~~566144~~ VCS/11/16
566114

Analysis Report for Preparation of the 2015 Culebra Potentiometric Surface Contour Map

Task Number: 4.4.2.3.1

Report Date: 5/10/2016

Author: _____ 5-10-2016
Matthew A. Thomas, 6932 — Date
Repository Performance Department

Technical Review: Matthew A. Thomas for Kristopher L. Kuhlman: _____ 5-10-2016
Kristopher L. Kuhlman, 6224 — Date
Applied Systems Analysis and Research Department

QA Review: _____ 5-10-16
Shelly R. Nielsen, 6930 — Date
Carlsbad Programs Group

Management Review: _____ 5-10-16
Christi D. Leigh, 6932 — Date
Manager, Repository Performance Department

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

Contents

1 Introduction

This report documents the preparation of the 2015 potentiometric contour map and associated particle track for the Culebra Member of the Rustler Formation (Culebra) 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 New Mexico Environment Department (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 (2015); the same analysis is performed on data from January 2015, rather than January 2014 data. January 2015 data for contouring were obtained from the WIPP Management and Operations contractor (Seal, 2016).

Beginning with the ensemble of 100 calibrated MODFLOW-2000 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-2000 to simulate equivalent freshwater heads within and around the WIPP Land Withdrawal Act (LWA) boundary. For 2015, PEST is used to adjust a subset of the boundary conditions in the averaged MODFLOW-2000 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-2000 model is both contoured and used to compute the 2015 advective particle track forward from the WIPP waste-handling shaft.

The effects of pumping activities at the Mills Ranch have significantly affected water levels in the Culebra monitoring network, especially south of the LWA boundary. The procedure for adjusting the boundary conditions of the averaged MODFLOW-2000 model cannot match the significant drawdown observed in WIPP Culebra wells (Thomas, 2016).

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-2000 transmissivity (T), anisotropy (A), and recharge (R) parameter fields 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-2000 models, the original MODFLOW-2000 model, which consists of 100 realizations of calibrated parameter fields (Hart et al., 2009), will be referred to as the "PA MODFLOW-2000 model." The model derived here from the PA MODFLOW-2000 model, calibrated using PEST, and used to construct the resulting contour map and particle track, is referred to as the "averaged MODFLOW-2000 model." The PA MODFLOW-2000 model *T*, *A* and *R* input fields are appropriately averaged across 100 realizations, producing a single averaged MODFLOW-2000 flow model. This averaged MODFLOW-2000 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-2000 model by adjusting the spatial distribution of the transmissivity (T), anisotropy (A), recharge (R), and storativity (S) parameter fields; MODFLOW-2000 boundary conditions were fixed. The calibration targets for PEST in the PA MODFLOW-2000 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-2000 model. PEST was used to modify a subset of the MODFLOW-2000 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-2000 model were the January 2015 measured annual freshwater heads at Culebra monitoring wells. In the averaged MODFLOW-2000 model, boundary conditions were modified while holding model

parameters (*T*, *A*, and *R*) constant. In contrast to this, the PA MODFLOW-2000 model used fixed boundary conditions and made adjustments to *T*, *A*, and *R* parameter fields.

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	2.7.2	Scripting language (file manipulation)	PA cluster	Commercial off the shelf
Python	2.7.9	Scripting language (plotting)	Windows desktop	Commercial off the shelf
Bash	4.1.17	Scripting language (file manipulation)	PA cluster	Commercial off the shelf

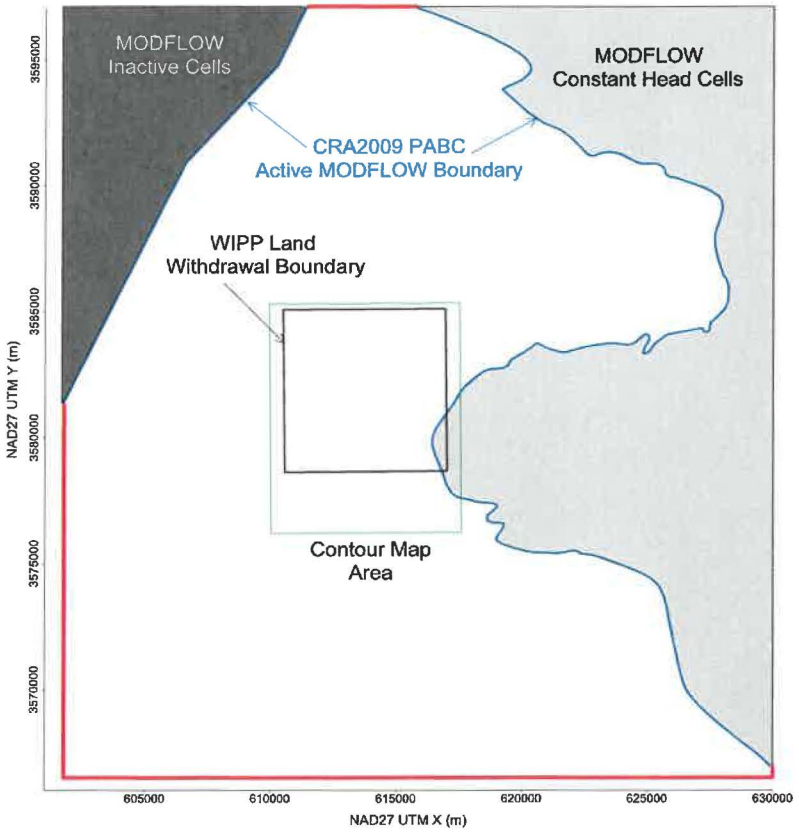

Figure 1. MODFLOW-2000 model domain. Adjusted boundary conditions shown in red. Contour area outlined in green.

The resulting heads from the PEST-calibrated averaged MODFLOW-2000 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-2000 model domain; see the green rectangle surrounding the LWA boundary

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 LWA boundary. The particle track is computed from the MODFLOW-2000 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-2000 model and observed Culebra freshwater heads. MODFLOW-2000, 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 Windows 7 Enterprise, Service Pack 1.

2.2 Creating Average MODFLOW-2000 Simulation

An averaged MODFLOW-2000 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-2000 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 (Wagner and Thomas, 2016), 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-2000 model grid is a single 7.75-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-2000 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-2000 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-2000 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-2000 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-2000 model. All constant-head cells (specified by a value of -1 in the MODFLOW-2000 IBOUND array from the PA MODFLOW-2000 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-2000, 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-2000 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-2000 model setup derived from the PA MODFLOW-2000 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-2000 model will be adjusted in the inverse simulation. Freshwater head values from January 2015 used as targets for the PEST calibration from Seal (2016) are listed in Table 2, and specified along with weights in the PEST configuration files.

Table 2. Freshwater head calibration targets used in PEST, from Seal (2016). Green text indicates wells not used because of long-term recovery. Red text indicates wells clearly influenced by pumping at the Mills Ranch.

Well	Measurement Date	Freshwater Head (m AMSL)	Specific Gravity (g/cm^3)
C-2737	01/13/15	911.42	1.024
ERDA-9	01/07/15	917.42	1.072
H-02b2	01/13/15	924.13	1.012
H-03b2	01/07/15	905.73	1.027
H-04bR	01/07/15	898.53	1.027
H-05b	01/06/15	938.75	1.089
H-06bR	01/03/15	935.73	1.038
H-07b1	01/05/15	913.90	1.009
H-09bR	01/06/15	903.33	1.004
H-10c	01/06/15	926.04	1.096
H-11b4R	01/07/15	901.02	1.077
H-15R	01/07/15	907.67	1.118
H-16	01/13/15	924.73	1.035
H-17	01/07/15	901.32	1.134
H-19b0	01/07/15	906.27	1.067
IMC-461	01/05/15	927.82	1.000
SNL-01	01/05/15	938.23	1.030
SNL-02	01/05/15	936.29	1.010
SNL-03	01/13/15	937.93	1.027
SNL-05	01/05/15	936.54	1.008
SNL-06	01/06/15	1007.92	1.246
SNL-08	01/06/15	931.62	1.095
SNL-09	01/05/15	930.55	1.018
SNL-10	01/06/15	929.42	1.010
SNL-12	01/06/15	900.21	1.007
SNL-13	01/06/15	908.28	1.022
SNL-14	01/07/15	900.02	1.046
SNL-15	01/07/15	929.40	1.230
SNL-16	01/05/15	918.89	1.012
SNL-17	01/05/15	911.72	1.007
SNL-18	01/05/15	937.04	1.009
SNL-19	01/05/15	936.30	1.006
WIPP-11	01/09/15	938.76	1.038
WIPP-13	01/09/15	937.27	1.037
WIPP-19	01/07/15	931.62	1.053
WQSP-1	01/09/15	937.05	1.050
WQSP-2	01/07/15	939.59	1.047
WQSP-3	01/07/15	935.35	1.146
WQSP-4	01/15/15	906.58	1.076
WQSP-5	01/07/15	906.78	1.027
WQSP-6	01/07/15	912.89	1.017

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:

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

where $\mathrm{abs}(y)$ is absolute value and $\mathrm{sign}(y)$ 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 $-1 \le \{x,y\} \le 1$. 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-2000 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-2000 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-2000 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-2000 output), the template files (how to write the MODFLOW-2000 input), and the PEST control file (listing the ranges and initial values for the estimable parameters and the values and weights associated with observations). In Kuhlman (2015), the observed heads used in the 2014 PEST calibration were separated into four groups (1) wells inside the LWA boundary, (2) wells outside, but near the LWA boundary, (3) wells distant to the LWB, and (4) wells significantly impacted by pumping activities at the Mills Ranch. A different weighting factor was assigned to each group. For this analysis report, residuals are presented and discussed in terms of proximity to the LWA boundary, but the observed freshwater heads from the monitoring network were weighted equally. Additional observations representing the average heads north of the LWA boundary and south of the LWA boundary were used to help prevent over-smoothing of the estimated results across the LWA boundary. The additional observations are assigned to improve the fit in the area of interest (inside the WIPP LWA boundary), possibly at the expense of a somewhat poorer fit far from the LWA boundary and closer to the boundary conditions.

2.5 Figures Generated from Averaged MODFLOW-2000 Model

The MODFLOW-2000 model is run predictively using the averaged MODFLOW-2000 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 LWA boundary. Particle tracking stops when the particle crosses the LWA boundary. The Python script `convert_dtrkmf_output_for_surfer.py` converts the MODFLOW-2000 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-2000 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 LWA boundary, 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-2000 output by `mod2obs` are then merged into a common file for plotting using the Python script `merge_observed_modeled_heads.py`.

3 2015 Results

3.1 2015 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 stratigraphically bound by halite in the Rustler Formation. This region has high freshwater heads but extremely low transmissivities, essentially serving as a no-flow boundary for this area. Figure 2 shows the freshwater head contours are nearly perpendicular to the purple line.

