





564811

Analysis Report for Preparation of 2014 Culebra Potentiometric Surface Contour Map

Task Number: 4.4.2.3.1

Report Date: 9/28/2015

Author:	 _____ Kristopher L. Kuhlman, 6224 Applied Systems Analysis & Research Department	<u>9/22/2015</u> Date
Technical Review:	 _____ Michael Schuhen, 6212 Repository Performance Department	<u>9/22/15</u> Date
QA Review:	 _____ Shelly R. Nielsen, 6210 Carlsbad Programs Group	<u>9-22-15</u> Date
Management Review:	 _____ Christi D. Leigh, 6212 Manager, Repository Performance Department	<u>9-30-15</u> Date

WIPP:4.4.2.3.1:TD:QA-L:RECERT:549085

Information Only

Table of Contents

1	Introduction.....	2
2	Scientific Approach.....	3
2.1	Modeling Overview.....	3
2.2	Creating Average MODFLOW Simulation.....	6
2.3	Boundary Conditions.....	6
2.4	PEST Calibration of Averaged MODFLOW Model to Observations.....	7
2.5	Figures Generated from Averaged MODFLOW Model.....	9
3	2014 Results.....	10
3.1	2014 Equivalent Freshwater Head Contours.....	10
3.2	2014 Particle Track.....	12
3.3	2014 Measured vs. Modeled Fit.....	13
4	References.....	18
5	Run Control Narrative.....	19
6	Appendix: MODFLOW and Pest Files and Script Source Listings.....	26
6.1	Input File Listing.....	26
6.2	Output File Listing.....	27
6.3	Individual MODFLOW and Pest Script Listings.....	28

1 Introduction

This report documents the preparation of the 2014 potentiometric contour map and associated particle tracks for the Culebra Member of the Rustler Formation in the vicinity of the Waste Isolation Pilot Plant (WIPP). The driver for this analysis is the draft of the Stipulated Final Order sent to NMED on May 28, 2009 (Moody, 2009). This Analysis Report follows the procedure laid out in Sandia National Laboratories procedure SP 9-9 (Kuhlman, 2009), which reflects this NMED driver. This report is similar to Kuhlman (2014); the same analysis is performed on data from January 2014, rather than February 2013 data. January 2014 data for contouring were obtained from the WIPP Management & Operations contractor (Seal, 2015).

Beginning with the ensemble of 100 calibrated MODFLOW transmissivity (T), horizontal anisotropy (A), and areal recharge (R) fields (Hart et al., 2009) used in WIPP performance assessment (PA), average parameter fields were used as input to MODFLOW to simulate equivalent freshwater heads within and around the WIPP land withdrawal boundary (LWB). For 2014, PEST is used to adjust a subset of the boundary conditions in the averaged MODFLOW model to improve the match between the observed

freshwater heads and the model-predicted heads at Culebra well locations. The output of the averaged, PEST-calibrated MODFLOW model is both contoured and used to compute the 2014 advective particle track forward from the WIPP waste-handling shaft.

The effects of pumping at C-2492 (i.e., new Mills well) have significantly affected water levels in the Culebra monitoring network at WIPP, especially south of the WIPP facility. The procedure for adjusting the boundary conditions of the averaged MODFLOW model to match this year's observed freshwater heads cannot match the significant drawdown observed in WIPP Culebra wells (Kuhlman, 2015). We present a new figure in this report, showing the spatial distribution of the difference between the measured and modeled freshwater heads.

2 Scientific Approach

2.1 Modeling Overview

Steady-state groundwater flow simulations were carried out using similar software to what was used for the WIPP Compliance Recertification Application 2009 Performance Assessment Baseline Calculation (CRA-2009 PABC), as presented in the AP-114 Task 7 Analysis Report (Hart et al., 2009), and used in CRA-2014 (DOE, 2014). This setup was used to create the input calibrated fields. See Table 1 for a summary of software used in this analysis. The MODFLOW parameter fields (transmissivity (T), anisotropy (A), and recharge (R)) used in this analysis are ensemble averages of the 100 sets of Culebra parameter fields used for WIPP PA for the CRA-2009 PABC and CRA-2014. To clearly distinguish between the two MODFLOW models, the original MODFLOW model, which consists of 100 realizations of calibrated parameter fields (Hart et al., 2009), will be referred to as the "PA MODFLOW model." The model derived here from the PA MODFLOW model, calibrated using PEST, and used to construct the resulting contour map and particle track, is referred to as the "averaged MODFLOW model." The PA MODFLOW model T , A and R input fields are appropriately averaged across 100 realizations, producing a single averaged MODFLOW flow model. This averaged MODFLOW model was used to predict regional Culebra groundwater flow across the WIPP site.

For CRA-2009 PABC, PEST was used to construct 100 calibrated model realizations of the PA MODFLOW model by adjusting the spatial distribution of model parameters (T , A , and R); MODFLOW boundary conditions were fixed. The calibration targets for PEST in the PA MODFLOW model were both May 2007 freshwater heads (excluding AEC-7) and transient drawdown to large-scale pumping tests. Hart et al. (2009) described the calibration effort that went into the CRA-2009 PABC; DOE (2014) summarizes the model development and calibration results. An analogous but much simpler process was used here for the averaged MODFLOW model. PEST was used to modify a subset of the MODFLOW boundary conditions (see red boundaries in Figure 1). For simplicity the boundary conditions were modified (rather than the T , A , and R parameter fields), because re-calibrating the 100 T , A , and R parameter fields would be a significant effort (thousands of hours of computer time). The PEST calibration targets for the averaged MODFLOW model were the January 2014 measured annual freshwater heads at Culebra monitoring wells. In the averaged MODFLOW model, boundary conditions were modified while

holding model parameters (T , A , and R) constant. In contrast to this, the PA MODFLOW model used fixed boundary conditions and made adjustments to T , A , and R parameter fields.

Information Only

Table 1. Software

Software	Version	Description	Platform	Software QA status
MODFLOW-2000	1.07	Groundwater flow model	PA cluster	Acquired; qualified under NP 19-1 (Harbaugh et al., 2000)
PEST	9.12	Automatic parameter estimation code	PA cluster	Developed; qualified under NP 19-1 (Doherty, 2002)
DTRKMF	1.01	Particle tracker	PA cluster	Developed; qualified under NP 19-1
Python	2.7.9	Scripting language (file manipulation)	PA cluster	Commercial off the shelf
Python	2.7.6	Scripting language (plotting)	Linux desktop	Commercial off the shelf
Bash	4.1.17	Scripting language (file manipulation)	PA cluster	Commercial off the shelf

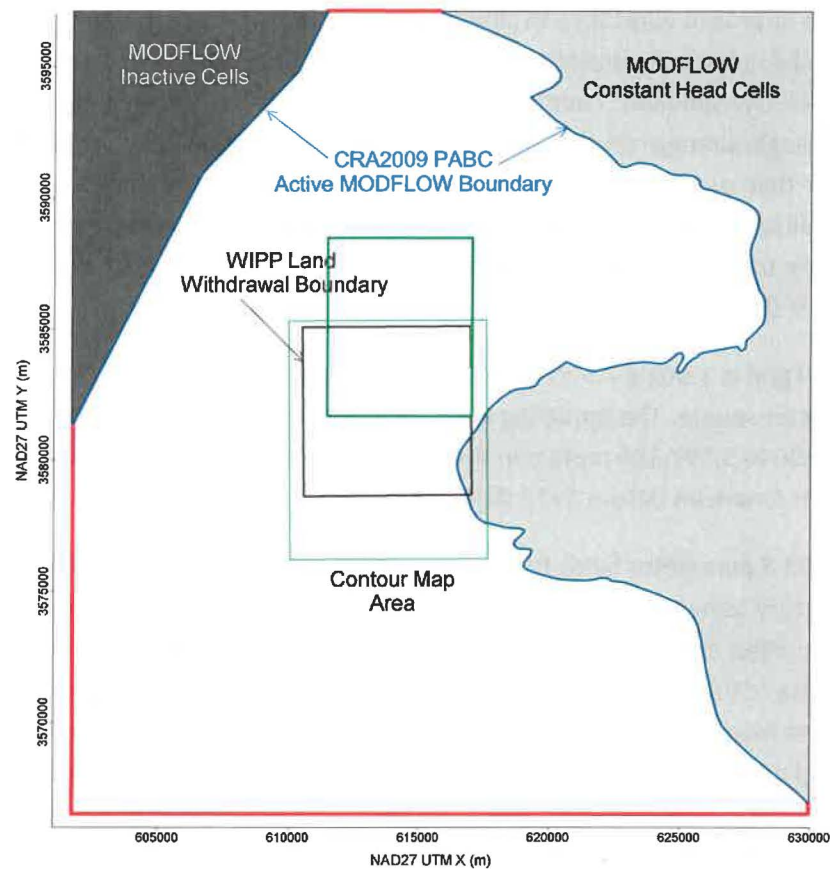


Figure 1. MODFLOW-2000 model domain, adjusted boundary conditions shown in red, contour area outlined in green.

The resulting heads from the PEST-calibrated averaged MODFLOW model were contoured over an area surrounding the WIPP site using matplotlib (a Python plotting library). The figure covers a subset of the complete MODFLOW model domain; see the green rectangle surrounding the WIPP LWB in Figure 1. We compute the path taken through the Culebra by a conservative (i.e., non-dispersive and non-reactive) particle from the waste-handling shaft to the WIPP LWB. The particle track is computed from the MODFLOW flow field using DTRKMF, these results are also plotted using matplotlib. Scatter plot

statistics were computed using NumPy (a Python array-functionality library), which summarize the quality of the fit between the averaged MODFLOW model and observed Culebra freshwater heads. MODFLOW, PEST, DTRKMF, and the Bash and Python input files and scripts written for this work were executed on the WIPP PA Solaris cluster (greenday.sandia.gov), while the creation of figures was done using Python scripts on an Intel-Xeon-equipped desktop computer running Kubuntu Linux, version 14.04.

2.2 Creating Average MODFLOW Simulation

An averaged MODFLOW model is used to compute the equivalent freshwater head and cell-by-cell flow solution. The computed heads are contoured and the flow solution is used to compute particle tracks. The ensemble-averaged inputs are used to create a single average simulation that produces a single averaged output, rather than averaging the 100 individual outputs of the Culebra flow model used for WIPP PA. This average approach was taken to simplify the contouring process, and create a single contour map that exhibits physically realistic patterns (i.e., its behavior is constrained by the physics embodied in the MODFLOW simulator code). An alternative approach would average outputs from 100 models to produce a single average result, but average result may be physically unrealistic. The choice to average inputs, rather than outputs, is a simplification (only one model must be calibrated using PEST, rather than all 100 realizations). This simplification results in “smooth” freshwater head contours and relatively faster particle tracks, compared to those predicted by the any one of the 100 fields calibrated as part of AP114 Task 7 (Hart et al., 2009).

The MODFLOW model grid is a single 7.5-m thick layer, comprising 307 rows and 284 columns; each model cell is a 100-meter square. The modeling area spans 601,700 to 630,000 meters in the east-west direction, and 3,566,500 to 3,597,100 meters in the north-south direction, both in Universal Transverse Mercator (UTM) North American Datum 1927 (NAD27) coordinates, zone 13 north.

The calibrated T , A , and R parameter fields from the PA MODFLOW model were checked out of the PA version control repository using the `checkout_average_run_modflow.sh` script (scripts are listed in the Appendix; input and output files are available from the WIPP version control system in the repository `/nfs/data/CVSLIB/Analyses/SP9_9`). Model inputs can be divided into two groups. The first group includes model inputs that are common across all 100 calibrated realizations; these include the model grid definition, the boundary conditions, and the model solver parameters. The second group includes the T , A , and R fields, which are different for each of the 100 realizations. The constant model inputs in the first group are used directly in the averaged MODFLOW model, while the inputs in the second group were averaged across all 100 calibrated model realizations using the Python script `average_realizations.py`. All three averaged parameters were geometrically averaged (i.e., the arithmetic average was computed from the \log_{10} values, which was then exponentiated to give the resulting value), since they vary over multiple orders of magnitude.

2.3 Boundary Conditions

The boundary conditions taken from the PA MODFLOW model are used as the initial condition from which PEST boundary condition calibration proceeds. There are two types of boundary conditions in the

WIPP MODFLOW model. The first type of condition includes geologic or hydrologic boundaries, which correspond to known physical features in the flow domain. The no-flow boundary along the axis of Nash Draw is a hydrologic boundary (the boundary along the dark gray region in the upper left of Figure 1). The constant-head boundary along the halite margin corresponds to a geologic boundary (the eastern irregular boundary adjoining the light gray region in the right of Figure 1). Physical boundaries are believed to be well known, and are not adjusted in this PEST calibration.

The second type of boundary condition includes the constant-head cells along the rest of the model domain. This type of boundary includes the straight-line southern, southwestern, and northern boundaries that coincide with the primary compass directions and the rectangular frame surrounding the model domain (shown as heavy red lines in Figure 1). The value of specified head assigned in boundary cells corresponds with this second boundary type and is adjusted in the PEST calibration process.

The Python script `boundary_types.py` is used to distinguish between the two different types of specified head boundary conditions based on the specified head value used in the PA MODFLOW model. All constant-head cells (specified by a value of -1 in the MODFLOW IBOUND array from the PA MODFLOW model) with a starting head value greater than 1000 meters above mean sea level (AMSL) are left fixed and not adjusted in the PEST optimization, because they correspond to no-flow constant head region to the east of WIPP. The remaining constant-head cells are distinguished by setting their IBOUND array value to -2 (which is still interpreted as a constant-head value by MODFLOW, but allows simpler discrimination between boundary conditions in Python scripts elsewhere in this analysis).

Using output from `boundary_types.py`, the Python script `surface_02_extrapolate.py` computes initial head at active model cells (IBOUND=1) and the specified constant-head at adjustable boundary condition cells (IBOUND=-2), given parameter values for the surface to extrapolate.

2.4 PEST Calibration of Averaged MODFLOW Model to Observations

There are two major types of inputs to PEST. The first input class is the “forward model”, which includes the entire MODFLOW model setup derived from the PA MODFLOW model and described in the previous section, along with any pre- or post-processing scripts or programs needed. These files comprise the forward model PEST runs repeatedly to estimate sensitivities of model outputs to model inputs. The second input type includes the PEST configuration files, which list parameter and observation groups, observation weights, and indicate which parameters in the MODFLOW model will be adjusted in the inverse simulation. Freshwater head values from January 2014 used as targets for the PEST calibration from Seal (2015) are listed in Table 2, and specified along with weights in the PEST configuration files. AEC-7R replaced previous AEC-7, but the surface casing has not been surveyed, so freshwater heads are not available from AEC-7R, and AEC-7 has been plugged and abandoned. SNL-13 was indicated in Seal (2015) as being an anomalous level, which should be excluded from mapping. Excluding this datapoint from the analysis resulted in high predicted water levels in all the wells on the southern portion of the WIPP site. Scientific judgment was used to include the well in the contour map generation exercise.

