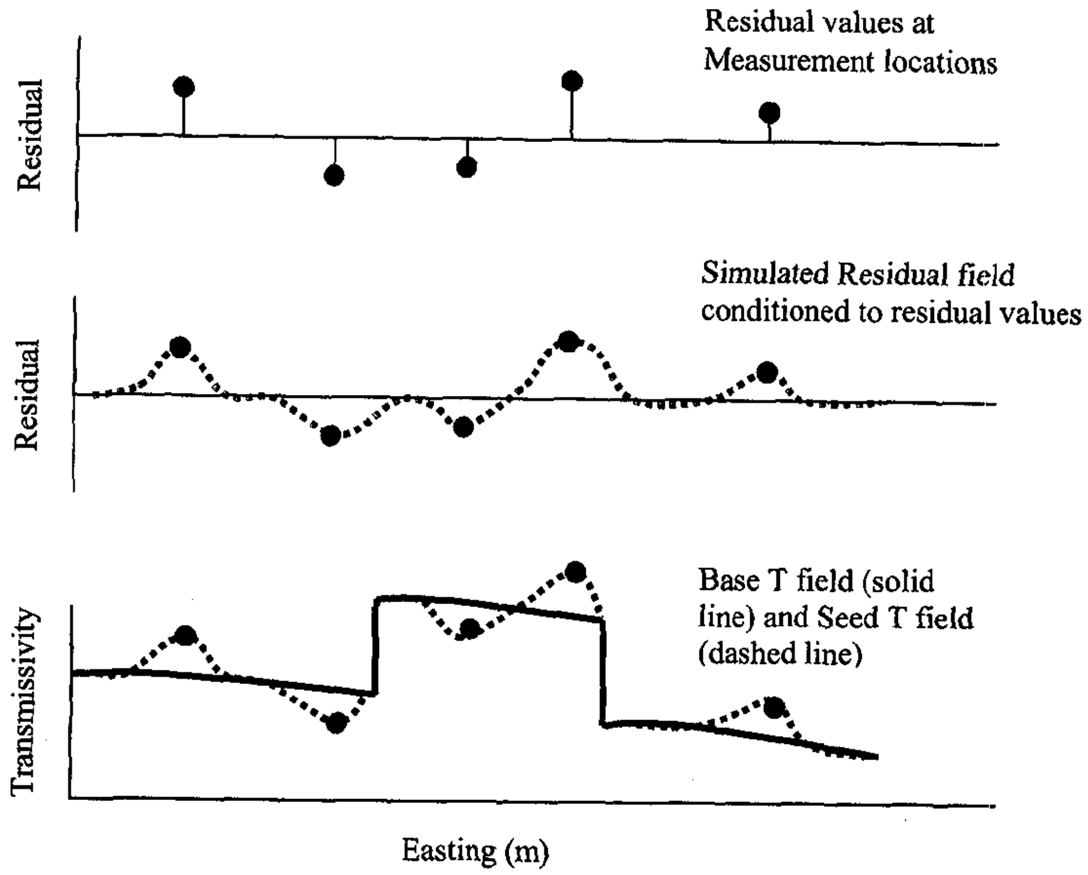

Residual values at Measurement locations
Residual
Simulated Residual field conditioned to residual values
Residual
Base T field (solid line) and Seed T field (dashed line)
Transmissivity
Easting (m)

Figure 10. Conceptual cross section showing the updating of the residual field and the base T field into the seed T field.

A total of 46 measured transmissivity values and corresponding residual data, both in units of log10 (m^2/s), are available (Holt and Yarbrough, 2002, ERMS# 523889). These data are shown in Table 6. For each pair of log10 T and residual data, the well name and the easting (X) and northing (Y) coordinates in UTM are also given. These data are contained in the Excel file *residuals.xls* (converted from the residuals.dat file from Holt and Yarbrough, 2002) included on the CD-ROM as part of this analysis package. Note that the number and locations of the transmissivity data are not the same as the number and location of head data for any of the four data sets.

The process of creating the residual fields is to use the residual data to generate variograms in the **VarioWin** software package and to then create conditional stochastic Gaussian geostatistical simulations of the residual field within the GSLIB program **sgsim**. To render the data set amenable to the variogram calculations in **VarioWin**, the data coordinates were translated by subtracting a value of 600,000 from each Easting coordinate and a value of 3,500,000 from each Northing coordinate. This translation was accomplished in the *residuals.xls* file and the locations of the well data both before and after the translation are shown in Figure 11.

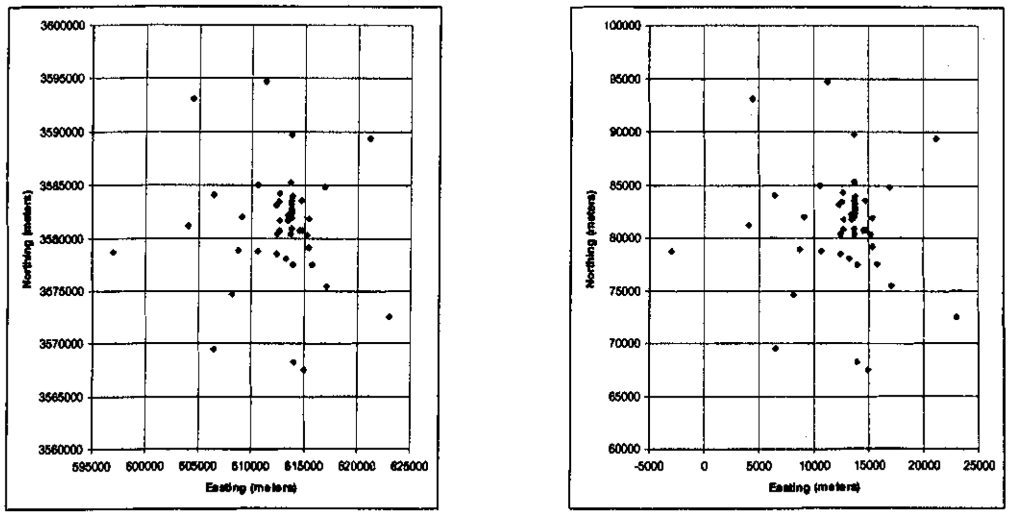


Figure 11. Locations of log10 T and residual data before (left) and after (right) translation to a temporary new coordinate system for variogram modeling.

Table 6. Log10 transmissivity data used in inverse calibration for all four data sets.

Well	Easting	Northing	log10 T	log10 T residual	Translated Easting	Translated Northing
ID	(UTM)	(UTM)	(m^2/s)	(m^2/s)	(UTM)	(UTM)
P-15	610624	3578747	-7	-0.95938	10624	78747
H-19b0	614514	3580716	-5.2	-0.62242	14514	80716
Engle	614953	3567454	-4.3	-0.51632	14953	67454
WIPP-12	613710	3583524	-7	-0.39627	13710	83524
WIPP-30	613721	3589701	-6.7	-0.35131	13721	89701
CB-1	613191	3578049	-6.5	-0.32943	13191	78049
WQSP-6	612605	3580736	-6.6	-0.32261	12605	80736
WQSP-4	614728	3580766	-4.9	-0.28895	14728	80766
H-14	612341	3580354	-6.5	-0.26934	12341	80354
H-9c	613974	3568234	-4	-0.22763	13974	68234
H-3b1	613729	3580895	-4.7	-0.22131	13729	80895
DOE-1	615203	3580333	-4.9	-0.21004	15203	80333
WQSP-3	614686	3583518	-6.8	-0.15139	14686	83518
WIPP-28	611266	3594680	-3.6	-0.15124	11266	94680
H-17	615718	3577513	-6.6	-0.14310	15718	77513
H-15	615315	3581859	-6.8	-0.12631	15315	81859
WIPP-29	596981	3578694	-3	-0.12497	-3019	78694
WIPP-21	613743	3582319	-6.6	-0.11148	13743	82319
AEC-7	621126	3589381	-6.8	-0.11078	21126	89381
H-12	617023	3575452	-6.7	-0.07647	17023	75452
WIPP-27	604426	3593079	-3.3	-0.03209	4426	93079
WQSP-2	613776	3583973	-4.7	-0.02729	13776	83973
H-6c	610610	3584983	-4.4	-0.01524	10610	84983
H-10b	622975	3572473	-7.4	-0.01484	22975	72473
WIPP-25	606385	3584028	-3.5	-0.01378	6385	84028
WQSP-1	612561	3583427	-4.5	0.01540	12561	83427
H-5c	616903	3584802	-6.7	0.02946	16903	84802
H-4c	612406	3578499	-6.1	0.05221	12406	78499
WIPP-18	613735	3583179	-6.5	0.06840	13735	83179
WIPP-22	613739	3582653	-6.4	0.10549	13739	82653
H-2c	612666	3581668	-6.2	0.13594	12666	81668
ERDA-9	613696	3581958	-6.3	0.15250	13696	81958
P-14	609084	3581976	-3.5	0.16212	9084	81976
WIPP-26	604014	3581162	-2.9	0.21598	4014	81162
P-17	613926	3577466	-6	0.24762	13926	77466
H-11b4	615301	3579131	-4.3	0.25314	15301	79131
D-268	608702	3578877	-5.7	0.27914	8702	78877
USGS-1	606462	3569459	-3.3	0.28998	6462	69459
WIPP-19	613739	3582782	-6.2	0.32598	13739	82782
H-16	613369	3582212	-6.1	0.34962	13369	82212
H-7c	608095	3574640	-2.8	0.39794	8095	74640
H-1	613423	3581684	-6	0.41295	13423	81684
WIPP-13	612644	3584247	-4.1	0.42180	12644	84247
WQSP-5	613668	3580353	-5.9	0.47178	13668	80353
DOE-2	613683	3585294	-4	0.69492	13683	85294
H-18	612264	3583166	-5.7	0.73159	12264	83166

To use the data in a Gaussian simulation algorithm, it is first necessary to transform the distribution of the raw residual data to a standard normal distribution. This is accomplished through a process called the "normal-score transform" where each transformed residual value is the "normal-score" of each original datum. The normal-score transform is a relatively simple two-step process. First the cumulative frequency of each original residual value, *cdf(i)*, is determined as:

$$cdf(i) = \frac{R(i) - 0.5}{N}$$

where *R(i)* is the rank (smallest to largest) of the i^{th} residual value and N is the total number of data (46 in this case). Then for each cumulative frequency value, the corresponding normal-score value is calculated from the inverse of the standard normal distribution. By definition, the standard normal distribution has a mean of 0.0 and a standard deviation of 1.0. Further details of the normal-score transform process can be found in Deutsch and Journel (1998).

The two-step normal-score transformation process is conducted in the Microsoft Excel spreadsheet file *residuals.xls*. First, the residual data are rank-ordered (smallest to largest) using the Sort command in Excel. Then the cumulative frequency of each datum is calculated from the preceding equation and finally the Excel NORMSINV() function is used to determine the normal-score of the datum. The resulting normal-score values are the distance from the mean as measured in standard deviations. The parameters describing the residual and normal-score transformed distributions are presented in Table 7.

Table 7. Statistical parameters describing the distributions of the raw and normal-score transformed residual data.

Parameter	Raw Residual Data	Normal-Score Transformed Residual Data
Mean	0.000	0.000
Median	-0.015	0.000
Standard Deviation	0.330	0.997
Minimum	-0.959	-2.295
Maximum	0.732	2.295

The normal-score residual data in the *residual.xls* file are copied into a text file used as input to the **VarioWin** variogram modeling software (Pannatier, 1996). The text file contains the translated Easting and Northing coordinates, the log10 T data, the log10 residual data and the normal-score transform of the log10 residual data. This new text file, *resid_ns.dat*, is also included on the CD-ROM as part of this analysis package. A five-line text header is added to this file to put it into the format required by **VarioWin**.

The omnidirection variogram is calculated with a 250-meter lag spacing. The experimental variogram is shown in Figure 12:

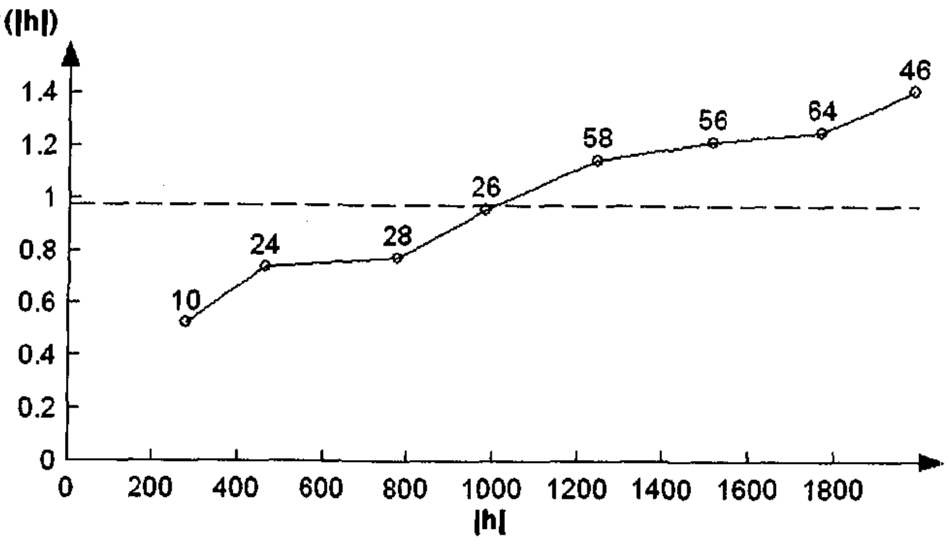


Figure 12. Experimental normal-score variogram of the transmissivity residuals. The numbers indicate the number of pairs of data compared to calculated each point of the variogram.

The model fit to this experimental variogram is Gaussian with a nugget of 0.2, a sill of 0.8 and a range of 1050 meters (Figure 13). The sum of the nugget and sill values is constrained to equal the theoretical variance of 1.0 by the **sgsim** (Deutsch and Journel, 1998) software that is used to create the spatially correlated residual fields.

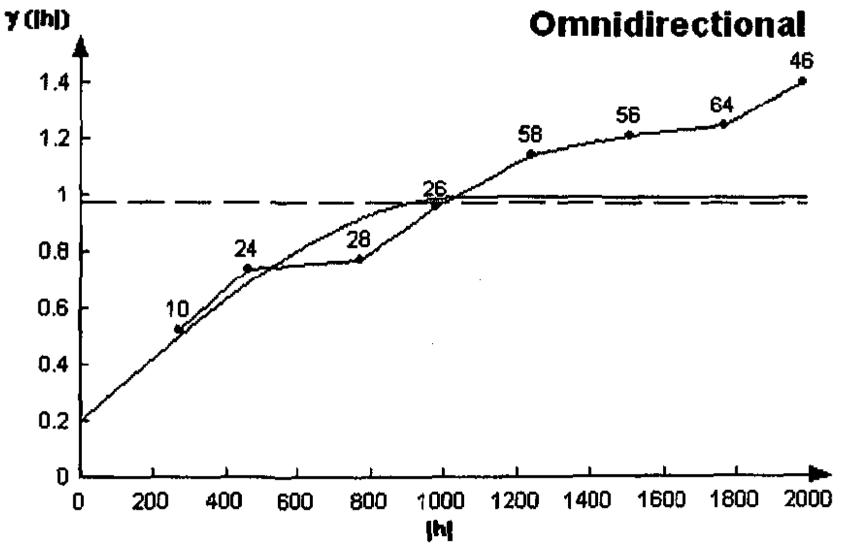


Figure 13. Omindirectional variogram model fit to the experimental variogram of the transmissivity residuals

The initial residual field is created through a stochastic geostatistical simulation process using the variogram calculated on the normal-score transformed residual values (Figure 13). Updates to this initial residual field are performed in the inverse modeling with an estimation (kriging) algorithm. Therefore it is necessary to calculate and model a variogram on the raw, not normal-score transformed, residuals for use in this kriging process. This variogram was also calculated with a 250-meter lag and is omnidirectional. A doubly nested spherical variogram model is fit to the experimental variogram. The variogram parameters are a nugget of 0.008, a first sill and range of 0.033 and 500 meters and a second sill and range of 0.067 and 1500 meters (Figure 14). This variogram model is used by PEST to propagate any pertubation to the original residual field made at the pilot point locations to the neighboring model grid cells.

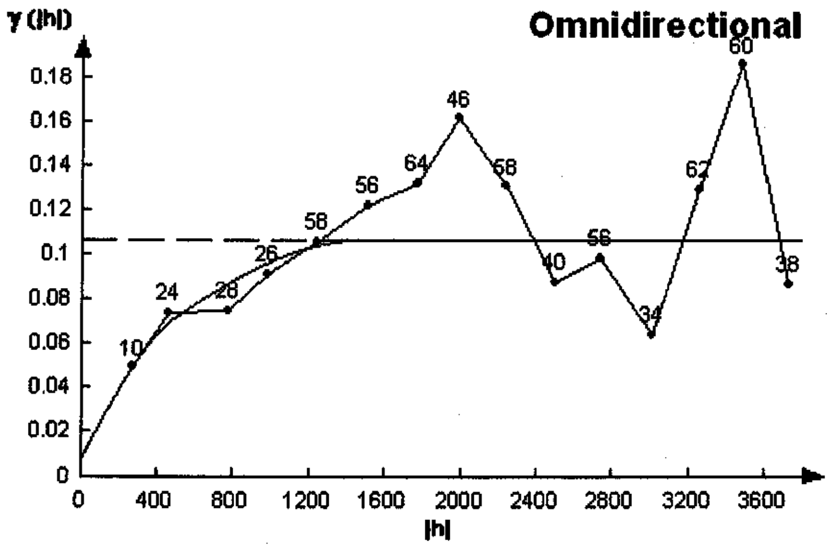


Figure 14. Experimental and model variograms for the raw-space (not normal-score transformed) transmissivity residual data.

The variogram parameters for the normal-score transformed residuals are used directly in the **sgsim** program to create 100 conditional realizations of the residual field. Each of these 100 residual fields is used as an initial residual field and each one is assigned to an individual base transmissivity field. An example of a realization of the residual field and its combination with a base transmissivity field is shown in Figure 15. From Figure 15, the effect of the residual field on the base transmissivity field can be seen. The residual field perturbs the transmissivities to match the measured transmissivities at the well locations. The discrete features that are part of the original base transmissivity field (e.g., high-transmissivity zones in the middle of the domain) are retained when the residual field is added to the base field.

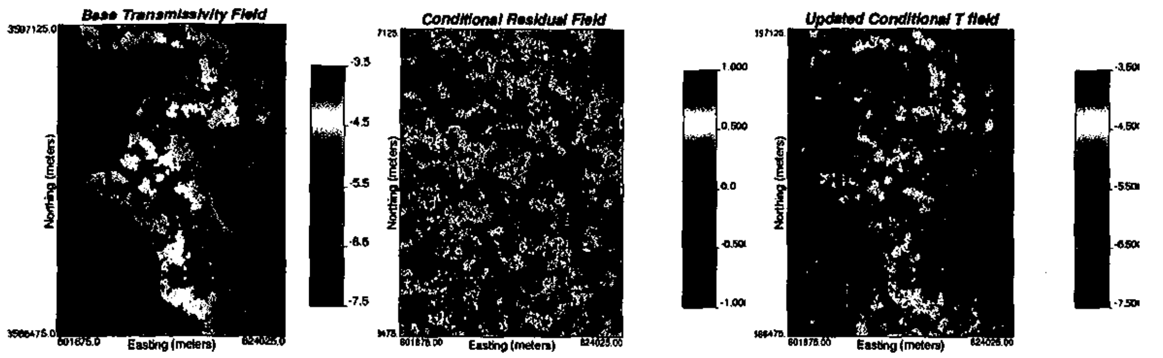


Figure 15. An example of the initial steps in creation of the calibrated transmissivity field. The base transmissivity field (left image) is combined with the initial residual field created through geostatistical simulation (center image) to produce the transmissivity field (right image). All three color scales denote the log10 (m^2/s) transmissivity value.

Subtask 3: Forward Modeling

As an initial test of the available data, boundary conditions, and the flow model setup, the seed realizations (combination of a base transmissivity field with an initial residual field) were used in a set of forward models. These forward models are not calibrated to the observed head data. Heads, fluxes, and particle travel times from these forward models are retained for comparison with the results obtained after the inverse modeling step.

FORWARD SIMULATIONS ON BASE TRANSMISSIVITY FIELDS

The initial and boundary head values generated in the previous step are used as input to **MF2K** for simulation of groundwater flow in the original base transmissivity fields created by Holt and Yarbrough (2002). These simulations are forward runs only and there is no calibration of the fields to match observed heads.

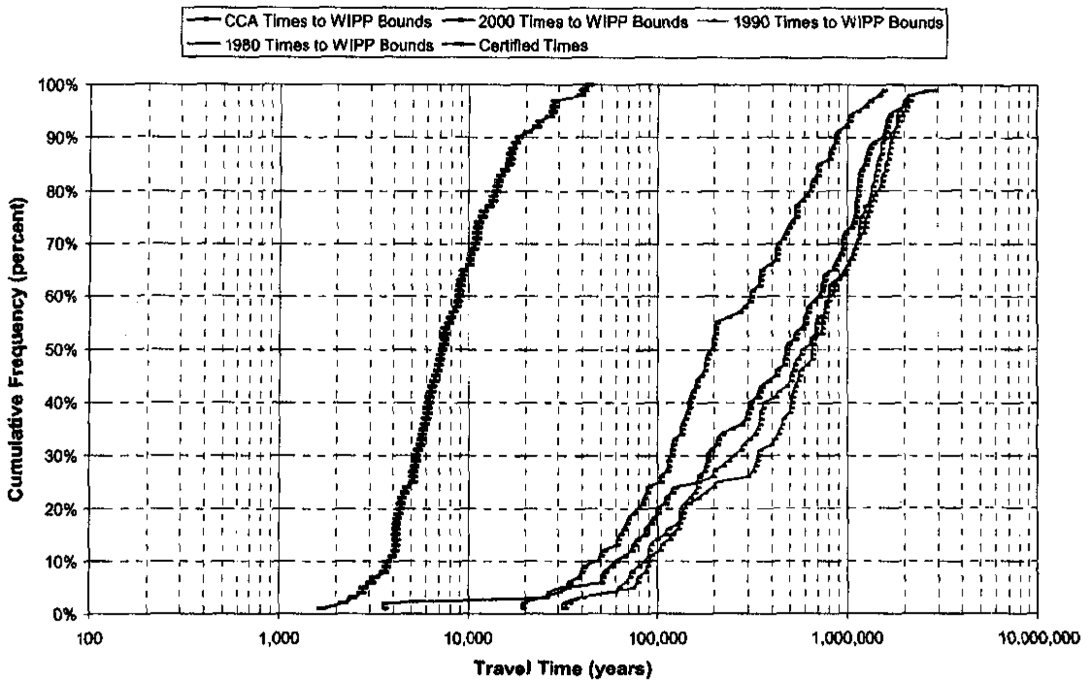
The first step in the forward modeling process is to create 100 subdirectories with the naming convention: /real### with "###" ranging from 001 to 100. These subdirectories are created from a base directory containing the generic input files for **MF2K** and the streamline particle-tracking code **DTRKMF**. The code **xform** (Appendix 4) is used to reformat each of the base transmissivity fields from the four-column ascii format supplied by Holt and Yarbrough (2002) to the **MF2K** format and these files are copied to each realization subdirectory and named: *base_tfield_name.trans* where "*base_tfield_name*" is the file name of the base transmissivity field (e.g., *b01r02, which is the 2nd realization for the first set of ten fields*).