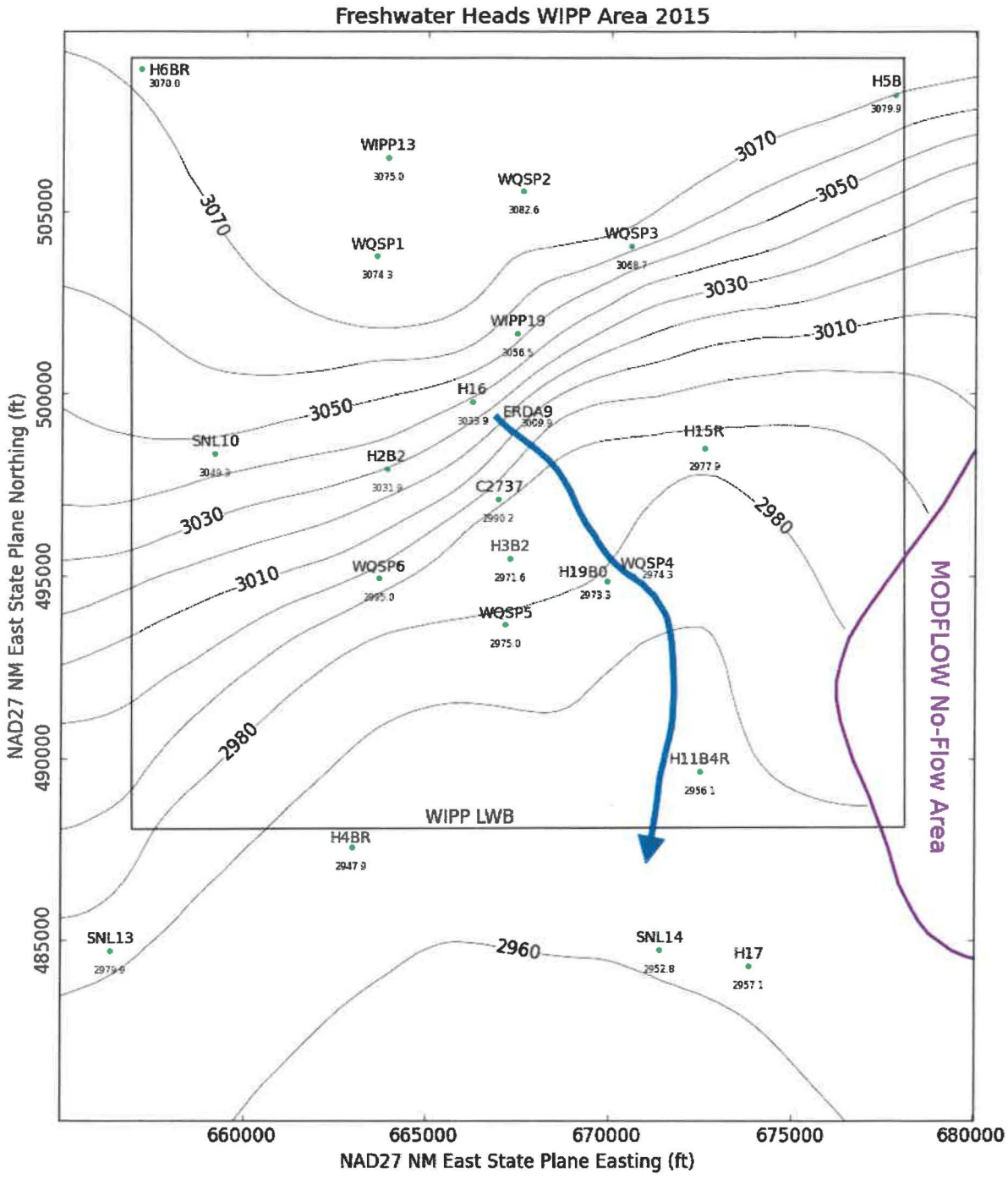

Figure 2. Model-generated January 2015 freshwater head contours (10 ft intervals) with observed head listed at each well. Blue line is water particle track from waste handling shaft to LWA boundary. Purple curve is Rustler halite margin.

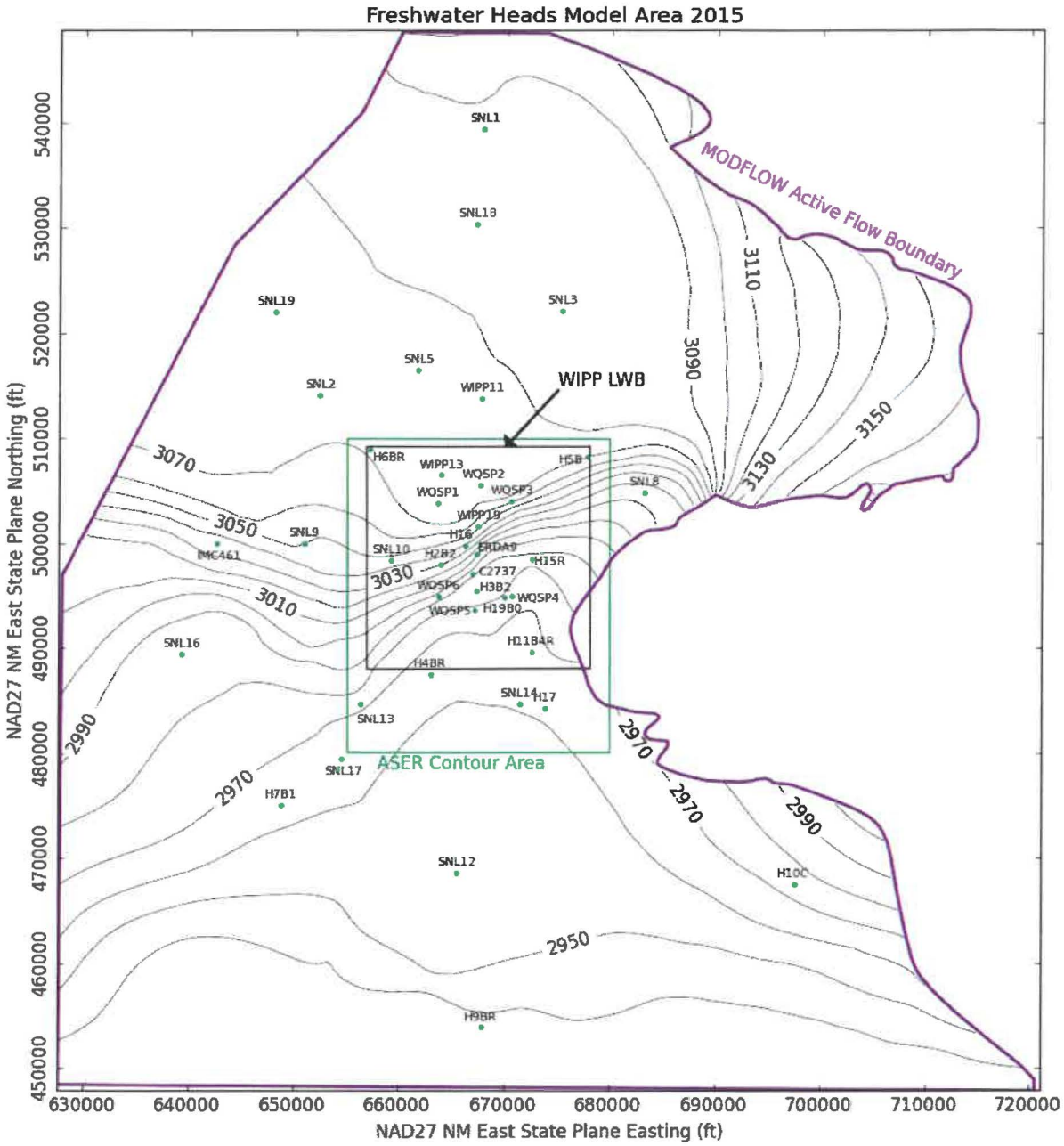

Figure 3. Model-generated January 2015 freshwater head contours (10 ft intervals) for entire model domain. Green rectangle indicates region contoured in Figure 2. Black square is LWA boundary.

3.2 2015 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 LWA boundary (a path length of 4109 m). Assuming the transmissive portion of the Culebra is four meters thick and has a constant porosity of 16%, the travel time to the LWA boundary is 3647 years (output from DTRKMF is

adjusted from an original 7.75-m Culebra thickness). This is an average velocity of 1.1 m/yr. This estimated flow velocity is higher than predicted in previous years (e.g., 0.70 m/yr in Kuhlman, 2015), because of the steeper gradient from the release point to the southern edge of the LWA boundary caused by pumping activities at the Mills Ranch.

3.3 2015 Measured versus 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 LWA boundary are represented by red crosses, wells outside but within 3 km of the LWA boundary are represented with green 'x's, and other wells within the MODFLOW-2000 model domain but distant from the WIPP site are indicated with blue stars. Additional observations representing the average heads north of the LWA boundary and south of the LWA boundary were used to help prevent over-smoothing of the estimated results across the LWA boundary. 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).

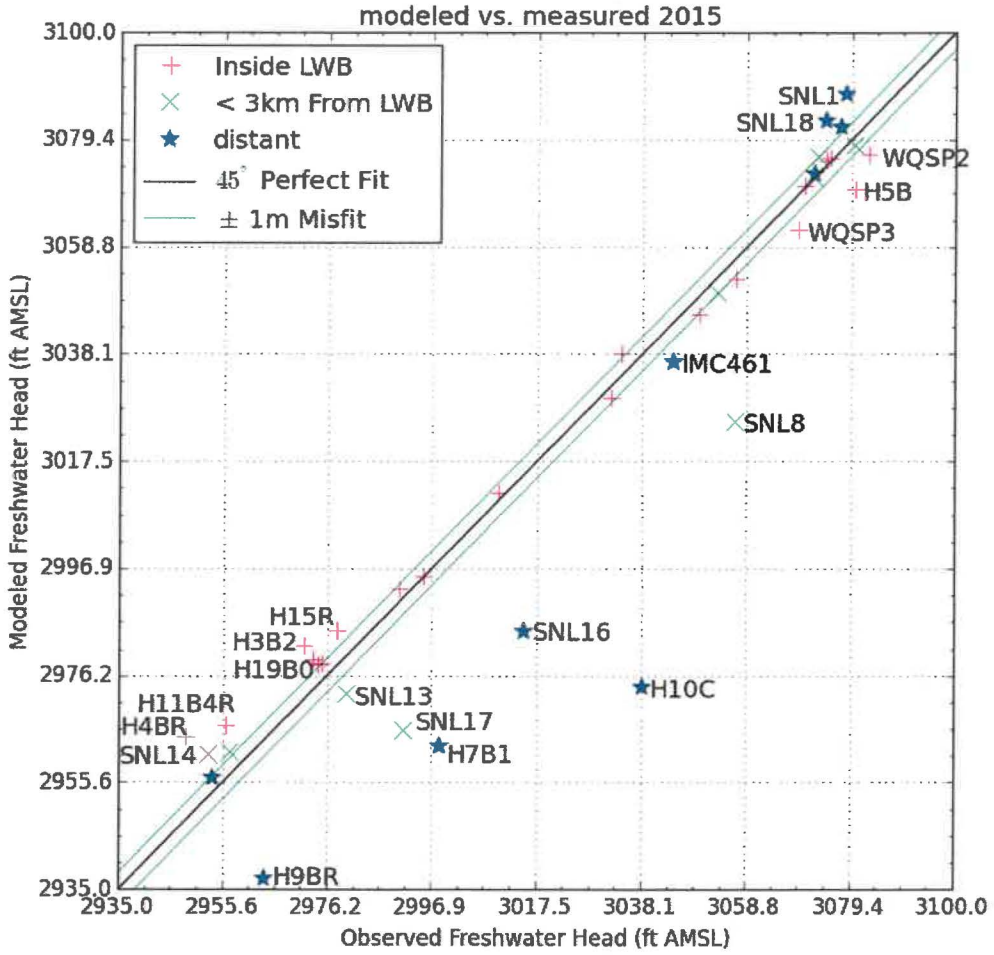

Figure 4. Measured versus modeled freshwater heads scatter plot.