Table 2. Freshwater Head Calibration Targets used in PEST, from Seal (2015). * (and green) indicates wells not used because of long-term recovery. ‡ (and red) indicates wells significantly influenced by Mills Ranch pumping (also see wells in red in Figure 7).

Well	Measurement Date	Freshwater Head Elevation (m AMSL)	Culebra Groundwater Density (g/cm ³)
C-2737 [‡]	01/09/14	917.48	1.023
ERDA-9	01/09/14	922.27	1.071
H-02b2	01/09/14	927.54	1.013
H-03b2 [‡]	01/08/14	913.33	1.032
H-04bR [‡]	01/08/14	901.07	1.017
H-05b	01/09/14	939.13	1.092
H-06bR	01/08/14	935.67	1.039
H-07b1	01/07/14	913.37	1.007
H-09bR [‡]	01/07/14	905.87	1.001
H-10c	01/07/14	924.45	1.095
H-11b4R [‡]	01/09/14	906.64	1.076
H-12	01/07/14	914.04	1.108
H-15R [‡]	01/08/14	914.32	1.118
H-16	01/08/14	927.30	1.036
H-17 [‡]	01/09/14	907.35	1.133
H-19b0 [‡]	01/07/14	913.20	1.066
IMC-461	01/07/14	927.82	1.000
SNL-01	01/07/14	938.23	1.030
SNL-02	01/07/14	935.94	1.009
SNL-03	01/08/14	937.94	1.028
SNL-05	01/07/14	936.12	1.009
SNL-06*	01/09/14	994.50	1.243
SNL-08	01/09/14	931.01	1.095
SNL-09	01/08/14	930.36	1.018
SNL-10	01/08/14	930.03	1.010
SNL-12 [‡]	01/07/14	903.12	1.006
SNL-13	01/08/14	913.96	1.017
SNL-14 [‡]	01/08/14	904.54	1.046
SNL-15*	01/09/14	925.24	1.229
SNL-16	01/07/14	917.49	1.008
SNL-17 [‡]	01/07/14	912.09	1.005
SNL-18	01/07/14	936.39	1.009
SNL-19	01/07/14	935.94	1.007
WIPP-11	01/08/14	938.60	1.038
WIPP-13	01/08/14	937.25	1.040
WIPP-19	01/09/14	933.02	1.052
WQSP-1	01/08/14	936.82	1.049
WQSP-2	01/09/14	939.01	1.047
WQSP-3	01/07/14	936.46	1.148
WQSP-4	01/07/14	913.50	1.076
WQSP-5	01/08/14	914.49	1.027
WQSP-6	01/08/14	919.51	1.015

Information Only

To minimize the number of estimable parameters, and to ensure a degree of smoothness in the specified constant-head boundary condition values, a parametric surface is used to extrapolate the heads to the estimable boundary conditions. The surface is of the same form described in the analysis report for AP-114 Task 7. The parametric surface is given by the following equation:

$$h(x, y) = A + B(y + D \text{sign}(y) \text{abs}(y)^\alpha) + C(Ex^3 + Fx^2 - x) \quad (1)$$

where $\text{abs}(y)$ is absolute value and $\text{sign}(y)$ is the function returning 1 for $y > 0$, -1 for $y < 0$ and 0 for $y = 0$ and x and y are coordinates scaled to the range $-1 \leq \{x, y\} \leq 1$. In Hart et al. (2009), the values $A = 928$, $B = 8$, $C = 1.2$, $D = 1$, $E = 1$, $F = -1$ and $\alpha = 0.5$ are used with the above equation to assign the boundary conditions in the PA MODFLOW model.

PEST was used to estimate the values of parameters A , B , C , D , E , F , and α given the observed heads in Table 2. The Python script `surface_02_extrapolate.py` was used to compute the MODFLOW starting head input file (which is also used to specify the constant-head values) from the parameters A - F and α . Each forward run of the model corresponded to a call to the Bash script `run_02_model`. This script called the `surface_02_extrapolate.py` script, the MODFLOW-2000 executable, and the PEST utility `mod2obs`, which is used to extract and interpolate model-predicted heads from the MODFLOW output files at observation well locations.

The PEST-specific input files were generated from the observed heads using the Python script `create_pest_02_input.py`. The PEST input files include the instruction file (how to read the MODFLOW output), the template files (how to write the MODFLOW input), and the PEST control file (listing the ranges and initial values for the estimable parameters and the values and weights associated with observations). The wells used in the 2014 PEST calibration were separated into four groups. Higher observation weights (2.5) were assigned to wells inside the LWB, lower observation weights (0.4) were assigned to wells distant to the WIPP site, wells near the WIPP LWB were assigned an intermediate weight (1.0), and wells impacted by Mills Ranch pumping were assigned a low weight (0.05). Additional observations representing the average heads north of the LWB and south of the LWB were used to help prevent over-smoothing of the estimated results across the LWB. The additional observations and weights were assigned to improve the fit in the area of interest (inside the WIPP LWB), possibly at the expense of a somewhat poorer fit far from the WIPP LWB and closer to the boundary conditions.

2.5 Figures Generated from Averaged MODFLOW Model

The MODFLOW model is run predictively using the averaged MODFLOW model parameters, along with the PEST-calibrated boundary conditions. The resulting cell-by-cell flow budget is then used by DTRKMF to compute a particle track from the waste-handling shaft to the WIPP LWB. Particle tracking stops when the particle crosses the WIPP LWB. The Python script `convert_dtrkmf_output_for_surfer.py` converts the MODFLOW cell-indexed results of DTRKMF into a UTM x and y coordinate system, saving the results in the Surfer blanking file format to facilitate plotting results. The heads in the binary MODFLOW output file are converted to an ASCII matrix file format using the Python script `head_bin2ascii.py`.

The resulting particle track and contours of the model-predicted head are plotted using a matplotlib Python script for an area including the WIPP LWB, corresponding to the region shown in previous versions of the ASER (e.g., see Figure 6.11 in DOE (2008)), specifically the green box in Figure 1. The modeled heads extracted from the MODFLOW output by `mod2obs` are then merged into a common file for plotting using the Python script `merge_observed_modeled_heads.py`.

3 2014 Results

3.1 2014 Equivalent Freshwater Head Contours

The model-generated freshwater head contours are given in Figure 2 and Figure 3. There is a roughly east-west trending band of steeper gradients, corresponding to lower Culebra transmissivity. The uncontoured region to the right of the purple line in the eastern part of the figures corresponds to the portion of the Culebra that is located stratigraphically between halite and other members of the Rustler Formation (Tamarisk Member above and Los Medaños Member below). This region east of the “halite margin” has a high freshwater head but extremely low transmissivity, essentially serving as a no-flow boundary in this area.

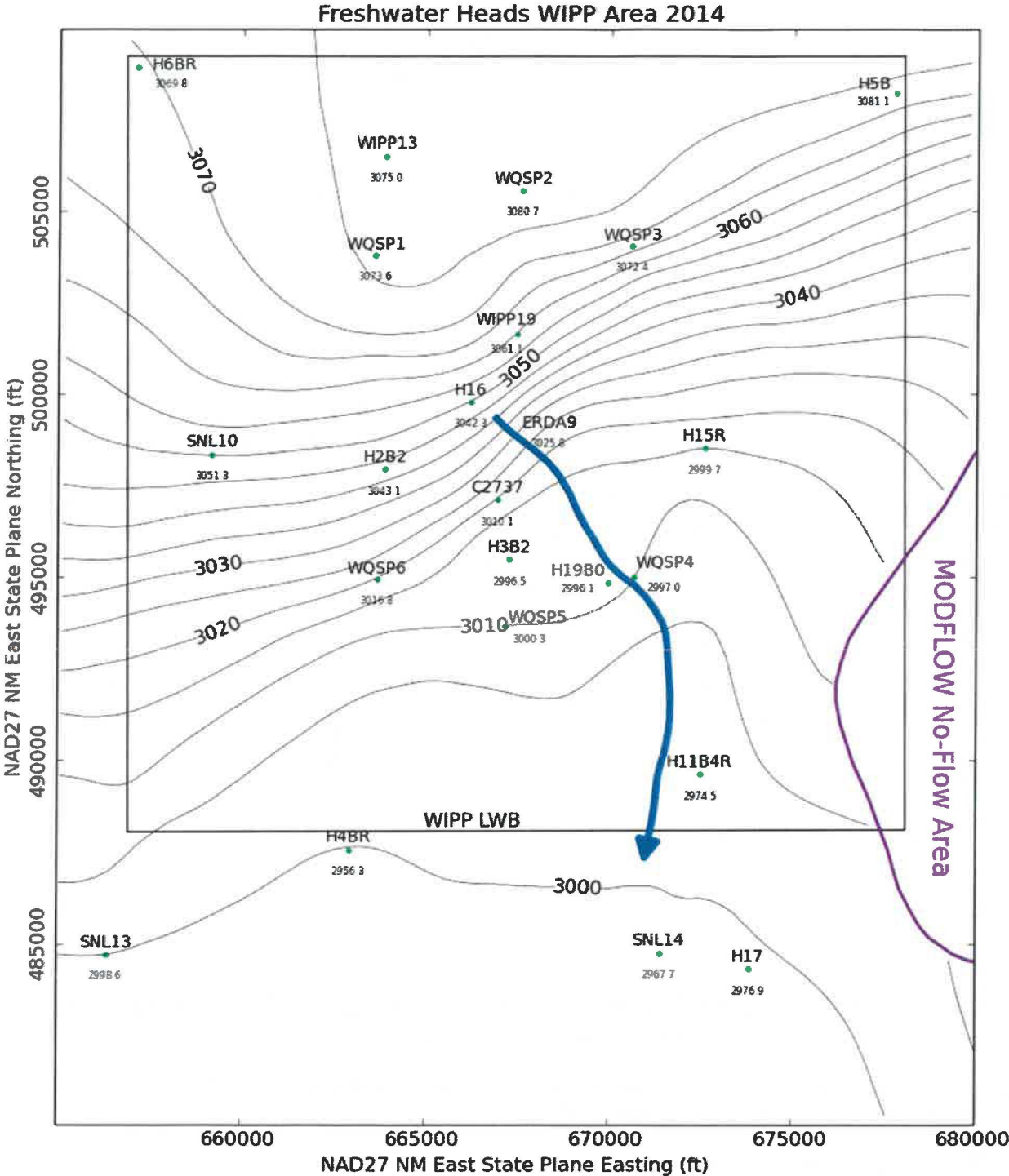


Figure 2. Model-generated January 2014 freshwater head contours with observed head listed at each well (5-foot contour interval) with blue water particle track from waste handling shaft to WIPP LWB. Purple curve is Rustler halite margin.

Information Only

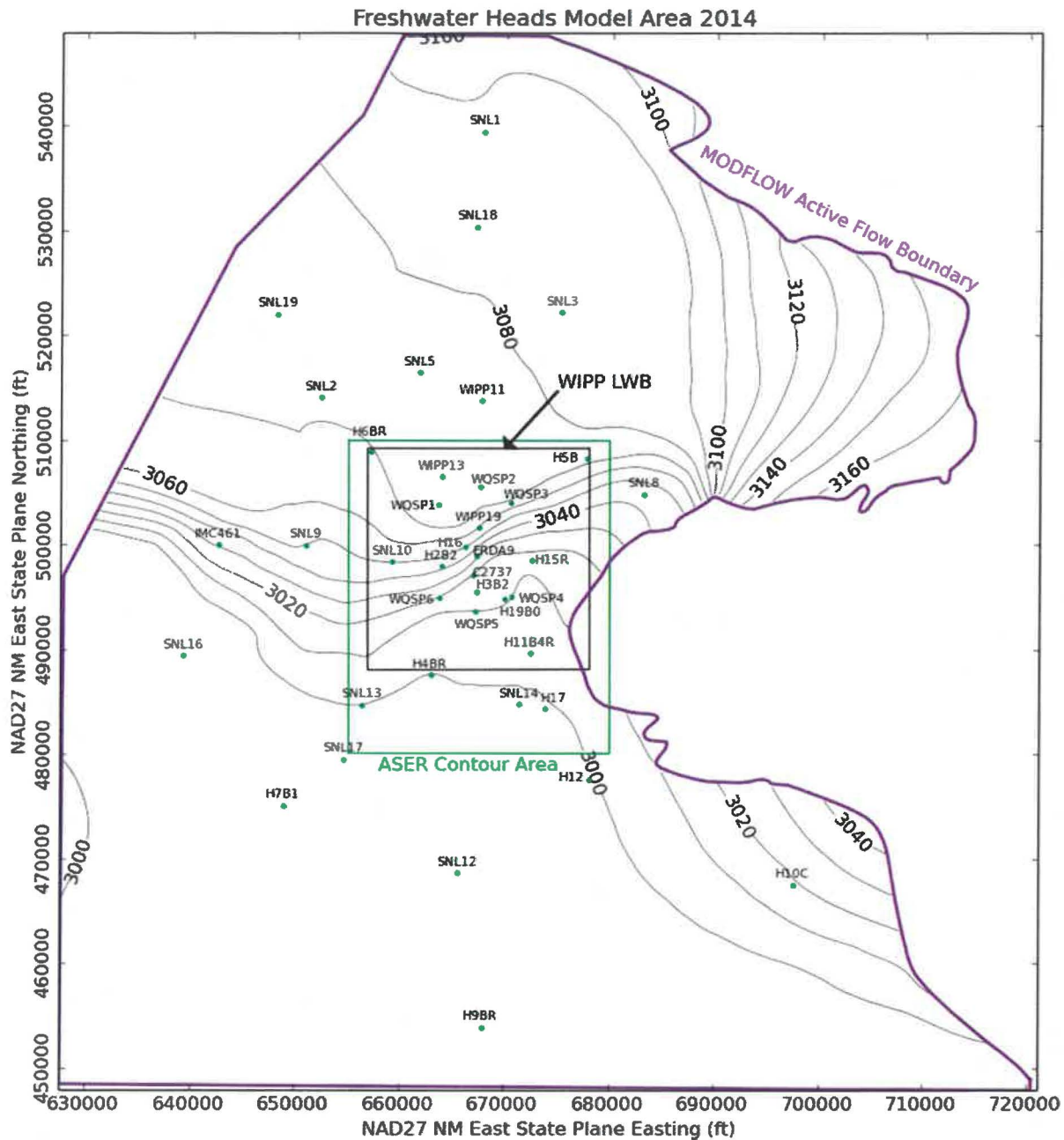


Figure 3. MODFLOW-modeled January 2014 heads for entire model domain (10-foot contour interval). Green rectangle indicates region contoured in Figure 2, black square is WIPP LWB.