The forward modeling is performed for a single mean transmissivity field using a shell script that runs **MF2K** and **DTRKMF** with all four initial heads and boundary conditions (1980, 1990, CCA and 2000). There are 100 shell scripts needed for the forward simulations – one for each base transmissivity field. An example shell script for realization number three is given in Appendix 5. The only differences between the 100 shell scripts are the names of the input and output files and the subdirectories in which they reside.

The modeling process begins by setting up **MF2K** for each of the four different data sets with the appropriate initial and fixed-head boundary values. For each of thedata sets, the same base transmissivity field is input to **MF2K** resulting in a single flow solution for each data set for each base transmissivity field. The resulting heads are saved to the **.lst* file and **DTRKMF** is run to track a single particle from the starting location (shown in Figure 1) to the WIPP boundary. The **DTRKMF** output is reformatted using the intrinsic UNIX command language "*awk*" for visualization in the UNCERT program and saved to the **.lbl* file. The **get-heads** program (Appendix 6) is used to extract the modeled heads at the well locations from the **.lst* file. The modeled and observed heads are written to an output file (e.g., *calc_heads_b01r03.out*).

FORWARD SIMULATION RESULTS

For each of the 400 forward runs, there are two results that are saved to files: the calculated heads at each of the observed head locations, and the information on the particle track from the starting point to the point where it exits the WIPP boundary. These results are summarized in Figure 16.



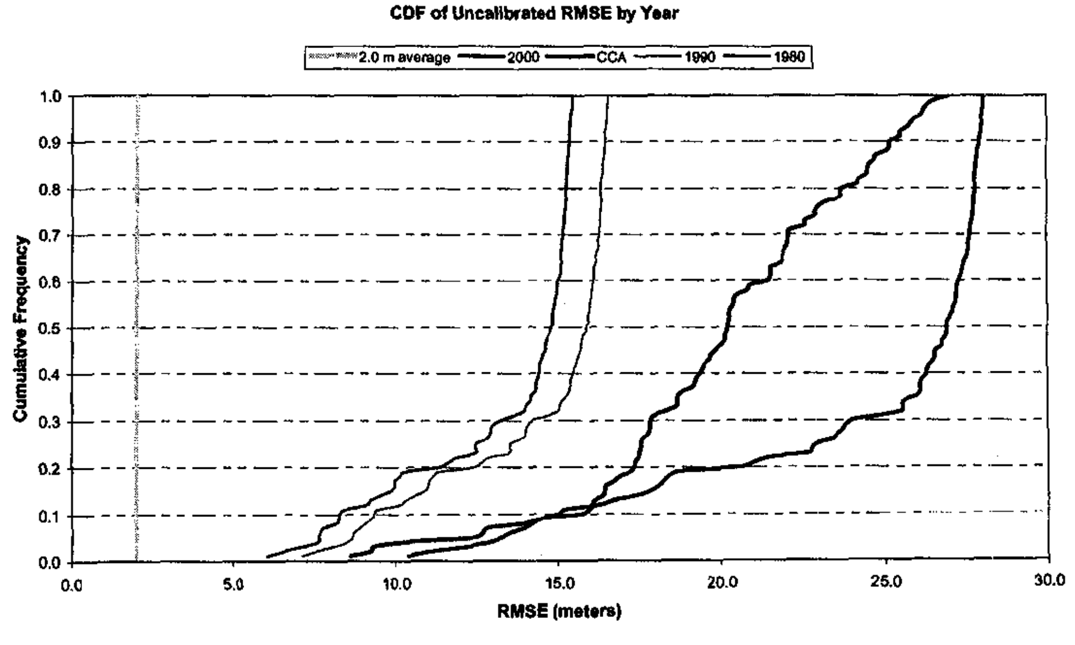


Figure 16. Results of the forward simulations foreach data set. The particle travel times to the WIPP boundary are shown in the upper image and the RMSE values between the measured and modeled heads are shown in the lower image.

The cumulative distribution function (cdf) of the particle travel times to the WIPP boundary is shown for each data set in Figure 16 (upper image). These cdfs are compared to the cdf of particle travel times from the calibrated transmissivity fields calculated for the CCA (Wallace, 1996). The cdf resulting from the calculations done for the CCA are shown as the "certified" times in Figure 16 to distinguish them from the results of the forward calculations made with the heads used in the CCA calculations. For comparison with the travel times calculated for the CCA (Figure 16, upper image), the travel times calculated as part of the current analysis have been reduced by a factor of 4.0/7.75 = 0.516. The current analysis used a thickness of 7.75 m (full Culebra thickness) because that is the average thickness contributing to T over the entire model domain. Results were scaled to a 4-meter thick Culebra to be consistent with the conservatism used in the CCA calculations. That conservatism was based on data from H-19 and elsewhere suggesting that most flow in high-T areas (but not necessarily low-T areas) is concentrated in the lower 4 m of Culebra.

In general, the heads collected at later time periods produce faster travel times with the 2000 heads producing significantly faster travel times than the other time periods. All of the times from the forward models are significantly longer than the times calculated as part of the CCA. However, the travel times for the CCA calculations are based on transmissivity fields calibrated to both steady-state and transient head data.

The heads resulting from the forward (uncalibrated) solution of the groundwater flow model are summarized for each realization as the root mean squared error (RMSE) between the calculated heads and the observed heads for all head observation points for that data set. The RMSE is:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n_{obs}} (H_i^{obs} - H_i^{calc})^2}{n_{obs}}}$$

where n_{obs} is the number of head observations for the data set and H^{obs} and H^{calc} are the values of the observed head and calculated head, respectively. The cdfs of the RMSE values for each of the four different data sets are shown in Figure 16 (lower image). For these forward runs, mismatch between the observed and calculated heads is expected and found to be quite high and the results in Figure 16 show a considerable amount of error. The RMSE values increase with time. The vertical line in Figure 16 (lower image) at an RMSE of 2.0 meters is given as a reference value based on the CCA calculations where the majority of the calibrated Tfields had heads that deviated from the observed heads by ± 2 meters.

Subtask 4: Steady-State Inverse Modeling

The base realizations created in Subtask 2 are input to the inverse model using the pilot point method. The number of pilot points and their locations are based on the results of Subtask 1 (Figure 1). The same 100 seed realizations are calibrated to each of the four different sets of steady-state head measurements. Results of this task include the T fields along with the heads and fluxes calculated on those T fields and data comparing the modeled and measured heads at the observation wells. The flow path and groundwater travel time for a particle released from a point above the center of the WIPP disposal panels to the WIPP site boundary has also been calculated for each T field. Ensemble average T fields show how the transmissivities vary across the four sets of steady-state calibrations. The cumulative distribution functions (cdfs) of travel times for each set of realizations and the cdfs of the head calibrations evaluated by the RMSE between observed and calculated heads are compared across the different time periods and to the CCA results.

The residuals and the T field calculations are done in log10 space so that a unit change in the residual equates to a one order of magnitude change in the value of the transmissivity. The initial values of the pilot points are equal to the value of the initial residual field at each pilot point location. The pilot points are constrained to have a maximum perturbation of $\pm$ 3.0 from the initial value except for those pilot points within the high-T zone in Nash Draw (Figure 1) (see Holt and Yarbrough, 2002) that are limited to perturbations of $\pm$ 1.0.

Figure 10 is updated as Figure 17 to show, conceptually, how the addition of two pilot points along the cross section can modify the residual field and then update the transmissivity field. The pilot points are shown as the open circles in Figure 17 and are used to modify the residual field before it is added to the base T field. Compare the shape of the dashed red and blue lines in Figure 17 to the same lines in Figure 10. The values of the residuals at the observation points are held fixed so any adjacent pilot points cannot modify them.

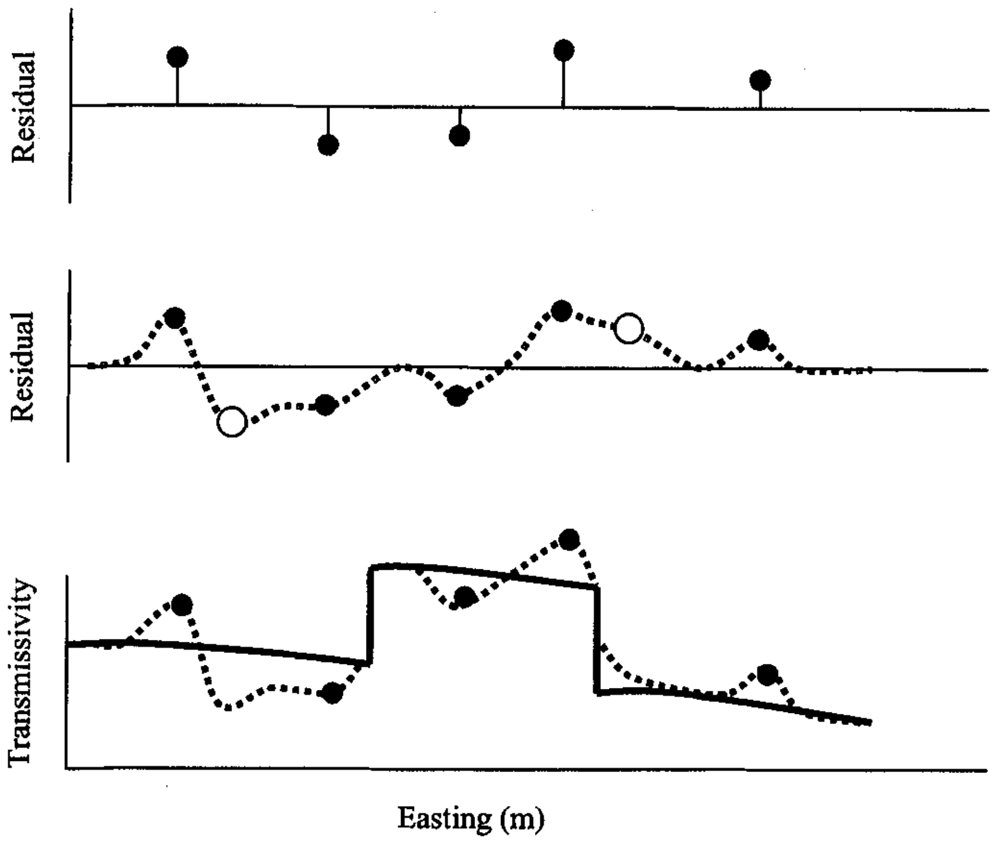


Figure 17. Conceptual cross section showing the addition of pilot points to the optimization process.

STOCHASTIC INVERSE CALIBRATION

The stochastic inverse calibration process uses multiple pre- and post-processor codes in addition to **PEST** and **MF2K**. The overall approach to the transmissivity field calibration is shown in Figure 18. The preprocessing steps from the top to the middle of Figure 18 are:

1) The conditional residual fields are created using the program **sgsim** (Deutsch and Journel, 1998).
2) The **get-data** code (Appendix 7) is used to specify the initial pilot point values by extracting the initial values of the residuals at the pilot point locations from the simulated residual field.
3) The initial pilot point values and the measured heads and transmissivities as well as the locations of these measurements are all entered into the **PEST** control file.
4) The code **ppk2fac** (Appendix 8) is part of the **PEST** software package that provides the transmissivity weighting factors for the locations surrounding each of the pilot points. These weighting factors are calculated from the variogram model information by invoking an assumption of second-order stationarity that specifies the spatial covariance as being the complement of the variogram. The covariance

function acts as a weight for updating transmissivity values surrounding a pilot point. For this analysis, the variogram of the raw residuals is used in **ppk2fac** to calculate the weighting factors. These weighting factors are stored in the file *factor.inf* and only need to be calculated once for each combination of pilot point locations, variograms, and grid size. The **ppk2fac** code also generates a table of standard deviations, and the algebraic regularization equations describing the relationship between pilot points. These equations are set up to minimize the weighted squared differences between pilot points where the weights are again based on the variogram model. The outputs of **ppk2fac** are the regularization equations that go into the control file and the kriging weight factors that are used by **PEST** in the optimization of the pilot points. More details on the mathematics used in **ppk2fac** are given in the **PEST** user's manual (Doherty, 2000).

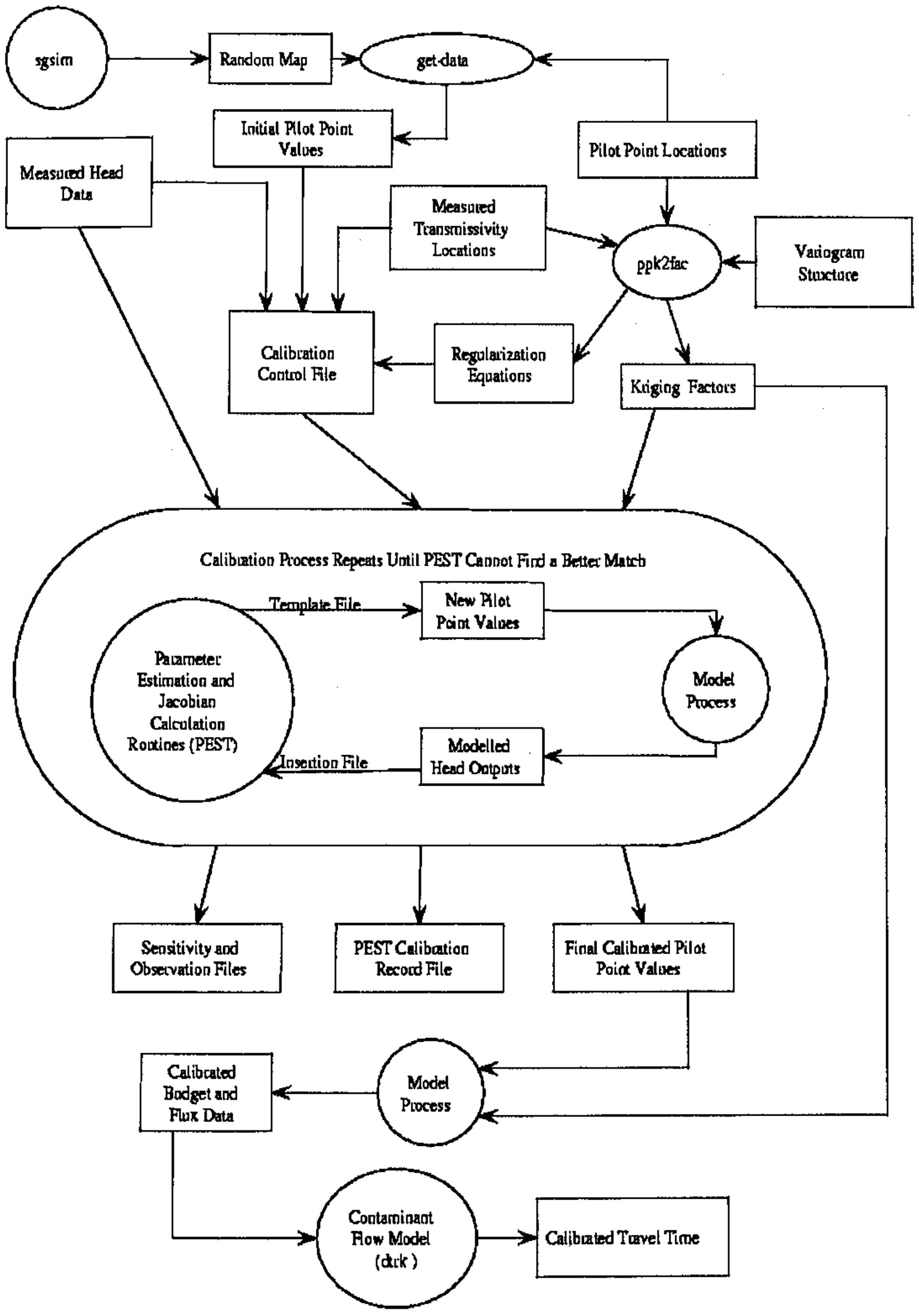
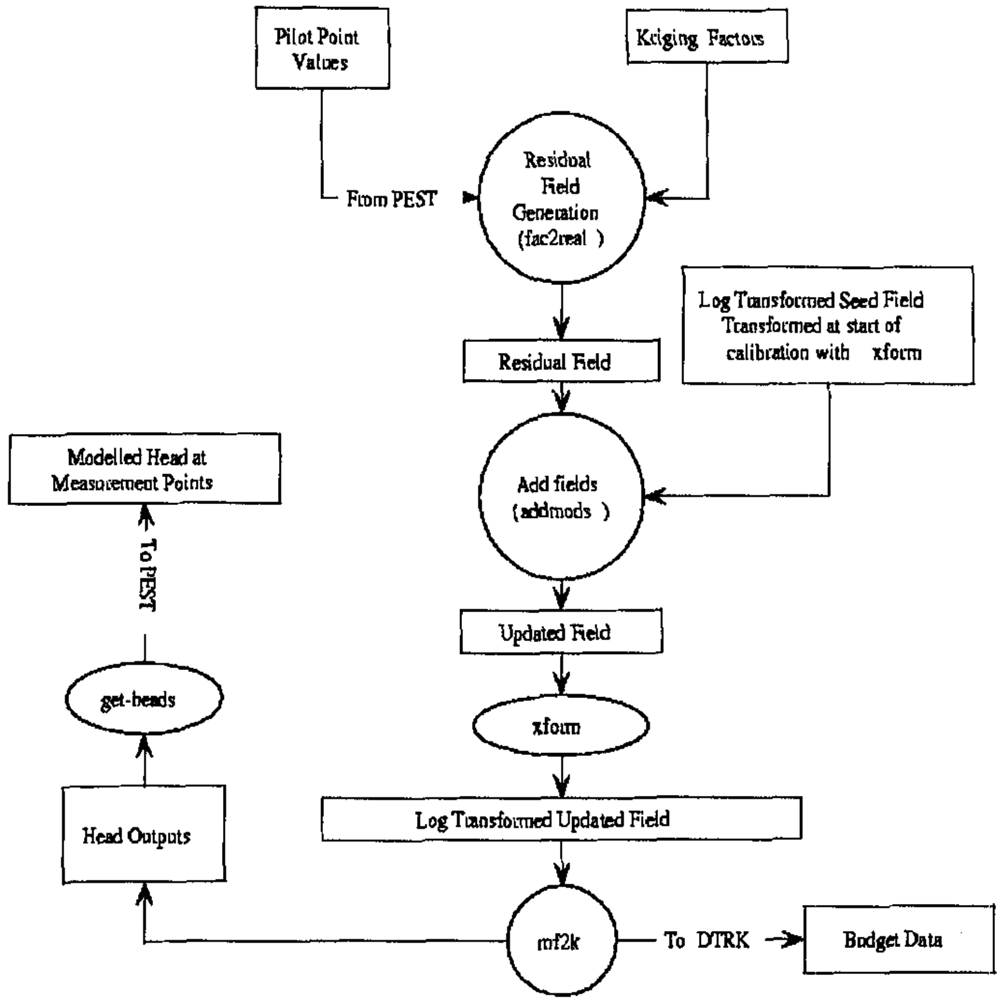

sgsim
Random Map
get-data
Initial Pilot Point Values
Pilot Point Locations
Measured Head Data
Measured Transmissivity Locations
ppk2fac
Variogram Structure
Calibration Control File
Regularization Equations
Kriging Factors
Calibration Process Repeats Until PEST Cannot Find a Better Match
Template File
New Pilot Point Values
Parameter Estimation and Jacobian Calculation Routines (PEST)
Model Process
Insertion File
Modelled Head Outputs
Sensitivity and Observation Files
PEST Calibration Record File
Final Calibrated Pilot Point Values
Calibrated Budget and Flux Data
Model Process
Contaminant Flow Model (dtrk)
Calibrated Travel Time

Figure 18. Flow chart of the steady-state stochastic inverse process used to create the calibrated transmissivity fields.

At the heart of the calibration process is the iterative adjustment of the residual field at the pilot points by **PEST** and the subsequent updates of the residual field at the locations surrounding the pilot points based on the shape of the variogram modeled on the raw residuals. The updated residual field is then combined with the base transmissivity field (see Figure 17) and then used in **MF2K** to calculate the current set of modeled heads. These modeled heads are then input to **PEST** for the next iteration. This process is shown as the "Model Process" within Figure 18 and is shown in detail in Figure 19.



Pilot Point Values
Kriging Factors
From PEST
Residual Field Generation (fac2real)
Log Transformed Seed Field Transformed at start of calibration with xform
Residual Field
Modelled Head at Measurement Points
To PEST
Add fields (addmods)
Updated Field
get-heads
xform
Head Outputs
Log Transformed Updated Field
mf2k
To DTRK
Budget Data

Figure 19. Flow chart of the core of the inversion process highlighting the connection between PEST and MODFLOW 2000.

Using Figure 18 as a guide, the steps in the calibration process are described as:

1) The initial pilot point values are obtained from the initial residual field. The value of the pilot points at locations that correspond to the actual transmissivity measurement locations are held as fixed values throughout the calibration process. The remaining 115 pilot point values will be adjusted by **PEST**. Forward from

the 2nd iteration, these initial pilot point values are those updated by **PEST** in the previous iteration.

The objective function minimized by **PEST** is a combination of the weighted sum of the squared residuals between the measured and observed head data and a second weighted sum of the squared differences in the estimated T value between pairs of pilot points.

$$\phi = \sum_{i=1}^{N_H^{obs}} W_{ii}^{H} (H_i^{obs} - H_i^{calc})^2 + \sum_{j=1}^{N_{PP}} \sum_{k=j}^{N_{PP}} W_{jk}^{R} (PP_j - PP_k)^2$$

The first weighted sum of squares is the measure of the difference between the measured, H_i^{obs}, and modeled, H_i^{calc}, head values. For this work, the weights on the head observations, W_{ii}^{H}, are constant. The second weighted sum of squares in the objective function is the regularization portion of the objective function. This weighted sum of squares is the difference in values between each pair of pilot points (PP_j - PP_k) and is designed to keep the transmissivity field as homogeneous as possible and to provide numerical stability when estimating more parameters than there are data. In this second weighted sum of squares, the weights, W_{jk}^{R}, are defined by the kriging factors and are a function of the distance between any two pilot points. Details on the formulation of the objective function can be found in Doherty (1998) and McKenna et al. (in press).