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 3.3 ft (1 m) misfit above or below the perfect fit. Wells more than 1.5 m from the 1:1 line are labeled. Figure 4 shows that H-7b1, H-9bR, H-10c, SNL-8, SNL-16, and SNL-17 exhibit large (>25 ft) misfits. Head is under-predicted for these wells, all of which are outside the ASER contour area (i.e., the area of interest), because they are situated in relatively close proximity to a large number wells experiencing significant drawdown due to pumping activities at the Mills Ranch. Despite their proximity to the pumping activities, pressures in these wells are impacted by other effects, as discussed in Thomas (2016). For example, pressures in H-7b1 may be influenced by precipitation and localized pumping. H-10c shows above-average pressures due to nearby hydrocarbon development activities. Pressures in SNL-8, a low transmissivity area, have been rising since 2011. SNL-16 is situated in Nash Draw and exhibits elevated pressures due to above-average precipitation in September 2014. SNL-17 closely follows pumping at the Mills Ranch well, but despite its proximity to the pumping well, exhibits less drawdown than other wells further from the pumping well. This inconsistency may be due to differences in Culebra permeability or connectivity in the vicinity of SNL-17.

The calibrated parameters for equation 1 were A = 919.4, B = 8.5, C = 0.1, D = 2, E = 2.6, F = -2.5, and α = 1.5. The parameters α, E, and F had the largest relative change (~150-200%) compared to the starting values. Parameter D was within 100% of its original value and C was 91% away. All other parameters were <10% different from their original values.

The squared correlation coefficient (R^2) for the measured versus modeled data is listed in

Table 3. The calibration protocol did not improve the correlation between measured and modeled head values for wells inside the LWA boundary and only marginally improved the correlation for wells within three kilometers of the LWA boundary. Figure 5 and Figure 6 show the distribution of errors resulting from the PEST-adjusted model fit to observed data. The wells within the LWA boundary have an R^2 greater than 99%. The calibration improved the fit for the wells in and near the LWA boundary at the expense of fit to wells distant from the LWA boundary.

Table 3. 2015 Measured versus modeled correlation coefficients.

	dataset	measured versus modeled R^2
Uncalibrated	wells inside LWA boundary	0.993
	wells <3km from LWA boundary	0.960
	all wells	0.906
Calibrated	wells inside LWA boundary	0.993
	wells <3km from LWA boundary	0.962
	all wells	0.896

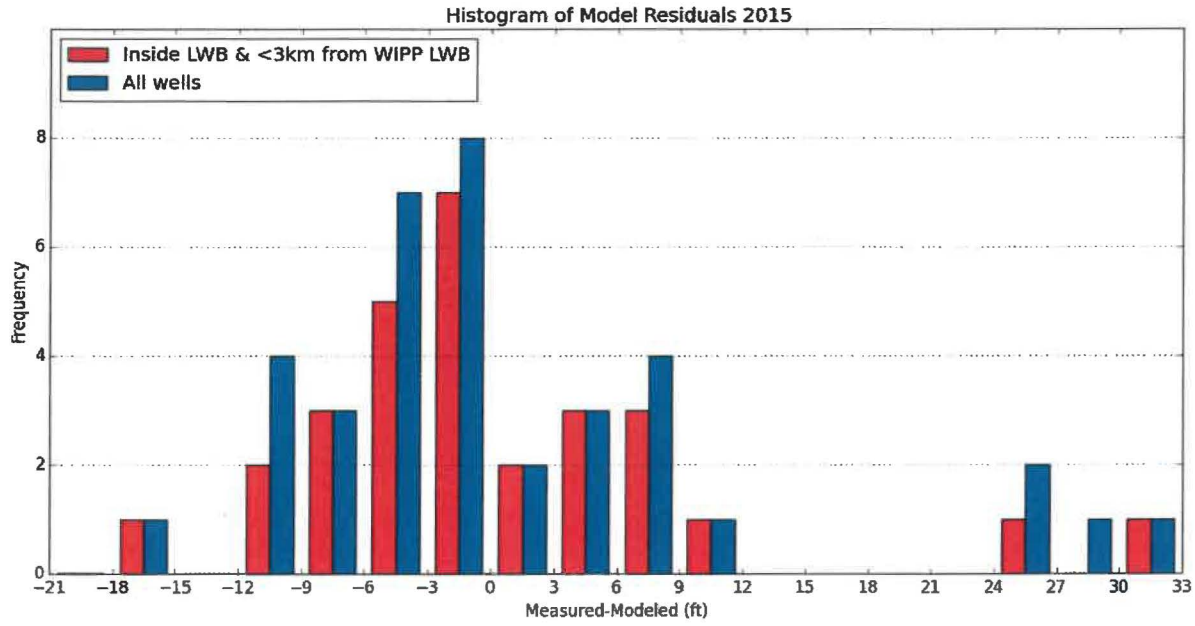

Figure 5. Histogram of measured minus modeled errors for 2015 (H-10c residual is off the right side of the figure at 63.9 ft).

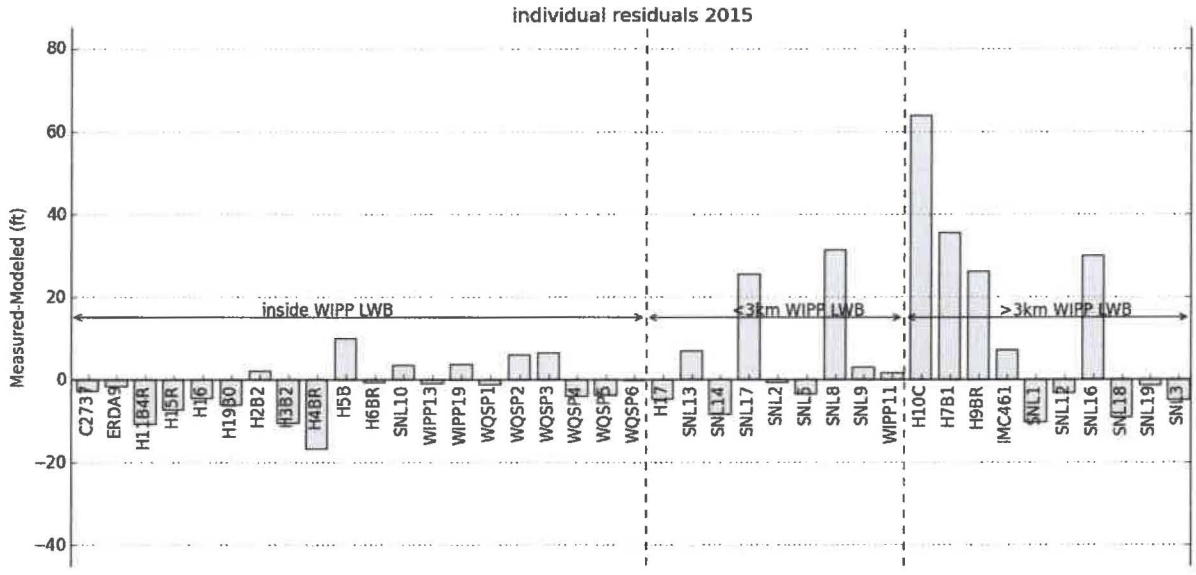

Figure 6. Measured minus modeled errors at each well location for 2015.

In Kuhlman (2015), an effort was made to not let PEST try to calibrate to wells significantly affected by pumping activities at the Mills Ranch. The weights for these select observations were minimized (i.e., set to 0.05) because they did not appear to be consistent with steady-state conditions. The difference between observed and modeled freshwater head within the LWA boundary was, on average, 9.4 ft and was as large as 40 ft. Drawdown at H-4bR had reached 50 ft since Mills Ranch pumping activities began in 2013 (Thomas, 2016). Approximately 15% of that drawdown was represented by the averaged

MODFLOW-2000 model generated from the calibration process. For this analysis report, lower weights were not assigned to observation wells impacted by Mills Ranch pumping activities because it would require minimizing at least 50% of the dataset (see Table 2).

The overall averaged MODFLOW-2000 model fit (i.e., inside, near, and distant to the LWA boundary) to the January 2015 observations is poor. However, Figure 6 shows that the fit improves with increasing proximity to the area of interest (i.e., within the LWA boundary). For example, the average residuals for locations distant, near, and inside the LWA boundary are 19.2, 9.5, and 5.1 ft, respectively. Figure 7 shows the difference between the modeled and observed freshwater heads is mainly due to pumping at the Mills Ranch (Thomas, 2016). The difference between observed and modeled freshwater head within the LWA boundary can be as large as 15 ft, particularly in the vicinity of H-4bR. Drawdown at H-4bR has reached 60 ft since Mills Ranch pumping activities began in 2013 (Thomas, 2016). Approximately 75% of that drawdown is captured by the averaged MODFLOW-2000 model generated from the calibration process followed this year. Unsurprisingly, the drawdown represented in the Culebra potentiometric surface contour map generated this year leads to a faster particle track time. Clearly, the averaged MODFLOW-2000 model cannot perfectly match the effects of the Mills Ranch pumping activities. In the future, SNL will explore modifications to the SP 9-9 modeling protocol that can accommodate annual water level fluctuations that do not appear to be consistent with steady-state conditions.

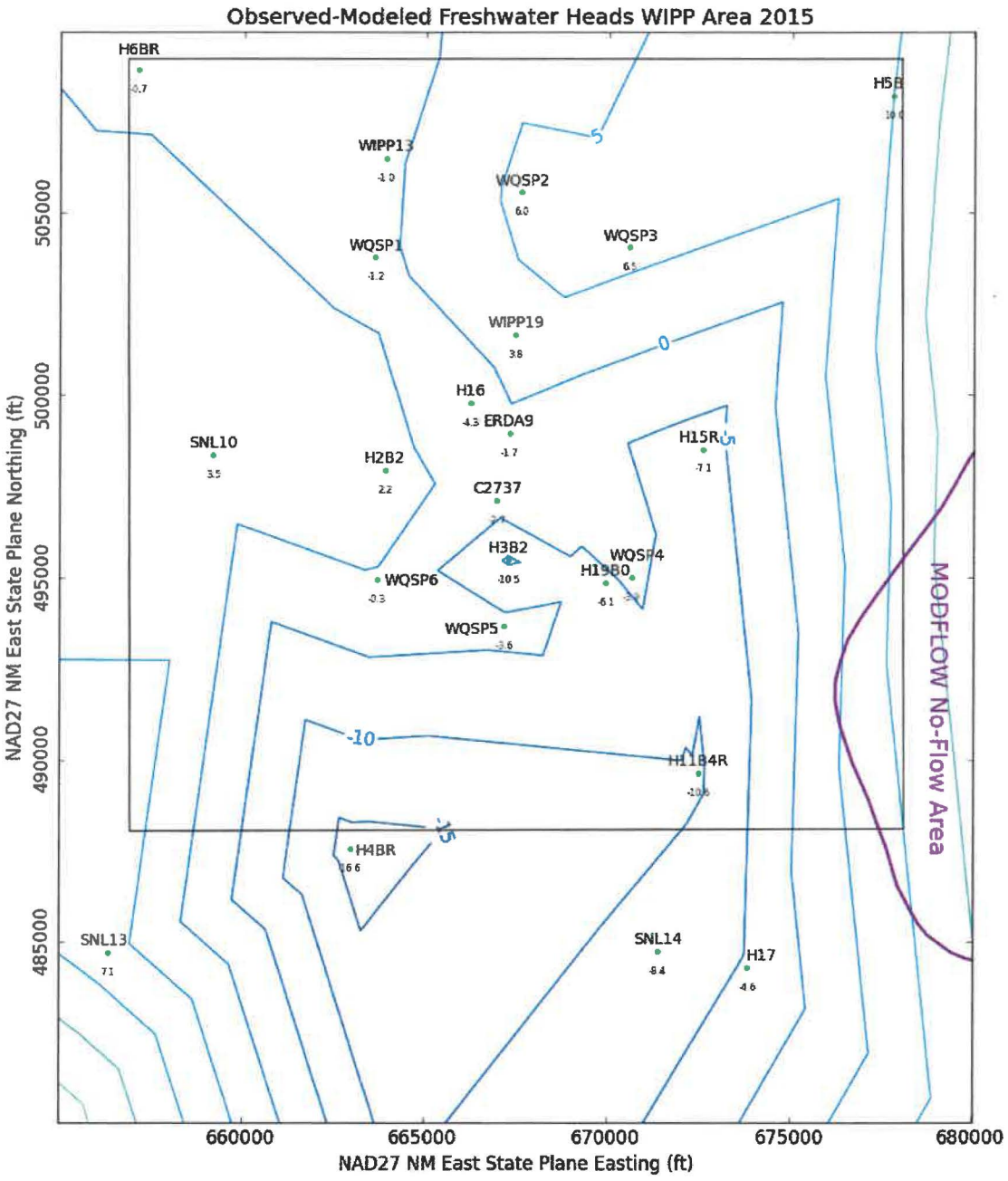

Figure 7. Triangulated contours (in five ft intervals) for measured minus modeled freshwater head.