3.2 2014 Particle Track

The blue arrow in Figure 2 shows the DTRKMF-calculated path a water particle would take through the Culebra from the coordinates corresponding to the WIPP waste handling shaft to the LWB (a path length of 4076 m). Assuming the transmissive portion of the Culebra is only 4-m thick, and assuming a constant porosity of 16%, the travel time to the WIPP LWB is 5829 years (output from DTRKMF is adjusted from an original 7.75-m Culebra thickness). This is an average velocity of 0.70 m/yr. This estimated flow velocity is higher than predicted in previous years (e.g., 0.65 m/yr in Kuhlman, 2014), because of the

steeper gradient from the release point to the southern edge of the WIPP LWB. This steeper gradient is partially due to matching heads in the south affected by pumping at C-2942.

3.3 2014 Measured vs. Modeled Fit

The scatter plot in Figure 4 shows measured and modeled freshwater heads at the observation locations used in the PEST calibration. The observations are divided into three groups, based on proximity to the WIPP site. Wells within the LWB are represented by red crosses, wells outside but within 3 km of the LWB are represented with green 'x's, and other wells within the MODFLOW model domain but distant from the WIPP site are indicated with blue stars. Additional observations representing the average heads north of the LWB and south of the LWB were used to help prevent over-smoothing of the estimated results across the LWB. This allowed PEST to improve the fit of the model to observed heads inside the area contoured in Figure 2, at the expense of fitting wells closer to the boundary conditions (i.e., wells not shown in Figure 2).

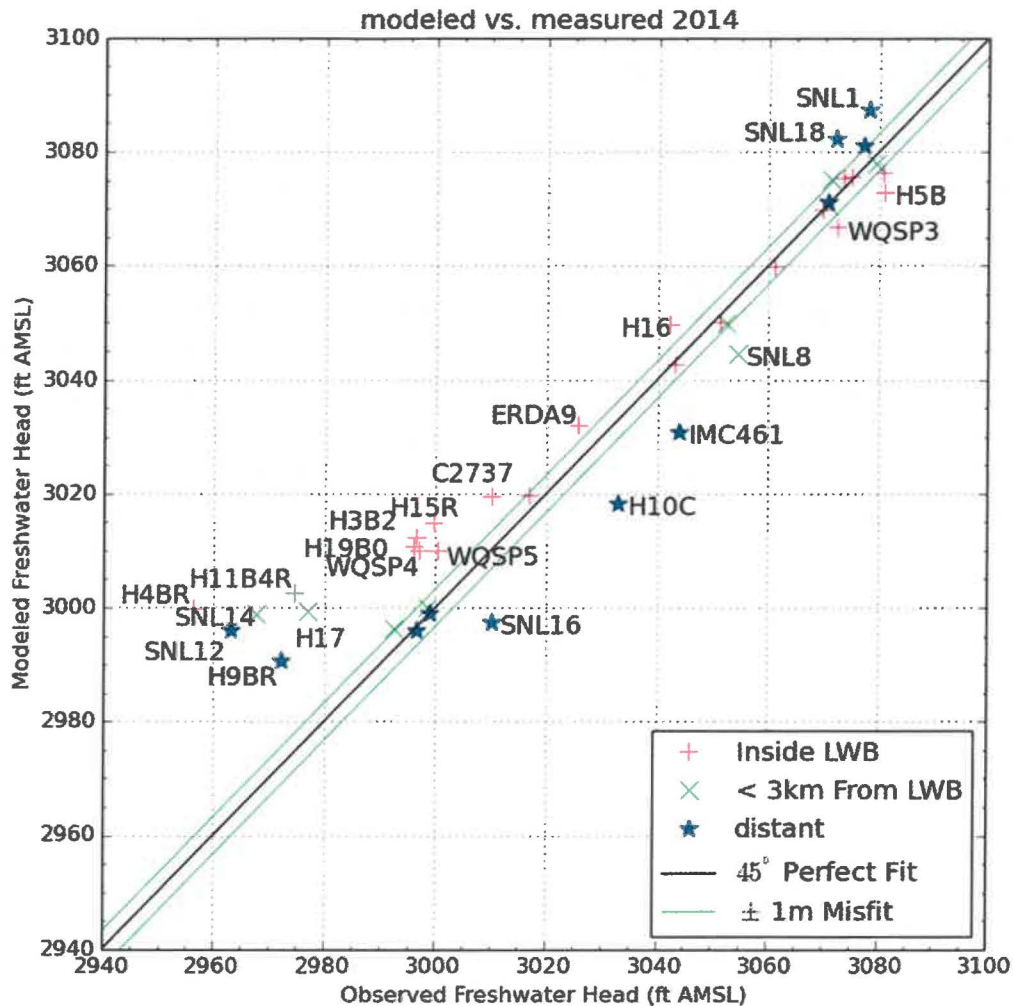


Figure 4. Measured vs. modeled scatter plot for averaged MODFLOW model generated heads and January 2014 observed freshwater heads

The central black diagonal line in Figure 4 represents a perfect model fit (1:1 or 45-degree slope); the two green lines on either side of this represent a 1-m misfit above or below the perfect fit. Wells more than 1.5 m from the 1:1 line are labeled. AEC-7 has a large misfit (13 m), for two reasons. First, this well has historically had an anomalously low freshwater head elevation, lower than wells around it in all directions. Secondly, it did not have a May 2007 observation (due to ongoing well reconfiguration activities) and therefore was not included as a calibration target in the PA MODFLOW model calibration. The ensemble-average *T*, *A*, and *R* fields used here were not calibrated to accommodate this observation. This well is situated in a low-transmissivity region, and near the constant-head boundary associated with the halite margin, therefore PEST will not be able to improve this fit solely through adjustment of the boundary conditions indicated with red in Figure 1.

The calibrated parameters (for equation 1) were $A = 929.4$, $B = 8.76$, $C = -1.036$, $D = 0.7920$, $E = -0.8611$, $F = -2.340$, and $\alpha = -0.4502$. The parameters α (exponent on y), C (coefficient on all x variability), and E (coefficient on x^3 variability) had the largest relative change (~186-190%) compared to the starting

values. Parameter F (coefficient on x^2) was within -134% of its original value, and D (coefficient on exponentiated y term) was 21% away. All other parameters were <10% different from their original values.

The squared correlation coefficient (R^2) for the measured vs. modeled data is listed in Table 3. Figure 5 and Figure 6 show the distribution of errors resulting from the PEST-adjusted model fit to observed data. The wells within and near the WIPP LWB have an R^2 of greater than 98%, and the calibration decreased the R^2 value very slightly when including all wells. The calibration improved the fit for the wells in and near the WIPP LWB at the expense of fit to wells distant from the LWB. The distribution of residuals in Figure 5 does not have a strong bias.

Table 3. 2014 Measured vs. Modeled correlation coefficients

	dataset	measured vs. modeled R^2
Uncalibrated	wells inside WIPP LWB	0.965
	wells <3km from WIPP LWB	0.958
	all wells	0.936
Calibrated	wells inside WIPP LWB	0.965
	wells <3km from WIPP LWB	0.953
	all wells	0.918

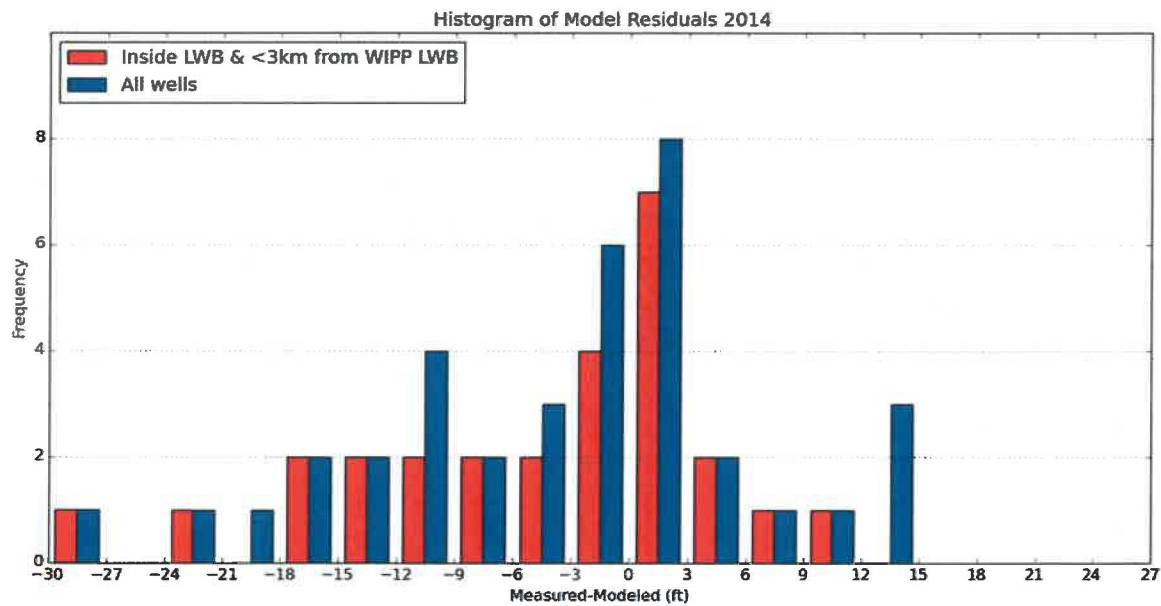


Figure 5. Histogram of Measured-Modeled errors for 2014 (H-04bR residual is off the left side of the figure at -43.6 ft)

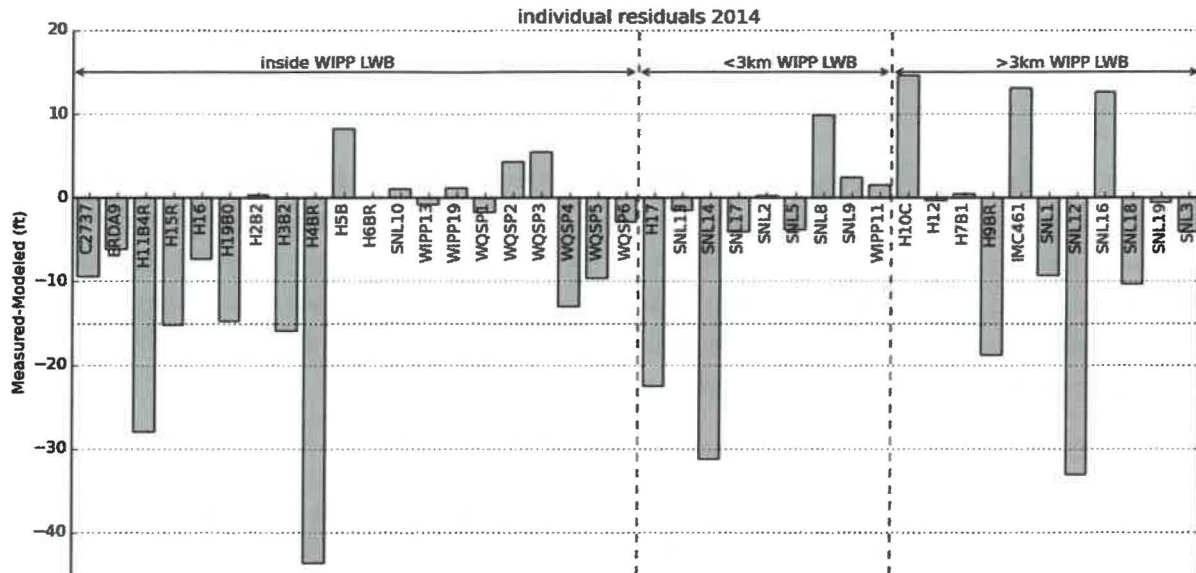


Figure 6. Measured-Modeled errors at each well location for 2014

The model fit to the January 2014 observations is poor. The averaged MODFLOW model captures the bulk steady-state Culebra flow behavior, but it cannot recreate the large drawdown centered on well C-2492 (also called the “new Mills” well; Kuhlman, 2015), which is located immediately south of WIPP Culebra well H-04bR. This well has been pumping since September 2013. By January 2014 impacts from this pumping were extending across the southern portion of the WIPP LWB.

The process for adjusting the boundary conditions of the averaged steady-state MODFLOW outlined in SP 9-9 (Kuhlman, 2009) cannot match the effects of this pumping. This report presents a contour map which represents the Culebra freshwater head piezometric surface as it would possibly look, if it were unperturbed by pumping. Figure 7 shows the difference between the unperturbed (i.e., modeled) freshwater head and the observed freshwater head drawdown which is mainly due to pumping at C-2492. The decision was made to use this approach, rather than modify the procedure, because it is believed the pumping at C-2492 does not represent the natural steady-state flow in the Culebra.

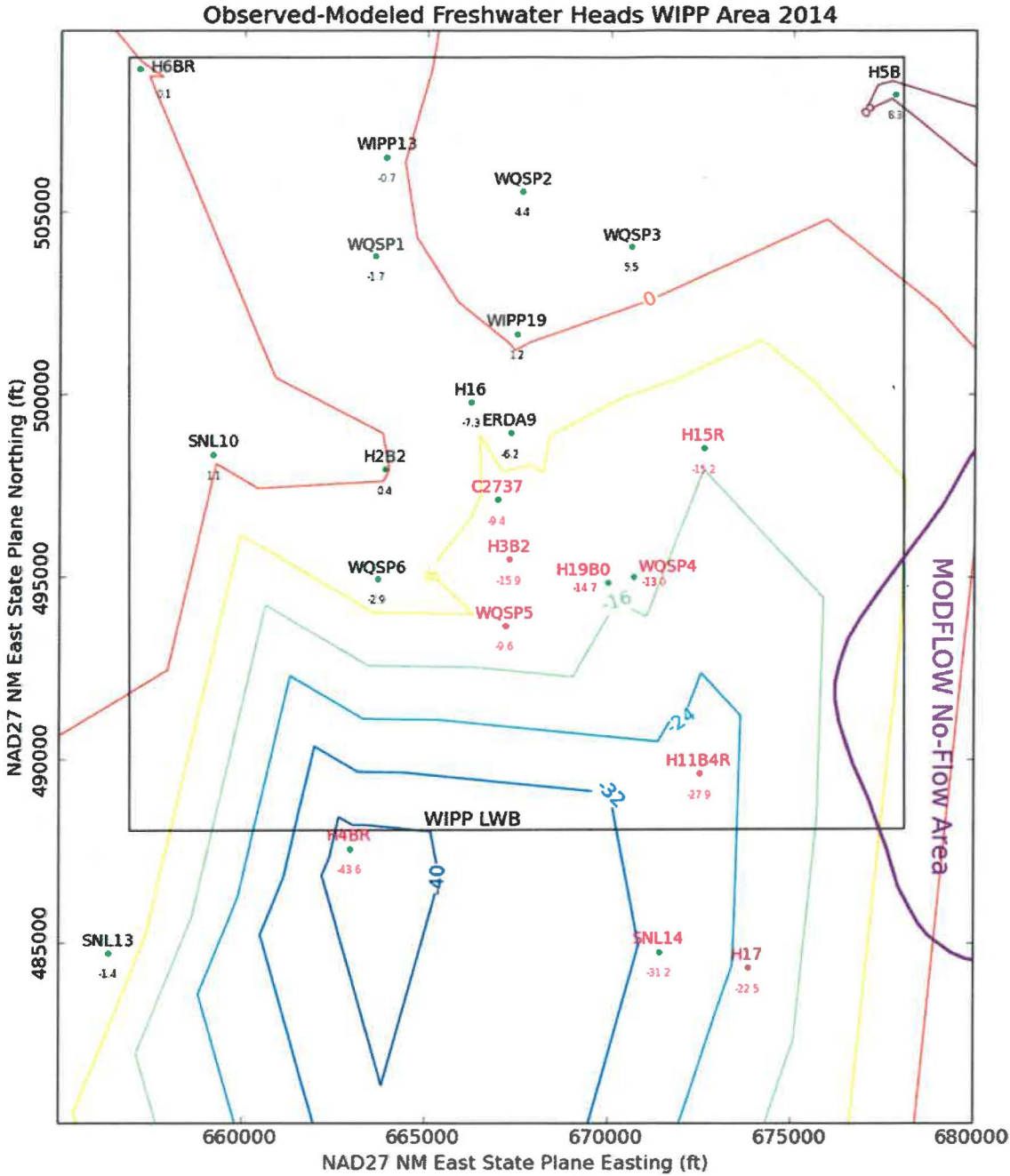


Figure 7. Triangulated contours for observed minus model-predicted freshwater head. Red labels indicate wells designated as significantly impacted by C-2492 pumping, which were assigned a smaller weight (0.05) in the calibration process (also see wells marked with ‡ in Table 2).