2) The kriging factors used to spread the influence of the pilot point values are calculated in the preprocessing step using **ppk2fac** (Appendix 8) and remain constant throughout the calibration process.
3) The pilot point values and the kriging factors are input into the **fac2real** program (Appendix 9). The **fac2real** code uses these inputs to generate an **MF2K** readable array of the residual field. At this step, this field is in log10 space. Note that the initial residual field created by the **sgsim** program will be considerably smoothed in the **fac2real** program. The **fac2real** code uses a kriging algorithm to spread the influence of the pilot points and kriging is an interpolator and therefore a smoothing process.
4) The updated residual field is then added to the base transmissivity field using the **addmods** program (Appendix 10) to create the updated log10 transmissivity field.
5) The program **xform** (Appendix 4) takes the log10 transmissivity field and converts it to raw space values that can be read by **MF2K**. **MF2K** does not read in log-transformed transmissivity values.
6) **MF2K** is run in forward mode and the resulting heads and the cell-by-cell volumetric fluxes, the "flow budget", are saved.

The final step in this process is to run the **get-heads** code (Appendix 6) on the **MF2K** output head file to get the calculated heads at the observation locations. These calculated heads are then compared to the observed heads within **PEST**.

The process shown in Figure 19 is run iteratively using the shell **model.sh** (Appendix 11) until at least one of three conditions are met: 1) the number of iterations reaches the maximum allowable number of 30; 2) the objective function reaches a predefined minimum value; or 3) there is less than a one-percent change in the value of the objective function across three consecutive iterations.

For these calibrations, the predefined minimum value of the objective function is determined using the measurement error of the heads for each time period. For each time period, the number of observation wells, the target Sum of Squared Errors (SSE) and the acceptable SSE are given in Table 8. Internally, PEST uses the SSE rather than the previously defined RMSE as a measure of how close the calculated heads are to the observed heads. The SSE is calculated as:

$$SSE = \sum_{i=1}^{N_{obs}} (H_i^{obs} - H_i^{calc})^2$$

PEST requires both a target value of the SSE and an acceptable value of the RMSE. Each measured head value also has an associated measurement error value (Beauheim, 2002a). The target SSE was set equal to the sum of these squared head measurement errors across the observation wells for each of the four data sets. The acceptable SSE was set to be 4 meters times the number of observation wells (N_{obs}). This acceptable SSE limit corresponds to a two-meter average error across all wells. Recall that a two-meter error encompassed the majority of the errors in the CCA calibration models (Lavenue, 1996).

Table 8. Target and acceptable SSE values for the four different data sets. The numbers of observation wells are only those wells within the modeling domain and therefore they are not the same as the numbers of wells used to create the boundary conditions.

Data Set	Number of Observations	Target SSE (m^2)	Acceptable SSE (m^2)
1980	13	47.0	52.0
1990	25	65.23	100.0
CCA	32	143.85	128.0
2000	35	76.74	140.0

For the CCA head data runs, the target SSE and acceptable SSE values were set to 1.00 and 1.10. These values were set in the default **PEST** input file and were not changed to the respective Culebra values of 143.85 and 128.0 prior to running the inverse models. By setting the target and acceptable values at 1.0 and 1.1, meeting an objective function value will not play a role in stopping the optimization process and the process will continue until the objective function can no longer be decreased or until the maximum number of iterations is reached. Nevertheless, results show that the RMSE values calculated for the CCA data set are consistent with those calculated for the other time periods.

The final piece of the calibration process is to do some post-processing on the results and to create and save the necessary files. The post-processing steps are shown in the bottom portion of Figure 18 and are as follows:

1) **PEST** writes the final calibrated pilot point values. These values are not necessarily the pilot point values that result from the final iteration of the **PEST** optimization, but are the set of pilot point values that created the lowest value of the objective function.
2) The final calibrated pilot point values are used in one last forward run through the model process to produce the ground water heads and budgets (cell by cell flux values) needed for the particle-tracking software (**DTRK**).
3) A particle track is calculated from the center of the repository area to the WIPP boundary and the time of the particle transport is recorded as the calibrated travel time for each T field.
4) Additional outputs from **PEST** that are saved for each realization are the sensitivity coefficient file and the record file.

All of the steps in the calibration process shown in Figures 18 and 19 are run using the **pest-setup.sh** shell (Appendix 12). This shell allows for initiation of the calibration process with a single command and this shell also calls the **model.sh** shell (Appendix 11) as part of the calibration process. An example of the final step in the creation of a calibrated transmissivity field is shown in Figure 20.

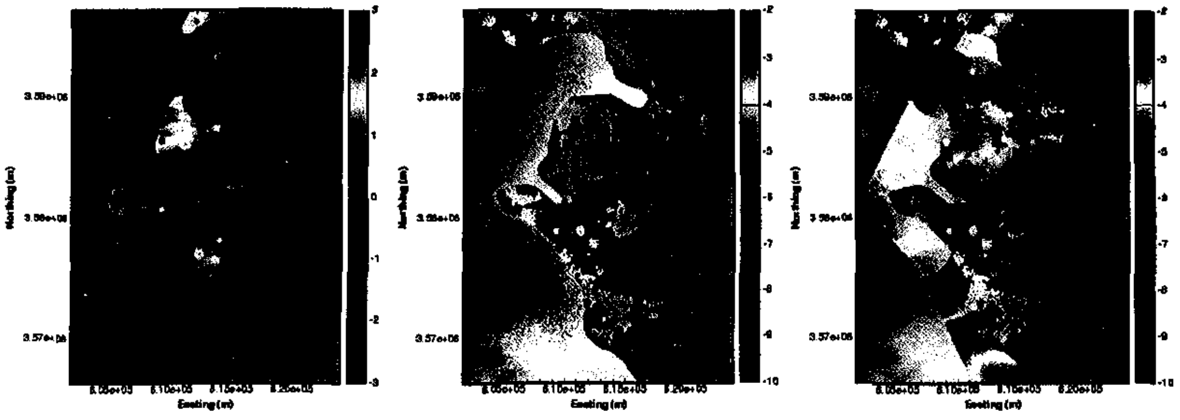


Figure 20. Example final steps in the creation of a calibrated transmissivity field. The calibrated residual field (left image) is added to the base transmissivity field (middle image) to get the final calibrated transmissivity field (right image). All color scales are in units of log10 (m^2/s) transmissivity.

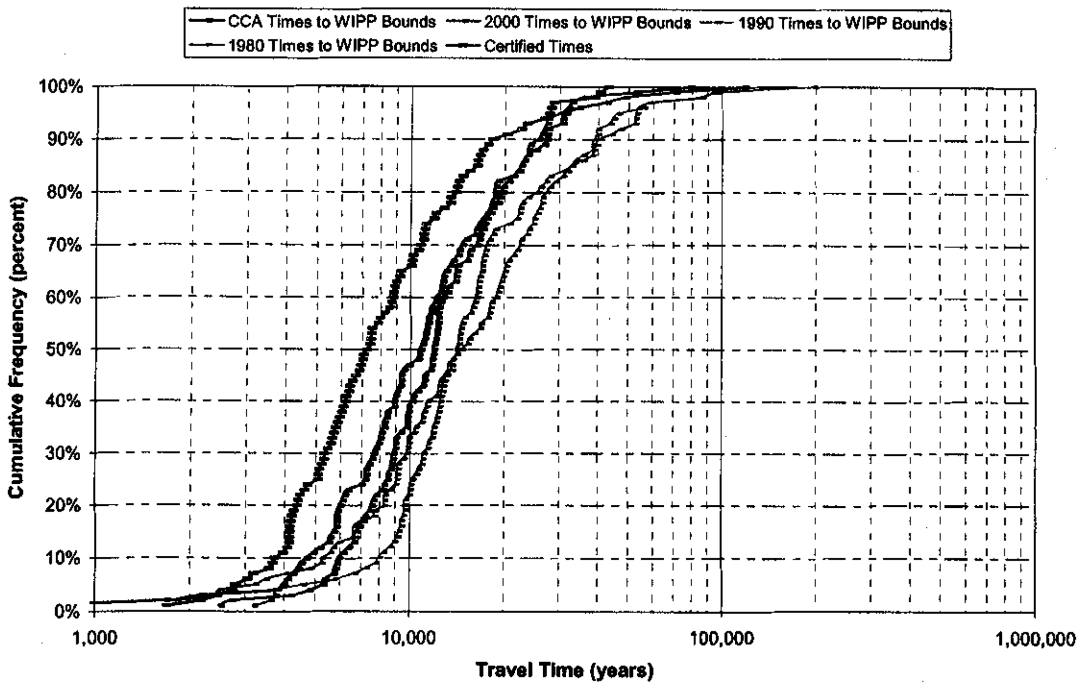
Inverse Modeling Results

For each of the 400 calibration runs, there are two results saved to files: the calculated heads at each of the observed head locations and the particle-tracking information. These results are summarized in Figure 21.

The inverse calibration process creates a significant change in the travel times modeled across the WIPP site. In general, the median travel times are reduced by a factor of almost 50,000 between uncalibrated and calibrated results (compare the upper images of Figures 21 and 16). The order of the travel time distributions seen in the uncalibrated results showed that, in general, the later the year, the shorter the travel times (Figure 16, upper image). This relation between the time period of the head measurements and the travel times is not apparent in the calibrated results (upper image of Figure 21).

The cdfs of the RMSE values shown in the lower image of Figure 21 are the "adjusted RMSE" values. The results demonstrated that for each of the four data sets, the calibration process was not able to reduce the fit to the measured heads at one of the wells in the data set. This problem occurred at well H-10b for the time periods where it is included in the data set and at well H-9b for the time periods when the H-10b well is not present. The coordinates of H-10b are (622975, 3572473) and the coordinates of the H-9b well are (613989, 3568261). Both of these wells are close to the boundary of the model domain and both of them had high residual values between the measured heads and the trend surfaces fit to the measured heads during the creation of the fixed-head boundary conditions (see Figures 4 and 5). The kriging process, in the creation of the boundary condition heads, forces the estimated head to match the observed head at all locations, but it does this with a relatively local perturbation to the total head field. The nearby boundary heads, that are fixed for all calculations, can be considerably different from the head assigned to the well location if that well location is not a good fit to the trend surface estimate. This situation causes conflict between the fixed-head boundary and the ability of PEST to fit the measured head at the nearby well. This conflict causes wells located near the model boundaries, such as H-10b and H-9b, to have anomalously high residuals at the end of the calibration process – PEST is unable to both fit the measured heads at these locations and meet the nearby fixed-head boundary condition. For this reason, the "adjusted RMSE" value is the RMSE calculated across all wells minus the one well, H-10b or H-9b that displayed this type of behavior for the time period being considered. The H-10b well is removed from the adjusted RMSE calculations forthe 1980, 1990 and CCA data sets and the H-9b well is removed for the 2000 data set. It is stressed that the calculations for the steady-state calibrations include the measured heads at wells H-9b or H-10b; it is only the summary value of the adjusted RMSE that does not.

The adjusted RMSE cdfs in the lower image of Figure 21 show that the majority of the T fields for the 1990, CCA and 2000 data sets are calibrated to an adjusted RMSE of less than 2.0 meters. These values are either at or very close to the average head measurement errors for those data sets indicating that a better calibration cannot be achieved. The 1980 adjusted RMSE values show that the majority of the RMSE values are greater than 2.0 but less than 3.0 meters. The adjusted RMSE values show that for all data sets there is a small proportion, 6 percent or less, of the realizations for which an adjusted RMSE of less than 3.0 meters cannot be obtained.



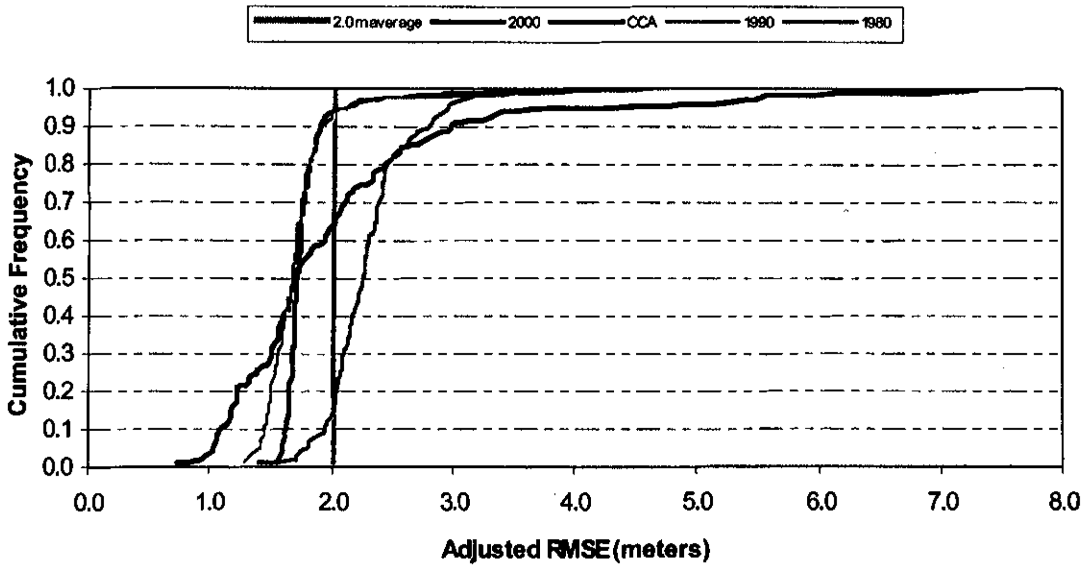


Figure 21. Results of the inverse simulations for the four different data sets. The particle travel times to the WIPP boundary are shown in the upper image and the adjusted RMSE values between the measured and modeled heads are shown in the lower image.

The variation in the adjusted RMSE values begs the question: What is the relationship between the adjusted RMSE and the calculated travel time for any given realization? For example do poor fits to the head data (large RMSE values) allow for significantly faster travel times? This relationship is shown in Figure 22 for all four data sets. Figure 22 shows that there is no relationship between the adjusted RMSE and the travel time for any of the four data sets and that those realizations with relatively high RMSE values do not produce extreme travel time results.

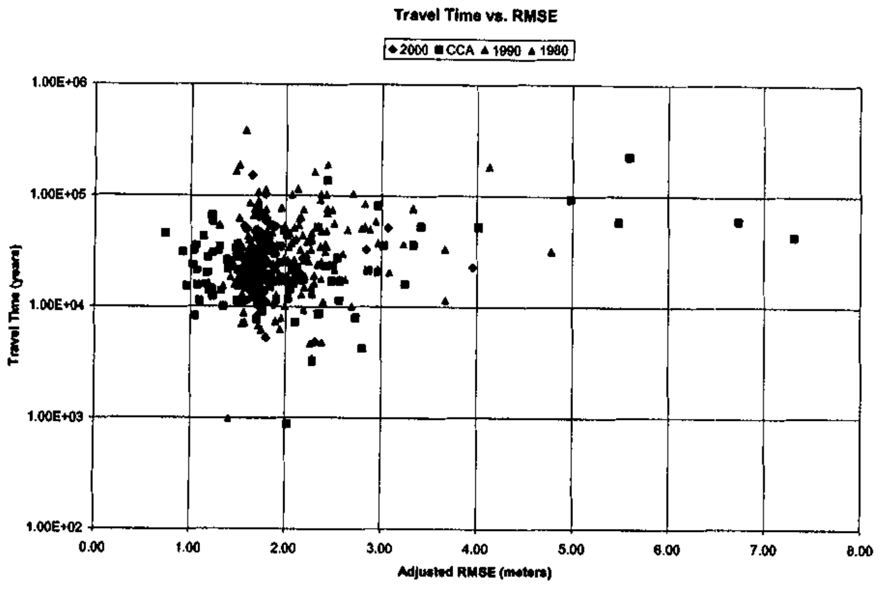


Figure 22. Particle travel time to the WIPP boundary as a function of the adjusted RMSE for all four data sets.

Summary

This analysis has shown that it is possible to develop T fields calibrated to heads measured at different time periods.

Points

1) Calibration yields a drastic change in the travel time distributions and generally reduces the median travel time by a factor of approximately 50,000 relative to the uncalibrated case.
2) Calibration also makes a major change in the fit of the modeled heads to the measured heads and can reduce the difference between measured and modeled heads to within the range of the head measurement error.

3) A calibrated solution is not a unique solution. These results indicate that the same level of calibration can produce travel times that range over an order of magnitude. As an example, for an adjusted RMSE of 1.80 meters, the travel times range from roughly 5000 to over 100,000 years (Figure 20).

References

Beauheim, R.L., 2002a, Calculation of Culebra Freshwater Heads in 1980, 1990, and 2000 for Use in T-Field Calibration, ERMS 522580.

Beauheim, R.L., 2002b, Analysis Plan for Evaluation of the Effects of Head Changes on Calibration of Culebra Transmissivity Fields, AP-088, Rev.1, 12/6/02, 11 pp.

Capilla, J.E., J.J. Gomez-Hernandez and A. Sahuquillo, 1998, Stochastic Simulation of Transmissivity Fields Conditional to Both Transmissivity and Piezometric Data- 3. Application to the Culebra Formation at the Waste Isolation Pilot Plant, New Mexico, U.S.A., *Journal of Hydrology*, 207, pp. 254-269.

Deutsch, C.V. and A.G. Journel, 1998, *GSLIB: Geostatistical Software Library and User's Guide, Second Edition*, Oxford University Press, New York, 369 pp.

DOE (U.S. Department of Energy). 1996. *Title 40 CFR Part 191 Compliance Certification Application for the Waste Isolation Pilot Plant.* DOE/CAO-1996-2184. Carlsbad, NM: U.S. DOE, Carlsbad Area Office.

Doherty, J., 1998, *PEST-Model independent parameter estimation, 2nd Edition*, Watermark Numerical Computing, Brisbane, Australia, 191 pp., http://members.ozemail.com.au/~wnc/wnc.htm.

Gomez-Hernandez, J.J, A. Sahuquillo and J.E. Capilla, 1997, Stochastic Simulation of Transmissivity Fields Conditional to Both Transmissivity and Piezometric Data – I. Theory, *Journal of Hydrology*, 203, pp. 162-174.

Harbaugh, A.W., E.R. Banta, M.C. Hill and M.G. McDonald, 2000, MODFLOW 2000: The U.S. Geological Survey Modular Ground-Water Model – User Guide to Modularization Concepts and the Ground-Water Flow Process, Open File Report 00-92, U.S. Geological Survey, Reston, Virginia, 121 pp.

Holt, R.M and L. Yarbrough, 2002, Analysis Report: Task 2 of AP-088: Estimating Base Transmissivity Fields, 69 pp., ERMS 523889.

Lavenue, A.M., 1996, Analysis of the Generation of Transmissivity Fields for the Culebra Dolomite, ERMS 240517.

McKenna, S.A., J. Doherty and D.B. Hart (in press) Non-Uniqueness of Inverse Transmissivity Field Calibration and Predictive Transport Modeling, accepted for publication in: *Journal of Hydrology*, Special Issue on Stochastic Inverse Methods, to be published summer of 2003.

Pannatier, Y., 1996, *VarioWin: Software Spatial Analysis in 2D*, Springer, New York, 91 pp.

Wallace, M.G., 1996, *Records Package For Screening Effort NS11: Subsidence Associated with Mining Inside or Outside the Controlled Area.* ERMS 412918.

Appendix 1: Guidelines for Pilot Point Selection

John Doherty, Watermark Numerical Computing

INTRODUCTION

Doherty (2001; 2003) describes a methodology for the use of pilot points in groundwater model calibration. Using that method, the values of aquifer hydraulic properties are estimated at the locations of a number of points spread throughout the model domain. Hydraulic properties are then assigned to the model grid through spatial interpolation from those points. In the software described by Doherty (2001), spatial interpolation is implemented through kriging on the basis of a user-specified variogram. The same variogram is then used to enforce a type of "uniformity condition" on the values assigned to pilot points. The uniformity condition is applied more strongly to points that are closer together, than to those that are further apart (the "strength" of this application being determined by the variogram). This condition is then used by the regularization functionality of PEST-ASP (Doherty, 2002) to achieve a numerically stable solution to the inverse problem of model calibration. It is because of the regularization algorithm implemented by PEST-ASP that so many parameters can be estimated through the model calibration process. In implementing this algorithm, PEST enforces parameter uniformity constraints as strongly as it can without violating the necessity for model outputs to match field data.

In an alternative calibration methodology, pilot points can be used in the estimation of "hydraulic property multipliers". Spatial interpolation from pilot points to the finite difference grid then allows the construction of a "warping array". A hydraulic property array (normally built by a stochastic field generator) is multiplied by this warping array on a cell-by-cell basis. Use of PEST's regularization functionality guarantees that departures from uniformity of the warping array are only as great as they need to be for the resulting warped property array to ensure a calibrated model.

When using pilot points to characterize the spatial variation of some hydraulic property (or property multiplier) prior to estimation of this property (or multiplier) through model calibration, the modeller must choose the locations of these points him/herself. While, ostensibly, this can introduce a certain amount of subjectivity into the calibration process, the proper placement of these points can, in fact, reduce the affects of this subjectivity in comparison to other methods of spatial parameterisation (for example those based on user-specified zonation patterns in situations were geological mapping is unable to provide much assistance in specification of zone boundaries). Furthermore, the more pilot points that are used to define spatial heterogeneity, the less pronounced is the element of subjectivity in the calibration process. However, as there will always be computational and numerical limits to the number of points that can be used, there will be occasions when the modeller must choose the locations of pilot points judiciously. This document is intended to act as a guide in this selection process.

Other methods of using pilot points in conjunction with PEST's regularization functionality are under continued development. It is possible that the following guidelines will be expanded somewhat as experience is gained in the development and implementation of these methods.

Number of Pilot Points

Conventional wisdom in environmental model calibration dictates adherence to a policy of parameter parsimony. This wisdom is based on the fact that if attempts are made to estimate too many parameters, the inverse problem becomes numerically unstable as parameter estimates are plagued by nonuniqueness resulting from parameter correlation and insensitivity. Use of a parameter set estimated on the basis of an improperly posed inverse problem can then lead to erroneous model predictions because of the tendency for spurious heterogeneity to be introduced into such an over-parameterized calibration process.

Limitations on the number of parameters which can be estimated through model calibration can be revised radically upwards when regularization is introduced to the parameter estimation process. This is because the regularization process is mathematically a "constrained minimization process" whereby parameter values are constrained to adhere as closely as possible to a "preferred system condition" described by the regularization equations. As implemented in the software described in Doherty (2001), the preferred system condition is one of uniformity of parameters or multipliers within one or a number of user-defined zones. However other regularization methods are possible with PEST-ASP such as adherence of parameters (or multipliers) to preferred values (which can be the same or different for different parameters). Hence if the calibration dataset does not possess the information content required for estimation of a particular parameter, that parameter will be assigned a value that is in accordance with the "preferred system state" as it pertains to that parameter. Thus, properly applied regularization ensures that, no matter how many parameters require simultaneous estimation through the calibration process, each of them can be assigned a unique value because none of them is insensitive, and none of them is excessively correlated with any other parameter.

In general, the more pilot points that are used to characterize the distribution of a spatially varying hydraulic property, the better will be the outcome of the calibration process. The principal advantage of using a multitude of pilot points is that they are more likely to be placed at locations "where they are needed" if there are many of them than if there are just a few of them. As is discussed below, the closer pilot points are situated to the locations at where hydraulic property heterogeneity exists within the model domain, the more likely it is that such points will be assigned realistic parameter or multiplier values. Improper placement of pilot points with respect to heterogeneity can result in out-of-range parameters as the latter are endowed with extreme values to compensate for the limited "leverage" they have in affecting properties at those locations within the model domain where property adjustment is most urgently needed.