4 References

Beauheim, R.L. 2008. *Analysis Plan for Evaluation and Recalibration of Culebra Transmissivity Fields*. AP-114, Revision 1. Sandia National Laboratories, Carlsbad, NM.

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*. DOE/WIPP-14-3503. US Department of Energy, Carlsbad, NM.

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*. Open-File Report 00-92. US Geological Survey, Reston, VA.

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.

Kuhlman, K.L. 2015. *Analysis Report for Preparation of 2014 Culebra Potentiometric Surface Contour Map*. Sandia National Laboratories, Carlsbad, NM.

Kuhlman, K.L. 2009. *Preparation of Culebra potentiometric surface contour maps*. Procedure SP 9-9, Revision 0. Sandia National Laboratories, Carlsbad, NM.

Moody, D.C. 2009. *Stipulated Final Order for Notice of Violation for Detection Monitoring Program*. Sandia National Laboratories, Carlsbad, NM.

Seal, B. 2016. *January 2015 Culebra ASER map data* (provided by Regulatory Environmental Services). Sandia National Laboratories, Carlsbad, NM.

Thomas, M.A. 2016. *WIPP Milestone Report: 2015 Culebra Groundwater Level Fluctuations* (memo to WIPP Records Center). Sandia National Laboratories, Carlsbad, NM.

Wagner, S. and M.A. Thomas, 2016, *Sandia National Laboratories Compliance Monitoring Parameter Assessment for 2015.* WBS 4.3.1. Sandia National Laboratories, Carlsbad, NM.

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-2000, 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-2000 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`.

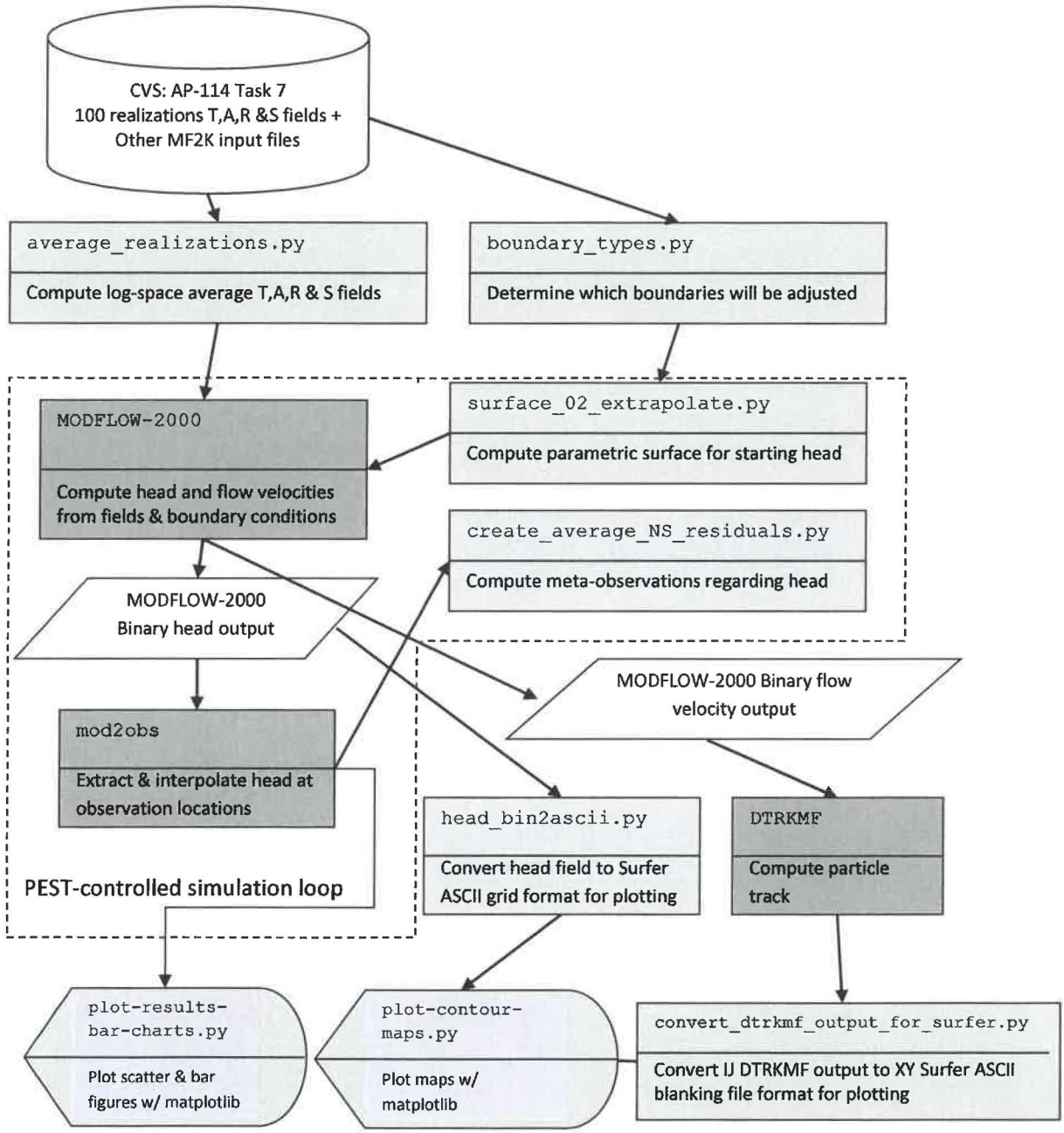

Figure 8. Process flowchart; dark gray indicates qualified programs, light gray are scripts written for this analysis

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.

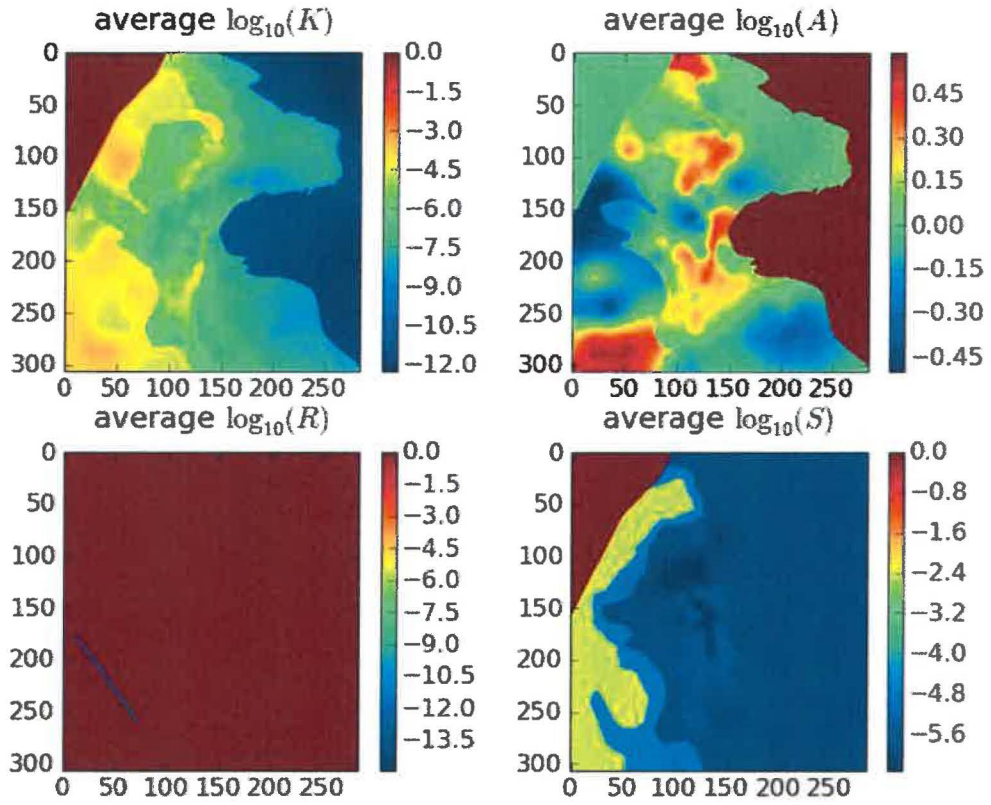

Figure 9. Plots of base-10 logarithms of average parameter fields; rows and columns are labeled on edges of figures.

Next, a subdirectory is created, and the averaged MODFLOW-2000 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 `dtrkmf.in` and `wippctrl.inp`, which utilizes the cell-by-cell budget file written by MODFLOW-2000 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 LWA boundary, since the calculation stops when the particle reaches the LWA boundary.

The Python script `head_bin2ascii.py` (§A-4.7) converts the MODFLOW-2000 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.bln`. 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.

A similar process is carried out in a new directory called `pest_02` (beginning line 146 of the driver script). The PEST calibration is carried out there, to keep it separate from the `original_average` simulation. Now the Python script `boundary_types.py` (§A-4.3) is also run, to create a new MODFLOW-2000 IBOUND array, where the two different types of boundary conditions are differentiated. This Python script uses the MODFLOW-2000 IBOUND array (`init_bnds_orig.inf` first ⅓ of Table 6) and the initial head array (`init_head_orig.mod` middle ⅓ of Table 6) as inputs, and writes a new MODFLOW-2000 IBOUND array (`init_bnds.inf` bottom ⅓ of Table 6) with constant-head nodes indicated in red in Figure 1 marked as -2 and other constant-head nodes remaining as -1 as output. The script differentiates between these two types of boundary conditions by checking if the starting head is <1000m. Starting heads >1000m are associated with the constant-head areas to the east of the halite margins (lighter gray areas in Figure 1).

Table 6. Input IBOUND, starting head, and output IBOUND array data corresponding to first row of MODFLOW-2000 model

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
943 943 943 943 943 943 943 943 943 943 943 943 943 943 943 943 943 943 943 943 943 943 943 943
943 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944
944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944
944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944
944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944
944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 944 1087 1085
1085 1082 1081 1080 1079 1078 1078 1077 1077 1077 1077 1077 1077 1078 1078 1077 1078 1079 1081
1083 1084 1086 1086 1088 1090 1092 1095 1096 1098 1099 1099 1100 1100 1100 1101 1101 1101 1102
1104 1104 1104 1104 1104 1104 1105 1105 1105 1105 1106 1106 1106 1107 1107 1109 1110 1110 1112
1113 1113 1114 1114 1115 1115 1115 1116 1116 1116 1117 1118 1119 1120 1121 1122 1123 1124 1125
1126 1127 1127 1129 1129 1130 1131 1132 1132 1133 1133 1134 1135 1136 1137 1137 1138 1139 1139
1140 1140 1140 1141 1142 1142 1142 1142 1142 1143 1143 1144 1144 1144 1145 1145 1146 1147 1147
1147 1147 1148 1148 1147 1146 1144 1143 1143 1145 1147 1148 1149 1150 1149 1149 1148 1149 1149
1151 1151 1151 1150 1152 1153 1154 1155
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -2
-2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2
-2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

Table 6 shows the data corresponding to the northernmost row of the MODFLOW-2000 model domain (284 entries long) for the two input files and one output file. In the top IBOUND array, the values are either 0 or -1, indicating either inactive (the region northwest of the no-flow area shown in dark gray in Figure 1) or constant head (both red and light gray cells in Figure 1). The first 284 values from the initial head file (reformatted from scientific notation to integers to facilitate printing) show a jump from approximately 944 (in blue) to >1000 (in red). These same cells are colored in the output, showing how the initial head value is used to distinguish between the two types of constant-head boundaries.