4 References

- Beauheim. 2008. Analysis Plan for Evaluation and Recalibration of Culebra Transmissivity Fields, AP-113, Revision 1. Carlsbad, NM: Sandia National Laboratories.
- DOE (US Department of Energy). 2014. *Title 40 CFR Part 191 Subparts B and C Compliance Recertification Application 2014 for the Waste Isolation Pilot Plant Appendix TFIELD-2014 Transmissivity Fields*. US Department of Energy: Carlsbad, NM, DOE/WIPP-14-3503.
- Doherty, J. 2002. *PEST: Model Independent Parameter Estimation*. Watermark Numerical Computing, Brisbane, Australia.
- 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*. U.S. Geological Survey Open-File Report 00-92.
- Hart, D.B., S.A. McKenna, and R.L. Beauheim. 2009. *Analysis Report for Task 7 of AP-114: Calibration of Culebra Transmissivity Fields*. Sandia National Laboratories: Carlsbad, NM, ERMS 552391.
- Kuhlman, K.L. 2014. Analysis Report for Preparation of 2013 Culebra Potentiometric Surface Contour Map, Sandia National Laboratories: Carlsbad, NM, ERMS 562110.
- Kuhlman, K.L. 2015. WIPP Milestone Report: 2014 Culebra Groundwater Level Fluctuations, Memo to WIPP Records Center. Carlsbad, NM: Sandia National Laboratories, ERMS XXXXXX. 564004
KED 9/30/15
- Kuhlman, K.L. 2009. Procedure SP 9-9, revision 0, Preparation of Culebra potentiometric surface contour maps. Carlsbad, NM: Sandia National Laboratories, ERMS 552306.
- Moody, D.C. 2009. *Stipulated Final Order for Notice of Violation for Detection Monitoring Program*, Sandia National Laboratories: Carlsbad, NM. WIPP Records Center, ERMS 551713.
- Seal, B. 2015. January 2014 Culebra ASER map data, Washington TRU Solutions, Carlsbad, NM. WIPP Records Center, ERMS 563914.

5 Run Control Narrative

This section is a narrative describing the calculation process mentioned in the text, which produced the figures given there.

Figure 8 gives an overview of the driver script `checkout_average_run_modflow.sh` (§A-4.1); this script first exports the 3 parameter fields (transmissivity (T), anisotropy (A), and recharge (R), and storativity (S)) from CVS version control for each of the 100 realizations of MODFLOW, listed in the file `keepers` (see lines 17-26 of script). Some of the realizations are inside the `Update` or `Update2` subdirectories in CVS, which complicates the directory structure. An equivalent list `keepers_short` is made from `keepers`, and the directories are moved to match the flat directory structure (lines 31-53). At this point, the directory structure has been modified but the MODFLOW input files checked out from CVS are unchanged.

Python script `average_realizations.py` (§A-4.2) is called, which first reads in the `keepers_short` list, then reads in each of the 400 input files and computes the geometric average at each cell across the 100 realizations. The 400 input files are each saved as flattened matrices, in row-major order. The average result is saved into 4 parameter files, each with the extension `.avg` instead of `.mod`. A single value from each file, corresponding to either the cell in the southeast corner of the domain (input file row 87188 = model row 307, model column 284 for K and A) or on the west edge of the domain (input file row 45157 = model row 161, model column 1 for R and S) is saved in the text file `parameter_representative_values.txt` to allow checking the calculation in Excel, comparing the results to the value given at the same row of the `.avg` file. The value in the right column of Table 4 can be found by taking the geometric average of the values in the text file, which are the values from the indicated line of each of the 100 realizations.

The input files used by this analysis, the output files from this analysis (including the plotting scripts) are checked into the WIPP version control system (CVS) under the repository `$CVSLIB/Analyses/SP9_9`.

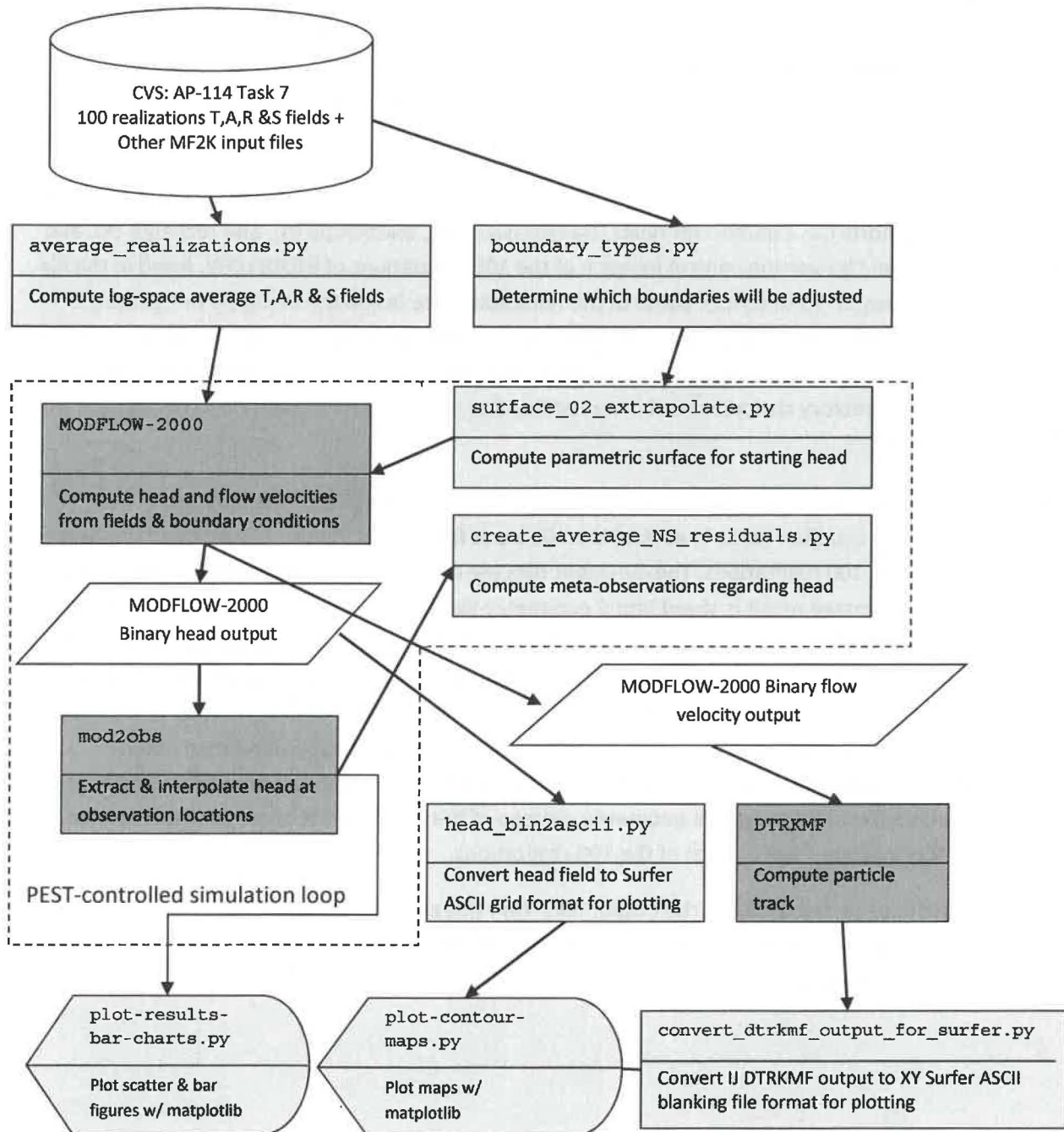


Figure 8. Process flowchart; dark gray indicates qualified programs, light gray are scripts written for this analysis

Table 4. Averaged values for representative model cells

Field	Input file row	Model row	Model column	Geometric average
K	87188	307	284	9.2583577E-09
A	87188	307	284	9.6317478E-01
R	45157	161	1	1.4970689E-19
S	45157	161	1	4.0388352E-03

Figure 9 shows plots of the average \log_{10} parameters, which compare with similar figures in Hart et al. (2009); inactive regions ($< 10^{-15}$) were reset to 1 to improve the plotted color scale. The rest of the calculations are done with these averaged fields.

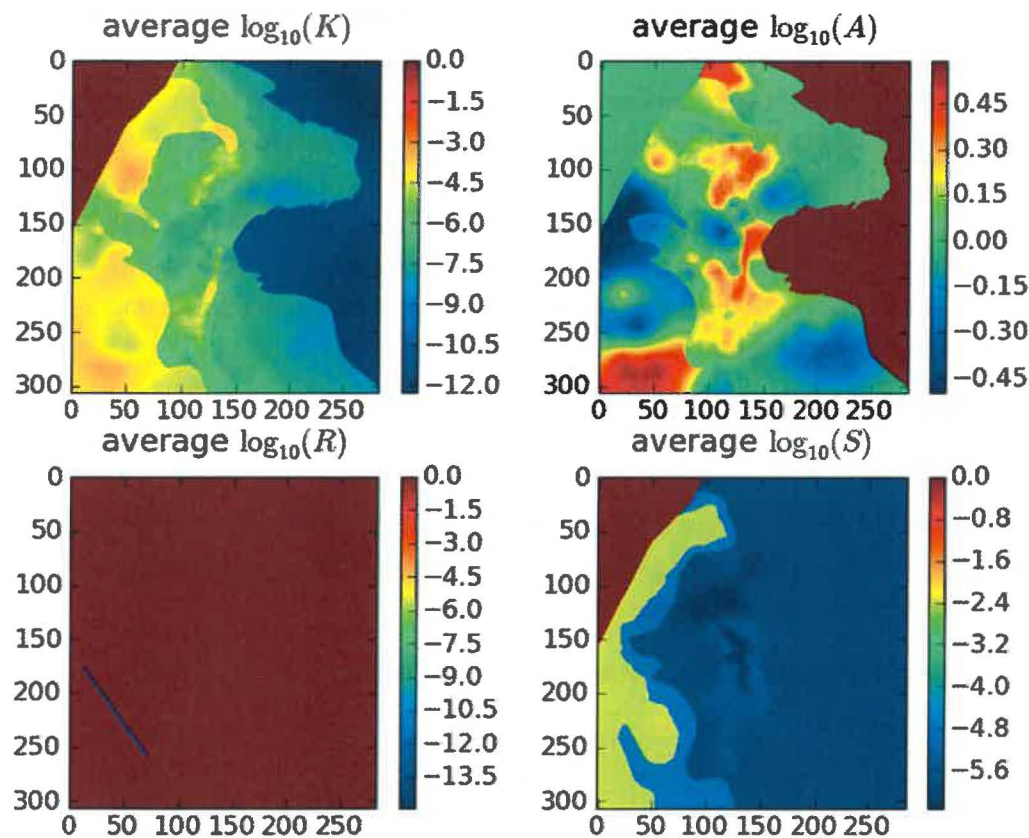


Figure 9. Plots of base-10 logarithms of average parameter fields; rows and columns are labeled on edges of figures.

Next, a subdirectory is created, and the averaged MODFLOW model is run without any modifications by PEST. Subsequently, another directory will be created where PEST will be run to improve the fit of the model to observed heads at well locations.

The next portion of the driving script `checkout_average_run_modflow.sh` links copies of the input files needed to run MODFLOW-2000 and DTRKMF into the `original_average` run directory. Then MODFLOW-2000 is run with the name file `mf2k_head.nam`, producing binary head (`modeled_head.bin`) and binary cell-by-cell flow budget (`modeled_flow.bud`) files, as well as a text listing file (`modeled_head.lst`). DTRKMF is then run with the input files `dtrkfmf.in` and `wippctrl.inp`, which utilizes the cell-by-cell budget file written by MODFLOW to generate a particle track output file, `dtrk.out`. The input file `wippctrl.inp` specifies the starting location of the particle in DTRKMF face-centered cell coordinates, the porosity of the aquifer (here 16%), and the

coordinates of the corners of the WIPP LWB, since the calculation stops when the particle reaches the LWB.

The Python script `head_bin2ascii.py` (§A-4.7) converts the MODFLOW binary head file, which includes the steady-state head at every element in the flow model domain (307 rows × 284 columns) into a Surfer ASCII grid file format. This file is simply contoured in Python using `matplotlib`, no interpolation or gridding is needed. The Python script `convert_dtrkmf_output_for_surfer.py` (§A-4.9) reads the DTRKMF output file `dtrk.out` and does two things. First it converts the row, column format of this output file to an *x, y* format suitable for plotting, and second it converts the effective thickness of the Culebra from 7.75 m to 4 m. The following table shows the first 10 lines of the `dtrk.out` and the corresponding output of the Python script `dtrk_output_original_average.blm`. The first three columns of `dtrk.out` (top half of Table 5) after the header are cumulative time (red), column (blue), and row (green). The three columns in the blanking file (second half of Table 5) after the header are UTM NAD27 X (blue), UTM NAD27 Y (green), and adjusted cumulative time (red, which is faster than the original cumulative travel time by the factor $7.75/4=1.9375$). The conversion from row, column to *x, y* is

$$X = 601700 + 100 * column$$

$$Y = 3597100 - 100 * row$$

since the I,J origin is the northwest corner of the model domain (601700, 3597100), while the X,Y origin is the southwest corner of the domain. The blanking file is plotted directly in Python using `matplotlib`, since it now has the same coordinates as the ASCII head file.