In practice, the number of pilot points that can be used in the parameter estimation process is limited by CPU time, and by internal numerical noise within PEST itself.

Experience has shown that once the number of parameters rises above 220, PEST's performance begins to suffer as result of the latter phenomenon. The former problem (i.e. excessive CPU times) results from the fact that a numerical derivative must be calculated for each parameter adjusted through the calibration process. Thus, during each optimization iteration, PEST must run the model at least as many times as there are adjustable parameters (sometimes twice this number). While overall PEST run times can be lowered significantly through parallelisation (and through other devices such as the use of the MODFLOW-2000 AMG solver), the fact still remains that the estimation of a large number of parameters is a computationally expensive process.

Experience has demonstrated that for a single-layer model in which only the hydraulic conductivity is estimated, use of 100 pilot points seems to provide a suitable compromise between the competing needs of pilot point density and adequate execution speed.

Model Domain Heterogeneity

As documented by Doherty (2001), pilot points can be combined with the use of zones in model calibration, to accommodate *mapped* heterogeneity. The following discussion pertains only to the use of pilot points in accommodating intra-zonal or *unmapped* heterogeneity.

As was mentioned above, one of the great advantages of using pilot points for spatial parameterization is that the modeller does not need to guess where unmapped heterogeneity might exist within a model domain ahead of the calibration process. Instead, the calibration process can itself determine where such heterogeneity exists, or where stochastic fields are best warped to accommodate this heterogeneity to achieve an optimal model fit to field data. Nevertheless, this gain in the robustness and efficiency of is possible if a few simple steps are taken.

If there are any indications of the existence of heterogeneity within a model domain, then it is best to place a number of points within suspected anomalous zones; the number of such points will depend on the spatial extent of each suspected anomalous region. The existence of heterogeneity can often be inferred from the patterns of piezometric contours. For example, one or a number of points should be placed in regions of locally high hydraulic gradient; if such a region is elongate, points should be placed at regular intervals along its strike.

Similar considerations apply to regions where piezometric contours are widely spaced. However, by their nature, such regions will tend to be defined over broader areas and will not require as great a pilot point density as narrower zones of high piezometric gradient. If possible a point should be placed at the centre of such a region, and at regular intervals along its inferred boundary.

Placement in Relation to Measurement Wells

Where there is a greater density of piezometric measurement points within a model domain, there is a greater potential for inferring hydraulic property heterogeneity from measured head data. Hence, to reflect the locally enhanced resolving power of the calibration dataset, pilot points can be placed with greater density in such information-rich areas.

Where the line joining a pair of measurement wells is roughly in the down-gradient direction, consideration should be given to placing a pilot point somewhere on, or close to, this line based on the premise that it is the hydraulic conductivity between the points that determines the observed head differential between them. Based on this same argument, a good pilot points placement strategy would be to position pilot points as much as possible between measurement wells (rather than coincident with them), with particular emphasis being placed on pairs of wells that are aligned in the direction of the local hydraulic gradient.

Placement in Relation to Hydraulically Tested Wells

If independent hydraulic property estimates are available at certain points within the model domain, then these estimates should be used in the calibration process. The manner in which hydraulic test data is best incorporated into the parameterization of a regional aquifer is the subject of current research. However, for the moment it will be assumed that hydraulic properties determined through hydraulic test analysis will be used in the calibration process either through the direct assignment of local hydraulic properties, or as "prior information" by which local hydraulic property estimation will be guided.

For implementation of either of these methods, a pilot point should be placed at the site of each tested well. In the former method, the parameter pertaining to each such pilot point should be fixed at a constant value (equal to that determined through hydraulic test analysis) throughout the inversion process. In the latter case, the pilot point located at the hydraulic test site will assume its normal role; that is, the parameter value with which it is associated will be estimated through the inversion process. Just like other pilot points, the parameter associated with this point will suffer constraints imposed by the regularization process. However, this parameter will also feature in an additional item of prior information, in which it is linked to the hydraulic property value determined through hydraulic test analysis. Deviations of this parameter value from the independent estimate will then incur a penalty in the overall objective function. The weight assigned to this prior information equation, and hence the penalty incurred by deviation of the parameter from its independently estimated value, should be carefully chosen by the user.

Placement in Relation to Model Boundaries

Unless observation wells are very close to outflow boundaries marking the lower end of a model domain, pilot points should be placed between each such boundary and the closest up-gradient wells. Furthermore, the longer is an outflow boundary, the more pilot points may need to be placed sub-parallel to this boundary, along a line forming the rough mid-position between the boundary and the first row of measurement points. Whatever the geometry of the system, enough pilot points should be placed between outflow points and head measurement points to allow the calibration process to calculate conductivity values that account for the piezometric drop between these two model entities.

Similar considerations apply to uphill inflow boundaries were the boundary condition is of the prescribed or general head type. The principal is the same; that is, it is the hydraulic conductivity of the material between the boundary and the closest observation well along any pathline that determines the potential drop along that pathline. Hence, enough pilot points must be placed between the boundary and the up-gradient set of observation points (with placement parallel or sub-parallel to the boundary) to allow PEST to assign hydraulic conductivity values to this part of the model domain, and to account for any lateral conductivity variations that may exist in a direction that is roughly parallel to the boundary.

In many modeling applications, up-gradient model boundaries are of the prescribed inflow type (including, possibly, no inflow). Also, such boundaries are sometimes placed at a considerable distance from the nearest set of observation wells to minimize their effect on that part of the model domain that is of most interest for predictive purposes. In cases such as this, the hydraulic properties pertaining to those parts of the model domain that lie beyond the outer set of observation wells will be virtually indeterminable from the calibration dataset alone (especially in a steady-state model). Fortunately, in many cases their effect on model predictions will be minor. Nevertheless some thought should be given to the characterization of hydraulic properties within such areas, and to the effect of pilot point placement on the representation of these properties in the model. If no pilot points are placed in such areas, then the kriging process by which hydraulic property values are assigned to model cells in these areas will be such that, the further such cells are removed from the nearest pilot point, the closer will their hydraulic property value be to the mean property value prevailing within the model domain. However, if a few pilot points are sprinkled in such areas, and if "smoothing regularization" is used (i.e. regularization which attempts to minimize hydraulic property differences between neighboring pilot points – see Doherty, 2001), then the hydraulic property values assigned to these uphill areas will tend to be more like those that prevail in the closest parts of the model domain for which hydraulic property values can be assigned on the basis of the calibration dataset. Hence it is often good practice to place a few pilot points in the "back blocks" of the model domain between the most up-gradient observation wells and the inflow/no-flow boundaries that form the uphill end of the model.

Similar considerations apply to areas at the sides of the model domain that may be far removed from observation wells.

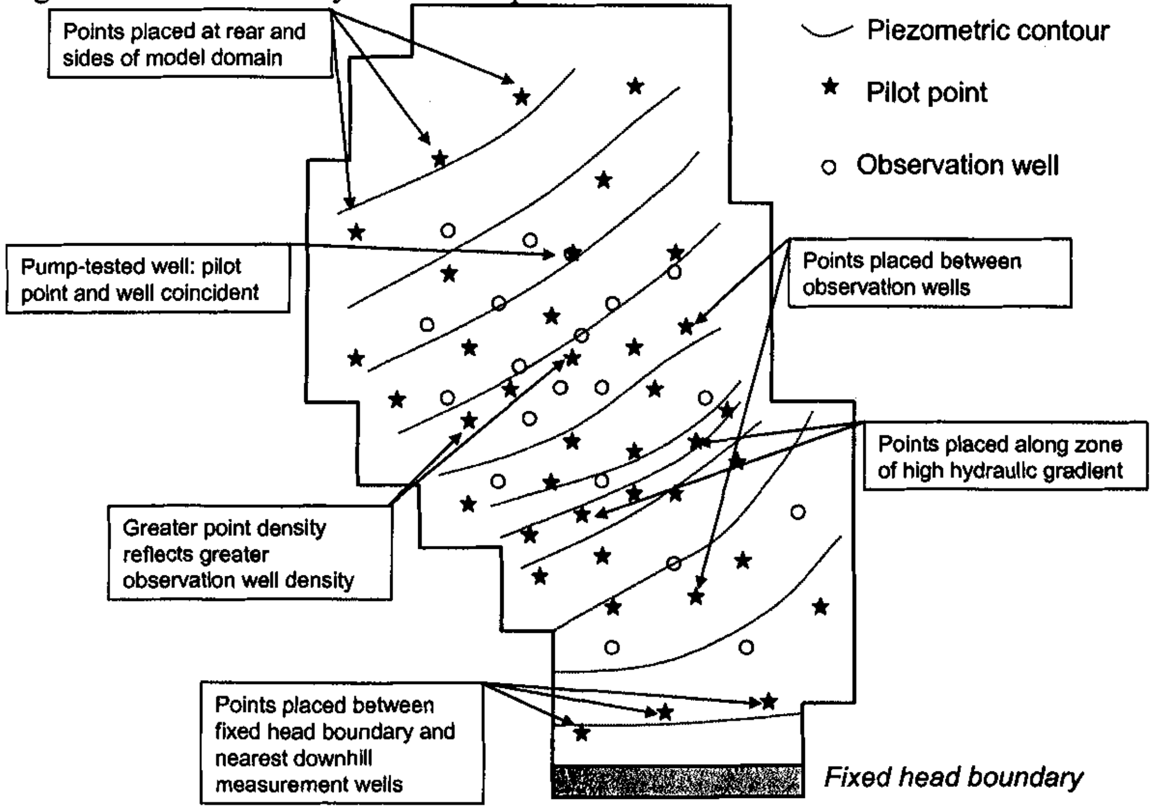
Areas of Special Interest

Consideration should be given to increasing pilot point density at locations within the model domain at which key predictions are to be made. Of particular interest in many instances of model usage are the paths taken by contaminants (or potential contaminants) from their points of entry into the model domain. The regularization process implemented by PEST-ASP will ensure that spurious heterogeneity will not be introduced into the model domain as a result of locally increased pilot-point density. However the introduction of extra pilot points to critical areas allows the calibration dataset to have "full sway" in detecting any heterogeneities which may exist at those places within the model domain where enhanced spatial resolution may be most important for the making of key model predictions, or of exploring the uncertainty associated with those predictions (especially those predictions that depend on "system fine detail" as do contaminant pathways).

FILLING IN

If, after all of the above suggestions have been followed, there are areas in the model domain which are devoid of pilot points, then points should be added to fill in the gaps. This process should continue until there are no remaining gaps, or until the maximum number of pilot points has been reached. However, as is further discussed below, the variogram should not be ignored when deciding on how many points are sufficient in any parameterisation context. After all pilot points have been assigned, the average distance between pilot points should be considerably less than the variogram range (a factor of 3 or 4 is suggested).

Figure A1 illustrates many of the concepts discussed above.



Points placed at rear and sides of model domain
Piezometric contour
Pilot point
Observation well
Pump-tested well: pilot point and well coincident
Points placed between observation wells
Points placed along zone of high hydraulic gradient
Greater point density reflects greater observation well density
Points placed between fixed head boundary and nearest downhill measurement wells
Fixed head boundary

Figure A1. Some examples of the rationale governing the placement of pilot points.

THE VALUE FOR PHIMLIM

PEST allows the user to set the "calibration threshold" for a particular model though choice of a value for the control variable PHIMLIM. This variable resides in the "regularization" section of the PEST control file. It is the objective function below which the model is deemed to be calibrated. PEST will not seek to reduce the objective function below this level during a regularized inversion process.

Care must be exercised in choosing a suitable value for PHIMLIM. If it is set too low, PEST will give little importance to regularization constraints as it attempts to create a perfect fit between model outputs and field data. While the assignment of a low value to PHIMLIM may result in a spectacular fit between model outputs and field measurements, it may also result in unbelievable parameter estimates as the calibration process "bends parameters to fit the noise". This, in turn, can lead to unrealistic model predictions that defy credibility. Alternatively, if PHIMLIM is set too high, PEST may calculate a hydraulic property field that is too smooth, or too close to the "default system condition" encapsulated in the regularization equations; in some situations this may give rise to model predictions that are too conservative.

In many instances of model deployment, it will not be possible to assign a suitable value to PHIMLIM until some calibration runs have been carried out to determine the extent to which the model is capable of fitting field measurements, given its present conceptual basis. When undertaking runs for this purpose, PEST is able to select an appropriate value for PHIMLIM itself on the basis of the user-supplied value for another control variable, viz. FRACPHIM. However this will often result in a PHIMLIM value that is too low. Thus, once PEST has been used in this manner in early calibration runs, the calibration process may need to be repeated with a more appropriate (higher) user-specified PHIMLIM setting.

Parameter Bounds

The occurrence of wild and aberrant parameter values can be precluded through the use of PEST's parameter bounds functionality. However, unless there is a good reason to do so, it is best to refrain from applying bounds to parameters when undertaking regularized parameter estimation. This is because the regularization process itself should ensure that parameters stay within a reasonable range. If they do not, then PHIMLIM may have to be increased, or some other measure may need to be taken to ensure that parameter values stay within realistic ranges. If parameters are kept within these ranges "artificially" by the imposition of bounds, this might prevent the user from gaining valuable insights into possible model inadequacy that may be forthcoming from the calibration process. (Having stated this however, it cannot be denied that there may be occasions when the imposition of bounds is important; no rule is universally applicable.)

Variogram

In the regularized calibration process described by Doherty (2001), the variogram performs two roles. The first is the determination of kriging factors by which model grid property values are calculated from pilot point property values. The second is the assignment of relative weights to the equations that encapsulate the regularization constraints. In neither of these roles is the assignment of variogram parameters critical. Furthermore, the more pilot points that are used for the characterization or spatial heterogeneity, the less critical do the variogram parameters become.

In spite of this, some care should be taken in choosing a suitable variogram. Experience suggests that use of the power and Gaussian variograms should be avoided, as kriging based on these variograms can lead to spurious hydraulic property values at grid cells after interpolation from pilot points, especially if the latter are placed too close together.

In many groundwater modelling contexts information available for variogram construction is very limited. Nevertheless, the modeller will often have some idea of the length scale of hydraulic property continuity, and hence of the variogram range. On some occasions it may be possible to estimate the range of the variogram as part of the inversion process. However, bounds should be placed on variogram range estimates made through the inversion process; in particular, it is important that its lower bound be such that the average inter-pilot-point distance is significantly less than the variogram range. Another issue to consider when estimating the variogram range is that kriging factors will need to be re-computed on every occasion that the range is altered by PEST. Efficiencies in this process can be gained through use of PEST's multiple-command-line functionality.

As is discussed in Doherty (2001), if a single variogram is used to characterise geostatistical structure within a model area, its sill has no effect on the pilot point parameterisation process. If multiple variograms are use to characterise a structure, then only the relative values of the individual sills are important, not their absolute values.

TROUBLESHOOTING

The section briefly outlines problems that may occur in a regularized calibration/warping process based on pilot points, and suggests steps that may be taken to rectify them.

Poor Fit Between Model Outputs and Field Measurements

This can result from flaws in the conceptual model that underpins the numerical model. Experience has shown that use of pilot points in a regularized calibration process allows such errors to be detected more quickly than would otherwise be possible, for this methodology allows rapid exploration of the effects of potential spatial heterogeneity on model outputs. Failure to achieve a better fit through enhanced heterogeneity necessitates a revision of other aspects of model design.

If the conceptual model is judged to be correct and a poor fit between model outcomes and field measurements remains, the user should consider using a greater number of pilot points, or shifting certain points to different locations. Areas of greatest model-to-measurement misfit are prime candidates for the introduction of new pilot points.

If the occurrence of a poor fit between model outcomes and field data is accompanied by the estimation of hydraulic property field that is too smooth, consideration should be given to lowering PHIMLIM.

Out-of-Range Parameter Values

This can be caused by flaws in the conceptual basis of the model (for example if the occurrence of broad areas of low piezometric gradient is attributed to high hydraulic conductivity rather than to enhanced recharge). However out-of-range parameter values can also be the result of improper placement of pilot points. If heterogeneity needs to exist at a certain location in order for the model to replicate measured heads in nearby wells, but the nearest pilot point is relatively far away, PEST will have no option but to adjust the property value assigned to the far-away point in an attempt to fit the heads; however it would be much better to adjust the value assigned to a point which is located within the actual heterogeneity. If there are few or no observation wells near the far-away pilot point, then there may be only weak calibration-imposed restraints on the parameter value assigned to that point. It may thus be assigned a value that is outside the normal value range for that parameter type.

Out-of-range parameter values can also result from "chasing noise" in field data. This occurs when PEST adjusts parameter values in order to fit every nuance of the calibration dataset, even when a component of each field measurement results from processes other than those simulated by the model. Before accepting any pilot-point based calibration, it is extremely important that the hydraulic property field (after interpolation from pilot points to the grid), and a complete set of model-generated heads over the entire model domain, be carefully inspected. Anomalies in either of these could indicate the assignment of spurious parameter values to one or more pilot points.

Problems associated with out-of-range parameter values can be rectified in a number of ways, including:

- adjustment of the conceptual basis for the model;
- assignment of more pilot points to areas of possible heterogeneity within the model domain;
- shifting offending pilot points to places where they are most needed;
- increasing the value of PHIMLIM;
- placing bounds on parameters (but see the above discussion on parameter bounds).

If using the Gaussian or power variogram (not suggested), it is possible for values interpolated to the model grid to be lower/higher than values assigned to pilot points. Utility software supplied for the implementation of pilot-point-based calibration using PEST allows the user to "clip" interpolated hydraulic fields at reasonable values. If using this functionality, be careful of its interaction with PEST's parameter bounds functionality. However it is best to avoid the problem altogether by using the exponential or spherical variogram.

Spurious Model Outputs

It sometimes occurs that while the fit between model outputs and field measurements is exceptionally good, the model may produce spurious heads (or other outputs) at locations within the model domain where there are no calibration targets. This is mostly a direct outcome of the occurrence of out-of range parameter values and can be rectified using one or a number of the measures discussed above.

Inability to Lower the Objective Function

A common occurrence in unregularized parameter estimation in an over-parameterized system is an inability on the part of PEST to lower the objective function. Meanwhile, one or more parameters may change by large amounts during each optimisation iteration (often limited by the factor or relative change limits FACPARMAX and RELPARMAX), and the Marquardt lambda may progressively rise as the optimisation process progresses.

Furthermore, an inspection of PEST's parameter sensitivity file (this has an extension of *.sen*) and/or its matrix file (which has an extension of *.mtt*) reveals a high degree of parameter correlation and/or insensitivity.

The same problems can occur in some instances of regularized inversion – especially in the final stages of a parameter estimation run in which the value selected for PHIMLIM is too low. In such a case, PEST may neglect regularization information and, in doing this, lose the numerical advantages of regularization. Fortunately, this problem is easily overcome by increasing PHIMLIM.

References

Doherty, J., 2001. PEST Groundwater Data Utilities. Watermark Numerical Computing, Australia.
Doherty, J., 2002. Manual for PEST; 5th Edition. Watermark Numerical Computing, Australia.
Doherty, J., 2003. "Ground Water Model Calibration using Pilot Points and Regularization". *Ground Water.* Vol. 41, no. 2, 170-177.

Appendix 2: Supplementary Material for Estimation of the Fixed Head Boundary Values

RESULTS OF FITTING THE GAUSSIAN TREND SURFACE TO THE 1980 HEADS

```
Nonlinear Regression

[Variables]
x = col(1)
y = col(2)
z = col(3)
[Parameters]
x0 = xatymax(x,z) "Auto {{previous: 626195}}
y0 = xatymax(y,z) "Auto {{previous: 4.14982e+006}}
a = max(z) "Auto {{previous: 1323.29}}
b = fwhm(x,z)/2.2 "Auto {{previous: 163929}}
c = fwhm(y,z)/2.2 "Auto {{previous: 674927}}
[Equation]
f=a*exp(-.5*( ((x-x0)/b)^2 + ((y-y0)/c)^2 ))
fit f to z
[Constraints]
[Options]
tolerance=0.000100
stepsize=100
iterations=100

R = 0.86434538  Rsqr = 0.74709293        Adj Rsqr = 0.65512672

Standard Error of Estimate = 6.3707

        Coefficient     Std. Error      t        P
x0      626195.3611     30694.8879      20.4006 <0.0001
y0      4149817.9394    9178912.7528    0.4521   0.6600
a       1323.2916       7416.9448       0.1784   0.8616
b       163929.4859     147739.8589     1.1096   0.2908
c       674926.8586     5645329.6128    0.1196   0.9070

Analysis of Variance:
                DF      SS              MS              F       P
Regression      4       1318.8112       329.7028        8.1236  0.0027
Residual 11     446.4460        40.5860
Total    15     1765.2572       117.6838

PRESS = 2378.1747

Durbin-Watson Statistic = 2.0941

Normality Test:  Passed   (P = 0.1570)

Constant Variance Test:  Passed   (P = 0.2922)

Power of performed test with alpha = 0.0500: 0.9971
```

Regression Diagnostics:

Row	Predicted	Residual	Std. Res.	Stud. Res.	Stud. Del. Res.
1	925.7026	-2.7895 -	0.4379	-0.4690	-0.4517
2	924.9243	-8.3480 -	1.3104	-1.4055	-1.4795
3	921.5283	-8.8928 -	1.3959	-1.4970	-1.5995
4	930.6170	3.3511	0.5260	0.5927	0.5743
5	928.1497	4.7473	0.7452	0.7937	0.7794
6	914.7788	-2.4295	-0.3814	-0.4159	-0.3997
7	902.2684	8.3891	1.3168	2.4564	3.4856
8	910.3936	-1.8616	-0.2922	-0.3414	-0.3272
9	917.6483	3.3026	0.5184	1.1628	1.1838
10	923.8082	3.6774	0.5772	0.6167	0.5985
11	920.9487	-2.8235	-0.4432	-0.4760	-0.4586
12	924.4544	3.7356	0.5864	0.6269	0.6087
13	919.4562	-3.1933	-0.5013	-0.5494	-0.5312
14	933.4058	8.9989	1.4125	1.8533	2.1307
15	939.6096	-3.9958	-0.6272	-0.9522	-0.9478
16	910.4777	-4.7834	-0.7508	-1.6956	-1.8811

Influence Diagnostics:

Row	Cook'sDist	Leverage	DFFITS
1	0.0065	0.1285	-0.1735
2	0.0594	0.1308	-0.5739
3	0.0673	0.1305	-0.6196
4	0.0189	0.2122	0.2981
5	0.0169	0.1185	0.2858
6	0.0066	0.1594	-0.1740
7	2.9924	0.7126	5.4888
8	0.0085	0.2673	-0.1976
9	1.0899	0.8012	2.3767
10	0.0108	0.1240	0.2252
11	0.0070	0.1331	-0.1797
12	0.0112	0.1251	0.2302
13	0.0122	0.1677	-0.2385
14	0.4956	0.4191	1.8098
15	0.2366	0.5661	-1.0826
16	2.3573	0.8039	-3.8087