MODFLOW-2000 treats any cells as constant head which have an IBOUND entry < 0, so both -2 and -1 are the same to MODFLOW-2000, 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_2015ASER.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_2015ASER.smp`), and the weights are read in from the input file (`obs_loc_2015ASER.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-2000 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 RES for the ASER.

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-2000 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.p
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-2000 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 2015 heads in mod2obs bore sample file format	meas_head_2015ASER.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-2000	modflow_files.dat
393	input	listing of wells and geographic groupings	obs_loc_2015ASER.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-2000 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-2000 heads	modeled_head_pest_02.grd
5.3K	script output	formatted DTRKMF particle	dtrk_output_pest_02.bln
12K	PEST output	matrix condition numbers	bc_adjust_2015ASER.cnd
2.7K	PEST output	binary intermediate file	bc_adjust_2015ASER.drf
7.3K	PEST output	binary intermediate file	bc_adjust_2015ASER.jac
7.4K	PEST output	binary intermediate file	bc_adjust_2015ASER.jco
9.8K	PEST output	binary intermediate file	bc_adjust_2015ASER.jst
3.8K	PEST output	parameter statistical matrices	bc_adjust_2015ASER.mtt
477	PEST output	parameter file	bc_adjust_2015ASER.par
56K	PEST output	optimization record	bc_adjust_2015ASER.rec
4.5K	PEST output	model outputs for last iteration	bc_adjust_2015ASER.rei
8.2K	PEST output	summary of residuals	bc_adjust_2015ASER.res
28	PEST output	binary restart file	bc_adjust_2015ASER.rst
22K	PEST output	relative parameter sensitivities	bc_adjust_2015ASER.sen
3.9K	PEST output	absolute parameter sensitivities	bc_adjust_2015ASER.seo
214K	png image	matplotlib plot (Fig. 2)	aser-area-contour-map2015.png
226K	png image	matplotlib plot (Fig. 3)	large-area-contour-map2015.png
161K	png image	matplotlib plot (Fig. 7)	aser-area-modobs-contour-map2015.png
34K	png image	matplotlib plot (Fig. 5)	model-error-histogram2015.png
25K	png image	matplotlib plot (Fig. 6)	model-error-residuals2015.png
117K	png image	matplotlib plot (Fig. 4)	scatter_pest_02_2015.png

Table 2: Listing of Output Files

6.3 Individual scripts

6.3.1 Bash shell script checkout_average_run_modflow.sh

```
#!/bin/bash

set -o nounset # explode if using an un-initialized variable
set -o errexit # exit on non-zero error status of sub-command

# this script makes the following directory substructure
#
#  current_dir \----- Outputs  (calibrated parameter fields - INPUTS)
#               \---- Inputs   (other modflow-2000 files - INPUTS)
#                \--- original_average  (foward sim using average fields)
#                 \-- bin      (MODFLOW-2000 and DTRKMF binaries)
#                  \- pest_0?  (pest-adjusted results)

set -o xtrace

echo " ^^^^^^^^^^^^^^^^^^^^ "
echo " checking out T fields"
echo " ^^^^^^^^^^^^^^^^^^^^ "

# these will checkout the calibrated parameter-field data into subdirectories
# checkout things that are different for each of the 100 realiztaions
for d in `cat keepers`
do
  cvs -d /nfs/data/CVSLIB/Tfields checkout Outputs/${d}/modeled_{K,A,R,S}_field.mod
done

# checkout MODFLOW-2000 input files that are constant for across all realizations
cvs -d /nfs/data/CVSLIB/Tfields checkout Inputs/data/elev_{top,bot}.mod
cvs -d /nfs/data/CVSLIB/Tfields checkout Inputs/data/init_{bnds.inf,head.mod}
cvs -d /nfs/data/CVSLIB/Tfields checkout Inputs/modflow/mf2k_culebra.{lmg,lpf}
cvs -d /nfs/data/CVSLIB/Tfields checkout Inputs/modflow/mf2k_head.{ba6,nam,oc,dis,rch}

# modify the path of "updated" T-fields, so they are all at the
# same level in the directory structure (simplifying scripts elsewhere)

if [ -a keepers_short ]
then
    rm keepers_short
fi
touch keepers_short

for d in `cat keepers`
do
  bn=`basename ${d}`
  # test whether it is a compount path
  if [ ${d} != ${bn} ]
      then
      dn=`dirname ${d}`
```

```
        mv ./Outputs/${d} ./Outputs/

        # put an empty file in the directory to indicate
        # what the directory was previously named
        touch ./Outputs/${bn}/${dn}
    fi

    # create a keepers list without directories
    echo ${bn} >> keepers_short
done

# ---------------------------------------------------

echo " ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ "
echo " perform averaging across all realizations "
echo " ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ "

python average_realizations.py

# ---------------------------------------------------

echo " ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ "
echo " setup copies of files constant between all realizations "
echo " ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ "

# directory for putting original base-case results in
od=original_average

if [ -d ${od} ]
then
    echo ${od}" directory exists: removing and re-creating"
    rm -rf ${od}
fi

mkdir ${od}
cd ${od}
echo `pwd`

# link to unchanged input files
for file in `cat ../modflow_files.dat`
do
  ln -sf ${file} .
done

# link to averaged files computed in previous step
for f in {A,R,K,S}
do
  ln -sf ../modeled_${f}_field.avg ./modeled_${f}_field.mod
done

ln -sf elev_top.mod fort.33
```

```
ln -sf elev_bot.mod fort.34

echo "^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^"
echo " run original MODFLOW-2000 and DTRKMF and export results for plotting"
echo "^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^"

# run MODFLOW-2000, producing average head and CCF
/utilities/modflow2000 mf2k_head.nam

# run DTRKMF, producing particle track (from ccf)
/utilities/dtrkmf `cat dtrkmf.in`

# convert binary MODFLOW-2000 head output to Surfer ascii grid file format
ln -sf ../head_bin2ascii.py .
python head_bin2ascii.py
mv modeled_head_asciihed.grd modeled_head_${od}.grd

# convert DTRKMF output from cells to X,Y and
# save in Surfer blanking file format
ln -sf ../convert_dtrkmf_output_for_surfer.py .
python convert_dtrkmf_output_for_surfer.py
mv dtrk_output.bln dtrk_output_${od}.bln

# extract head results at well locations and merge with observed
# head file for easy scatter plotting in Excel (tab delimited)
for file in `cat ../mod2obs_files.dat`
do
  ln -sf ${file} .
done

ln -sf ../meas_head_2015ASER.smp .
ln -sf ../obs_loc_2015ASER.dat .

/utilities/mod2obs <mod2obs_head.in
ln -sf ../merge_observed_modeled_heads.py
python merge_observed_modeled_heads.py
mv both_heads.smp modeled_vs_observed_head_${od}.txt

# go back down into root directory
cd ..
echo `pwd`

echo "^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^"
echo " setup and run PEST to optimize parametric surface to set BC "
echo "^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^"

for p in pest_02
do

  if [ -d ${p} ]
      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_2015ASER
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_asciihed.grd modeled_head_${p}.grd
211
212     ln -sf ../convert_dtrkmf_output_for_surfer.py .
213     python convert_dtrkmf_output_for_surfer.py
214     mv dtrk_output.bln dtrk_output_${p}.bln
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**

```
from math import log10,pow

nrow = 307
ncol = 284
nel = nrow*ncol
nfr = 100  # number of fields (realizations)
nft = 4    # number of field types

def floatload(filename):
    """Reads file (a list of strings, one per row) into a list of strings."""
    f = open(filename,'r')
    m = [float(line.rstrip()) for line in f]
    f.close()
    return m

types = ['K','A','R','S']

# get list of 100 best calibrated fields
flist = open('keepers_short','r')
runs = flist.read().strip().split('\n')
flist.close()

# initialize to help speed lists up a bit
# nfr (100) realizations of each
fields = []
for i in xrange(nft):
    fields.append([None]*nfr)
    for i in xrange(nfr):
        # each realization being nel (87188) elements
        fields[-1][i] = [None]*nel

# read in all realizations
print 'reading ...'
for i,run in enumerate(runs):
    print i,run
    for j,t in enumerate(types):
        fields[j][i][0:nel] = floatload('Outputs/'+ run +'/modeled_'+ t +'_field.mod')

# open up files for writing
fh = []
for t in types:
    fh.append(open('modeled_'+ t +'_field.avg','w'))

# transpose fields to allow slicing across realizations, rather than across cells
for j in range(len(types)):
    fields[j] = zip(*(fields[j]))

print 'writing ...'
# 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

```
nx = 284        # number columns in model grid
ny = 307        # number rows
nel = nx*ny

def intload(filename):
    """Reads file (a 2D integer array) as a list of lists.
    Outer list is rows, inner lists are columns."""
    f = open(filename,'r')
    m = [[int(v) for v in line.rstrip().split()] for line in f]
    f.close()
    return m

def intsave(filename,m):
    """Writes file as a list of lists as a 2D integer array, format '%3i'.
    Outer list is rows, inner lists are columns."""
    f = open(filename,'w')
    for row in m:
        f.write(' '.join(['%2i' % col for col in row]) + '\n')
    f.close()

def floatload(filename):
    """Reads file (a list of real numbers, one number each row) into a list of floats."""
    f = open(filename,'r')
    m = [float(line.rstrip()) for line in f]
    f.close()
    return m

def reshapev2m(v):
    """Reshape a vector that was previously reshaped in C-major order from a matrix,
    back into a matrix (here a list of lists)."""
    m = [None]*ny
    for i,(lo,hi) in enumerate(zip(xrange(0, nel-nx+1, nx), xrange(nx, nel+1, nx))):
        m[i] = v[lo:hi]
    return m

########################################

# read in original MODFLOW-2000 IBOUND array (only 0,1, and -1)
ibound = intload('init_bnds_orig.inf')

# read in initial heads
h = reshapev2m(floatload('init_head_orig.mod'))

# discriminate between two types of constant head boundaries
# -1) CH, where value > 1000 (area east of halite margin)
# -2) CH, where value < 1000 (single row/column of cells along edge of domain

for i,row in enumerate(ibound):
    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-2000
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**

```
prefix = '2015ASER'

################################################
## pest instruction file reads output from mod2obs
fin = open('meas_head_%s.smp' % prefix,'r')

# each well is a [name,head] pair
wells = [[line.split()[0],line.split()[3]] for line in fin]
fin.close()


fout = open('modeled_head.ins','w')
fout.write('pif @\n')
for i,well in enumerate(wells):
        fout.write("l1 [%s]39:46\n" % well[0])
fout.close()