Table 5. Comparison of first 10 lines of DTRKMF output and converted Surfer blanking file for `original_average`

1	159								
0.00000000E+00	118.79	150.21	1.18790000E+04	1.50210000E+04	0.00000000E+00	1.85168267E-01	1.59999996E-01	1.00000000E+00	
5.53946616E+01	118.86	150.29	1.18859872E+04	1.50285080E+04	1.02562574E+01	1.85130032E-01	1.59999996E-01	1.00000000E+00	
1.10789323E+02	118.93	150.36	1.18929942E+04	1.50359947E+04	2.05104788E+01	1.85094756E-01	1.59999996E-01	1.00000000E+00	
1.66017959E+02	119.00	150.43	1.19000000E+04	1.50434379E+04	3.07321029E+01	1.85062532E-01	1.59999996E-01	1.00000000E+00	
3.27990509E+02	119.21	150.62	1.19206651E+04	1.50624751E+04	5.88294962E+01	1.73534671E-01	1.59999996E-01	1.00000000E+00	
4.89963060E+02	119.42	150.81	1.19415109E+04	1.50813473E+04	8.69490492E+01	1.73684593E-01	1.59999996E-01	1.00000000E+00	
6.51450155E+02	119.62	151.00	1.19624759E+04	1.51000000E+04	1.15010608E+02	1.73860152E-01	1.59999996E-01	1.00000000E+00	
7.40581455E+02	119.75	151.10	1.19749757E+04	1.51102419E+04	1.31170520E+02	1.81333000E-01	1.59999996E-01	1.00000000E+00	
8.29712755E+02	119.87	151.20	1.19874963E+04	1.51204665E+04	1.47335525E+02	1.81390626E-01	1.59999996E-01	1.00000000E+00	
159, 1									
613579.0,	3582079.0,	0.00000000E+00							
613586.0,	3582071.0,	2.85907931E+01							
613593.0,	3582064.0,	5.71815861E+01							
613600.0,	3582057.0,	8.56866885E+01							
613621.0,	3582038.0,	1.69285424E+02							
613642.0,	3582019.0,	2.52884160E+02							
613662.0,	3582000.0,	3.36232338E+02							
613675.0,	3581990.0,	3.82235590E+02							
613687.0,	3581980.0,	4.28238841E+02							

The PEST utility script `mod2obs` is run to extract and interpolate the model-predicted heads at observation locations. The input files for `mod2obs.exe` were taken from AP-114 Task 7 in CVS. The observed head file has the wells and freshwater heads, but is otherwise the same as that used in the model calibration in AP-114 (Beauhiem, 2008; Hart et al., 2009). The Python script `merge_observed_modeled_heads.py` (§A-4.9) simply puts the results from `mod2obs` and the original observed heads in a single file together for easier plotting and later analysis.

cells as constant head which have an IBOUND entry < 0, so both -2 and -1 are the same to MODFLOW, but allow distinguishing between them in the Python script which extrapolates the heads to the boundaries.

The required PEST input files are created by the Python script `create_pest_02_input.py` (§A-4.4). This script writes **1**) the PEST instruction file (`modeled_head.ins`), which shows PEST how to extract the model-predicted heads from the `mod2obs.exe` output; **2**) the PEST template file (`surface_par_params.ptf`), which shows PEST how to write the input file for the surface extrapolation script; **3**) the PEST parameter file (`surface_par_params.par`), which lists the starting parameter values to use when checking the PEST input; **4**) the PEST control file (`bc_adjust_2014ASER.pst`), which has PEST-related parameters, definitions of extrapolation surface parameters, and the observations and weights that PEST is adjusting the model inputs to fit. The observed heads are read as an input file in the PEST borehole sample file format (`meas_head_2014ASER.smp`), and the weights are read in from the input file (`obs_loc_2014ASER.dat`).

PEST runs the “forward model” many times, adjusting inputs and reading the resulting outputs using the instruction and template files created above. The forward model actually consists of a Bash shell script (`run_02_model`) that simply calls a pre-processing Python script `surface_02_extrapolate.py` (§A-4.5), the MODFLOW-2000 executable, the Python script `create_average_NS_residuals.py`, and the PEST utility `mod2obs.exe` as a post-processing step. The script redirects the output of each step to `/dev/null` to minimize screen output while running PEST, since PEST will run the forward model many dozens of times.

The Python script `create_average_NS_residuals.py` takes the output from the PEST utility `mod2obs` and creates a meta-observation that consists of the average residual between measured and model-prediction, only averaged across the northern or southern WIPP wells (the wells in the center of the WIPP site are not included in either group). This was done to minimize cancelation of the errors north (where the model tended to underestimate heads) and south (where the model tended to overestimate heads) of the WIPP. The results of this script are read directly by PEST and incorporated as four additional observations (mean and median errors, both north and south of WIPP).

The pre-processing Python script `surface_02_extrapolate.py` reads the new IBOUND array created in a previous step (with -2 now indicating which constant-head boundaries should be modified), the initial head file used in AP-114 Task 7 (`init_head_orig.mod`), two files listing the relative X and Y coordinates of the model cells (`rel_{x,y}_coord.dat`), and an input file listing the coefficients of the parametric equation used to define the initial head surface. This script then cycles over the elements in the domain, writing the original starting head value if the IBOUND value is -1 or 0, and writing the value corresponding to the parametric equation if the IBOUND value is -2 or 1. Using the parameters corresponding to those used in AP-114 Task 7, the output starting head file should be identical to that used in AP-114 Task 7.

After PEST has converged to the optimum solution for the given observed heads and weights, it runs the forward model one more time with the optimum parameters. The post-processing Python scripts for creating the Surfer ASCII grid file and Surfer blanking file from the MODFLOW and DTRKMF output are run and the results are plotted using additional Python scripts that utilize the plotting and map coordinate projection functionality of the matplotlib library.

These two plotting scripts (`plot-contour-maps.py` and `plot-results-bar-charts.py`) are included in the appendix for completeness, but only draw the figures included in this report, and passed on to WRES for the ASER.

Information Only

6 Files and Script Source Listings

6.1 Input Files

bytes	file type	description	file name
1.5K	Python script	average 100 realizations	average_realizations.py
2.1K	Python script	distinguish different BC types	boundary_types.py
6.2K	Bash script	main routine: checkout files, run MODFLOW run PEST, call Python scripts	checkout_average-run.modflow.sh
624	Python script	convert DTRKMF IJ output to Surfer X,Y blanking format	convert_dtrkmf_output_for_surfer.py
2.8K	Python script	create meta observations of avg heat	create_average_NS_residuals.py
3.1K	Python script	create PEST input files from observed data	create_pest_02_input.py
48	input listing	responses to DTRKMF prompts	dtrkmf.in
4.0K	Python script	convert MODFLOW binary output to Surfer ASCII grid format	head_bin2ascii.py
1.1K	input	listing of 100 realizations from CVS	keepers
1.4K	input	observed January 2014 heads in mod2obs bore sample file format	meas_head_2014ASER.smp
968	Python script	paste observed head and model-generated heads into one file	merge_observed_modeled_heads.py
76	file listing	files needed to run mod2obs	mod2obs_files.dat
139	input listing	responses to mod2obs prompts	mod2obs_head.in
372	file listing	files needed to run MODFLOW	modflow_files.dat
393	input	listing of wells and geographic groupings	obs_loc_2014ASER.dat
215	file listing	files needed to run PEST	pest_02_files.dat
2.3M	input	relative coordinate $1 \leq x \leq 1$	rel_x_coord.dat
2.3M	input	relative coordinate $1 \leq y \leq 1$	rel_y_coord.dat
490	Bash script	PEST model: execute MODFLOW and do pre- and post-processing	run_02_model
26	input	mod2obs input file	settings.fig
47	input	mod2obs input file	spec_domain.spc
1.8K	input	mod2obs input file	spec_wells.crd
2.4K	Python script	compute starting head from parameter and coordinate inputs	surface_02_extrapolate.py
506	input	DTRKMF input file	wippctrl.inp

Table 1: Input Files

6.2 Output Files

bytes	file type	description	file name
19K	DTRKMF output	particle track results	dtrk.out
16K	DTRKMF output	particle track debug	dtrk.dbg
1.9K	script output	heads at well locations	modeled_vs_observed_head_pest_02.txt
1.1M	script output	formatted MODFLOW heads	modeled_head_pest_02.grd
5.3K	script output	formatted DTRKMF particle	dtrk_output_pest_02.blm
12K	PEST output	matrix condition numbers	bc_adjust_2014ASER.cnd
2.7K	PEST output	binary intermediate file	bc_adjust_2014ASER.drf
7.3K	PEST output	binary intermediate file	bc_adjust_2014ASER.jac
7.4K	PEST output	binary intermediate file	bc_adjust_2014ASER.jco
9.8K	PEST output	binary intermediate file	bc_adjust_2014ASER.jst
3.8K	PEST output	parameter statistical matrices	bc_adjust_2014ASER.mtt
477	PEST output	parameter file	bc_adjust_2014ASER.par
56K	PEST output	optimization record	bc_adjust_2014ASER.rec
4.5K	PEST output	model outputs for last iteration	bc_adjust_2014ASER.rei
8.2K	PEST output	summary of residuals	bc_adjust_2014ASER.res
28	PEST output	binary restart file	bc_adjust_2014ASER.rst
22K	PEST output	relative parameter sensitivities	bc_adjust_2014ASER.sen
3.9K	PEST output	absolute parameter sensitivities	bc_adjust_2014ASER.seo
214K	png image	matplotlib plot (Fig. 2)	aser-area-contour-map2014.png
226K	png image	matplotlib plot (Fig. 3)	large-area-contour-map2014.png
161K	png image	matplotlib plot (Fig. 7)	aser-area-modobs-contour-map2014.png
34K	png image	matplotlib plot (Fig. 5)	model-error-histogram2014.png
25K	png image	matplotlib plot (Fig. 6)	model-error-residuals2014.png
117K	png image	matplotlib plot (Fig. 4)	scatter_pest_02_2014.png

Table 2: Listing of Output Files

6.3 Individual scripts

6.3.1 Bash shell script `checkout_average_run_modflow.sh`

```
1  #!/bin/bash
2
3  set -o nounset # explode if using an un-initialized variable
4  set -o errexit # exit on non-zero error status of sub-command
5
6  # this script makes the following directory substructure
7  #
8  #  current_dir \----- Outputs  (calibrated parameter fields - INPUTS)
9  #                \----- Inputs   (other modflow files - INPUTS)
10 #                 \--- original_average (foward sim using average fields)
11 #                  \-- bin           (MODFLOW and DTRKMF binaries)
12 #                   \- pest_0?      (pest-adjusted results)
13
14 set -o xtrace
15
16 echo " ~~~~~ "
17 echo " checking out T fields"
18 echo " ~~~~~ "
19
20 # these will checkout the calibrated parameter-field data into subdirectories
21 # checkout things that are different for each of the 100 realiztaions
22 for d in `cat keepers`
23 do
24     cvs -d /nfs/data/CVSLIB/Tfields checkout Outputs/${d}/modeled_{K,A,R,S}_field.mod
25 done
26
27 # checkout MODFLOW input files that are constant for across all realizations
28 cvs -d /nfs/data/CVSLIB/Tfields checkout Inputs/data/elev_{top,bot}.mod
29 cvs -d /nfs/data/CVSLIB/Tfields checkout Inputs/data/init_{bnds.inf,head.mod}
30 cvs -d /nfs/data/CVSLIB/Tfields checkout Inputs/modflow/mf2k_culebra.{lmg,lpf}
31 cvs -d /nfs/data/CVSLIB/Tfields checkout Inputs/modflow/mf2k_head.{ba6,nam,oc,dis,rch}
32
33 # modify the path of "updated" T-fields, so they are all at the
34 # same level in the directory structure (simplifying scripts elsewhere)
35
36 if [ -a keepers_short ]
37 then
38     rm keepers_short
39 fi
40 touch keepers_short
41
42 for d in `cat keepers`
43 do
44     bn=`basename ${d}`
45     # test whether it is a compount path
46     if [ ${d} != ${bn} ]
47     then
48         dn=`dirname ${d}`
```

```

49     mv ./Outputs/${d} ./Outputs/
50
51     # put an empty file in the directory to indicate
52     # what the directory was previously named
53     touch ./Outputs/${bn}/${dn}
54 fi
55
56 # create a keepers list without directories
57 echo ${bn} >> keepers_short
58 done
59
60 # -----
61
62 echo " ~~~~~ "
63 echo " perform averaging across all realizations "
64 echo " ~~~~~ "
65
66 python average_realizations.py
67
68 # -----
69
70 echo " ~~~~~ "
71 echo " setup copies of files constant between all realizations "
72 echo " ~~~~~ "
73
74 # directory for putting original base-case results in
75 od=original_average
76
77 if [ -d ${od} ]
78 then
79     echo ${od}" directory exists: removing and re-creating"
80     rm -rf ${od}
81 fi
82
83 mkdir ${od}
84 cd ${od}
85 echo `pwd`
86
87 # link to unchanged input files
88 for file in `cat ../modflow_files.dat`
89 do
90     ln -sf ${file} .
91 done
92
93 # link to averaged files computed in previous step
94 for f in {A,R,K,S}
95 do
96     ln -sf ../modeled_${f}_field.avg ./modeled_${f}_field.mod
97 done
98
99 ln -sf elev_top.mod fort.33

```

```

100 ln -sf elev_bot.mod fort.34
101
102 echo " ~~~~~"
103 echo " run original MODFLOW and DTRKMF and export results for plotting"
104 echo " ~~~~~"
105
106 # run MODFLOW, producing average head and CCF
107 /utilities/modflow2000 mf2k_head.nam
108
109 # run DTRKMF, producing particle track (from ccf)
110 /utilities/dtrkmf `cat dtrkmf.in`
111
112 # convert binary MODFLOW head output to Surfer ascii grid file format
113 ln -sf ../head_bin2ascii.py .
114 python head_bin2ascii.py
115 mv modeled_head_asciished.grd modeled_head_${od}.grd
116
117 # convert DTRKMF output from cells to X,Y and
118 # save in Surfer blanking file format
119 ln -sf ../convert_dtrkmf_output_for_surfer.py .
120 python convert_dtrkmf_output_for_surfer.py
121 mv dtrk_output.blm dtrk_output_${od}.blm
122
123 # extract head results at well locations and merge with observed
124 # head file for easy scatter plotting in Excel (tab delimited)
125 for file in `cat ../mod2obs_files.dat`
126 do
127     ln -sf ${file} .
128 done
129
130 ln -sf ../meas_head_2014ASER.smp .
131 ln -sf ../obs_loc_2014ASER.dat .
132 /utilities/mod2obs <mod2obs_head.in
133 ln -sf ../merge_observed_modeled_heads.py
134 python merge_observed_modeled_heads.py
135 mv both_heads.smp modeled_vs_observed_head_${od}.txt
136
137
138 # go back down into root directory
139 cd ..
140 echo `pwd`
141
142 echo " ~~~~~"
143 echo " setup and run PEST to optimize parametric surface to set BC "
144 echo " ~~~~~"
145
146 for p in pest_02
147 do
148
149     if [ -d ${p} ]
150     then