95% Confidence:

Row	Predicted	Regr. 5%	Regr. 95%	Pop. 5%	Pop. 95%
1	925.7026	920.6762	930.7289	910.8070	940.5981
2	924.9243	919.8534	929.9953	910.0137	939.8350
3	921.5283	916.4631	926.5934	906.6196	936.4369
4	930.6170	924.1574	937.0766	915.1787	946.0552
5	928.1497	923.3228	932.9767	913.3203	942.9791
6	914.7788	909.1813	920.3763	899.6810	929.8766
7	902.2684	890.4317	914.1052	883.9185	920.6184
8	910.3936	903.1444	917.6428	894.6087	926.1785
9	917.6483	905.0971	930.1994	898.8296	936.4670
10	923.8082	918.8706	928.7458	908.9424	938.6740
11	920.9487	915.8332	926.0643	906.0229	935.8746
12	924.4544	919.4949	929.4138	909.5813	939.3274
13	919.4562	913.7140	925.1985	904.3042	934.6083
14	933.4058	924.3285	942.4831	916.7022	950.1094
15	939.6096	929.0595	950.1597	922.0621	957.1571

16	910.4777	897.9055	923.0498	891.6450	929.3104

RESULTS OF FITTING THE GAUSSIAN TREND SURFACE TO THE 1990 HEADS

Nonlinear Regression

```
[Variables]
x = col(1)
y = col(2)
z = col(3)
[Parameters]
x0 = xatymax(x,z) "Auto {{previous: 615692}}
y0 = xatymax(y,z) "Auto {{previous: 3.92718e+006}}
a = max(z) "Auto {{previous: 1155.98}}
b = fwhm(x,z)/2.2 "Auto {{previous: 124127}}
c = fwhm(y,z)/2.2 "Auto {{previous: 517625}}
[Equation]
f=a*exp(-.5*( ((x-x0)/b)^2 + ((y-y0)/c)^2 ))
fit f to z
[Constraints]
[Options]
tolerance=0.000100
stepsize=100
iterations=100
```

```
R = 0.78629497   Rsqr = 0.61825978        Adj Rsqr = 0.55187017

Standard Error of Estimate = 6.4936

        Coefficient     Std. Error      t        P
x0      615691.5112     6208.8006       99.1643  <0.0001
y0      3927177.2298    3469314.6321    1.1320   0.2693
a       1155.9754       2562.6688       0.4511   0.6562
b       124127.3319     57104.2611      2.1737   0.0403
c       517624.7100     2694099.9615    0.1921   0.8493

Analysis of Variance:
              DF      SS              MS             F        P
Regression    4       1570.7405       392.6851       9.3126   0.0001
Residual 23   969.8428        42.1671
Total    27   2540.5833       94.0957

PRESS = 3524.8490

Durbin-Watson Statistic = 1.7968

Normality Test:   Passed   (P = 0.1370)

Constant Variance Test:   Passed   (P = 0.3971)

Power of performed test with alpha = 0.0500: 0.9996

Regression Diagnostics:
Row   Predicted     Residual    Std. Res.   Stud. Res.      Stud. Del. Res.
1     933.3732      -1.3462  -0.2073  -0.2736  -0.2680
2     920.6080      -7.9509  -1.2244  -1.2659  -1.2836
```

INFORMATION ONLY

```
3        920.3274          -5.0278  -0.7743  -0.8082  -0.8019
4        923.5232          -7.7813  -1.1983  -1.2369  -1.2521
5        929.3160          5.7979   0.8929   0.9261   0.9231
6        920.9876          -5.8036  -0.8937  -0.9244  -0.9213
7        928.8107          5.1243   0.7891   0.8278   0.8219
8        928.3153          3.6371   0.5601   0.5821   0.5736
9        915.0014          -1.8079  -0.2784  -0.2945  -0.2886
10       901.7762          12.0544  1.8563   3.3856   4.6749
11       908.8775          2.7877   0.4293   0.4923   0.4840
12       912.5035          8.3009   1.2783   1.8421   1.9513
13       922.0596          -7.6108  -1.1721  -1.2121  -1.2252
14       917.6198          -1.8492  -0.2848  -0.2998  -0.2938
15       923.2190          -7.4887  -1.1532  -1.1897  -1.2011
16       925.3482          -8.9607  -1.3799  -1.4238  -1.4583
17       920.1514          -5.9513  -0.9165  -0.9514  -0.9494
18       926.5562          7.1590   1.1025   1.1367   1.1443
19       924.1817          3.3322   0.5132   0.5328   0.5243
20       920.8641          -3.1466  -0.4846  -0.5029  -0.4946
21       920.0019          -6.0951  -0.9386  -0.9710  -0.9698
22       927.2173          5.6357   0.8679   0.8943   0.8903
23       927.9152          6.0751   0.9356   0.9663   0.9648
24       925.3302          4.8698   0.7499   0.7888   0.7821
25       920.4386          -0.6700  -0.1032  -0.1114  -0.1090
26       934.7525          4.3932   0.6765   0.9828   0.9820
27       911.1579          -5.6189  -0.8653  -1.8883  -2.0090
28       934.5272          0.8391   0.1292   0.1445   0.1414

Influence Diagnostics:
Row      Cook'sDist        Leverage         DFFITS
1        0.0111   0.4257   -0.2307
2        0.0221   0.0644   -0.3368
3        0.0117   0.0822   -0.2399
4        0.0201   0.0615   -0.3206
5        0.0130   0.0705   0.2543
6        0.0119   0.0652   -0.2433
7        0.0138   0.0912   0.2604
8        0.0054   0.0743   0.1625
9        0.0021   0.1063   -0.0995
10       5.3325   0.6994   7.1301
11       0.0153   0.2396   0.2717
12       0.7306   0.5184   2.0246
13       0.0204   0.0649   -0.3229
14       0.0019   0.0977   -0.0967
15       0.0182   0.0603   -0.3044
16       0.0262   0.0607   -0.3709
17       0.0141   0.0721   -0.2646
18       0.0163   0.0593   0.2872
19       0.0044   0.0723   0.1464
20       0.0039   0.0717   -0.1375
21       0.0132   0.0656   -0.2570
22       0.0099   0.0583   0.2215
23       0.0125   0.0626   0.2493
24       0.0132   0.0962   0.2552
25       0.0004   0.1430   -0.0445
26       0.2144   0.5261   1.0346
27       2.6828   0.7900   -3.8968
```

28 0.0010 0.2004 0.0708

95% Confidence:

Row	Predicted	Regr. 5%	Regr. 95%	Pop. 5%	Pop. 95%
1	933.3732	924.6089	942.1375	917.3339	949.4125
2	920.6080	917.1983	924.0177	906.7489	934.4670
3	920.3274	916.4767	924.1781	906.3533	934.3015
4	923.5232	920.1916	926.8547	909.6831	937.3632
5	929.3160	925.7490	932.8830	915.4174	943.2146
6	920.9876	917.5575	924.4176	907.1235	934.8516
7	928.8107	924.7540	932.8674	914.7785	942.8430
8	928.3153	924.6543	931.9763	914.3923	942.2383
9	915.0014	910.6219	919.3810	900.8725	929.1304
10	901.7762	890.5425	913.0100	884.2650	919.2875
11	908.8775	902.3024	915.4525	893.9216	923.8334
12	912.5035	902.8314	922.1757	895.9507	929.0564
13	922.0596	918.6363	925.4830	908.1972	935.9221
14	917.6198	913.4206	921.8189	903.5457	931.6939
15	923.2190	919.9191	926.5190	909.3866	937.0515
16	925.3482	922.0373	928.6591	911.5131	939.1833
17	920.1514	916.5444	923.7585	906.2425	934.0603
18	926.5562	923.2858	929.8267	912.7308	940.3817
19	924.1817	920.5696	927.7937	910.2714	938.0919
20	920.8641	917.2660	924.4623	906.9575	934.7707
21	920.0019	916.5603	923.4435	906.1350	933.8689
22	927.2173	923.9745	930.4602	913.3984	941.0363
23	927.9152	924.5548	931.2756	914.0682	941.7622
24	925.3302	921.1640	929.4965	911.2659	939.3945
25	920.4386	915.3592	925.5179	906.0773	934.7999
26	934.7525	925.0093	944.4957	918.1580	951.3470
27	911.1579	899.2182	923.0975	893.1856	929.1301
28	934.5272	928.5133	940.5411	919.8094	949.2451

Results of Fitting the Gaussian Trend Surface to the CCA Heads

Nonlinear Regression

```
[Variables]
x = col(1)
y = col(2)
z = col(3)
[Parameters]
x0 = xatymax(x,z) "Auto {{previous: 497048}}
y0 = xatymax(y,z) "Auto {{previous: 3.71273e+006}}
a = max(z) "Auto {{previous: 1024.17}}
b = fwhm(x,z)/2.2 "Auto {{previous: 4.37843e+006}}
c = fwhm(y,z)/2.2 "Auto {{previous: 287105}}
[Equation]
f=a*exp(-.5*( ((x-x0)/b)^2 + ((y-y0)/c)^2 ))
fit f to z
[Constraints]
[Options]
tolerance=0.000100
stepsize=100
iterations=100
```

```
R = 0.82414590  Rsqr = 0.67921646        Adj Rsqr = 0.63497045

Standard Error of Estimate = 5.9760

          Coefficient       Std. Error          t        P
x0        497048.4153       143182271.7998 0.0035   0.9973
y0        3712731.9513      505763.5357    7.3408   <0.0001
a         1024.1661         619.7038       1.6527   0.1092
b         4378431.6009      2306530015.0373         0.0019   0.9985
c         287104.5604       552804.8242    0.5194   0.6075

Analysis of Variance:
                    DF       SS               MS                F        P
Regression          4        2192.8705        548.2176          15.3509 <0.0001
Residual 29         1035.6592        35.7124
Total    33         3228.5297        97.8342

PRESS = 2205.3919

Durbin-Watson Statistic = 1.6163

Normality Test:   Passed   (P = 0.1576)

Constant Variance Test:   Passed   (P = 0.7521)

Power of performed test with alpha = 0.0500: 1.0000

Regression Diagnostics:
Row       Predicted         Residual           Std. Res.          Stud. Res.          Stud. Del. Res.
1         933.4974          -1.4974  -0.2506  -0.3223  -0.3172
2         917.1318          -6.0318  -1.0093  -1.0359  -1.0372
3         918.3941          -3.1941  -0.5345  -0.5569  -0.5502
4         920.5204          -6.2204  -1.0409  -1.0681  -1.0708
5         927.7549          6.9451   1.1622   1.1979   1.2073
6         922.5199          -0.9199  -0.1539  -0.1577  -0.1550
7         922.4728          2.3272   0.3894   0.3989   0.3930
8         921.3582          -6.5582  -1.0974  -1.1238  -1.1291
9         921.3746          -6.5746  -1.1002  -1.1266  -1.1320
10        917.7858          -6.3858  -1.0686  -1.0964  -1.1004
11        927.0284          7.1716   1.2001   1.2530   1.2660
12        927.3615          4.6385   0.7762   0.7999   0.7948
13        912.0128          0.6872   0.1150   0.1225   0.1204
14        902.0521          4.3479   0.7276   1.0166   1.0172
15        908.5864          12.7136  2.1274   3.4111   4.3316
16        918.7002          -6.3002  -1.0543  -1.0824  -1.0858
17        918.7362          -6.3362  -1.0603  -1.0885  -1.0921
18        913.1894          0.3106   0.0520   0.0544   0.0535
19        920.5674          -3.6674  -0.6137  -0.6287  -0.6220
20        922.7657          -6.6657  -1.1154  -1.1461  -1.1525
21        916.3131          -5.3131  -0.8891  -0.9159  -0.9133
22        924.6902          7.7098   1.2901   1.3231   1.3412
23        922.9716          3.9284   0.6574   0.6815   0.6751
24        918.1897          -0.3897  -0.0652  -0.0671  -0.0660
25        916.2526          -6.9526  -1.1634  -1.1956  -1.2048
26        925.2018          8.3982   1.4053   1.4422   1.4708
27        926.2540          7.4460   1.2460   1.2798   1.2946
```

28	924.7008	5.7992	0.9704	0.9954	0.9952
29	925.9716	2.7284	0.4566	0.4987	0.4921
30	921.8036	-3.3036	-0.5528	-0.7154	-0.7093
31	938.6943	-0.5943	-0.0995	-0.1334	-0.1311
32	940.8266	-3.3266	-0.5567	-0.8000	-0.7949
33	933.9874	0.1126	0.0188	0.0204	0.0200
34	903.9805	5.8195	0.9738	1.2325	1.2441

Influence Diagnostics:

Row	**Cook'sDist**	**Leverage**	**DFFITS**
1	0.0136	0.3954	-0.2566
2	0.0114	0.0506	-0.2394
3	0.0053	0.0789	-0.1610
4	0.0121	0.0503	-0.2465
5	0.0179	0.0588	0.3018
6	0.0002	0.0467	-0.0343
7	0.0016	0.0469	0.0872
8	0.0123	0.0463	-0.2488
9	0.0123	0.0463	-0.2494
10	0.0127	0.0502	-0.2529
11	0.0283	0.0827	0.3801
12	0.0080	0.0585	0.1981
13	0.0004	0.1192	0.0443
14	0.1968	0.4878	0.9926
15	3.6557	0.6110	5.4290
16	0.0127	0.0514	-0.2527
17	0.0128	0.0511	-0.2534
18	0.0001	0.0875	0.0166
19	0.0039	0.0471	-0.1384
20	0.0146	0.0528	-0.2720
21	0.0103	0.0577	-0.2260
22	0.0181	0.0492	0.3052
23	0.0070	0.0697	0.1848
24	0.0001	0.0560	-0.0161
25	0.0160	0.0530	-0.2851
26	0.0221	0.0505	0.3391
27	0.0180	0.0521	0.3035
28	0.0103	0.0496	0.2272
29	0.0096	0.1618	0.2162
30	0.0691	0.4030	-0.5827
31	0.0028	0.4438	-0.1171
32	0.1364	0.5158	-0.8205
33	0.0000	0.1425	0.0082
34	0.1829	0.3758	0.9653

95% Confidence:

Row	**Predicted**	**Regr. 5%**	**Regr. 95%**	**Pop. 5%**	**Pop. 95%**
1	933.4974	925.8115	941.1833	919.0593	947.9354
2	917.1318	914.3834	919.8802	904.6043	929.6592
3	918.3941	914.9611	921.8270	905.6988	931.0893
4	920.5204	917.7785	923.2622	907.9943	933.0464
5	927.7549	924.7912	930.7185	915.1785	940.3313
6	922.5199	919.8788	925.1609	910.0155	935.0242
7	922.4728	919.8265	925.1190	909.9673	934.9782
8	921.3582	918.7277	923.9888	908.8561	933.8604
9	921.3746	918.7445	924.0047	908.8726	933.8766

10	917.7858	915.0482	920.5235	905.2607	930.3109
11	927.0284	923.5134	930.5435	914.3108	939.7461
12	927.3615	924.4054	930.3176	914.7868	939.9362
13	912.0128	907.7923	916.2333	899.0824	924.9433
14	902.0521	893.5161	910.5881	887.1442	916.9601
15	908.5864	899.0325	918.1404	893.0732	924.0997
16	918.7002	915.9295	921.4710	906.1678	931.2326
17	918.7362	915.9733	921.4992	906.2056	931.2669
18	913.1894	909.5736	916.8052	900.4435	925.9353
19	920.5674	917.9135	923.2214	908.0604	933.0745
20	922.7657	919.9584	925.5730	910.2252	935.3062
21	916.3131	913.3776	919.2487	903.7433	928.8830
22	924.6902	921.9784	927.4019	912.1707	937.2096
23	922.9716	919.7454	926.1979	910.3307	935.6125
24	918.1897	915.2974	921.0820	905.6299	930.7496
25	916.2526	913.4377	919.0674	903.7104	928.7948
26	925.2018	922.4559	927.9478	912.6749	937.7288
27	926.2540	923.4641	929.0439	913.7174	938.7907
28	924.7008	921.9800	927.4216	912.1794	937.2223
29	925.9716	921.0551	930.8881	912.7976	939.1457
30	921.8036	914.0451	929.5621	907.3268	936.2804
31	938.6943	930.5516	946.8370	924.0080	953.3806
32	940.8266	932.0482	949.6049	925.7785	955.8746
33	933.9874	929.3741	938.6006	920.9234	947.0513
34	903.9805	896.4883	911.4726	889.6446	918.3163

RESULTS OF FITTING THE GAUSSIAN TREND SURFACE TO THE 2000 HEADS

```
Nonlinear Regression

[Variables]
x = col(1)
y = col(2)
z = col(3)
[Parameters]
x0 = xatymax(x,z) "Auto {{previous: 611012}}
y0 = xatymax(y,z) "Auto {{previous: 3.78089e+006}}
a = max(z) "Auto {{previous: 1134.61}}
b = fwhm(x,z)/2.2 "Auto {{previous: 73559.4}}
c = fwhm(y,z)/2.2 "Auto {{previous: 313474}}
[Equation]
f=a*exp(-.5*( ((x-x0)/b)^2 + ((y-y0)/c)^2 ))
fit f to z
[Constraints]
[Options]
tolerance=0.000100
stepsize=100
iterations=100

R = 0.84940930  Rsqr = 0.72149616        Adj Rsqr = 0.68668318

Standard Error of Estimate = 5.5471

        Coefficient     Std. Error      t           P
x0      611011.8967     1480.3846       412.7386    <0.0001
```

```
y0      3780891.5012    1052646.9742    3.5918  0.0011
a       1134.6135       1213.4826       0.9350  0.3568
b       73559.3533      12971.0833      5.6710  <0.0001
c       313474.4090     829108.9913     0.3781  0.7079

Analysis of Variance:
                DF      SS              MS              F       P
Regression      4       2550.8316       637.7079        20.7249 <0.0001
Residual32      984.6434        30.7701
Total   36      3535.4750       98.2076

PRESS = 22345.6338

Durbin-Watson Statistic = 1.9526

Normality Test:  Passed   (P = 0.2217)

Constant Variance Test:   Passed   (P = 0.7532)

Power of performed test with alpha = 0.0500: 1.0000

Regression Diagnostics:
Row     Predicted       Residual        Std. Res.       Stud. Res.      Stud. Del. Res.
1       932.6014        0.5934  0.1070  0.2710  0.2670
2       923.1180        -6.5698 -1.1844 -1.2771 -1.2903
3       933.2947        6.7384  1.2148  1.3332  1.3502
4       927.0594        -5.4669 -0.9855 -1.0519 -1.0537
5       926.6641        0.5284  0.0953  0.1018  0.1002
6       926.8633        -0.2405 -0.0433 -0.0464 -0.0457
7       925.0796        -7.9207 -1.4279 -1.5288 -1.5629
8       920.9577        -5.4032 -0.9741 -1.0800 -1.0829
9       930.0371        6.2218  1.1216  1.2514  1.2630
10      933.3632        0.8401  0.1514  0.1683  0.1657
11      913.0971        0.7597  0.1370  0.1853  0.1825
12      900.7080        10.8670 1.9591  2.5789  2.8519
13      920.7787        -5.3089 -0.9571 -1.0478 -1.0494
14      912.2902        2.3718  0.4276  0.5457  0.5396
15      924.5063        -4.2673 -0.7693 -0.8310 -0.8269
16      925.9052        -6.0349 -1.0879 -1.1654 -1.1722
17      917.3946        -2.0229 -0.3647 -0.4152 -0.4098
18      929.8066        7.4164  1.3370  1.4396  1.4652
19      924.2918        -7.1656 -1.2918 -1.3848 -1.4058
20      918.4636        -3.2670 -0.5890 -0.6660 -0.6601
21      929.9850        5.3120  0.9576  1.0276  1.0285
22      931.7313        3.4423  0.6206  0.6731  0.6673
23      929.3286        6.7544  1.2176  1.3036  1.3186
24      928.5840        4.0766  0.7349  0.7854  0.7806
25      927.7147        -0.7113 -0.1282 -0.1369 -0.1348
26      928.3424        2.6150  0.4714  0.5036  0.4976
27      929.7111        2.9889  0.5388  0.6054  0.5993
28      921.9985        -0.9374 -0.1690 -0.1921 -0.1892
29      944.4095        -3.4013 -0.6132 -3.9904 -5.5410
30      904.9051        0.4562  0.0822  0.2244  0.2211
31      941.4068        -4.5235 -0.8155 -1.0984 -1.1021
32      930.2232        5.4162  0.9764  1.0514  1.0532
33      930.7921        8.0232  1.4464  1.5579  1.5951
```

```
34      929.4395         6.4514  1.1630  1.2483  1.2597
35      924.2540        -6.7656 -1.2197 -1.3079 -1.3231
36      924.0527        -6.8350 -1.2322 -1.3245 -1.3409
37      925.1606        -5.1385 -0.9264 -0.9965 -0.9964

Influence Diagnostics:
Row     Cook'sDist       Leverage        DFFITS
1       0.0796  0.8442  0.6215
2       0.0531  0.1399  -0.5204
3       0.0727  0.1698  0.6106
4       0.0308  0.1222  -0.3932
5       0.0003  0.1238  0.0377
6       0.0001  0.1289  -0.0176
7       0.0684  0.1277  -0.5979
8       0.0535  0.1866  -0.5186
9       0.0767  0.1967  0.6250
10      0.0013  0.1903  0.0803
11      0.0057  0.4540  0.1664
12      -3.6351 1.5771  (+inf)
13      0.0436  0.1656  -0.4676
14      0.0375  0.3861  0.4279
15      0.0230  0.1430  -0.3377
16      0.0401  0.1286  -0.4502
17      0.0102  0.2286  -0.2230
18      0.0661  0.1375  0.5850
19      0.0572  0.1299  -0.5431
20      0.0247  0.2179  -0.3484
21      0.0320  0.1316  0.4003
22      0.0160  0.1501  0.2804
23      0.0497  0.1276  0.5042
24      0.0175  0.1244  0.2942
25      0.0005  0.1224  -0.0503
26      0.0072  0.1236  0.1869
27      0.0192  0.2078  0.3069
28      0.0022  0.2262  -0.1023
29      -138.0564       1.0236  (+inf)
30      0.0650  0.8657  0.5614
31      0.1965  0.4488  -0.9945
32      0.0353  0.1376  0.4208
33      0.0778  0.1381  0.6384
34      0.0474  0.1319  0.4911
35      0.0513  0.1303  -0.5122
36      0.0545  0.1345  -0.5286
37      0.0312  0.1358  -0.3950

95% Confidence:
Row     Predicted        Regr. 5%        Regr. 95%       Pop. 5%         Pop. 95%
1       932.6014         922.2201        942.9827        917.2573        947.9454
2       923.1180         918.8917        927.3444        911.0544        935.1816
3       933.2947         928.6391        937.9502        921.0741        945.5152
4       927.0594         923.1093        931.0095        915.0898        939.0290
5       926.6641         922.6887        930.6394        914.6861        938.6420
6       926.8633         922.8069        930.9196        914.8582        938.8683
7       925.0796         921.0422        929.1170        913.0809        937.0783
8       920.9577         916.0772        925.8381        908.6496        933.2657
9       930.0371         925.0259        935.0482        917.6767        942.3975
```