# exponential surface used to set initial head everywhere
# except east of the halite margins, where the land surface is used.
# initial guesses come from AP-114 Task report
params = [928.0, 8.0, 1.2, 1.0, 1.0, -1.0, 0.5]
pnames = ['a',   'b', 'c', 'd', 'e', 'f', 'exp']

fout = open('avg_NS_res.ins','w')
fout.write("""pif @
l1 [medianN]1:16
l1 [medianS]1:16
l1 [meanN]1:16
l1 [meanS]1:16
""")
fout.close()


##############################
## pest template file
ftmp = open('surface_par_params.ptf','w')
ftmp.write('ptf @\n')
for n in pnames:
        ftmp.write('@      %s       @\n' % n)
ftmp.close()


##################
## pest parameter file

fpar = open('surface_par_params.par','w')
fpar.write('double  point\n')
for n,p in zip(pnames,params):
    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  lines = fin.readlines()
90  location = [line.rstrip().split()[1] for line in lines]
91  groups = [line.rstrip().split()[2] for line in lines]
92  fin.close()
93
94  weights = []
95
96  numnorth = 0.0
97  numsouth = 0.0
98
99  for l,g in zip(location,groups):
100     if "M" in g:
```

```
            # wells affected by Mills ranch pumping
            # don't include them in count for weighting averages
            weights.append(1.0)
        else:
            # inside LWB
            if l == '0':
                weights.append(1.0)
            # near LWB
            elif l == '1':
                weights.append(1.0)
            # distant to LWB
            elif l == '2':
                weights.append(1.0)
            elif l == '99':
                weights.append(1.0) # AEC-7

            if "N" in g:
                numnorth += 1.0
            elif "S" in g:
                # not including ones with "M"
                numsouth += 1.0

for name,head,w in zip(zip(*wells)[0],zip(*wells)[1],weights):
    f.write('%s  %s  %.3f   ss_head\n' % (name,head,w))

# weight the averages by the number of wells in average


# split the weight between the mean and median
f.write("""medianN  0.0  %.2f  avg_head
medianS  0.0   %.2f  avg_head
meanN    0.0   %.2f  avg_head
meanS    0.0   %.2f  avg_head
""" % (numnorth,numsouth,numnorth,numsouth))

f.write("""* model command line
./run_02_model
* model input/output
surface_par_params.ptf surface_par_params.in
modeled_head.ins modeled_head.smp
avg_NS_res.ins avg_NS_res.smp
""")
f.close()
```

6.3.5 Python script **surface_02_extrapolate.py**

```
from itertools import chain
from math import sqrt

def matload(filename):
    """Reads file (a 2D string array) as a list of lists.
    Outer list is rows, inner lists are columns."""
    f = open(filename,'r')
    m = [line.rstrip().split() for line in f]
    f.close()
    return m

def floatload(filename):
    """Reads file (a list of real numbers, one number each row) into a list of floats."""
    f = open(filename,'r')
    m = [float(line.rstrip()) for line in f]
    f.close()
    return m

def reshapem2v(m):
    """Reshapes a rectangular matrix into a vector in same fashion as numpy.reshape().
    which is C-major order"""
    return list(chain(*m))

def sign(x):
    """ sign function"""
    if x<0:
        return -1
    elif x>0:
        return +1
    else:
        return 0

########################################

# read in modified IBOUND array, with the cells to modify set to -2
ibound = reshapem2v(matload('init_bnds.inf'))

h = floatload('init_head_orig.mod')

# these are relative coordinates, -1 <= x,y < +1
x = floatload('rel_x_coord.dat')
y = floatload('rel_y_coord.dat')

# unpack surface parameters (one per line)
# z = A + B*(y + D*sign(y)*sqrt(abs(y)))+C*(E*x**3 - F*x**2 - x)

finput = open('surface_par_params.in','r')
try:
    a,b,c,d,e,f,exp = [float(line.rstrip()) for line in finput]
```

```
except ValueError:
    # python doesn't like 'D' in 1.2D-4 notation used by PEST sometimes.
    finput.seek(0)
    lines =  [line.rstrip() for line in finput]
    for i in range(len(lines)):
        lines[i] = lines[i].replace('D','E')
    a,b,c,d,e,f,exp = [float(line) for line in lines]

finput.close()

# file to output initial/boundary head for MODFLOW-2000 model
fout = open('init_head.mod','w')
for i in xrange(len(ibound)):
    if ibound[i] == '-2' or ibound[i] == '1':
        # apply exponential surface to active cells (ibound=1) -> starting guess
        # and non-geologic boundary conditions (ibound=-2) -> constant head value
        if y[i] == 0:
            fout.write('%.7e \n' % (a + c*(e*x[i]**3 + f*x[i]**2 - x[i])))
        else:
            fout.write('%.7e \n' % (a + b*(y[i] + d*sign(y[i])*abs(y[i])**exp) +
                                    c*(e*x[i]**3 + f*x[i]**2 - x[i])))
    else:
        # use land surface at constant head east of halite boundary
        # ibound=0 doesn't matter (inactive)
        fout.write('%.7e\n' % h[i])

fout.close()
```

6.3.6 Bash shell script `run_02_model`

```
#!/bin/bash

#set -o xtrace

#echo 'step 1: surface extrapolate'
python surface_02_extrapolate.py

# run modflow-2000
#echo 'step 2: run modflow-2000'
##../bin/mf2k/mf2k_1.6.release mf2k_head.nam  >/dev/null
/utilities/modflow2000 mf2k_head.nam  >/dev/null

# run mod2obs
#echo 'step 3: extract observations'
##../bin/Builds/Linux/mod2obs.exe < mod2obs_head.in  >/dev/null
/utilities/mod2obs <mod2obs_head.in  >/dev/null

# create meta-observations of N vs. S
python create_average_NS_residuals.py
```

6.3.7 Python script **head_bin2ascii.py**

```
import struct
from sys import argv,exit

class FortranFile(file):
    """ modified from May 2007 Enthought-dev mailing list post by Neil Martinsen-Burrell"""

    def __init__(self,fname, mode='r', buf=0):
        file.__init__(self, fname, mode, buf)
        self.ENDIAN = '<'  # little endian
        self.di = 4  # default integer (could be 8 on 64-bit platforms)

    def readReals(self, prec='f'):
        """Read in an array of reals (default single precision) with error checking"""
        # read header (length of record)
        l = struct.unpack(self.ENDIAN+'i',self.read(self.di))[0]
        data_str = self.read(l)
        len_real = struct.calcsize(prec)
        if l % len_real != 0:
            raise IOError('Error reading array of reals from data file')
        num = l/len_real
        reals = struct.unpack(self.ENDIAN+str(num)+prec,data_str)
        # check footer
        if struct.unpack(self.ENDIAN+'i',self.read(self.di))[0] != l:
            raise IOError('Error reading array of reals from data file')
        return list(reals)

    def readInts(self):
        """Read in an array of integers with error checking"""
        l = struct.unpack('i',self.read(self.di))[0]
        data_str = self.read(l)
        len_int = struct.calcsize('i')
        if l % len_int != 0:
            raise IOError('Error reading array of integers from data file')
        num = l/len_int
        ints = struct.unpack(str(num)+'i',data_str)
        if struct.unpack(self.ENDIAN+'i',self.read(self.di))[0] != l:
            raise IOError('Error reading array of integers from data file')
        return list(ints)

    def readRecord(self):
        """Read a single fortran record (potentially mixed reals and ints)"""
        dat = self.read(self.di)
        if len(dat) == 0:
            raise IOError('Empy record header')
        l = struct.unpack(self.ENDIAN+'i',dat)[0]
        data_str = self.read(l)
        if len(data_str) != l:
            raise IOError('Didn''t read enough data')
        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-2000 model (or at least only one layer
84
85      # format of MODFLOW-2000 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
```

```
            kstp,kper,pertim,totim,text,ncol,nrow,ilay = struct.unpack(fmt,h)

            # print status/confirmation to terminal
            print kstp,kper,pertim,totim,text,ncol,nrow,ilay

            h = ff.readReals()

    ff.close()

    xmin, xmax = (601700.0,630000.0)
    ymin, ymax = (3566500.0,3597100.0)
    hmin = min(h)
    hmax = max(h)

    # write output in Surfer ASCII grid format
    f = open(outfn,'w')
    f.write("""DSAA
    %i %i
    %.1f %.1f
    %.1f %.1f
    %.8e %.8e
    """ %(ncol,nrow,xmin,xmax,ymin,ymax,hmin,hmax) )
    hmat = reshapev2m(h,ncol,nrow)

    # MODFLOW-2000 starts data in upper-left corner
    # Surfer expects data starting in lower-left corner
    # flip array in row direction

    floatmatsave(f,hmat[::-1])
    f.close()
```

6.3.8 Python script **merge_observed_modeled_heads.py**

```
fobs = open('meas_head_2015ASER.smp','r') # measured head
fmod = open('modeled_head.smp','r')       # modeled head
fwgt = open('obs_loc_2015ASER.dat','r')   # weights
fdb = open('spec_wells.crd','r')          # x/y coordinates

fout = open('both_heads.smp','w')         # resulting file

# read in list of x/y coordinates, key by well name
wells = {}
for line in fdb:
    well,x,y = line.split()[0:3] # ignore last column
    wells[well.upper()] = [x,y]
fdb.close()

fout.write('\t'.join(['#NAME','UTM-NAD27-X','UTM-NAD27-Y',
                      'OBSERVED','MODELED','OBS-MOD','WEIGHT'])+'\n')

for sobs,smod,w in zip(fobs,fmod,fwgt):
    obs = float(sobs.split()[3])
    mod = float(smod.split()[3])
    name = sobs.split()[0].upper()
    fout.write('\t'.join([name,wells[name][0],wells[name][1],
                          str(obs),str(mod),str(obs-mod),
                          w.rstrip().split()[1]])+'\n')