```

```

151     echo ${p}" directory exists: removing and re-creating"
152     rm -rf ${p}
153 fi
154
155 mkdir ${p}
156 cd ${p}
157 echo `pwd`
158
159 # link to unchanged input files
160 for file in `cat ../modflow_files.dat`
161 do
162     ln -sf ${file} .
163 done
164
165 # link to averaged files computed in previous step
166 for f in {A,R,K,S}
167 do
168     ln -sf ../modeled_${f}_field.avg ./modeled_${f}_field.mod
169 done
170
171 # link to mod2obs files (needed for pest)
172 for file in `cat ../mod2obs_files.dat`
173 do
174     ln -sf ${file} .
175 done
176
177 # link to pest files
178 for file in `cat ../${p}_files.dat`
179 do
180     ln -s ${file} .
181 done
182
183 # rename 'original' versions of files to be modified by pest
184 rm init_head.mod
185 ln -sf ../Inputs/data/init_head.mod ./init_head_orig.mod
186 rm init_bnds.inf
187 ln -sf ../Inputs/data/init_bnds.inf ./init_bnds_orig.inf
188
189 # create new ibound array for easier modification during PEST
190 # optimization iterations
191 python boundary_types.py
192
193 # create the necessary input files from observations
194 python create_${p}_input.py
195
196 # run pest
197 /utilities/pest bc_adjust_2014ASER
198
199 # last output files should be best run
200 # extract all the stuff from that output
201 #####

```

```
202
203 ln -sf elev_top.mod fort.33
204 ln -sf elev_bot.mod fort.34
205
206 /utilities/dtrkmf `cat dtrkmf.in`
207
208 ln -sf ../head_bin2ascii.py .
209 python head_bin2ascii.py
210 mv modeled_head_asciimed.grd modeled_head_${p}.grd
211
212 ln -sf ../convert_dtrkmf_output_for_surfer.py .
213 python convert_dtrkmf_output_for_surfer.py
214 mv dtrk_output.blm dtrk_output_${p}.blm
215
216 for file in `cat ../mod2obs_files.dat`
217 do
218     ln -sf ${file} .
219 done
220
221 /utilities/mod2obs <mod2obs_head.in
222 ln -sf ../merge_observed_modeled_heads.py
223 python merge_observed_modeled_heads.py
224 mv both_heads.smp modeled_vs_observed_head_${p}.txt
225
226 cd ..
227 done
```


6.3.2 Python script `average_realizations.py`

```
1 from math import log10, pow
2
3 nrow = 307
4 ncol = 284
5 nel = nrow*ncol
6 nfr = 100 # number of fields (realizations)
7 nft = 4   # number of field types
8
9 def floatload(filename):
10     """Reads file (a list of strings, one per row) into a list of strings."""
11     f = open(filename, 'r')
12     m = [float(line.rstrip()) for line in f]
13     f.close()
14     return m
15
16 types = ['K', 'A', 'R', 'S']
17
18 # get list of 100 best calibrated fields
19 flist = open('keepers_short', 'r')
20 runs = flist.read().strip().split('\n')
21 flist.close()
22
23 # initialize to help speed lists up a bit
24 # nfr (100) realizations of each
25 fields = []
26 for i in xrange(nft):
27     fields.append([None]*nfr)
28     for i in xrange(nfr):
29         # each realization being nel (87188) elements
30         fields[-1][i] = [None]*nel
31
32 # read in all realizations
33 print 'reading ...'
34 for i,run in enumerate(runs):
35     print i,run
36     for j,t in enumerate(types):
37         fields[j][i][0:nel] = floatload('Outputs/'+ run + '/modeled_'+ t + '_field.mod')
38
39 # open up files for writing
40 fh = []
41 for t in types:
42     fh.append(open('modeled_'+ t + '_field.avg', 'w'))
43
44 # transpose fields to allow slicing across realizations, rather than across cells
45 for j in range(len(types)):
46     fields[j] = zip(*(fields[j]))
47
48 print 'writing ...'
49 # do averaging across 100 realizations
```

```
50 for i in xrange(nel):
51     if i%10000 == 0:
52         print i
53     for h,d in zip(fh,fields):
54         h.write('%18.11e\n' % pow(10.0, sum(map(log10,d[i]))/nfr) )
55
56 for h in fh:
57     h.close()
```

6.3.3 Python script `boundary-types.py`

```
1 nx = 284          # number columns in model grid
2 ny = 307          # number rows
3 nel = nx*ny
4
5 def intload(filename):
6     """Reads file (a 2D integer array) as a list of lists.
7     Outer list is rows, inner lists are columns."""
8     f = open(filename, 'r')
9     m = [[int(v) for v in line.rstrip().split()] for line in f]
10    f.close()
11    return m
12
13 def intsave(filename,m):
14    """Writes file as a list of lists as a 2D integer array, format '%3i'.
15    Outer list is rows, inner lists are columns."""
16    f = open(filename, 'w')
17    for row in m:
18        f.write(' '.join(['%2i' % col for col in row]) + '\n')
19    f.close()
20
21 def floatload(filename):
22    """Reads file (a list of real numbers, one number each row) into a list of floats."""
23    f = open(filename, 'r')
24    m = [float(line.rstrip()) for line in f]
25    f.close()
26    return m
27
28 def reshapev2m(v):
29    """Reshape a vector that was previously reshaped in C-major order from a matrix,
30    back into a matrix (here a list of lists)."""
31    m = [None]*ny
32    for i,(lo,hi) in enumerate(zip(xrange(0, nel-nx+1, nx), xrange(nx, nel+1, nx))):
33        m[i] = v[lo:hi]
34    return m
35
36 #####
37
38 # read in original MODFLOW IBOUND array (only 0,1, and -1)
39 ibound = intload('init_bnds_orig.inf')
40
41 # read in initial heads
42 h = reshapev2m(floatload('init_head_orig.mod'))
43
44 # discriminate between two types of constant head boundaries
45 # -1) CH, where value > 1000 (area east of halite margin)
46 # -2) CH, where value < 1000 (single row/column of cells along edge of domain)
47
48 for i,row in enumerate(ibound):
49     for j,val in enumerate(row):
```

```
50     # is this constant head and is starting head less than 1000m ?
51     if ibound[i][j] == -1 and h[i][j] < 1000.0:
52         ibound[i][j] = -2
53
54     # save new IBOUND array that allows easy discrimination between types in python script dur.
55     # PEST optimization runs, and is still handled the same by MODFLOW
56     # since all ibound values < 0 are treated as constant head.
57     intsave('init_bnds.inf',ibound)
```

6.3.4 Python script create_pest.02_input.py

```
1 prefix = '2014ASER'
2
3 #####
4 ## pest instruction file reads output from mod2obs
5 fin = open('meas_head_%s.smp' % prefix, 'r')
6
7 # each well is a [name,head] pair
8 wells = [[line.split()[0],line.split()[3]] for line in fin]
9 fin.close()
10
11
12 fout = open('modeled_head.ins', 'w')
13 fout.write('pif @\n')
14 for i,well in enumerate(wells):
15     fout.write("l1 [%s]39:46\n" % well[0])
16 fout.close()
17
18 # exponential surface used to set initial head everywhere
19 # except east of the halite margins, where the land surface is used.
20 # initial guesses come from AP-114 Task report
21 params = [928.0, 8.0, 1.2, 1.0, 1.0, -1.0, 0.5]
22 pnames = ['a', 'b', 'c', 'd', 'e', 'f', 'exp']
23
24 fout = open('avg_NS_res.ins', 'w')
25 fout.write("""pif @
26 l1 [medianN]1:16
27 l1 [medianS]1:16
28 l1 [meanN]1:16
29 l1 [meanS]1:16
30 """)
31 fout.close()
32
33
34 #####
35 ## pest template file
36 ftmp = open('surface_par_params.ptf', 'w')
37 ftmp.write('ptf @\n')
38 for n in pnames:
39     ftmp.write('@ %s @\n' % n)
40 ftmp.close()
41
42
43 #####
44 ## pest parameter file
45
46 fpar = open('surface_par_params.par', 'w')
47 fpar.write('double point\n')
48 for n,p in zip(pnames,params):
49     fpar.write('%s %.2f 1.0 0.0\n' % (n,p))
```

```

50 fpar.close()
51
52
53 #####
54 ## pest control file
55
56 f = open('bc_adjust_%s.pst' % prefix, 'w')
57
58 f.write("""pcf
59 * control data
60 restart estimation
61 %i %i 1 0 2
62 1 2 double point 1 0 0
63 5.0 2.0 0.4 0.001 10
64 3.0 3.0 1.0E-3
65 0.1
66 30 0.001 4 4 0.0001 4
67 1 1 1
68 * parameter groups
69 bc relative 0.005 0.0001 switch 2.0 parabolic
70 """ % (len(params), len(wells)+4))
71
72 f.write('* parameter data\n')
73 for n,p in zip(pnames,params):
74     if p > 0:
75         f.write('%s none relative %.3f %.3f %.3f bc 1.0 0.0 1\n' %
76                 (n, p, -2.0*p, 3.0*p))
77     else:
78         f.write('%s none relative %.3f %.3f %.3f bc 1.0 0.0 1\n' %
79                 (n, p, 3.0*p, -2.0*p))
80
81 f.write("""* observation groups
82 ss_head
83 avg_head
84 * observation data
85 """)
86
87 ## read in observation weighting group definitions
88 fin = open('obs_loc_%s.dat' % prefix, 'r')
89 location = [line.rstrip().split()[1] for line in fin]
90 fin.close()
91
92 weights = []
93
94 for l in location:
95     # inside LWB
96     if l == '0':
97         weights.append(2.5)
98     # near LWB
99     if l == '1':
100        weights.append(1.0)

```

```

101     # distant to LWB
102     if l == '2':
103         weights.append(0.4)
104     if l == '99':
105         weights.append(0.01) # AEC-7
106
107     j=1
108     for name,head,w in zip(zip(*wells)[0],zip(*wells)[1],weights):
109         f.write('%s %s %.3f ss_head\n' % (name,head,w))
110
111     # one fewer N observation (WIPP-25 removed), there were 13
112     # there are 12 N observations in the average and 11 S, therefore
113     # split the weight between the mean and median
114     f.write("""medianN 0.0 18.0 avg_head
115 medianS 0.0 16.5 avg_head
116 meanN 0.0 18.0 avg_head
117 meanS 0.0 16.5 avg_head
118 """)
119
120     f.write("""* model command line
121 ./run_02_model
122 * model input/output
123 surface_par_params.ptf surface_par_params.in
124 modeled_head.ins modeled_head.smp
125 avg_NS_res.ins avg_NS_res.smp
126 """)
127     f.close()

```

6.3.5 Python script `surface_02_extrapolate.py`

```
1 from itertools import chain
2 from math import sqrt
3
4 def matload(filename):
5     """Reads file (a 2D string array) as a list of lists.
6     Outer list is rows, inner lists are columns."""
7     f = open(filename, 'r')
8     m = [line.rstrip().split() for line in f]
9     f.close()
10    return m
11
12 def floatload(filename):
13     """Reads file (a list of real numbers, one number each row) into a list of floats."""
14     f = open(filename, 'r')
15     m = [float(line.rstrip()) for line in f]
16     f.close()
17     return m
18
19 def reshapem2v(m):
20     """Reshapes a rectangular matrix into a vector in same fashion as numpy.reshape().
21     which is C-major order"""
22     return list(chain(*m))
23
24 def sign(x):
25     """ sign function"""
26     if x<0:
27         return -1
28     elif x>0:
29         return +1
30     else:
31         return 0
32
33 #####
34
35 # read in modified IBOUND array, with the cells to modify set to -2
36 ibound = reshapem2v(matload('init_bnds.inf'))
37
38 h = floatload('init_head_orig.mod')
39
40 # these are relative coordinates, -1 <= x,y < +1
41 x = floatload('rel_x_coord.dat')
42 y = floatload('rel_y_coord.dat')
43
44 # unpack surface parameters (one per line)
45 # z = A + B*(y + D*sign(y)*sqrt(abs(y)))+C*(E*x**3 - F*x**2 - x)
46
47 finput = open('surface_par_params.in', 'r')
48 try:
49     a,b,c,d,e,f,exp = [float(line.rstrip()) for line in finput]
```



```

50 except ValueError:
51     # python doesn't like 'D' in 1.2D-4 notation used by PEST sometimes.
52     finput.seek(0)
53     lines = [line.rstrip() for line in finput]
54     for i in range(len(lines)):
55         lines[i] = lines[i].replace('D', 'E')
56     a,b,c,d,e,f,exp = [float(line) for line in lines]
57
58 finput.close()
59
60 # file to output initial/boundary head for MODFLOW model
61 fout = open('init_head.mod', 'w')
62 for i in xrange(len(ibound)):
63     if ibound[i] == '-2' or ibound[i] == '1':
64         # apply exponential surface to active cells (ibound=1) -> starting guess
65         # and non-geologic boundary conditions (ibound=-2) -> constant head value
66         if y[i] == 0:
67             fout.write('%.7e \n' % (a + c*(e*x[i]**3 + f*x[i]**2 - x[i])))
68         else:
69             fout.write('%.7e \n' % (a + b*(y[i] + d*sign(y[i])*abs(y[i])**exp) +
70                                     c*(e*x[i]**3 + f*x[i]**2 - x[i])))
71     else:
72         # use land surface at constant head east of halite boundary
73         # ibound=0 doesn't matter (inactive)
74         fout.write('%.7e\n' % h[i])
75
76 fout.close()

```

6.3.6 Bash shell script `run_02_model`

```
1 #!/bin/bash
2
3 #set -o xtrace
4
5 #echo 'step 1: surface extrapolate'
6 python surface_02_extrapolate.py
7
8 # run modflow
9 #echo 'step 2: run modflow'
10 ##../bin/mf2k/mf2k_1.6.release mf2k_head.nam >/dev/null
11 /utilities/modflow2000 mf2k_head.nam >/dev/null
12
13 # run mod2obs
14 #echo 'step 3: extract observations'
15 ##../bin/Builds/Linux/mod2obs.exe < mod2obs_head.in >/dev/null
16 /utilities/mod2obs <mod2obs_head.in >/dev/null
17
18 # create meta-observations of N vs. S
19 python create_average_NS_residuals.py
```