10	933.3632	928.4346	938.2918	921.0360	945.6904
11	913.0971	905.4839	920.7103	899.4725	926.7217
12	900.7080	886.5185	914.8975	882.5694	918.8466
13	920.7787	916.1801	925.3773	908.5797	932.9777
14	912.2902	905.2695	919.3109	898.9876	925.5927
15	924.5063	920.2338	928.7788	912.4265	936.5861
16	925.9052	921.8540	929.9565	913.9019	937.9086
17	917.3946	911.9928	922.7964	904.8707	929.9185
18	929.8066	925.6171	933.9961	917.7559	941.8573
19	924.2918	920.2199	928.3638	912.2815	936.3022
20	918.4636	913.1893	923.7378	905.9942	930.9330
21	929.9850	925.8868	934.0832	917.9657	942.0043
22	931.7313	927.3535	936.1091	919.6138	943.8488
23	929.3286	925.2930	933.3642	917.3305	941.3267
24	928.5840	924.5993	932.5687	916.6029	940.5651
25	927.7147	923.7623	931.6672	915.7443	939.6851
26	928.3424	924.3696	932.3153	916.3653	940.3196
27	929.7111	924.5608	934.8615	917.2936	942.1286
28	921.9985	916.6246	927.3724	909.4866	934.5104
29	944.4095	932.9778	955.8411	928.3362	960.4828
30	904.9051	894.3920	915.4182	889.4716	920.3386
31	941.4068	933.8370	948.9765	927.8064	955.0071
32	930.2232	926.0312	934.4151	918.1716	942.2748
33	930.7921	926.5937	934.9905	918.7383	942.8459
34	929.4395	925.3356	933.5435	917.4183	941.4608
35	924.2540	920.1751	928.3329	912.2413	936.2668
36	924.0527	919.9090	928.1964	912.0179	936.0876
37	925.1606	920.9964	929.3249	913.1187	937.2026

Appendix 3

EXAMPLE SOURCE FOR *ADD_TREND.C*

```
#include <stdio.h>
#include <math.h>
#include <string.h>

/*
   Sean A. McKenna                                        June 2002
   Geohydrology Department
   Sandia National Laboratories
   Albuquerque, NM  87185-0735

   ph: 505 844-2450
   em: samcken@sandia.gov

   Code to read in a single GeoEAS Formatted output file from kt3d
where the
   first column is a kriged residual field and the second column is the
   kriging variance.  This file then adds a trend surface to the
residuals
   and writes a new file of the trend+residuals and the kriging
variance in
   GeoEAS format.

*/
/*---------------------------------------------------------*/
char *read_line (fp)
FILE *fp;
{
    static char string[256];

   /*  This routine reads in a line of data from the given
       inout stream.  It however returns only lines that do
       not start with an '!', this symbol is used to denote a
       comment line.  The maximum line length is 256 characters.*/

    string[0] = '\0';
    do
      fgets (string, 256, fp);
      while ((string[0] == '!') && !feof (fp));

      return (string);
}

/*---------------------------------------------------------*/

main ()
{
      FILE *stream1,*stream2;
      char string[256],title[80],value_title[80],file1[80],file2[80];
      int i,j,nx,ny,data_col;
      double resid,krig_var,currx,curry,y0,x0,coeff_a,coeff_b,coeff_c;
      double delx,dely,o_x,o_y,trend,first,second;

      /* set constants */
      nx = 447;
      ny = 613;
      delx = 50.0;
      dely = 50.0;
      o_x = 601700.0;
      o_y = 3566500.0;

            x0 = 626195.36;
      y0 = 4149817.94;
      coeff_a = 1323.29;
      coeff_b = 163929.49;
      coeff_c = 674926.86;


      /* open input and output files */
    printf ("Enter the name of the GeoEAS formatted residual file
\n");
    gets (file1);
    stream1 = fopen(file1,"r");

    printf ("Enter the name of the GeoEAS formatted output file  \n");
    gets (file2);
    stream2 = fopen(file2,"w");


    /* Read and Write file header information */
      sprintf (string, "%s", read_line (stream1));
    sscanf (string, "%s", &title);
    sprintf (string, "%s", read_line (stream1));
    sscanf (string, "%d", &data_col);
      sprintf (string, "%s", read_line (stream1));
    sscanf (string, "%s", &value_title);
      sprintf (string, "%s", read_line (stream1));
    sscanf (string, "%s", &value_title);

    fprintf (stream2,"Starting Head Field\n");
      fprintf (stream2,"2\n");
      fprintf (stream2,"Trend plus residual\n");
      fprintf (stream2,"Kriging Variance\n");


      /* read in residuals, calculate and add trend surface, write
output */
      for (j=1;j<=ny;j++) {
        curry = (o_y+(float)j*dely)-(dely/2.0);
        for (i=1;i<=nx;i++) {
          currx = (o_x+(float)i*delx)-(delx/2.0);
                fscanf (stream1,"%lf %lf",&resid, &krig_var);
            if (resid < 1.0E-09) resid = 0.0;
            first  = ((currx-x0)/coeff_b)*((currx-x0)/coeff_b);
            second = ((curry-y0)/coeff_c)*((curry-y0)/coeff_c);
                        trend = coeff_a*exp(-0.5*(first+second));
            if ((i==1)&&(j<=10))
```

```
                              printf ("j = %3d, trend = %7.2f X = %9.1f
Y = %9.1f\n",
                                  j, trend,currx,curry);
            fprintf (stream2," %7.2lf  %7.3lf\n",
(trend+resid),krig_var);
        }
      }

            fclose (stream1);
      fclose (stream2);

}
```

Appendix 4: xform source code

Description:
The program **xform** was written to log transform a MODFLOW formatted array. It has the ability to transform between formats or to perform simple log-10 transforms on the array, moving the array in and out of log-space. This last function is what was used in the model process.

Input:
A MODFLOW formatted array (typically hydraulic conductivity or transmissivity)

Output:
A MODFLOW formatted array after transform

Platform:
1.9 GHz AMD Athlon, Red Hat Linux 7.2

Program Execution:
xform <file1> [mod] <file2> [mod] [log | none | real]

Source Files:
- xform.c (Attached)
- includes.h (See addmods)
- bool.h (See addmods)
- bool.c (See addmods)
- Globals.h (See addmods)
- Grid_Util.h (See addmods)
- Grid_Util.c (See addmods)
- Check_Flags.h (See addmods)
- Read_Files.h (See addmods)
- Write_Files.h (See addmods)
- Read_Files.c (See addmods)
- Write_Files.c (See addmods)

Program Listings: xform.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "includes.h"

void printErr(void)
{
    printf("Please enter:  xform <infile> <format> <outfile> <format>
<xform> [--head n]\n");
    exit(-1);
```

INFORMATION ONLY

```
}

int main (int argc, char *argv[])
{
  int i,nHead;
  double data[274011];
  char inType, outType, XForm;
  char *inFile, *outFile;
  nx = 447;
  ny = 613;
  nz = 1;
  delx = 50.0;
  dely = 50.0;
  delz = 7.75;
  xO = 601700.0;
  yO = 3566500.0;
  zO = 900.0;
  if (argc < 6) printErr();
  for (i = 0; i < 274011; i++) data[i] = 0;
  if (strcmp(argv[2],"gs")==0) inType = 'G';
  else if (strcmp(argv[2],"mod")==0) inType = 'M';
  else if (strcmp(argv[2],"srf")==0) inType = 'S';
  else if (strcmp(argv[2],"ai")==0) inType = 'A';
  else if (strcmp(argv[2],"xyzd")==0) inType = 'L';
  else printErr();
  if (strcmp(argv[4],"mod")==0) outType = 'M';
  else if (strcmp(argv[4],"srf")==0) outType = 'S';
  else if (strcmp(argv[4],"ai")==0) outType = 'A';
  else if (strcmp(argv[4],"gs")==0) outType = 'G';
  else printErr();
  if (strcmp(argv[5],"log")==0) XForm = 'L';
  else if (strcmp(argv[5],"real")==0) XForm = 'P';
  else if (strcmp(argv[5],"pow")==0) XForm = 'P';
  else if (strcmp(argv[5],"none")==0) XForm= 'N';
  else printErr();
  inFile = argv[1];
  outFile = argv[3];
  if (inFile == NULL || outFile == NULL) printErr();
  if (inType == 'G' || inType == 'A' || outType == 'G') {
    if (strcmp(argv[6],"--head") != 0) printErr();
    if (argc < 8) printErr();
    nHead = atoi(argv[7]);
  }
  switch (inType) {
  case 'G':
    Read_GS(inFile,data,1,1,nHead);
    break;
  case 'M':
    Read_MOD(inFile,data);
    break;
  case 'S':
    Read_SRF(inFile,data);
    break;
  case 'A':
    Read_AI(inFile,data,1,1,nHead);
    break;
    case 'L':
      Read_XYZD(inFile,data);
      break;
    }
    switch (XForm) {
    case 'L':
      for (i = 0; i < 274011; i++) data[i] = log10(data[i]);
      break;
    case 'P':
      for (i = 0; i < 274011; i++) data[i] = pow(10,data[i]);
      break;
    }
    switch (outType) {
    case 'M':
      Write_MOD(outFile,data);
      break;
    case 'S':
      Write_SRF(outFile,data);
      break;
    case 'G':
      Write_GS(outFile,data,nHead);
      break;
    case 'A':
      Write_AI(outFile,data,1);
      break;
    }
}
```

Appendix 5: Example Shell Script for Forward Runs

This is the *b01r03.sh* shell used to accomplish the forward runs using base T field number 3.

```
echo STARTING TO PROCESS
time mf2k b01r03_1980
mv -f b01r03.lst b01r03_1980.lst
time dtrkcdb control.inp b01r03_1980.bud b01r03_1980.trk dtrk1980.dbg
time cat *1980.trk | awk '{printf("%8.2f\t%8.2f\t%8.2E years\t%8.2E
   m\n",(601700+(50.0*$2)),(3597100-(50.0*$3)),$1,$6)}' >
   part_b01r03_1980.lbl
time get-heads heads_b01r03_1980.out measured_head_1980.xyz
   calc_heads_b01r03.1980
time mod2srf heads_b01r03_1980.out heads_b01r03_1980.srf
time mf2k b01r03_1990
mv -f b01r03.lst b01r03_1990.lst
time dtrkcdb control.inp b01r03_1990.bud b01r03_1990.trk dtrk1990.dbg
time cat *1990.trk | awk '{printf("%8.2f\t%8.2f\t%8.2E years\t%8.2E
   m\n",(601700+(50.0*$2)),(3597100-(50.0*$3)),$1,$6)}' >
   part_b01r03_1990.lbl
time get-heads heads_b01r03_1990.out measured_head_1990.xyz
   calc_heads_b01r03.1990
time mod2srf heads_b01r03_1990.out heads_b01r03_1990.srf
time mf2k b01r03_2000
mv -f b01r03.lst b01r03_2000.lst
time dtrkcdb control.inp b01r03_2000.bud b01r03_2000.trk dtrk2000.dbg
time cat *2000.trk | awk '{printf("%8.2f\t%8.2f\t%8.2E years\t%8.2E
   m\n",(601700+(50.0*$2)),(3597100-(50.0*$3)),$1,$6)}' >
   part_b01r03_2000.lbl
time get-heads heads_b01r03_2000.out measured_head_2000.xyz
   calc_heads_b01r03.2000
time mod2srf heads_b01r03_2000.out heads_b01r03_2000.srf
time mf2k b01r03_CCA
mv -f b01r03.lst b01r03_CCA.lst
time dtrkcdb control.inp b01r03_CCA.bud b01r03_CCA.trk dtrkCCA.dbg
time cat *CCA.trk | awk '{printf("%8.2f\t%8.2f\t%8.2E years\t%8.2E
   m\n",(601700+(50.0*$2)),(3597100-(50.0*$3)),$1,$6)}' >
   part_b01r03_CCA.lbl
time get-heads heads_b01r03_CCA.out measured_head_CCA.xyz
   calc_heads_b01r03.CCA
time mod2srf heads_b01r03_CCA.out heads_b01r03_CCA.srf
```

Appendix 6: Source code for get-heads program

The get-heads program is used to extract the necessary head information from MODFLOW output for comparison with measured heads.

Input Files:
- Tupdate.hed
- heads.measured

Output Files:
- heads.out

Platform:
1.9 GHz AMD Athlon, Red Hat Linux 7.2

Program Execution:
get-heads Tupdate.hed heads.measured heads.out

Source Files:
- Grab_Heads.c (Attached)
- includes.h (See addmods)
- bool.h (See addmods)
- bool.c (See addmods)
- Globals.h (See addmods)
- Grid_Util.h (See addmods)
- Grid_Util.c (See addmods)
- Read_Files.h (See addmods)
- Write_Files.h (See addmods)
- Read_Files.c (See addmods)
- Write_Files.c (See addmods)

Program Listing: Grab_Heads.c

```
#include <stdio.h>
#include <stdlib.h>
#include "bool.h"
#include "Globals.h"
#include "Grid_Util.h"
#include "Read_Files.h"
#include "Check_Flags.h"

int main(int argc, char *argv[])
{
  char *gridFile,*headFile,*outFile;
  FILE *fOUT;
 double heads[50][4],data[274011],newHeads[50];
  int i,lines;
  nx = 447;
  ny = 613;
  nz = 1;
  delx = 50.0;
  dely = 50.0;
  delz = 7.75;
  xO = 601700.0;
  yO = 3566500.0;
  zO = 900.0;
  gridFile = argv[1];
  headFile = argv[2];
  outFile = argv[3];
  if (gridFile == NULL || headFile == NULL || outFile == NULL) {
    printf("Please use the format: getHeads <mf_out_file>
<head_loc_file> <out_file>\n");
    return;
  }
  Read_MOD(gridFile,data);
  lines = Read_XYZD_Array(headFile,heads);
  fOUT = fopen(outFile,"w");
  fprintf(fOUT,"#X\t\tY\t\tZ\t\tMeasured\tCalculated\n");
  for (i = 0; i < lines; i++) {
    newHeads[i] = Get_Data(heads[i][0],heads[i][1],heads[i][2],data);

fprintf(fOUT,"%8.2f\t%8.2f\t%8.2f\t%8.2f\t%8.2f\n",heads[i][0],heads[i]
[1],heads[i][2],heads[i][3],newHeads[i]);
  }
  fclose(fOUT);

}
```

Appendix 7: Source code for the get-data program

Description:
get-data was written to extract data values from a MODFLOW readable array by x y z location data. The data it returns is in generic format, where the program **get-heads** is specifically looking for head data.

Input Files:
- Pilot-points.coord (attached)
- Modflow array

Output Files:
- Pilot-points.dat (attached)

Platform:
1.9 GHz AMD Athlon, Red Hat Linux 7.2

Program Execution:
```
get-data pilot-points.coord <file1> pilot-points.dat
```

Source Files:
- Grab_XYD.c (Attached)
- includes.h (See addmods)
- bool.h (See addmods)
- bool.c (See addmods)
- Globals.h (See addmods)
- Check_Flags.h (See addmods)
- Check_Flags.c (Attached)
- Grid_Util.h (See addmods)
- Grid_Util.c (See addmods)
- Read_Files.h (See addmods)
- Write_Files.h (See addmods)
- Read_Files.c (See addmods)
- Write_Files.c (See addmods)

Input File: Pilot-points.coord
```
612658        3567490      1       1       pp001
610600        3568940      1       1       pp002
612300        3569660      1       1       pp003
609977        3572370      1       1       pp004
606576        3578170      1       1       pp005
604278        3583430      1       1       pp006
607199        3587030      1       1       pp007
615300        3590400      1       1       pp008
611500        3590500      1       1       pp009
613440        3591110      1       1       pp010
```

608923	3591190	1	1	pp011
611300	3592250	1	1	pp012
617050	3567420	1	2	pp013
618990	3567460	1	2	pp014
621600	3567460	1	2	pp015
620280	3568160	1	2	pp016
622970	3568200	1	2	pp017
616260	3568290	1	2	pp018
617960	3568460	1	2	pp019
614770	3568820	1	2	pp020
621560	3569000	1	2	pp021
619410	3569250	1	2	pp022
616590	3569380	1	2	pp023
622680	3569770	1	2	pp024
618040	3569950	1	2	pp025
620770	3570030	1	2	pp026
613814	3570320	1	2	pp027
615520	3570320	1	2	pp028
619490	3570770	1	2	pp029
616720	3570910	1	2	pp030
622970	3571040	1	2	pp031
621640	3571120	1	2	pp032
618170	3571380	1	2	pp033
614000	3571470	1	2	pp034
612500	3571690	1	2	pp035
615430	3571690	1	2	pp036
620610	3572260	1	2	pp037
618820	3572290	1	2	pp038
616920	3572500	1	2	pp039
614070	3573010	1	2	pp040
615770	3573200	1	2	pp041
612540	3573210	1	2	pp042
615460	3574380	1	2	pp043
613900	3574400	1	2	pp044
612590	3574440	1	2	pp045
611710	3575040	1	2	pp046
613280	3575250	1	2	pp047
614570	3575450	1	2	pp048
611420	3576070	1	2	pp049
612740	3576230	1	2	pp050
614600	3576500	1	2	pp051
613640	3576620	1	2	pp052
610775	3576990	1	2	pp053
612257	3577090	1	2	pp054
610166	3577710	1	2	pp055
611800	3577810	1	2	pp056
614699	3577900	1	2	pp057
613885	3578250	1	2	pp058
614976	3578580	1	2	pp059
611573	3578650	1	2	pp060
613063	3578760	1	2	pp061
615509	3578850	1	2	pp062
609681	3578880	1	2	pp063
614377	3579010	1	2	pp064
613641	3579340	1	2	pp065
612395	3579460	1	2	pp066
610770	3579480	1	2	pp067

```
615574      3579780     1      2      pp068
614469      3579890     1      2      pp069
611620      3580160     1      2      pp070
613064      3580240     1      2      pp071
614068      3580480     1      2      pp072
613667      3580640     1      2      pp073
609056      3580810     1      2      pp074
615449      3580850     1      2      pp075
613285      3580930     1      2      pp076
607600      3581000     1      2      pp077
610610      3581080     1      2      pp078
612647      3581240     1      2      pp079
614877      3581310     1      2      pp080
613713      3581370     1      2      pp081
614167      3581370     1      2      pp082
613081      3581680     1      2      pp083
611600      3582400     1      2      pp084
614690      3582650     1      2      pp085
615676      3582710     1      2      pp086
612911      3583020     1      2      pp087
617020      3583460     1      2      pp088
610000      3583750     1      2      pp089
614154      3583780     1      2      pp090
608460      3583960     1      2      pp091
615752      3584130     1      2      pp092
611427      3584230     1      2      pp093
613425      3584760     1      2      pp094
618741      3585010     1      2      pp095
615660      3585750     1      2      pp096
613640      3586070     1      2      pp097
611050      3586110     1      2      pp098
609100      3586400     1      2      pp099
620720      3586590     1      2      pp100
619020      3587120     1      2      pp101
613700      3587500     1      2      pp102
616143      3587780     1      2      pp103
610832      3588420     1      2      pp104
618620      3588950     1      2      pp105
618693      3590810     1      2      pp106
621050      3591110     1      2      pp107
617090      3591410     1      2      pp108
615620      3592670     1      2      pp109
613800      3593400     1      2      pp110
617000      3594380     1      2      pp111
615140      3594770     1      2      pp112
612000      3596220     1      2      pp113
613700      3596250     1      2      pp114
615380      3596260     1      2      pp115
```

Output File: pilot-points.dat

```
pp001  612658      3567490      1         3.5410e-01
pp002  610600      3568940      1         2.6670e-01
pp003  612300      3569660      1        -1.2600e-01
pp004  609977      3572370      1         4.1790e-01
pp005  606576      3578170      1         9.1700e-01
```

pp006	604278	3583430	1	1.1630e-01
pp007	607199	3587030	1	2.8510e-01
pp008	615300	3590400	1	2.1920e-01
pp009	611500	3590500	1	-1.4800e-02
pp010	613440	3591110	1	3.6040e-01
pp011	608923	3591190	1	4.0260e-01
pp012	611300	3592250	1	-4.9700e-02
pp013	617050	3567420	2	3.4060e-01
pp014	618990	3567460	2	4.0160e-01
pp015	621600	3567460	2	-4.3020e-01
pp016	620280	3568160	2	2.7400e-01
pp017	622970	3568200	2	3.5930e-01
pp018	616260	3568290	2	-2.0400e-02
pp019	617960	3568460	2	1.0320e-01
pp020	614770	3568820	2	3.4250e-01
pp021	621560	3569000	2	4.1700e-02
pp022	619410	3569250	2	4.7760e-01
pp023	616590	3569380	2	2.5320e-01
pp024	622680	3569770	2	-2.2390e-01
pp025	618040	3569950	2	-3.6180e-01
pp026	620770	3570030	2	-1.1100e-01
pp027	613814	3570320	2	1.3960e-01
pp028	615520	3570320	2	2.4090e-01
pp029	619490	3570770	2	1.5430e-01
pp030	616720	3570910	2	5.6760e-01
pp031	622970	3571040	2	9.5000e-03
pp032	621640	3571120	2	-1.4460e-01
pp033	618170	3571380	2	-1.5140e-01
pp034	614000	3571470	2	-2.4250e-01
pp035	612500	3571690	2	3.3850e-01
pp036	615430	3571690	2	-2.0290e-01
pp037	620610	3572260	2	-4.8000e-03
pp038	618820	3572290	2	-4.0700e-02
pp039	616920	3572500	2	7.0290e-01
pp040	614070	3573010	2	2.8450e-01
pp041	615770	3573200	2	1.2760e-01
pp042	612540	3573210	2	-6.7600e-01
pp043	615460	3574380	2	-2.6640e-01
pp044	613900	3574400	2	-4.0000e-03
pp045	612590	3574440	2	2.7920e-01
pp046	611710	3575040	2	-1.5200e-02
pp047	613280	3575250	2	3.0080e-01
pp048	614570	3575450	2	-2.6900e-02
pp049	611420	3576070	2	-2.8930e-01
pp050	612740	3576230	2	1.6150e-01
pp051	614600	3576500	2	-9.9420e-01
pp052	613640	3576620	2	-2.8680e-01
pp053	610775	3576990	2	6.6500e-02
pp054	612257	3577090	2	3.2080e-01
pp055	610166	3577710	2	8.6800e-02
pp056	611800	3577810	2	-1.5140e-01
pp057	614699	3577900	2	-3.1720e-01
pp058	613885	3578250	2	-3.9350e-01
pp059	614976	3578580	2	5.4600e-02
pp060	611573	3578650	2	5.6900e-02
pp061	613063	3578760	2	-1.2520e-01
pp062	615509	3578850	2	-6.4400e-02