fobs.close()
fmod.close()
fwgt.close()
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.bln','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

```
import numpy as np
from scipy.interpolate import griddata

manualFix = False
simpleContours = True

if not manualFix:
    import matplotlib
    matplotlib.use('Agg')

import matplotlib.pyplot as plt
from mpl_toolkits.basemap import pyproj

# http://spatialreference.org/ref/epsg/26713/
# http://spatialreference.org/ref/epsg/32012/
putm = pyproj.Proj(init='epsg:26713') # UTM Zone 13N NAD27 (meters)
pstp = pyproj.Proj(init='epsg:32012') # NM state plane east NAD27 (meters)

def transform(xin,yin):
    """does the default conversion from utm -> state plane
    then also convert to feet from meters"""
    xout,yout = pyproj.transform(putm,pstp,xin,yin)
    xout /= M2FT
    yout /= M2FT
    return xout,yout

year = '2015'
fprefix = 'pest_02/'
mprefix = '../../../well-location-maps/wipp-polyline-data/'
cfname = fprefix + 'modeled_head_pest_02.grd'
pfname = fprefix + 'dtrk_output_pest_02.bln'
wfname = fprefix + 'modeled_vs_observed_head_pest_02.txt'
wf13name = '../../2013_ASER/ASER_2013/'+wfname

M2FT = 0.3048

# read in well-related things
# %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# load in observed, modeled, obs-mod, (all in meters)
res = np.loadtxt(wfname,skiprows=1,usecols=(3,4,5))
res /= M2FT # convert heads to feet
wellutmx,wellutmy,obs,obsmod = np.loadtxt(wfname,skiprows=1,usecols=(1,2,3,5),unpack=True)
wellutmx13,wellutmy13,obs13 = np.loadtxt(wf13name,skiprows=1,usecols=(1,2,3),unpack=True)
wellx,welly = transform(wellutmx,wellutmy)
wellx13,welly13 = transform(wellutmx13,wellutmy13)
obs /= M2FT
obs13 /= M2FT
obsmod /= M2FT
names = np.loadtxt(wfname,skiprows=1,usecols=(0,),dtype='|S6')
```

```
names13 = np.loadtxt(wf13name,skiprows=1,usecols=(0,),dtype='|S6')

weights = {}
fh = open('obs_loc_%sASER.dat' % year, 'r')
lines = fh.readlines()
fh.close()
for line in lines:
    f = line.split()
    # inout is integer flag: 0=inside LWB, 1=near LWB, 2=far from LWB
    # ns is character flag: N=north, C=central, S=south, M=imacted by Mill's ranch pumping
    weights[f[0].upper()] = {'inout':int(f[1]), 'ns':f[2]}

#print 'DEBUG well coordinates'
#for n,ux,uy,x,y in zip(names,wellutmx,wellutmy,wellx,welly):
#    print n,ux,uy,'::',x,y

# read in head-related things
# %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
h = np.loadtxt(cfname,skiprows=5) # ASCII matrix of modeled head in meters AMSL
h[h<0.0] = np.NaN  # no-flow zone in northeast
h[h>1000.0] = np.NaN # constant-head zone in east
h /= M2FT # convert elevations to feet

# surfer grid is implicit in header
# create grid from min/max UTM NAD27 coordinates (meters)
utmy,utmx = np.mgrid[3566500.0:3597100.0:307j, 601700.0:630000.0:284j]

# head contour coords
hx,hy = transform(utmx,utmy)
del utmx,utmy

# read in particle-related things
# %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
px,py = transform(*np.loadtxt(pfname,skiprows=1,delimiter=',',usecols=(0,1),unpack=True))
part = np.loadtxt(pfname,skiprows=1,delimiter=',',usecols=(2,))

# read in MODFLOW-2000 model, WIPP LWB & ASER contour domain (UTM X & Y)
# %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
modx,mody =   transform(*np.loadtxt(mprefix+'total_boundary.dat',unpack=True))
wipputmx,wipputmy = np.loadtxt(mprefix+'wipp_boundary.dat',
                               usecols=(0,1),unpack=True)
wippx,wippy = transform(wipputmx,wipputmy)
aserx,asery = transform(*np.loadtxt(mprefix+'ASER_boundary.csv',
                               delimiter=',',usecols=(1,2),unpack=True))

#print 'DEBUG WIPP coordinates'
for ux,uy,x,y in zip(wipputmx,wipputmy,wippx,wippy):
    print ux,uy,'::',x,y

a = []
```

```
# plot contour map of entire model area
# ***************************************
fig = plt.figure(1,figsize=(12,16))
ax = fig.add_subplot(111)
lev = 3000 + np.arange(17)*10
CS = ax.contour(hx,hy,h,levels=lev,colors='k',linewidths=0.5)
ax.clabel(CS,lev[::2],fmt='%i')
if simpleContours:
    lev = 2900 + np.arange(27)*10
    hZ = griddata((wellx,welly),obs,(hx,hy),method="cubic")
    CS = ax.contour(hx,hy,hZ,levels=lev,linestyles='--',colors='gray',linewidth=0.5)
    #CS = ax.tricontour(wellx,welly,obs,levels=lev,linestyles='--',colors='gray',linewidth
    ax.clabel(CS,lev[::2],fmt='%i')
ax.plot(wippx,wippy,'k-')
ax.plot(aserx,asery,'g-')
ax.plot(modx,mody,'-',color='purple',linewidth=2)
ax.plot(wellx,welly,linestyle='none',marker='.',
        markeredgecolor='green',markerfacecolor='green')
ax.set_xticks(630000 + np.arange(10.0)*10000)
ax.set_yticks(450000 + np.arange(10.0)*10000)
labels = ax.get_yticklabels()
for label in labels:
    label.set_rotation(90)
for x,y,n in zip(wellx,welly,names):
    # plot just above
    a.append(plt.annotate(n,xy=(x,y),xytext=(0,5),
                textcoords='offset points',
                horizontalalignment='center',
                fontsize=8))
plt.axis('image')
ax.set_title('Freshwater Heads Model Area '+year)
ax.set_xlabel('NAD27 NM East State Plane Easting (ft)')
ax.set_ylabel('NAD27 NM East State Plane Northing (ft)')

# compute travel time and path length to WIPP LWB
# ***************************************

# compute incremental distance between times
pd = M2FT*np.sqrt((px[1:]-px[:-1])**2 + (py[1:]-py[:-1])**2)

ax.text(688000,537000,'MODFLOW-2000 Active Flow Boundary',size=12,rotation=-26,color='purp
ax.annotate('WIPP LWB',xy=(670000,509200),xytext=(675000,515000),
            fontsize=12,arrowprops=dict(facecolor='black',width=1))
ax.annotate('ASER Contour Area',xy=(658000,478500),fontsize=12,color='green')

print 'particle length:',pd.sum(),' (meters);  travel time:',part[-1],' (years); ',
print '  avg speed:',pd.sum()/part[-1],'(m/yr)'

if manualFix:
    # manually fix labels
    for lab in a:
```

```
            lab.draggable()
        plt.show()
    else:
        if simpleContours:
            fnout = 'large-area-contour-map_with_linear_'+year+'.png'
        else:
            fnout = 'large-area-contour-map'+year+'.png'
        plt.savefig(fnout)
    plt.close(1)

    del lev,CS
    mask = np.logical_and(np.logical_and(hx>aserx.min(),hx<aserx.max()),
                          np.logical_and(hy>asery.min(),hy<asery.max()))
    h[~mask] = np.NaN

    a = []

    # plot contour map of ASER-figure area
    # ************************************
    fig = plt.figure(1,figsize=(12,16))
    ax = fig.add_subplot(111)
    lev = 3000 + np.arange(17)*5
    CS = ax.contour(hx,hy,h,levels=lev,colors='k',linewidths=0.5)
    ax.clabel(CS,lev[::2],fmt='%i',inline_spacing=2)
    if simpleContours:
        lev = 2900 + np.arange(37)*5
        hZ = griddata((wellx,welly),obs,(hx,hy),method="cubic")
        CS = ax.contour(hx,hy,hZ,levels=lev,linestyles='--',colors='gray',linewidth=0.5)

        #CS = ax.tricontour(wellx,welly,obs,levels=lev,linestyles='--',colors='gray',linewidth
        ax.clabel(CS,lev[::2],fmt='%i')

    ax.plot(wippx,wippy,'k-')
    ax.plot(modx,mody,'-',color='purple',linewidth=2)
    ax.plot(wellx,welly,linestyle='none',marker='.',
            markeredgecolor='green',markerfacecolor='green')
    ax.plot(px,py,linestyle='solid',color='blue',linewidth=4)
    plt.arrow(x=px[-3],y=py[-3],dx=-10,dy=-50,
              linewidth=4,color='blue',head_length=500,head_width=500)
    plt.axis('image')
    ax.set_xlim([aserx.min(),aserx.max()])
    ax.set_ylim([asery.min(),asery.max()])

    ax.set_xticks(660000 + np.arange(5.0)*5000)
    ax.set_yticks(485000 + np.arange(5.0)*5000)
    labels = ax.get_yticklabels()
    for label in labels:
        label.set_rotation(90)
    for j,(x,y,n) in enumerate(zip(wellx,welly,names)):
        # only plot labels of wells inside the figure area
        if aserx.min()<x<aserx.max() and asery.min()<y<asery.max():
```

```
203         # name above
204         a.append(plt.annotate(n,xy=(x,y),xytext=(0,5),
205                     textcoords='offset points',
206                     horizontalalignment='center',
207                     fontsize=10))
208         # observed FW head below
209         a.append(plt.annotate('%.1f'%res[j,0],xy=(x,y),xytext=(0,-15),
210                     textcoords='offset points',
211                     horizontalalignment='center',
212                     fontsize=6))
213 ax.set_title('Freshwater Heads WIPP Area '+ year)
214 ax.set_xlabel('NAD27 NM East State Plane Easting (ft)')
215 ax.set_ylabel('NAD27 NM East State Plane Northing (ft)')
216
217 ax.annotate('WIPP LWB',xy=(665000,488200),fontsize=12)
218 ax.text(678700,495000,'MODFLOW-2000 No-Flow Area',size=16,rotation=-90,color='purple')
219
220 if manualFix:
221     # manually fix labels>>>>
222     for lab in a:
223         lab.draggable()
224     plt.show()
225 else:
226     if simpleContours:
227         fnout = 'aser-area-contour-map_with_linear_'+year+'.png'
228     else:
229         fnout = 'aser-area-contour-map'+year+'.png'
230     plt.savefig(fnout)
231 plt.close(1)
232
233
234 # plot contour map of measured-modeled residual
235 # ************************************
236 fig = plt.figure(1,figsize=(12,16))
237 ax = fig.add_subplot(111)
238 CS = ax.tricontour(wellx,welly,obsmod,linestyles='-') # colors='k',
239 ax.plot(wippx,wippy,'k-')
240 ax.plot(modx,mody,'-',color='purple',linewidth=2)
241 ax.plot(wellx,welly,linestyle='none',marker='.',
242         markeredgecolor='green',markerfacecolor='green')
243 plt.axis('image')
244 ax.set_xlim([aserx.min(),aserx.max()])
245 ax.set_ylim([asery.min(),asery.max()])
246 ax.clabel(CS,fmt='%i',inline_spacing=2)
247 ax.set_xticks(660000 + np.arange(5.0)*5000)
248 ax.set_yticks(485000 + np.arange(5.0)*5000)
249 labels = ax.get_yticklabels()
250 for label in labels:
251     label.set_rotation(90)
252
253 for j,(x,y,n) in enumerate(zip(wellx,welly,names)):
```

```
254     # only plot labels of wells inside the figure area
255     if aserx.min()<x<aserx.max() and asery.min()<y<asery.max():
256         if 'M' in weights[n]['ns']:
257             color = 'red'
258         else:
259             color = 'black'
260 
261         a.append(plt.annotate(n,xy=(x,y),xytext=(0,5),
262                               textcoords='offset points',
263                               horizontalalignment='center',
264                               fontsize=10,color=color))
265 
266         a.append(plt.annotate('%.1f'%obsmod[j],xy=(x,y),xytext=(0,-15),
267                               textcoords='offset points',
268                               horizontalalignment='center',
269                               fontsize=6,color=color))
270 ax.set_title('Observed-Modeled Freshwater Heads WIPP Area '+ year)
271 ax.set_xlabel('NAD27 NM East State Plane Easting (ft)')
272 ax.set_ylabel('NAD27 NM East State Plane Northing (ft)')
273 
274 ax.annotate('WIPP LWB',xy=(665000,488200),fontsize=12)
275 ax.text(678700,495000,'MODFLOW-2000 No-Flow Area',size=16,rotation=-90,color='purple')
276 
277 if manualFix:
278     # manually fix labels>>>>
279     for lab in a:
280         lab.draggable()
281     plt.show()
282 else:
283     plt.savefig('aser-area-modobs-contour-map'+year+'.png')
284 plt.close(1)
285 
286 # plot contour map of showing 2013 & 2015 contours
287 # ****************************************
288 fig = plt.figure(1,figsize=(12,16))
289 ax = fig.add_subplot(111)
290 lev = 2900 + np.arange(15)*20
291 
292 hZ = griddata((wellx,welly),obs,(hx,hy),method="cubic")
293 CS = ax.contour(hx,hy,hZ,linestyles='-',levels=lev,colors='red')
294 #CS = ax.tricontour(wellx,welly,obs,linestyles='-',colors='red') # colors='k',
295 ax.clabel(CS,fmt='%i') #,inline_spacing=2)
296 
297 hZ = griddata((wellx13,welly13),obs13,(hx,hy),method="cubic")
298 CS = ax.contour(hx,hy,hZ,linestyles='-',levels=lev,colors='green')
299 #CS = ax.tricontour(wellx13,welly13,obs13,linestyles='-',colors='green') # colors='k',
300 ax.clabel(CS,fmt='%i') #,inline_spacing=2)
301 
302 ax.plot(wippx,wippy,'k-')
303 ax.plot(modx,mody,'-',color='purple',linewidth=2)
304 ax.plot(wellx,welly,linestyle='none',marker='.',
```

```
305             markeredgecolor='green',markerfacecolor='green')
306 plt.axis('image')
307 ax.set_xlim([aserx.min(),aserx.max()])
308 ax.set_ylim([asery.min(),asery.max()])
309
310 ax.set_xticks(660000 + np.arange(5.0)*5000)
311 ax.set_yticks(485000 + np.arange(5.0)*5000)
312 labels = ax.get_yticklabels()
313 for label in labels:
314     label.set_rotation(90)
315 for j,(x,y,n) in enumerate(zip(wellx,welly,names)):
316     # only plot labels of wells inside the figure area
317     if aserx.min()<x<aserx.max() and asery.min()<y<asery.max():
318         a.append(plt.annotate(n,xy=(x,y),xytext=(0,5),
319                               textcoords='offset points',
320                               horizontalalignment='center',
321                               fontsize=10,color='black'))
322         a.append(plt.annotate('%.1f'%obs[j],xy=(x,y),xytext=(-12,-15),
323                               textcoords='offset points',
324                               horizontalalignment='center',
325                               fontsize=6,color='red'))
326
327 for j,(x,y,n) in enumerate(zip(wellx13,welly13,names13)):
328     # only plot labels of wells inside the figure area
329     if aserx.min()<x<aserx.max() and asery.min()<y<asery.max():
330         a.append(plt.annotate('%.1f'%obs13[j],xy=(x,y),xytext=(12,-15),
331                               textcoords='offset points',
332                               horizontalalignment='center',
333                               fontsize=6,color='green'))
334 ax.set_title('Freshwater Heads WIPP Area; 2015=red, 2013=green')
335 ax.set_xlabel('NAD27 NM East State Plane Easting (ft)')
336 ax.set_ylabel('NAD27 NM East State Plane Northing (ft)')
337
338 ax.annotate('WIPP LWB',xy=(665000,488200),fontsize=12)
339 ax.text(678700,495000,'MODFLOW-2000 No-Flow Area',size=16,rotation=-90,color='purple')
340
341 if manualFix:
342     # manually fix labels>>>>
343     for lab in a:
344         lab.draggable()
345     plt.show()
346 else:
347     plt.savefig('aser-area-2013-vs-2015-contour-map'+year+'.png')
348 plt.close(1)
```