6.3.7 Python script head.bin2ascii.py

```
1 import struct
2 from sys import argv,exit
3
4 class FortranFile(file):
5     """ modified from May 2007 Enthought-dev mailing list post by Neil Martinsen-Burrell"""
6
7     def __init__(self, fname, mode='r', buf=0):
8         file.__init__(self, fname, mode, buf)
9         self.ENDIAN = '<' # little endian
10        self.di = 4 # default integer (could be 8 on 64-bit platforms)
11
12    def readReals(self, prec='f'):
13        """Read in an array of reals (default single precision) with error checking"""
14        # read header (length of record)
15        l = struct.unpack(self.ENDIAN+'i', self.read(self.di))[0]
16        data_str = self.read(l)
17        len_real = struct.calcsize(prec)
18        if l % len_real != 0:
19            raise IOError('Error reading array of reals from data file')
20        num = l/len_real
21        reals = struct.unpack(self.ENDIAN+str(num)+prec, data_str)
22        # check footer
23        if struct.unpack(self.ENDIAN+'i', self.read(self.di))[0] != 1:
24            raise IOError('Error reading array of reals from data file')
25        return list(reals)
26
27    def readInts(self):
28        """Read in an array of integers with error checking"""
29        l = struct.unpack('i', self.read(self.di))[0]
30        data_str = self.read(l)
31        len_int = struct.calcsize('i')
32        if l % len_int != 0:
33            raise IOError('Error reading array of integers from data file')
34        num = l/len_int
35        ints = struct.unpack(str(num)+'i', data_str)
36        if struct.unpack(self.ENDIAN+'i', self.read(self.di))[0] != 1:
37            raise IOError('Error reading array of integers from data file')
38        return list(ints)
39
40    def readRecord(self):
41        """Read a single fortran record (potentially mixed reals and ints)"""
42        dat = self.read(self.di)
43        if len(dat) == 0:
44            raise IOError('Empty record header')
45        l = struct.unpack(self.ENDIAN+'i', dat)[0]
46        data_str = self.read(l)
47        if len(data_str) != l:
48            raise IOError('Didn't read enough data')
49        check = self.read(self.di)
```

```

50     if len(check) != 4:
51         raise IOError('Didn't read enough data')
52     if struct.unpack(self.ENDIAN+'i',check)[0] != 1:
53         raise IOError('Error reading record from data file')
54     return data_str
55
56 def reshapev2m(v,nx,ny):
57     """Reshape a vector that was previously reshaped in C-major order from a matrix,
58     back into a C-major order matrix (here a list of lists)."""
59     m = [None]*ny
60     n = nx*ny
61     for i,(lo,hi) in enumerate(zip(xrange(0, n-nx+1, nx), xrange(nx, n+1, nx))):
62         m[i] = v[lo:hi]
63     return m
64
65 def floatmatsave(filehandle,m):
66     """Writes array to open filehandle, format '568%e12.5'.
67     Outer list is rows, inner lists are columns."""
68
69     for row in m:
70         f.write(''.join([' %12.5e' % col for col in row]) + '\n')
71
72 if __name__ == "__main__":
73     # open file and set endian-ness
74     try:
75         infn,outfn = argv[1:3]
76     except:
77         print '2 command-line arguments not given, using default in/out filenames'
78         infn = 'modeled_head.bin'
79         outfn = 'modeled_head_asciihed.grd'
80
81     ff = FortranFile(infn)
82
83     # currently this assumes a single-layer MODFLOW model (or at least only one layer of o
84
85     # format of MODFLOW header in binary layer array
86     fmt = '<2i2f16s3i'
87     # little endian, 2 integers, 2 floats,
88     # 16-character string (4 element array of 4-byte strings), 3 integers
89
90     while True:
91         try:
92             # read in header
93             h = ff.readRecord()
94
95         except IOError:
96             # exit while loop
97             break
98
99     else:
100        # unpack header

```

```

101         kstp,kper,pertim,totim,text,ncol,nrow,ilay = struct.unpack(fmt,h)
102
103         # print status/confirmation to terminal
104         print kstp,kper,pertim,totim,text,ncol,nrow,ilay
105
106         h = ff.readReals()
107
108     ff.close()
109
110     xmin, xmax = (601700.0,630000.0)
111     ymin, ymax = (3566500.0,3597100.0)
112     hmin = min(h)
113     hmax = max(h)
114
115     # write output in Surfer ASCII grid format
116     f = open(outfn,'w')
117     f.write("""DSAA
118     %i %i
119     %.1f %.1f
120     %.1f %.1f
121     %.8e %.8e
122     """ %(ncol,nrow,xmin,xmax,ymin,ymax,hmin,hmax) )
123     hmat = reshapev2m(h,ncol,nrow)
124
125     # MODFLOW starts data in upper-left corner
126     # Surfer expects data starting in lower-left corner
127     # flip array in row direction
128
129     floatmatsave(f,hmat[::-1])
130     f.close()

```

6.3.8 Python script merge_observed_modeled_heads.py

```
1 fobs = open('meas_head_2014ASER.smp', 'r') # measured head
2 fmod = open('modeled_head.smp', 'r')     # modeled head
3 fwgt = open('obs_loc_2014ASER.dat', 'r')  # weights
4 fdb = open('spec_wells.crd', 'r')        # x/y coordinates
5
6 fout = open('both_heads.smp', 'w')        # resulting file
7
8 # read in list of x/y coordinates, key by well name
9 wells = {}
10 for line in fdb:
11     well,x,y = line.split()[0:3] # ignore last column
12     wells[well.upper()] = [x,y]
13 fdb.close()
14
15 fout.write('\t'.join(['#NAME', 'UTM-NAD27-X', 'UTM-NAD27-Y',
16                       'OBSERVED', 'MODELED', 'OBS-MOD', 'WEIGHT']))+'\n')
17
18 for sobs,smod,w in zip(fobs,fmod,fwgt):
19     obs = float(sobs.split()[3])
20     mod = float(smod.split()[3])
21     name = sobs.split()[0].upper()
22     fout.write('\t'.join([name,wells[name][0],wells[name][1],
23                           str(obs),str(mod),str(obs-mod),
24                           w.rstrip().split()[1]])+'\n')
25
26 fobs.close()
27 fmod.close()
28 fwgt.close()
29 fout.close()
```

6.3.9 Python script `convert_dtrkmf_output_for_surfer.py`

```
1
2 # grid origin for dtrkmf cell -> x,y conversion
3 x0 = 601700.0
4 y0 = 3597100.0
5
6 dx = 100.0
7 dy = 100.0
8
9 fout = open('dtrk_output.blm', 'w')
10
11 # read in all results for saving particle tracks
12 fin = open('dtrk.out', 'r')
13 results = [l.split() for l in fin.readlines()[1:]]
14 fin.close()
15
16 npts = len(results)
17
18 # write Surfer blanking file header
19 fout.write('%i,1\n' % npts)
20
21 # write x,y location and time
22 for pt in results:
23     x = float(pt[1])*dx + x0
24     y = y0 - float(pt[2])*dy
25     t = float(pt[0])/7.75*4.0 # convert to 4m Cuelbra thickness
26     fout.write('%.1f,%.1f,%.8e\n' % (x,y,t))
27
28 fout.close()
```

6.3.10 Python script plot-contour-maps.py

```
1 import numpy as np
2
3 manualFix = True
4
5 if not manualFix:
6     import matplotlib
7     matplotlib.use('Agg')
8
9 import matplotlib.pyplot as plt
10 from mpl_toolkits.basemap import pyproj
11
12
13
14 # http://spatialreference.org/ref/epsg/26713/
15 # http://spatialreference.org/ref/epsg/32012/
16 putm = pyproj.Proj(init='epsg:26713') # UTM Zone 13N NAD27 (meters)
17 pstp = pyproj.Proj(init='epsg:32012') # NM state plane east NAD27 (meters)
18
19 def transform(xin,yin):
20     """does the default conversion from utm -> state plane
21     then also convert to feet from meters"""
22     xout,yout = pyproj.transform(putm,pstp,xin,yin)
23     xout /= M2FT
24     yout /= M2FT
25     return xout,yout
26
27 year = '2014'
28 fprefix = 'pest_02/'
29 mprefix = '../../well-location-maps/wipp-polyline-data/'
30 cfname = fprefix + 'modeled_head_pest_02.grd'
31 pfname = fprefix + 'dtrk_output_pest_02.blm'
32 wfname = fprefix + 'modeled_vs_observed_head_pest_02.txt'
33
34 M2FT = 0.3048
35
36 # read in well-related things
37 # %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
38 # load in observed, modeled, obs-mod, (all in meters)
39 res = np.loadtxt(wfname, skiprows=1, usecols=(3,4,5))
40 res /= M2FT # convert heads to feet
41 wellutm,wellutmy,obsmod = np.loadtxt(wfname, skiprows=1, usecols=(1,2,5), unpack=True)
42 wellx,wellly = transform(wellutm,wellutmy)
43 obsmod /= M2FT
44 names = np.loadtxt(wfname, skiprows=1, usecols=(0,), dtype='|S6')
45
46 weights = {}
47 fh = open('obs_loc_%sASER.dat' % year, 'r')
48 lines = fh.readlines()
49 fh.close()
```



```

50 for line in lines:
51     f = line.split()
52     # inout is integer flag: 0=inside LWB, 1=near LWB, 2=far from LWB
53     # ns is character flag: N=north, C=central, S=south, M=imacted by Mill's ranch pumping
54     weights[f[0].upper()] = {'inout':int(f[1]), 'ns':f[2]}
55
56 #print 'DEBUG well coordinates'
57 #for n,ux,uy,x,y in zip(names,wellutm,wellutmy,wellx,welly):
58 #     print n,ux,uy,'::',x,y
59
60 # read in head-related things
61 # %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
62 h = np.loadtxt(cfname,skiprows=5) # ASCII matrix of modeled head in meters AMSL
63 h[h<0.0] = np.NaN # no-flow zone in northeast
64 h[h>1000.0] = np.NaN # constant-head zone in east
65 h /= M2FT # convert elevations to feet
66
67 # surfer grid is implicit in header
68 # create grid from min/max UTM NAD27 coordinates (meters)
69 utmy,utm, = np.mgrid[3566500.0:3597100.0:307j, 601700.0:630000.0:284j]
70
71 # head contour coords
72 hx,hy = transform(utm,utmy)
73 del utm,utmy
74
75 # read in particle-related things
76 # %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
77 px,py = transform(*np.loadtxt(pfname,skiprows=1,delimiter=',',usecols=(0,1),unpack=True))
78 part = np.loadtxt(pfname,skiprows=1,delimiter=',',usecols=(2,))
79
80 # read in MODFLOW model, WIPP LWB & ASER contour domain (UTM X & Y)
81 # %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
82 modx,mody = transform(*np.loadtxt(mprefix+'total_boundary.dat',unpack=True))
83 wipputm,wipputm, = np.loadtxt(mprefix+'wipp_boundary.dat',
84                             usecols=(0,1),unpack=True)
85 wippx,wippy = transform(wipputm,wipputm)
86 aserx,asery = transform(*np.loadtxt(mprefix+'ASER_boundary.csv',
87                                 delimiter=',',usecols=(1,2),unpack=True))
88
89 #print 'DEBUG WIPP coordinates'
90 for ux,uy,x,y in zip(wipputm,wipputm,wippx,wippy):
91     print ux,uy,'::',x,y
92
93 a = []
94
95 # plot contour map of entire model area
96 # *****
97 fig = plt.figure(1,figsize=(12,16))
98 ax = fig.add_subplot(111)
99 lev = 3000 + np.arange(17)*10
100 CS = ax.contour(hx,hy,h,levels=lev,colors='k',linewidths=0.5)

```

```

101 ax.clabel(CS,lev[:,2],fmt='%i')
102 ax.plot(wippx,wippy,'k-')
103 ax.plot(aserx,asery,'g-')
104 ax.plot(modx,mody,'-',color='purple',linewidth=2)
105 ax.plot(wellx,welly,linestyle='none',marker='.',
106         markeredgecolor='green',markerfacecolor='green')
107 ax.set_xticks(630000 + np.arange(10.0)*10000)
108 ax.set_yticks(450000 + np.arange(10.0)*10000)
109 labels = ax.get_yticklabels()
110 for label in labels:
111     label.set_rotation(90)
112 for x,y,n in zip(wellx,welly,names):
113     # plot just above
114     a.append(plt.annotate(n,xy=(x,y),xytext=(0,5),
115                          textcoords='offset points',
116                          horizontalalignment='center',
117                          fontsize=8))
118 plt.axis('image')
119 ax.set_title('Freshwater Heads Model Area '+year)
120 ax.set_xlabel('NAD27 NM East State Plane Easting (ft)')
121 ax.set_ylabel('NAD27 NM East State Plane Northing (ft)')
122
123 # compute travel time and path length to WIPP LWB
124 # *****
125
126 # compute incremental distance between times
127 pd = M2FT*np.sqrt((px[1:]-px[:-1])**2 + (py[1:]-py[:-1])**2)
128
129 ax.text(688000,537000,'MODFLOW Active Flow Boundary',size=12,rotation=-26,color='purple')
130 ax.annotate('WIPP LWB',xy=(670000,509200),xytext=(675000,515000),
131            fontsize=12,arrowprops=dict(facecolor='black',width=1))
132 ax.annotate('ASER Contour Area',xy=(658000,478500),fontsize=12,color='green')
133
134 print 'particle length:',pd.sum(),'(meters); travel time:',part[-1],'(years); ',
135 print ' avg speed:',pd.sum()/part[-1],'(m/yr)'
136
137 if manualFix:
138     # manually fix labels
139     for lab in a:
140         lab.draggable()
141     plt.show()
142 else:
143     plt.savefig('large-area-contour-map'+year+'.png')
144 plt.close(1)
145
146 del lev,CS
147 mask = np.logical_and(np.logical_and(hx>aserx.min(),hx<aserx.max()),
148                       np.logical_and(hy>asery.min(),hy<asery.max()))
149 h[~mask] = np.NaN
150
151 a = []

```

```

152
153 # plot contour map of ASER-figure area
154 # *****
155 fig = plt.figure(1, figsize=(12,16))
156 ax = fig.add_subplot(111)
157 lev = 3000 + np.arange(17)*5
158 CS = ax.contour(hx,hy,h,levels=lev,colors='k',linewidths=0.5)
159 ax.plot(wippx,wippy,'k-')
160 ax.plot(modx,mody,'-',color='purple',linewidth=2)
161 ax.plot(wellx,welly,linestyle='none',marker='.',
162         markeredgecolor='green',markerfacecolor='green')
163 ax.plot(px,py,linestyle='solid',color='blue',linewidth=4)
164 plt.arrow(x=px[-3],y=py[-3],dx=-10,dy=-50,
165          linewidth=4,color='blue',head_length=500,head_width=500)
166 plt.axis('image')
167 ax.set_xlim([aserx.min(),aserx.max()])
168 ax.set_ylim([asery.min(),asery.max()])
169 ax.clabel(CS,lev[:,2],fmt='%i',inline_spacing=2)
170 ax.set_xticks(660000 + np.arange(5.0)*5000)
171 ax.set_yticks(485000 + np.arange(5.0)*5000)
172 labels = ax.get_yticklabels()
173 for label in labels:
174     label.set_rotation(90)
175 for j, (x,y,n) in enumerate(zip(wellx,welly,names)):
176     # only plot labels of wells inside the figure area
177     if aserx.min()<x<aserx.max() and asery.min()<y<asery.max():
178         # name above
179         a.append(plt.annotate(n,xy=(x,y),xytext=(0,5),
180                             textcoords='offset points',
181                             horizontalalignment='center',
182                             fontsize=10))
183         # observed FW head below
184         a.append(plt.annotate('%1f'%res[j,0],xy=(x,y),xytext=(0,-15),
185                             textcoords='offset points',
186                             horizontalalignment='center',
187                             fontsize=6))
188 ax.set_title('Freshwater Heads WIPP Area '+ year)
189 ax.set_xlabel('NAD27 NM East State Plane Easting (ft)')
190 ax.set_ylabel('NAD27 NM East State Plane Northing (ft)')
191
192 ax.annotate('WIPP LWB',xy=(665000,488200),fontsize=12)
193 ax.text(678700,495000,'MODFLOW No-Flow Area',size=16,rotation=-90,color='purple')
194
195 if manualFix:
196     # manually fix labels>>>>
197     for lab in a:
198         lab.draggable()
199     plt.show()
200 else:
201     plt.savefig('aser-area-contour-map'+year+'.png')
202 plt.close(1)