```
pp063  609681        3578880      2         3.7310e-01
pp064  614377        3579010      2        -1.9800e-02
pp065  613641        3579340      2         6.7000e-03
pp066  612395        3579460      2         5.6900e-02
pp067  610770        3579480      2         3.1600e-02
pp068  615574        3579780      2        -9.9790e-01
pp069  614469        3579890      2        -1.9660e-01
pp070  611620        3580160      2        -1.3750e-01
pp071  613064        3580240      2         5.1650e-01
pp072  614068        3580480      2        -2.8200e-01
pp073  613667        3580640      2         2.1000e-02
pp074  609056        3580810      2        -3.2500e-01
pp075  615449        3580850      2        -4.0210e-01
pp076  613285        3580930      2         1.3700e-01
pp077  607600        3581000      2        -9.7390e-01
pp078  610610        3581080      2        -1.1510e-01
pp079  612647        3581240      2         3.1750e-01
pp080  614877        3581310      2        -9.9530e-01
pp081  613713        3581370      2        -5.8200e-02
pp082  614167        3581370      2        -4.2470e-01
pp083  613081        3581680      2         5.2720e-01
pp084  611600        3582400      2         3.1320e-01
pp085  614690        3582650      2        -1.5000e-02
pp086  615676        3582710      2        -5.2860e-01
pp087  612911        3583020      2         5.5610e-01
pp088  617020        3583460      2         4.0360e-01
pp089  610000        3583750      2         4.1660e-01
pp090  614154        3583780      2        -1.4490e-01
pp091  608460        3583960      2         8.5000e-03
pp092  615752        3584130      2         2.9210e-01
pp093  611427        3584230      2        -1.4150e-01
pp094  613425        3584760      2        -1.7400e-02
pp095  618741        3585010      2         2.2760e-01
pp096  615660        3585750      2         7.2020e-01
pp097  613640        3586070      2         4.1780e-01
pp098  611050        3586110      2         2.4860e-01
pp099  609100        3586400      2        -3.3360e-01
pp100  620720        3586590      2        -1.4900e-02
pp101  619020        3587120      2        -2.3970e-01
pp102  613700        3587500      2        -1.2700e-01
pp103  616143        3587780      2         6.9540e-01
pp104  610832        3588420      2        -2.1910e-01
pp105  618620        3588950      2        -4.9700e-01
pp106  618693        3590810      2         2.3870e-01
pp107  621050        3591110      2        -6.7900e-02
pp108  617090        3591410      2         8.2580e-01
pp109  615620        3592670      2         4.1300e-01
pp110  613800        3593400      2         6.9850e-01
pp111  617000        3594380      2        -2.3200e-02
pp112  615140        3594770      2        -1.5200e-01
pp113  612000        3596220      2        -2.2210e-01
pp114  613700        3596250      2         3.4740e-01
pp115  615380        3596260      2         2.6100e-02
```

Program Listing: Grab_XYD.c

```
#include <stdio.h>
#include <stdlib.h>
#include "bool.h"
#include "Globals.h"
#include "Grid_Util.h"
#include "Read_Files.h"
#include "Check_Flags.h"

int main(int argc, char *argv[])
{
  char *coordFile,*modFile,*outFile;
  FILE *fOUT;
  double info[500][4],data[274011],newInfo[500];
  int i,lines;
  Check_Flags(argc,argv);
  coordFile = argv[1];
  modFile = argv[2];
  outFile = argv[3];
  if (coordFile == NULL || modFile == NULL || outFile == NULL) {
    printf("Please use the format: get-data <coord_file> <mod_file>
<out_file>\n");
    return;
  }
  Read_MOD(modFile,data);
  lines = Read_XYZD_Array(coordFile,info);
  fOUT = fopen(outFile,"w");
  for (i = 0; i < lines; i++) {
    newInfo[i] = Get_Data(info[i][0],info[i][1],info[i][2],data);

fprintf(fOUT,"pp%.3d\t%.0f\t%.0f\t%d\t%12.4e\n",i+1,info[i][0],info[i][
1],(int)info[i][3],newInfo[i]);
  }
}
```

Program Listing: Check_Flags.c

```
#include <stdio.h>
#include <stdlib.h>
#include "bool.h"
#include "Globals.h"

int Check_For_Flag(char *sz_arg, int argc, char *argv[])
{
  int arg_num;
  for (arg_num = 1; arg_num < argc; arg_num++)
    if (strcmp(argv[arg_num],sz_arg) == 0) return arg_num;
  return false;
}

char *szGet_Flag_Arg(int flag_num, int argc, char *argv[])
{
  if (flag_num > argc) return NULL;
  return argv[flag_num+1];
}

int iGet_Flag_Arg(int flag_num, int argc, char *argv[])
```

```
{
  if (flag_num > argc) return -9999999;
  return atoi(argv[flag_num+1]);
}

double fGet_Flag_Arg(int flag_num, int argc, char *argv[])
{
  if (flag_num > argc) return -9999999.9999;
  return atof(argv[flag_num+1]);
}

void Check_Flags(int argc, char *argv[])
{
  int flag;

  if ( (flag = Check_For_Flag("--pcg",argc,argv)) != 0 ) {
    bPCG = true;
    bAMG = false;
  } else {
    bPCG = false;
  }

  if ( (flag = Check_For_Flag("--amg",argc,argv)) != 0 ) {
    bAMG = true;
    bPCG = false;
  } else {
    bAMG = false;
  }

  bUserGrid = true;
  if ( (flag = Check_For_Flag("--nx",argc,argv)) != 0 ) {
    nx = iGet_Flag_Arg(flag,argc,argv);
  } else {
    nx = 447;
    bUserGrid = false;
  }

  if ( (flag = Check_For_Flag("--ny",argc,argv)) != 0 ) {
    ny = iGet_Flag_Arg(flag,argc,argv);
  } else {
    ny = 613;
    bUserGrid = false;
  }

  if ( (flag = Check_For_Flag("--nz",argc,argv)) != 0 ) {
    nz = iGet_Flag_Arg(flag,argc,argv);
  } else {
    nz = 1;
    bUserGrid = false;
  }

  if ( (flag = Check_For_Flag("--delx",argc,argv)) != 0 ) {
    delx = fGet_Flag_Arg(flag,argc,argv);
  } else {
    delx = 50.0;
    bUserGrid = false;
  }
```

```
if ( (flag = Check_For_Flag("--dely",argc,argv)) != 0 ) {
  dely = fGet_Flag_Arg(flag,argc,argv);
} else {
  dely = 50.0;
  bUserGrid = false;
}

if ( (flag = Check_For_Flag("--delz",argc,argv)) != 0 ) {
  delz = fGet_Flag_Arg(flag,argc,argv);
} else {
  delz = 7.75;
  bUserGrid = false;
}

if ( (flag = Check_For_Flag("--xO",argc,argv)) != 0 ) {
  xO = fGet_Flag_Arg(flag,argc,argv);
} else {
  xO = 601700.0;
  bUserGrid = false;
}

if ( (flag = Check_For_Flag("--yO",argc,argv)) != 0 ) {
  yO = fGet_Flag_Arg(flag,argc,argv);
} else {
  yO = 3566500.0;
  bUserGrid = false;
}

if ( (flag = Check_For_Flag("--zO",argc,argv)) != 0 ) {
  zO = fGet_Flag_Arg(flag,argc,argv);
} else {
  zO = 800.0;
  bUserGrid = false;
}

if ( (flag = Check_For_Flag("--problem-name",argc,argv)) != 0 ) {
  szProblem = szGet_Flag_Arg(flag,argc,argv);
  if (szProblem != NULL) bAskProblem = false;
  else {
    bAskProblem = true;
    szProblem = calloc(256,sizeof(char));
    sprintf(szProblem,"problem");
  }
} else {
  bAskProblem = true;
  szProblem = calloc(256,sizeof(char));
  sprintf(szProblem,"problem");
}

if ( (flag = Check_For_Flag("--ibound-file",argc,argv)) != 0 ) {
  szIBound = szGet_Flag_Arg(flag,argc,argv);
  if (szIBound != NULL) bUserIBound = true;
  else {
    bUserIBound = false;
    szIBound = calloc(256,sizeof(char));
    sprintf(szIBound,"%s.ibd",szProblem);
    }
  } else {
    bUserIBound = false;
    szIBound = calloc(256,sizeof(char));
    sprintf(szIBound,"%s.ibd",szProblem);
  }

  if ( (flag = Check_For_Flag("--initial-heads",argc,argv)) != 0 ) {
    szIHeads = szGet_Flag_Arg(flag,argc,argv);
    if (szIHeads != NULL) bUserIHeads = true;
    else {
      bUserIHeads = false;
      szIHeads = calloc(256,sizeof(char));
      sprintf(szIHeads,"%s.ihead",szProblem);
    }
  } else {
    bUserIHeads = false;
    szIHeads = calloc(256,sizeof(char));
    sprintf(szIHeads,"%s.ihead",szProblem);
  }

  if ( (flag = Check_For_Flag("--trans-file",argc,argv)) != 0 ) {
    szTrans = szGet_Flag_Arg(flag,argc,argv);
    if (szTrans != NULL) bUserTrans = true;
    else {
      bUserTrans = false;
      szTrans = calloc(256,sizeof(char));
      sprintf(szTrans,"%s.trans",szProblem);
    }
  } else {
    bUserTrans = false;
    szTrans = calloc(256,sizeof(char));
    sprintf(szTrans,"%s.trans",szProblem);
  }

  if ( (flag = Check_For_Flag("--aq-top",argc,argv)) != 0 ) {
    szAQTop = szGet_Flag_Arg(flag,argc,argv);
    if (szIHeads == NULL) {
      szAQTop = calloc(256,sizeof(char));
      sprintf(szAQTop,"%s.top",szProblem);
    }
  } else {
    szAQTop = calloc(256,sizeof(char));
    sprintf(szAQTop,"%s.top",szProblem);
  }

  if ( (flag = Check_For_Flag("--aq-bot",argc,argv)) != 0 ) {
    szAQBot = szGet_Flag_Arg(flag,argc,argv);
    if (szAQBot == NULL) {
      szAQBot = calloc(256,sizeof(char));
      sprintf(szAQBot,"%s.bot",szProblem);
    }
  } else {
    szAQBot = calloc(256,sizeof(char));
    sprintf(szAQBot,"%s.bot",szProblem);
  }
  if ( (flag = Check_For_Flag("--flow-file",argc,argv)) != 0 ) {
    szFlow = szGet_Flag_Arg(flag,argc,argv);
    if (szFlow == NULL) {
      szFlow = calloc(256,sizeof(char));
      sprintf(szFlow,"/h/dbhart/wipp/data/flow.ai");
    }
  } else {
    szFlow = calloc(256,sizeof(char));
    sprintf(szFlow,"/h/dbhart/wipp/data/flow.ai");
  }

  if ( (flag = Check_For_Flag("--budget-file",argc,argv)) != 0 ) {
    szBudget = szGet_Flag_Arg(flag,argc,argv);
    if (szBudget == NULL) {
      szBudget = calloc(256,sizeof(char));
      sprintf(szBudget,"%s.bud",szProblem);
    }
  } else {
    szBudget = calloc(256,sizeof(char));
    sprintf(szBudget,"%s.bud");
  }

  if ( (flag = Check_For_Flag("--heads-out",argc,argv)) != 0 ) {
    szOHeads = szGet_Flag_Arg(flag,argc,argv);
    if (szOHeads == NULL) {
      szOHeads = calloc(256,sizeof(char));
      sprintf(szOHeads,"%s_out.hed",szProblem);
    }
  } else {
    szOHeads = calloc(256,sizeof(char));
    sprintf(szOHeads,"%s_out.hed");
  }

  if ( (flag = Check_For_Flag("--defaults",argc,argv)) != 0 ) {
    if (!bPCG && !bAMG) bAMG = true;
    bAskSolver = false;
    bAskProblem = false;
    bUserTrans = true;
    bUserIBound = true;
    bUserIHeads = true;
    bUserGrid = true;
  }
}
```

Appendix 8: ppk2fac program

Description:
The program **ppk2fac** is a standard utility that comes with **PEST**. It takes the grid data along with a variogram structure and a list of pilot points to produce a kriging table for other **PEST** utilities. As defined by the variogram and the pilot points, the algebraic formula determining the hydraulic conductivity (K) for every point in the model grid is developed. This table, stored in *factor.inf*, only needs to be calculated once for each combination of pilot point locations, variograms, and grid size. **ppk2fac** also generates a table of standard deviations, and the algebraic regularization equations describing the relationship between pilot points.

The **ppk2fac** program is one of the utility codes tested as part of the software qualification of the **PEST** code. Therefore, the source code is not provided here for additional review.

Input:
- ppk2fac.in (attached)
- pilot-points.dat (See Pilot Points)
- files.fig (attached)
- settings.fig (attached)
- culebra.spc (attached)
- zones.inf (attached)
- variogram.str (attached)

Output:
- factor.inf (attached)
- stdev.inf (unused)
- regular.inf (attached)

Data Sources:
See Pilot Points.

Platform:
1.9 GHz AMD Athlon, Red Hat Linux 7.2

Program Execution:
```
ppk2fac < ppk2fac.in
```

Input File: ppk2fac.in

```
pilot-points.dat
0
zones.inf
```

```
variogram.str

culebra
o
3000
1
99
culebra
o
3000
1
99
factor.inf
f
stdev.inf
f
regular.inf
```

Input File: culebra.spc

```
613 447
601700 3597100 0.0
447*50.0
613*50.0
```

Input File: files.fig

```
grid_specification_file=culebra.spc
pilot_points_file=points.dat
```

Input File: settings.fig

```
date=yyyy/mm/dd
colrow=no
```

Input File: variogram.str

```
STRUCTURE culebra
  NUGGET  8.0E-3
  TRANSFORM  none
  NUMVARIOGRAM  2
  VARIOGRAM var1 3.3E-2
  VARIOGRAM var2 6.7E-2
END STRUCTURE

VARIOGRAM var1
  VARTYPE 1
  BEARING 0.0
  A 500
  ANISOTROPY 1.0
```

```
END VARIOGRAM

VARIOGRAM var2
  VARTYPE 2
  BEARING 0.0
  A 1500
  ANISOTROPY 1.0
END VARIOGRAM
```

Input File: zones.inf
Please see attachment

Output File: factor.inf
Please see attachment

Output File: regular.dat
Please see attachment

Appendix 9: fac2real program

The **ppk2fac** program is one of the standard utility programs that comes with the **PEST** software package. The program **fac2real** is used to take the output of **ppk2fac** and transform it into a **MODFLOW** readable array. This uses both the *factor.inf* file output from **ppk2fac** and the *points.dat* file generated by **PEST** to assign actual values to every point in the grid. This process is repeated each time the pilot points are updated.

The **ppk2fac** program is one of the utility codes tested as part of the software qualification of the **PEST** code. Therefore, the source code is not provided here for additional review.

Input Files

- factor.inf (See ppk2fac)
- points.dat (See Pilot Points)
- lower.inf (Attached)
- upper.inf (Attached)
- settings.fig (See ppk2fac)
- files.fig (See ppk2fac)

Output Files

- residT.log.mod

Data Sources:
See Pilot Points.

Platform:
1.9 GHz AMD Athlon, Red Hat Linux 7.2

Program Execution:
```
fac2real < fac2real.in
```

Input File: fac2real.in

```
factor.inf
f
points.dat
a
lower.inf
f
a
upper.inf
f
residT.log.mod
```

```
f
0.0
```

Input File: lower.inf
Please see Attached

Input File: upper.inf
Please see Attached

Output File: residT.log.mod
Please see attached electronic files

Appendix 10: Source Code for the addmods program

Description:
The addmods program is a short C code written to add two **MODFLOW** formatted arrays together. It is used in the model process to add together the true and residual fields. The program **addmods** reads in two MODFLOW formatted arrays, adds them together, and outputs the final values to another MODFLOW readable array. The grid dimensioning variables relating to the culebra transmissivity field grid were hard-coded into the program.

Input:
Two MODFLOW-readable files.

Ouput:
A MODFLOW-readable array.

Platform:
1.9 GHz AMD Athlon, Red Hat Linux 7.2

Program Execution:
addmods inputfile1 inputfile2 outputfile

Source Files:
- addmods.c (Attached)
- includes.h (Attached)
- bool.h (Attached)
- bool.c (Attached)
- Globals.h (Attached)
- Grid_Util.h (Attached)
- Grid_Util.c (Attached)
- Check_Flags.h (Attached)
- Read_Files.h (Attached)
- Write_Files.h (Attached)
- Read_Files.c (Attached)
- Write_Files.c (Attached)

Program Listing: addmods.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "includes.h"

void printErr(void)
{
    printf("Please enter:  addmods <infile1> <infile2> <outfile>\n");
    exit(-1);
}

int main (int argc, char *argv[])
{
  int i,nHead;
  double data1[274011], data2[274011];
  char inType, outType, XForm;
  char *inFile1, *inFile2, *outFile;
  nx = 447;
  ny = 613;
  nz = 1;
  delx = 50.0;
  dely = 50.0;
  delz = 7.75;
  xO = 601700.0;
  yO = 3566500.0;
  zO = 900.0;
  if (argc < 4) printErr();
  for (i = 0; i < 274011; i++) data1[i] = data2[i] = 0;
  inFile1 = argv[1];
  inFile2 = argv[2];
  outFile = argv[3];
  if (inFile1 == NULL || inFile2 == NULL || outFile == NULL)
printErr();
  Read_MOD(inFile1,data1);
  Read_MOD(inFile2,data2);
  for (i = 0; i < 274011; i++) data1[i] += data2[i];
  Write_MOD(outFile,data1);
}
```

Program Listing: includes.h

```
#include "bool.h"
#include "Globals.h"
#include "Grid_Util.h"
#include "Check_Flags.h"
#include "Read_Files.h"
#include "Write_Files.h"
```

Program Listing: bool.h

```
typedef unsigned short bool;
const bool true, false;
```

Program Listing: bool.c

```
#include "bool.h"
const bool true = 1;
const bool false = 0;
```

Program Listing: Globals.h

```
#define BOOL unsigned short
int nx, ny, nz;          // These are the gloabl variables for grid
size
double delx, dely, delz; // ... grid spacing
double xO, yO, zO;       // ... grid origins capital O, not 0.

// These are MF2K specific variables
char *szProblem, *szIBound, *szTrans, *szIHeads, *szAQTop;
char *szAQBot, *szFlow, *szBudget, *szOHeads;
BOOL bPCG, bAMG;
BOOL bAskSolver, bAskProblem, bUserTrans, bUserIBound;
BOOL bUserIHeads, bUserGrid;
```

Program Listing: Grid_Util.h

```
int V2D(int x, int y);
int V3D(int x, int y, int z);
double Get_Data(double x, double y, double z, double Data[]);
void Put_Data(double x, double y, double z, double d, double Data[]);
```

Program Listing: Grid_Util.c

```
#include "bool.h"
#include "Globals.h"

int V2D(int x, int y)
{
  return ( ( (y) * nx ) + x );
}

int V3D(int x, int y, int z)
{
  return ( ( (z) * (nx * ny) ) + ( ( (y) * nx ) + x ) );
}

double Get_Data(double x, double y, double z, double Data[])
{
  int xid, yid, zid;
  xid = (x - xO + (delx/2) )/delx; // calculate the nearest grid point
  yid = (y - yO + (dely/2) )/dely; // the g_delD/2 allows for round-up
  zid = (z - zO + (delz/2) )/delz; // for floats above .5, instead of
down
  if (nz < 2) zid = 0;
  if (xid > nx || yid > ny || zid > nz) return -99999999999.99999;
  if (xid < 0 || yid < 0 || zid < 0) return -99999999999.99999;
  return Data[V3D(xid,yid,zid)];
}

void Put_Data(double x, double y, double z, double d, double Data[])
{
  int xid, yid, zid;
  xid = (x - xO + (delx/2))/delx; // calculate the nearest grid point
  yid = (y - yO + (dely/2))/dely; // the g_delD/2 allows for round-up
  zid = (z - z0 + (delz/2))/delz; // for floats above .5, instead of
down
  if (nz < 2) zid = 0;
  if (xid > nx || yid > ny || zid > nz) return;
  if (xid < 0 || yid < 0 || zid < 0) return;
  Data[V3D(xid,yid,zid)] = d;
}
```

Program Listing: Check_Flags.h

```
int Check_For_Flag(char *sz_arg, int argc, char *argv[]);
char *szGet_Flag_Arg(int flag_num, int argc, char *argv[]);
int iGet_Flag_Arg(int flag_num, int argc, char *argv[]);
double fGet_Flag_Arg(int flag_num, int argc, char *argv[]);
void Check_Flags(int argc, char *argv[]);
```

Program Listing: Read_Files.h

```
bool Read_GS(char *sz_F_in, double data[], int ncol, int datacol, int
nhead);
bool Read_AI(char *sz_F_in, double data[], int ncol, int datacol, int
nhead);
bool Read_MOD(char *sz_F_in, double data[]);
bool Read_SRF(char *sz_F_in, double data[]);
bool Read_XYZD(char *sz_F_in, double data[]);
int Read_XYZD_Array(char *sz_F_in, double data[4][]);
```