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

```
import numpy as np

manualFix = True

if not manualFix:
    import matplotlib
    matplotlib.use('Agg')

import matplotlib.pyplot as plt

fprefix = 'pest_02/'
mprefix = '../../../well-location-maps/wipp-polyline-data/'
fname = fprefix + 'modeled_vs_observed_head_pest_02.txt'

ofname = 'original_average/modeled_vs_observed_head_original_average.txt'

M2FT = 0.3048
year = '2015'

# load in observed, modeled, obs-mod, (all in meters)
res = np.loadtxt(fname,skiprows=1,usecols=(3,4,5))
ores = np.loadtxt(ofname,skiprows=1,usecols=(3,4,5))

# load in weights
weights = np.loadtxt(fname,skiprows=1,usecols=(6,),dtype='int')
# load in names
names = np.loadtxt(fname,skiprows=1,usecols=(0,),dtype='|S6')

# load in N/S/C/X zones
zones = np.loadtxt('obs_loc_%sASER.dat' % year,usecols=(2,),dtype='|S2')

## checking locations / zones
# ******************************************
wipp = np.loadtxt(mprefix+'wipp_boundary.dat')
x,y = np.loadtxt(fname,skiprows=1,usecols=(1,2),unpack=True)

fig = plt.figure(2,figsize=(18,12))
ax1 = fig.add_subplot(121)
ax1.plot(x,y,'k*') # wells
ax1.plot(wipp[:,0],wipp[:,1],'r-') # WIPP LWB
buff = np.loadtxt(mprefix+'wipp_boundary.dat')
buff[1:3,0] -= 3000.0
buff[0,0] += 3000.0
buff[3:,0] += 3000.0
buff[2:4,1] -= 3000.0
buff[0:2,1] += 3000.0
buff[-1,1] += 3000.0
colors = {'N':'red','S':'blue','C':'green','X':'gray','SM':'magenta'}
ax1.plot(buff[:,0],buff[:,1],'g--') # WIPP LWB+3km
```

```
for xv,yv,n,w,z in zip(x,y,names,weights,zones):
    print xv,yv,n,w,z
    plt.annotate('%s %i'%(n,w),xy=(xv,yv),fontsize=8,color=colors[z])
plt.axis('image')
ax1.set_xlim([x.min()-1000,x.max()+1000])
ax1.set_ylim([y.min()-1000,y.max()+1000])
ax2 = fig.add_subplot(122)
ax2.plot(x,y,'k*') # wells
ax2.plot(wipp[:,0],wipp[:,1],'r-') # WIPP LWB
ax2.plot(buff[:,0],buff[:,1],'g--') # WIPP LWB+3km
for xv,yv,n,w,z in zip(x,y,names,weights,zones):
    plt.annotate('%s %i '%(n,w),xy=(xv,yv),fontsize=8,color=colors[z])
plt.axis('image')
ax2.set_xlim([wipp[:,0].min()-100,wipp[:,0].max()+100])
ax2.set_ylim([wipp[:,1].min()-100,wipp[:,1].max()+100])
plt.suptitle('well weights check '+year)
plt.savefig('check-well-weights-'+year+'.png')

# convert lengths to feet
res /= M2FT
ores /= M2FT

# create the histogram of residuals for ASER
# ****************************************

# -10,-9,...8,9,10
bins = 3*np.arange(-10,10)
rectfig = (15,7)
squarefig = (8.5,8.5)

fig = plt.figure(1,figsize=rectfig)
ax = fig.add_subplot(111)
# all the data, all but distant wells
ax.hist([res[weights<2,2],res[:,2]],bins=bins,range=(-30,30.0),
        rwidth=0.75,align='mid',
        color=['red','blue'],
        label=['Inside LWB & <3km from WIPP LWB','All wells'])
ax.set_xlabel('Measured-Modeled (ft)')
ax.set_ylabel('Frequency')
ax.set_xticks(bins)
ax.set_ylim([0,10])
ax.set_yticks(np.arange(0,10,2))
plt.grid()
ax.yaxis.grid(True,which='major')
ax.xaxis.grid(False)
plt.legend(loc='upper left')
plt.title('Histogram of Model Residuals '+year)
#plt.annotate('AEC-7 @ %.1f'%res[0,2],xy=(-9.75,5.0),xytext=(-8.5,5.0),
#             arrowprops={'arrowstyle':'->'},fontsize=16)
plt.savefig('model-error-histogram-'+year+'.png')
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')
```

```
ax.axhline(y=-15,color='black',linestyle='dotted')
plt.grid()
ax.yaxis.grid(True,which='major')
ax.xaxis.grid(False)
ax.set_xlim([0,res.shape[0]])

plt.annotate('',xy=(0.0,15.0),xycoords='data',
             xytext=(nin,15.0),textcoords='data',
             arrowprops={'arrowstyle':'<->'})
plt.annotate('inside WIPP LWB',xy=(nin/3.0,15.5),xycoords='data')

plt.annotate('',xy=(nin,15.0),xycoords='data',
             xytext=(nin+nnear,15.0),textcoords='data',
             arrowprops={'arrowstyle':'<->'})
plt.annotate('<3km WIPP LWB',xy=(nin+nnear/3.0,15.5),xycoords='data')

plt.annotate('',xy=(nin+nnear,15.0),xycoords='data',
             xytext=(nin+nnear+nfar,15.0),textcoords='data',
             arrowprops={'arrowstyle':'<->'})
plt.annotate('>3km WIPP LWB',xy=(nin+nnear+nfar/3.0,15.5),xycoords='data')

ax.set_ylabel('Measured-Modeled (ft)')
ax.set_title('individual residuals '+year)
#plt.annotate('AEC-7 @ %.1f'%res[0,2],xy=(nin+nnear+1.0,-14.5),xycoords='data')

plt.savefig('model-error-residuals-'+year+'.png')
plt.close(1)


# create scatter plot of measured vs. modeled
# ***************************************
m = 1.0/M2FT
sr = [2940,3100]

fh = open('calibration-statistics-%s.csv' % year,'w')

fh.write('wellgroup,calibrated,uncalibrated\n')
fh.write('"all wells",%.4f,'  % np.corrcoef(res[:,0],res[:,1])[1,0]**2)
fh.write('%.4f\n' % np.corrcoef(ores[:,0],ores[:,1])[1,0]**2)

fh.write('"wells inside 3km of LWB",%.4f,' % np.corrcoef(res[weights<2,0], res[weights<2,1
fh.write('%.4f\n' % np.corrcoef(ores[weights<2,0], ores[weights<2,1])[1,0]**2)

fh.write('"wells ~inside LWB",%.4f,' % np.corrcoef(res[weights==0,0],res[weights==0,1])[1,(
fh.write('%.4f\n' % np.corrcoef(ores[weights==0,0],ores[weights==0,1])[1,0]**2)

fh.close()

fig = plt.figure(1,figsize=squarefig)
ax = fig.add_subplot(111)
ax.plot(res[m0,0],res[m0,1],color='red',markersize=10,
```

```
        marker='+',linestyle='none',label='Inside LWB')
ax.plot(res[m1,0],res[m1,1],color='green',markersize=10,
        marker='x',linestyle='none',label='< 3km From LWB')
ax.plot(res[m2,0],res[m2,1],color='blue',markersize=10,
        marker='*',linestyle='none',label='distant')
ax.plot(sr,sr,'k-',label='$45^{\\degree}$ Perfect Fit')
ax.plot([sr[0],sr[1]],[sr[0]+m,sr[1]+m],'g-',linewidth=0.5,label='$\\pm$ 1m Misfit')
ax.plot([sr[0],sr[1]],[sr[0]-m,sr[1]-m],'g-',linewidth=0.5,label='__nolegend__')
ax.set_xticks(np.linspace(sr[0],sr[1],9))
ax.set_yticks(np.linspace(sr[0],sr[1],9))
ax.set_xlim(sr)
ax.set_ylim(sr)
plt.minorticks_on()
plt.legend(loc='lower right',scatterpoints=1,numpoints=1)
plt.grid()
a = []
for j,lab in enumerate(names):
    if res[j,2] < -1.5*m:
        # plot labels to left of value far above 45-degree line
        a.append(plt.annotate(lab,xy=(res[j,0],res[j,1]),
                              xytext=(res[j,0]-(2.9*len(lab)),
                                      res[j,1]-2.0),fontsize=14))
    elif res[j,2] > 1.5*m:
        # plot labels to right of value far below 45-degree line
        a.append(plt.annotate(lab,xy=(res[j,0],res[j,1]),
                              xytext=(res[j,0]+2.0,
                                      res[j,1]-2.0),fontsize=14))
ax.set_xlabel('Observed Freshwater Head (ft AMSL)')
ax.set_ylabel('Modeled Freshwater Head (ft AMSL)')
ax.set_title('modeled vs. measured '+year)

if manualFix:
    # manually fix labels>>>>
    for lab in a:
        lab.draggable()
    plt.show()
else:
    plt.savefig('scatter_pest_02_'+year+'.png')
plt.close(1)
```