```

```

203
204 # plot contour map of measured-modeled residual
205 # *****
206 fig = plt.figure(1,figsize=(12,16))
207 ax = fig.add_subplot(111)
208 CS = ax.tricontour(wellx,welly,obsmod,linestyles='-') # colors='k',
209 ax.plot(wippx,wippy,'k-')
210 ax.plot(modx,mody,'-',color='purple',linewidth=2)
211 ax.plot(wellx,welly,linestyle='none',marker='.',
212         markeredgecolor='green',markerfacecolor='green')
213 plt.axis('image')
214 ax.set_xlim([asex.min(),asex.max()])
215 ax.set_ylim([asery.min(),asery.max()])
216 ax.clabel(CS,fmt='%i',inline_spacing=2)
217 ax.set_xticks(660000 + np.arange(5.0)*5000)
218 ax.set_yticks(485000 + np.arange(5.0)*5000)
219 labels = ax.get_yticklabels()
220 for label in labels:
221     label.set_rotation(90)
222 for j,(x,y,n) in enumerate(zip(wellx,welly,names)):
223     # only plot labels of wells inside the figure area
224     if asex.min()<x<asex.max() and asery.min()<y<asery.max():
225         if 'M' in weights[n]['ns']:
226             color = 'red'
227         else:
228             color = 'black'
229
230     a.append(plt.annotate(n,xy=(x,y),xytext=(0,5),
231                          textcoords='offset points',
232                          horizontalalignment='center',
233                          fontsize=10,color=color))
234
235     a.append(plt.annotate('%0.1f'%obsmod[j],xy=(x,y),xytext=(0,-15),
236                          textcoords='offset points',
237                          horizontalalignment='center',
238                          fontsize=6,color=color))
239 ax.set_title('Observed-Modeled Freshwater Heads WIPP Area '+ year)
240 ax.set_xlabel('NAD27 NM East State Plane Easting (ft)')
241 ax.set_ylabel('NAD27 NM East State Plane Northing (ft)')
242
243 ax.annotate('WIPP LWB',xy=(665000,488200),fontsize=12)
244 ax.text(678700,495000,'MODFLOW No-Flow Area',size=16,rotation=-90,color='purple')
245
246 if manualFix:
247     # manually fix labels>>>>
248     for lab in a:
249         lab.draggable()
250     plt.show()
251 else:
252     plt.savefig('aser-area-modobs-contour-map'+year+'.png')
253 plt.close(1)

```

6.3.11 Python script plot-results-bar-charts.py

```
1 import numpy as np
2
3 manualFix = True
4
5 if not manualFix:
6     import matplotlib
7     matplotlib.use('Agg')
8
9 import matplotlib.pyplot as plt
10
11 fprefix = 'pest_02/'
12 mprefix = '../..../well-location-maps/wipp-polyline-data/'
13 fname = fprefix + 'modeled_vs_observed_head_pest_02.txt'
14
15 ofname = 'original_average/modeled_vs_observed_head_original_average.txt'
16
17 M2FT = 0.3048
18 year = '2014'
19
20 # load in observed, modeled, obs-mod, (all in meters)
21 res = np.loadtxt(fname, skiprows=1, usecols=(3,4,5))
22 ores = np.loadtxt(ofname, skiprows=1, usecols=(3,4,5))
23
24 # load in weights
25 weights = np.loadtxt(fname, skiprows=1, usecols=(6,), dtype='int')
26 # load in names
27 names = np.loadtxt(fname, skiprows=1, usecols=(0,), dtype='|S6')
28
29 # load in N/S/C/X zones
30 zones = np.loadtxt('obs_loc_%sASER.dat' % year, usecols=(2,), dtype='|S2')
31
32 ## checking locations / zones
33 # *****
34 wipp = np.loadtxt(mprefix+'wipp_boundary.dat')
35 x,y = np.loadtxt(fname, skiprows=1, usecols=(1,2), unpack=True)
36
37 fig = plt.figure(2, figsize=(18,12))
38 ax1 = fig.add_subplot(121)
39 ax1.plot(x,y,'k*') # wells
40 ax1.plot(wipp[:,0],wipp[:,1],'r-') # WIPP LWB
41 buff = np.loadtxt(mprefix+'wipp_boundary.dat')
42 buff[1:3,0] -= 3000.0
43 buff[0,0] += 3000.0
44 buff[3:,0] += 3000.0
45 buff[2:4,1] -= 3000.0
46 buff[0:2,1] += 3000.0
47 buff[-1,1] += 3000.0
48 colors = {'N':'red', 'S':'blue', 'C':'green', 'X':'gray', 'SM':'magenta'}
49 ax1.plot(buff[:,0],buff[:,1],'g--') # WIPP LWB+3km
```

```

50 for xv,yv,n,w,z in zip(x,y, names, weights, zones):
51     print xv,yv,n,w,z
52     plt.annotate('%s %i'%(n,w), xy=(xv,yv), fontsize=8, color=colors[z])
53 plt.axis('image')
54 ax1.set_xlim([x.min()-1000,x.max()+1000])
55 ax1.set_ylim([y.min()-1000,y.max()+1000])
56 ax2 = fig.add_subplot(122)
57 ax2.plot(x,y,'k*') # wells
58 ax2.plot(wipp[:,0],wipp[:,1],'r-') # WIPP LWB
59 ax2.plot(buff[:,0],buff[:,1],'g--') # WIPP LWB+3km
60 for xv,yv,n,w,z in zip(x,y, names, weights, zones):
61     plt.annotate('%s %i'%(n,w), xy=(xv,yv), fontsize=8, color=colors[z])
62 plt.axis('image')
63 ax2.set_xlim([wipp[:,0].min()-100,wipp[:,0].max()+100])
64 ax2.set_ylim([wipp[:,1].min()-100,wipp[:,1].max()+100])
65 plt.suptitle('well weights check '+year)
66 plt.savefig('check-well-weights-'+year+'.png')
67
68 # convert lengths to feet
69 res /= M2FT
70 ores /= M2FT
71
72 # create the histogram of residuals for ASER
73 # *****
74
75 # -10,-9,...8,9,10
76 bins = 3*np.arange(-10,10)
77 rectfig = (15,7)
78 squarefig = (8.5,8.5)
79
80 fig = plt.figure(1,figsize=rectfig)
81 ax = fig.add_subplot(111)
82 # all the data, all but distant wells
83 ax.hist([res[weights<2,2],res[:,2]],bins=bins,range=(-30,30.0),
84         rwidth=0.75,align='mid',
85         color=['red','blue'],
86         label=['Inside LWB & <3km from WIPP LWB','All wells'])
87 ax.set_xlabel('Measured-Modeled (ft)')
88 ax.set_ylabel('Frequency')
89 ax.set_xticks(bins)
90 ax.set_ylim([0,10])
91 ax.set_yticks(np.arange(0,10,2))
92 plt.grid()
93 ax.yaxis.grid(True,which='major')
94 ax.xaxis.grid(False)
95 plt.legend(loc='upper left')
96 plt.title('Histogram of Model Residuals '+year)
97 #plt.annotate('AEC-7 @ %.1f'%res[0,2],xy=(-9.75,5.0),xytext=(-8.5,5.0),
98 #            arrowprops={'arrowstyle':'->'},fontsize=16)
99 plt.savefig('model-error-histogram-'+year+'.png')
100 plt.close(1)

```

```

101
102 # create bar chart plot of individual residual for ASER
103 # *****
104
105 m0 = weights==0
106 m1 = weights==1
107 m2 = np.logical_or(weights==2,weights==99)
108
109 # separate wells into groups
110 resin = res[m0,2]
111 resnear = res[m1,2]
112 resfar = res[m2,2]
113
114 nin = resin.size
115 nnear = resnear.size
116 nfar = resfar.size
117
118 # separate names into groups
119 namin = names[m0]
120 namnear = names[m1]
121 namfar = names[m2]
122
123 # get indices that sort vectors
124 ordin = np.argsort(namin)
125 ordnear = np.argsort(namnear)
126 ordfar = np.argsort(namfar)
127
128 # put vectors back together (groups adjacent and sorted inside each group)
129 resagg = np.concatenate((resin[ordin],resnear[ordnear],resfar[ordfar]),axis=0)
130 namagg = np.concatenate((namin[ordin],namnear[ordnear],namfar[ordfar]),axis=0)
131
132 fig = plt.figure(1,figsize=rectfig)
133 ax = fig.add_subplot(111)
134
135 wid = 0.6
136 shift = 0.5 - wid/2.0
137 ab = np.arange(res.shape[0])
138
139 print ab.shape
140 print ab
141
142 ax.bar(left=ab+shift,height=resagg,width=0.75,bottom=0.0,color='lightgray')
143 ax.set_ylim([-45.0,20.0])
144 ax.spines['bottom'].set_position('zero')
145 ax.spines['top'].set_color('none')
146 ax.xaxis.set_ticks_position('bottom')
147 plt.xticks(ab+wid,namagg,rotation=90)
148 # vertical lines dividing groups
149 ax.axvline(x=nin,color='black',linestyle='dashed')
150 ax.axvline(x=nin+nnear,color='black',linestyle='dashed')
151 ax.axhline(y=0,color='black',linestyle='solid')

```

```

152 ax.axhline(y=-15,color='black',linestyle='dotted')
153 plt.grid()
154 ax.yaxis.grid(True,which='major')
155 ax.xaxis.grid(False)
156 ax.set_xlim([0,res.shape[0]])
157
158 plt.annotate('',xy=(0.0,15.0),xycoords='data',
159             xytext=(nin,15.0),textcoords='data',
160             arrowprops={'arrowstyle':'<->'})
161 plt.annotate('inside WIPP LWB',xy=(nin/3.0,15.5),xycoords='data')
162
163 plt.annotate('',xy=(nin,15.0),xycoords='data',
164             xytext=(nin+nnear,15.0),textcoords='data',
165             arrowprops={'arrowstyle':'<->'})
166 plt.annotate('<3km WIPP LWB',xy=(nin+nnear/3.0,15.5),xycoords='data')
167
168 plt.annotate('',xy=(nin+nnear,15.0),xycoords='data',
169             xytext=(nin+nnear+nfar,15.0),textcoords='data',
170             arrowprops={'arrowstyle':'<->'})
171 plt.annotate('>3km WIPP LWB',xy=(nin+nnear+nfar/3.0,15.5),xycoords='data')
172
173 ax.set_ylabel('Measured-Modeled (ft)')
174 ax.set_title('individual residuals '+year)
175 #plt.annotate('AEC-7 @ %.1f'%res[0,2],xy=(nin+nnear+1.0,-14.5),xycoords='data')
176
177 plt.savefig('model-error-residuals-'+year+'.png')
178 plt.close(1)
179
180
181 # create scatter plot of measured vs. modeled
182 # *****
183 m = 1.0/M2FT
184 sr = [2940,3100]
185
186 fh = open('calibration-statistics-%s.csv' % year,'w')
187
188 fh.write('wellgroup,calibrated,uncalibrated\n')
189 fh.write('"all wells",%.4f,' % np.corrcoef(res[:,0],res[:,1])[1,0]**2)
190 fh.write('%.4f\n' % np.corrcoef(ores[:,0],ores[:,1])[1,0]**2)
191
192 fh.write('"wells inside 3km of LWB",%.4f,' % np.corrcoef(res[weights<2,0],res[weights<2,1])
193 fh.write('%.4f\n' % np.corrcoef(ores[weights<2,0],ores[weights<2,1])[1,0]**2)
194
195 fh.write('"wells ~inside LWB",%.4f,' % np.corrcoef(res[weights==0,0],res[weights==0,1])[1,0]**2)
196 fh.write('%.4f\n' % np.corrcoef(ores[weights==0,0],ores[weights==0,1])[1,0]**2)
197
198 fh.close()
199
200 fig = plt.figure(1,figsize=squarefig)
201 ax = fig.add_subplot(111)
202 ax.plot(res[m0,0],res[m0,1],color='red',markersize=10,

```



```

203     marker='+', linestyle='none', label='Inside LWB')
204 ax.plot(res[m1,0],res[m1,1],color='green',markersize=10,
205         marker='x',linestyle='none',label='< 3km From LWB')
206 ax.plot(res[m2,0],res[m2,1],color='blue',markersize=10,
207         marker='*',linestyle='none',label='distant')
208 ax.plot(sr,sr,'k-',label='$45^{\degree}$ Perfect Fit')
209 ax.plot([sr[0],sr[1]],[sr[0]+m,sr[1]+m],'g-',linewidth=0.5,label='$\pm$ 1m Misfit')
210 ax.plot([sr[0],sr[1]],[sr[0]-m,sr[1]-m],'g-',linewidth=0.5,label='__nolegend__')
211 ax.set_xticks(np.linspace(sr[0],sr[1],9))
212 ax.set_yticks(np.linspace(sr[0],sr[1],9))
213 ax.set_xlim(sr)
214 ax.set_ylim(sr)
215 plt.minorticks_on()
216 plt.legend(loc='lower right',scatterpoints=1,numpoints=1)
217 plt.grid()
218 a = []
219 for j,lab in enumerate(names):
220     if res[j,2] < -1.5*m:
221         # plot labels to left of value far above 45-degree line
222         a.append(plt.annotate(lab,xy=(res[j,0],res[j,1]),
223                               xytext=(res[j,0]-(2.9*len(lab)),
224                                         res[j,1]-2.0),fontsize=14))
225     elif res[j,2] > 1.5*m:
226         # plot labels to right of value far below 45-degree line
227         a.append(plt.annotate(lab,xy=(res[j,0],res[j,1]),
228                               xytext=(res[j,0]+2.0,
229                                         res[j,1]-2.0),fontsize=14))
230 ax.set_xlabel('Observed Freshwater Head (ft AMSL)')
231 ax.set_ylabel('Modeled Freshwater Head (ft AMSL)')
232 ax.set_title('modeled vs. measured '+year)
233
234 if manualFix:
235     # manually fix labels>>>>
236     for lab in a:
237         lab.draggable()
238     plt.show()
239 else:
240     plt.savefig('scatter_pest_02_'+year+'.png')
241 plt.close(1)

```