Program Listing: Read_Files.c

```
#include <stdio.h>
#include <stdlib.h>
#include "Grid_Util.h"
#include "bool.h"
#include "Globals.h"
#define BUFFSIZE 512

//////////////////////////////////////////////////////////////////////
//////
//////////////////////////////////////////////////////////////////////
//////
//  The array of data that is read in  is mapped in such a way that the
array
//  matrix looks as below, with increasing z out of screen.
//
//  y    *    *    *    *              *    *    *
// y-1   *    *    *    *              *    *    *
// y-2   *    *    *    *              *    *    *
// y-3   *    *    *    *              *    *    *
//   .
//   .
//   .
//  3    *    *    *    *              *    *    *
//  2    *    *    *    *              *    *    *
//  1    *    *    *    *              *    *    *
```

```
//(0,0)  1    2    3    4    .    .    . x-2 x-1  x
*************************/
//
//
//  This is the array format after data extraction.   The reason for
this
//  orientation is so that an cell can be indexed by (x,y,z),
regardless of
//  the order the data that was read in.  Also, indexing starts with 0

bool Read_GS(char *sz_F_in, double data[], int ncol, int datacol, int
nhead)
{
  FILE *fIN, *fOUT;
  int i,j,k,n;
  char buffer[256];
  fIN = fopen(sz_F_in,"r");
  if (fIN == NULL || data == NULL) return false;  // Error Reporting

  // header deletion
  if (nhead > 0) for (n = 0; n < nhead ; n++) fgets(buffer,
sizeof(buffer), fIN);

  // data loading
  for (k = 0; k < nz; k++) {
    for (j = 0; j < ny; j++) {
      for (i = 0; i < nx; i++) {
      for (n = 1; n <= ncol; n++) {
        fscanf(fIN, "%s", &buffer);
        if (n == datacol) data[V3D(i,j,k)] = atof(buffer); }}}}

  fclose(fIN);
  return true;
}

bool Read_AI(char *sz_F_in, double data[], int ncol, int datacol, int
nhead)
{
  FILE *fIN, *fOUT;
  int i,j,k,n;
  char buffer[256];
  fIN = fopen(sz_F_in,"r");
  if (fIN == NULL || data == NULL) return false;  // Error Reporting

  // header deletion
  if (nhead > 0) for (n = 0; n < nhead ; n++) fgets(buffer,
sizeof(buffer), fIN);

  // data loading
  for (k = 0; k < nz; k++) {
    for (j = ny-1; j >= 0; j--) {
      for (i = 0; i < nx; i++) {
      for (n = 1; n <= ncol; n++) {
        fscanf(fIN, "%s", &buffer);
        if (n == datacol) data[V3D(i,j,k)] = atof(buffer); }}}}

  fclose(fIN);
  return true;
}

bool Read_MOD(char *sz_F_in, double data[])
{
  FILE *fIN, *fOUT;
  int i,j,k,n;
  char buffer[256];
  fIN = fopen(sz_F_in,"r");
  if (fIN == NULL || data == NULL) return false;  // Error Reporting

  // data loading
  for (k = 0; k < nz; k++) {
    for (j = ny-1; j >= 0; j--) {
      for (i = 0; i < nx; i++) {
      fscanf(fIN, "%s", &buffer);
      data[V3D(i,j,k)] = atof(buffer); }}}

  fclose(fIN);
  return true;
}

bool Read_SRF(char *sz_F_in, double data[])
{
  FILE *fIN, *fOUT;
  int i,j,k,n;
  char buffer[256];
  fIN = fopen(sz_F_in,"r");
  if (fIN == NULL || data == NULL) return false;  // Error Reporting

  // read in x/y/z information
  fgets(buffer, sizeof(buffer), fIN); //erase top line header
  fscanf(fIN, "%s", &buffer);
  nx = atoi(buffer);
  fscanf(fIN, "%s", &buffer);
  ny = atoi(buffer);
  fscanf(fIN, "%s", &buffer);
  nz = atoi(buffer);
  fscanf(fIN, "%s", &buffer);
  delx = atof(buffer) / nx;
  fscanf(fIN, "%s", &buffer);
  dely = atof(buffer) / ny;
  fscanf(fIN, "%s", &buffer);
  delz = atof(buffer) / nz;
  fscanf(fIN, "%s", &buffer);
  xO = atof(buffer);
  fscanf(fIN, "%s", &buffer);
  yO = atof(buffer);
  fscanf(fIN, "%s", &buffer);
  zO = atof(buffer);

  // data loading
  for (k = 0; k < nz; k++) {
    for (j = ny-1; j >= 0; j--) {
      for (i = 0; i < nx; i++) {
      fscanf(fIN, "%s", &buffer);
      data[V3D(i,j,k)] = atof(buffer); }}}
  fclose(fIN);
  return true;
}

bool Read_XYZD(char *sz_F_in, double data[])
{
  FILE *fIN, *fOUT;
  double x,y,z,d;
  char buffer[BUFFSIZE], *strdata;
  fIN = fopen(sz_F_in,"r");
  if (fIN == NULL || data == NULL) return false;  // Error Reporting

  while ( fgets(buffer, sizeof(buffer), fIN) != NULL) {
    if ( buffer[0] != '#' ) {
      strdata = (char*)strtok(buffer,"\t ,;");
      x = atof(strdata);
      strdata = (char*)strtok(NULL,"\t ,;");
      y = atof(strdata);
      strdata = (char*)strtok(NULL,"\t ,;");
      z = atof(strdata);
      strdata = (char*)strtok(NULL,"\t ,;");
      d = atof(strdata);
      Put_Data(x,y,z,d,data);
    }
  }
  fclose(fIN);
}

int Read_XYZD_Array(char *sz_F_in, double data[][4])
{
  FILE *fIN, *fOUT;
  double x,y,z,d;
  int lines;
  char buffer[BUFFSIZE], *strdata;
  fIN = fopen(sz_F_in,"r");
  if (fIN == NULL || data == NULL) return false;  // Error Reporting
  lines = 0;
  while ( fgets(buffer, sizeof(buffer), fIN) != NULL) {
    if ( buffer[0] != '#' ) {
      strdata = (char*)strtok(buffer,"\t ,;");
      x = atof(strdata);
      strdata = (char*)strtok(NULL,"\t ,;");
      y = atof(strdata);
      strdata = (char*)strtok(NULL,"\t ,;");
      z = atof(strdata);
      strdata = (char*)strtok(NULL,"\t ,;");
      d = atof(strdata);
      data[lines][0] = x;
      data[lines][1] = y;
      data[lines][2] = z;
      data[lines][3] = d;
      lines++;
    }
  }
  fclose(fIN);
  return lines;
```

```
}
```

Program Listing: Write_Files.h

```
bool Write_MOD(char *sz_Out_File, double data[]);
bool Write_SRF(char *sz_Out_File, double data[]);
void Write_NAM(void);
void Write_DIS(void);
void Write_OC(void);
void Write_BAS6(void);
void Write_BCF6(void);
void Write_LMG(void);
void Write_PCG(void );
void Write_HED(float top_head, float gradient, float bottom_head);
void Write_IBD(void);
bool Write_GS(char *sz_Out_File, double data[], int nHead);
bool Write_AI(char *sz_Out_File, double data[], int nHead);
```

Program Listing: Write_Files.c

```
#include <stdio.h>
#include <stdlib.h>
#include "bool.h"
#include "Globals.h"

bool Write_MOD(char *sz_Out_File, double data[])
{
  FILE *fOUT;
  int i,j,k;
  fOUT = fopen(sz_Out_File, "w");
  for (k = 0; k < nz; k++) {
    for (j = ny-1; j >= 0; j--) {
      for (i = 0; i < nx; i++) {
      fprintf(fOUT,"%10.4E  ",data[V3D(i,j,k)]); }
      fprintf(fOUT,"\n"); }}
  fclose(fOUT);
}

bool Write_SRF(char *sz_Out_File, double data[])
{
  FILE *fOUT;
  int i,j,k;
  fOUT = fopen(sz_Out_File, "w");
  fprintf(fOUT,"NODE CENTERED GRID\n");
  fprintf(fOUT," %5d %5d %5d %10.2f %10.2f %10.2f %10.2f %10.2f
%10.2f\n",
        nx, ny, nz,
        (nx*delx), (ny*dely), (nz*delz),
        xO-(delx/2), yO-(dely/2), zO-(delz/2));

  for (k = 0; k < nz; k++) {
    for (j = ny-1; j >= 0; j--) {
      for (i = 0; i < nx; i++) {
      fprintf(fOUT,"%10.4E  ",data[V3D(i,j,k)]); }
      fprintf(fOUT,"\n"); }}
  fclose(fOUT);
}

bool Write_GS(char *sz_Out_File, double data[], int nHead)
{
  FILE *fOUT;
  int i,j,k;
  fOUT = fopen(sz_Out_File,"w");
  for (i = 0; i < nHead; i++) fprintf(fOUT,"GSLIB Header: nx=%d ny=%d
nz=%d\n",nx,ny,nz);
  for (k = 0; k < nz; k++) {
    for (j = 0; j < ny; j++) {
      for (i = 0; i < nx; i++) {
      fprintf(fOUT," %10.4E\n",data[V3D(i,j,k)]); }
    }}
  fclose(fOUT);
}

bool Write_AI(char *sz_Out_File, double data[], int nHead)
{
  FILE *fOUT;
  int i,j,k;
  fOUT = fopen(sz_Out_File,"w");
  for (i = 0; i < nHead; i++) fprintf(fOUT,"X, Y, Data\n");
  for (k = 0; k < nz; k++) {
    for (j = ny-1; j >= 0; j--) {
      for (i = 0; i < nx; i++) {
      fprintf(fOUT,"%.2f, %.2f,
%10.4E\n",xO+(delx*i),yO+(dely*j),data[V3D(i,j,k)]); }
    }}
  fclose(fOUT);
}

void Write_NAM(void)
{
  char szNAMfile[256];
  FILE* fpNAM;
  sprintf (szNAMfile,"%s.nam",szProblem);
  fpNAM = fopen(szNAMfile,"w");

  fprintf (fpNAM,"LIST    40    %s.lst\n", szProblem); /* listing file
*/
  fprintf (fpNAM,"DIS     41    %s.dis\n", szProblem); /*
discretization file */
  fprintf (fpNAM,"BAS6    1     %s.ba6\n", szProblem); /* basic file */
  fprintf (fpNAM,"BCF6    11    %s.bc6\n", szProblem); /* block
centered flow */
  fprintf (fpNAM,"OC      42    %s.oc\n", szProblem);  /* output
control */

  if (bPCG) fprintf (fpNAM,"PCG     9     %s.pcg\n", szProblem); /*
preconditioned conjugate gradients */
  if (bAMG) fprintf (fpNAM,"LMG     8     %s.lmg\n", szProblem); /* AMG
Solver Routine */

  fprintf (fpNAM,"data    45    %s\n", szIHeads); /* starting heads */
  fprintf (fpNAM,"data    47    %s\n", szIBound); /* IBOUND Array */
  fprintf (fpNAM,"data    30    %s\n", szTrans); /* transmissivity
field */
  fprintf (fpNAM,"data    33    %s\n", szAQTop); // top of aquifer
  fprintf (fpNAM,"data    34    %s\n", szAQBot); // bottom of aquifer
  fprintf (fpNAM,"data    17    %s\n", szOHeads);  /* OUTPUT heads */
  fprintf (fpNAM,"data(binary)    15    %s\n", szBudget);  /* budget
file for paths */
  fclose (fpNAM);
}

void Write_DIS(void)
{
  /* open and write the DIS array file */
  char szDISfile[256];
  FILE* fpDIS;
  sprintf (szDISfile,"%s.dis",szProblem);
  fpDIS = fopen(szDISfile,"w");
  fprintf (fpDIS,"# Discretization file for example problem\n");
  fprintf (fpDIS,"    1  %3d  %3d    1    1    2\n",ny,nx);  /* num
rows, num columns */
  fprintf (fpDIS,"    0\n");               /* LAYCBD Flag for bottom
layer */
  fprintf (fpDIS,"CONSTANT    %7.2f\n", dely);  /* DELR */
  fprintf (fpDIS,"CONSTANT    %7.2f\n", delx);  /* DELC */
  fprintf (fpDIS,"EXTERNAL   33  1.0  (FREE)  -1\n");   /* "Top"top of
aquifer */
  fprintf (fpDIS,"EXTERNAL   34  1.0  (FREE)  -1\n");   /* "BOTM"
bottom of aquifer */
  fprintf (fpDIS,"   1.0  1  1.0E+00   SS\n");  /* PERLEN  NSTP  TSMULT
and Ss/tr */
  fclose (fpDIS);
}

void Write_OC(void)
{
  char szOCfile[56];
  FILE* fpOC;
  sprintf (szOCfile,"%s.oc",szProblem);
  fpOC = fopen(szOCfile,"w");
  fprintf (fpOC,"head print format 0\n");
  fprintf (fpOC,"head save format (%dF10.2)\n",nx);
  fprintf (fpOC,"head save unit 17\n");
  fprintf (fpOC,"compact budget files\n\n");
  fprintf (fpOC,"period 1 step 1\n");
  fprintf (fpOC,"   save head\n");
  fprintf (fpOC,"   save budget\n");
  fclose(fpOC);
}

void Write_BAS6(void)
{
  /* open and write the BASIC file */
  char szBAS6file[256];
  FILE* fpBAS6;
  sprintf (szBAS6file,"%s.ba6",szProblem);
  fpBAS6 = fopen(szBAS6file,"w");
  fprintf (fpBAS6,"# Basic file for heterogeneous transmissivity
field\n");
  fprintf (fpBAS6,"# Initial file for 447x613 grid\n");
  fprintf (fpBAS6,"FREE\n");
  fprintf (fpBAS6,"EXTERNAL  47  1   (FREE)  -1\n");   /* IBOUND ARRAY
*/
  fprintf (fpBAS6,"-999.00\n");                       /* HNOFLO */
  fprintf (fpBAS6,"EXTERNAL  45  1.0 (FREE)  -1\n");   /* STRT Head
ARRAY */
  fclose (fpBAS6);
}

void Write_BCF6(void)
{
  /* open and write the BCF file */
  char szBCF6file[256];
  FILE* fpBCF6;
  sprintf (szBCF6file,"%s.bc6",szProblem);
  fpBCF6 = fopen(szBCF6file,"w");
  fprintf (fpBCF6,"15     999.0     0     1.0     1     0\n");
  fprintf (fpBCF6,"00\n");
  fprintf (fpBCF6,"constant  1.0\n"); // anisotropy
  fprintf (fpBCF6,"EXTERNAL  30  1.0 (FREE)  -1\n");  //transmissivity
field
  fclose(fpBCF6);
}

void Write_LMG(void)
{
  /* open and write the LMG file */
  char szLMGfile[256];
  FILE* fpLMG;
  sprintf (szLMGfile,"%s.lmg",szProblem);
  fpLMG = fopen(szLMGfile,"w");
  fprintf (fpLMG,"3.0      2.2     5.4     0 \n");
  fprintf (fpLMG,"20     50     1.0E-13    1.0    1 \n");
  fclose(fpLMG);
}

void Write_PCG(void )
{
  /* open and write the PCG file */
  char szPCGfile[256];
  FILE* fpPCG;
  sprintf (szPCGfile,"%s.pcg",szProblem);
  fpPCG = fopen(szPCGfile,"w");
  fprintf (fpPCG,"       50       30       1\n");
  fprintf (fpPCG,"  5.00E-06  1.00E-13      1.0       0      15       1
0\n");
  fclose(fpPCG);
}

void Write_HED(float top_head, float gradient, float bottom_head)
{
  /* open and write the starting heads file */
  char szHEDfile[256];
  FILE* fpHED;
```

```
    float current_head;
    int i,j;
    sprintf (szHEDfile,"%s.hed",szProblem);
    fpHED = fopen(szHEDfile,"w");

    for (i=1;i<=nx;i++)
      fprintf (fpHED," %7.3f", top_head);
    fprintf(fpHED,"\n");

    for (j=1;j<=(ny-2);j++) {
      current_head = top_head - gradient*((float)j*dely);
      for (i=1;i<=nx;i++) {
        fprintf (fpHED," %7.3f",current_head);
      }
      fprintf(fpHED,"\n");
    }

    for (i=1;i<=nx;i++)
      fprintf (fpHED,"% 7.3f", bottom_head);
    fprintf(fpHED,"\n");

    fprintf(fpHED,"      1.00          1    1.0000\n");

    fclose (fpHED);
}

void Write_IBD(void)
{
    /* open and write the IBOUND array file */
    char szIBDfile[256];
    FILE *fpIBD, *fpFLOW;
    int cell_flag,i,j;
    float flow_flag;
    sprintf (szIBDfile,"%s.ibd",szProblem);
    fpIBD = fopen(szIBDfile,"w");
    {
      fpFLOW = fopen(szFlow,"r");

      /* top line is all -1 for fixed head along top of model */
      cell_flag = -1;
      for (i=1;i<=nx;i++) {
        fscanf (fpFLOW,"%f", &flow_flag);
        fprintf (fpIBD,"%3d", cell_flag*(int)flow_flag);
      }
      fprintf(fpIBD,"\n");

      /* top-1 to bottom+1 lines are all "1" for active cells */
      //  except for edges edge -- dbhart 02
      for (j=1;j<=(ny-2);j++) {
        cell_flag = -1;
        fscanf (fpFLOW,"%f", &flow_flag);
        fprintf (fpIBD,"%3d", cell_flag*(int)flow_flag);
        for (i=2;i<nx;i++)
        {
          cell_flag = 1;
          fscanf (fpFLOW,"%f", &flow_flag);
          fprintf (fpIBD,"%3d", cell_flag*(int)flow_flag);
      }
      cell_flag = -1;
      fscanf (fpFLOW,"%f", &flow_flag);
      fprintf (fpIBD,"%3d", cell_flag*(int)flow_flag);
      fprintf(fpIBD,"\n");
    }

    /* bottom line is -1 for fixed head */
    cell_flag = -1;
    for (i=1;i<=nx;i++) {
      fscanf (fpFLOW,"%f", &flow_flag);
      fprintf (fpIBD,"%3d", cell_flag*(int)flow_flag);
    }

    fprintf(fpIBD,"\n");

    fclose(fpFLOW);
    fclose(fpIBD);
  }
}
```

Appendix 11: The model.sh shell

Description:
The model process as called by PEST needs to be a single command. Because of this, a shell script was written to call all steps of the model process. It is conveniently unnecessary to change to model script since all files are named the same once they are in different directories.

Platform:
1.9 GHz AMD Athlon, Red Hat Linux 7.2

Program Execution:
`model.sh`

Program Listing: model.sh

```
#!/bin/bash

runMF2K() {
    trap "echo 'S'" SEGV

    # Step 1:  Run FAC2REAL to get the field
    echo -n 'F'
    fac2real < fac2real.in > /dev/null


    # Step 2:  Add the residual field to the log10()Transmissivity
field
    #          to get the t-update field
    echo -n 'A'
    addmods meanT.log.mod residT.log.mod Tupdate.log.mod


    # Step 3:  Transform t-update field back into real space from log10
space
    echo -n 'X'
    xform Tupdate.log.mod mod Tupdate.mod mod real


    # Step 4:  Run modflow2k on the updated field
    echo -n 'M'
    mf2k Tupdate.nam > /dev/null


    # Step 5:  Strip out the heads
    echo -n 'G'
   get-heads Tupdate.hed heads.measured heads.out
}

runMF2K

if [ ! -e "heads.out" ]
then
```

```
    echo -n 'E'
    runMF2K
fi

if [ ! -e "heads.out" ]
then
    echo -e '\n' "MAJOR MAJOR MAJOR ERRORS!!!!!  EXITING"
    pstop
    exit
fi

echo -n '.'
```

Appendix 12: The pest-setup.sh shell

Description:
The program pest-setup.sh is used to run all the pre-processing directives, the PEST calibration model, and then the post-processing. This allows the entire sequence to be run with one command, and the output from all results can be piped to the same file. It is re-named as the realization number in each directory, allowing each directory to contain its own copy of all commands executed.

Platform:
1.9 GHz AMD Athlon, Red Hat Linux 7.2

Program Execution:
`pest-setup.sh`

Program Listing: pest-setup.sh
```
#!/bin/bash

pestRun() {

xform b*r*T.mod mod meanT.log.mod mod log

get-data pilot-points.coord sgsim.*.mod pilot-points.dat

cat pilot-points.dat | awk '{ if ($4 == 2) printf("%s  none  factor
%.4f  0.01  6.00  zone2  1.00  -3.00  1\n",$1,$5+3.0); else printf("%s
none  factor  %.4f  2.00  4.00  zone1  1.00  -3.00  1\n",$1,$5+3.0); }'
> pcf.vpp

cat fixed-points.dat >> pilot-points.dat

cat pcf.top pcf.vpp pcf.fpp pcf.bot > pestmf2k.pst

pest pestmf2k

}

ROOTDIR=/h/dbhart/wipp/pest-1980
RUNDIR=/home/scratch/temp/pest
STARTDIR=`pwd`

echo "STARTING IN DIRECTORY $STARTDIR"
echo "ROOT DIRECTORY IS $ROOTDIR"
echo "IF THIS IS WRONG, YOU HAVE 30s TO STOP RUN"
sleep 30s

if [ ! -d ${RUNDIR} ]
    then
    mkdir ${RUNDIR}
fi

if [ ! -d ${RUNDIR} ]
```

```
    then
    echo "MAJOR ERRORS IN RUN!!!!!"
    echo "NO DIRECTORY"
    echo "RUNNING PEST OVER THE NETWORK"
    RUNDIR=${STARTDIR}
fi

if [ $RUNDIR != $STARTDIR ]
then
    cp -f ${STARTDIR}/*.mod  ${RUNDIR}
    cp -f ${STARTDIR}/*.sh  ${RUNDIR}
fi

cp -f ${ROOTDIR}/def_mf2k/*  ${RUNDIR}
cp -f ${ROOTDIR}/def_pest/*  ${RUNDIR}
cp -f ${ROOTDIR}/def_gslib/*  ${RUNDIR}


cd ${RUNDIR}

CURDIR=`pwd`
if [ $RUNDIR != $CURDIR ]
    then
    echo "MAJOR ERRORS IN RUN!!!!!"
    echo "EXITING WITH ERRORS"
    echo " I AM LOST ! ! !"
    echo
    exit
fi

pestRun

tempchek points.tpl points.dat pestmf2k.par
./model.sh
cp -f ${ROOTDIR}/def_dtrk/* ${RUNDIR}
./model.sh
mv -f culebra.top fort.33
mv -f culebra.bot fort.34
dtrkcdb control.inp Tupdate.bud 1980.trk dtrk.dbg
dtrkcdb wippctrl.inp Tupdate.bud 1980-wipp.trk dtrk.dbg

mv -f ${RUNDIR}/*  ${STARTDIR}
rm ${STARTDIR}/culebra.* ${STARTDIR}/fort.* ${STARTDIR}/*.inf
${STARTDIR}/*.dbg

echo "ALL FINISHED!"
```

Chavez, Mario Joseph Mario Chavez

From:	James, Scott C
Sent:	Tuesday, May 20, 2003 10:02 AM
To:	Chavez, Mario Joseph
Subject:	RE: Task 3 AP-88 Conditioning of Base T-Fields to Steady State Heads

Hi Mario,
I give you signature authority for Task 3 AP-88 Conditioning of Base T-Fields to Steady State Heads.
Thanks,
Scott James
======================

Scott C. James, Ph.D.
Sandia National Laboratories
Geohydrology Department
P.O. Box 5800
Albuquerque, NM 87185-0735
Phone: (505) 845-7227
Fax: (505) 844-7354
======================

-----Original Message-----

From:	Chavez, Mario Joseph
Sent:	Tuesday, May 20, 2003 10:46 AM
To:	James, Scott C
Subject:	Task 3 AP-88 Conditioning of Base T-Fields to Steady State Heads

Scott,

Would you please send me signature authority on the subject document--thanks.

Mario

[illegible]

Chavez, Mario Joseph [illegible]

From:	McKenna, Sean A
Sent:	Monday, May 19, 2003 1:55 PM
To:	Chavez, Mario Joseph
Subject:	Task 3 Analysis Package

Mario, I hereby grant you authority to sign for me on the Task 3 Analysis Package.

Sean

Sean A. McKenna Ph.D.
Geohydrology Department
Sandia National Laboratories
PO Box 5800 MS 0735
Albuquerque, NM 87185-0735
ph: 505 844-2450

INFORMATION ONLY

Chavez, Mario Joseph Mario Chavez

From: Hart, David Blaine
Sent: Monday, May 19, 2003 2:13 PM
To: Chavez, Mario Joseph
Subject: Analysis Package

Mario, I hereby grant you authority to sign for me on the Task 3 Analysis Package

David

David Hart
dbhart@sandia.gov
dbhart@cc.usu.edu

INFORMATION ONLY