557494

Analysis Report for Preparation of 2005-2007 Culebra Potentiometric Surface Contour Maps
Revision 1

Task Number: 1.4.2.3

Report Date: 4/26/2012

Author:	[illegible] Kristopher L. Kuhlman, 6212 Repository Performance Department	4/26/2012 Date
Technical Review:	[illegible] Kevin S. Barnhart, 6212 Repository Performance Department	4/26/2012 Date
QA Review:	Shelly R Nielsen Shelly R. Nielsen, 6210 Carlsbad Programs Group	4-30-12 Date
Management Review:	[illegible] Christi D. Leigh, 6212 Manager, Repository Performance Department	5/1/12 Date

WIPP:1.4.2.3:TD:QA-L:RECERT:549085

Table of Contents

1 Introduction

This report documents the preparation of three historic potentiometric contour maps and associated particle tracks for the Culebra Member of the Rustler Formation in the vicinity of the Waste Isolation Pilot Plant (WIPP), for submittal to the New Mexico Environment Department (NMED). The driver for this analysis is the draft of the Stipulated Final Order sent to NMED on May 28, 2009 (Moody, 2009). This Analysis Report follows the procedure laid out in procedure SP 9-9 (Kuhlman, 2009), which is based upon this NMED driver. This report is a similar to Kuhlman (2011), the same analysis is performed on data from 2005, 2006 and 2007, rather than 2010 data.

Historic data were taken from the Annual Site Environmental Reports (ASERs) to plot freshwater head and density[1] at Culebra wells through time; see (DOE, 2005) through (DOE, 2011) in references. This additional step of plotting time series at each well was done to pick an appropriate month with relatively undisturbed conditions and to assign consistent density data for 2005-2007. This revision includes this additional step because the current procedures related to Culebra densities and contour map creation (SP 9-9 and SP 9-11) were not in place when the historic potentiometric surface contour maps were created.

Beginning with the ensemble of 100 calibrated MODFLOW transmissivity (T), horizontal anisotropy (A), and areal recharge (R) fields (Hart et al., 2009) used in WIPP performance assessment (PA), three average parameter fields are used as input to MODFLOW to simulate freshwater heads within and around the WIPP land withdrawal boundary (LWB). For each year (2005-2007) PEST is used to adjust a subset of the boundary conditions in the averaged MODFLOW model to obtain the best-fit match between the observed freshwater heads and the model-predicted heads. The output of the averaged, PEST-calibrated MODFLOW model is both contoured and used to compute each year's advective particle track forward from the WIPP waste-handling shaft.

This revision (1) fixes an error in the 2007 contour map generation. WQSP-1 was incorrectly included as a well in the group >3 km from the WIPP LWB (zone 2), when it is actually inside the WIPP LWB. WQSP-1 incorrectly had a small weight (0.4), when it should have had a large weight (2.5). After fixing this, the calculations were re-run and the figures for 2007 were all regenerated. The tables of results and predictions of particle path lengths were revised. In general the resulting changes were very small compared to the initial report (revision 0, February 20, 2012). A second minor fix was a correction of the statements regarding boundary conditions that were impacted most by which x- or y-direction related variables. The results were correct, but the text discussion of the results was incorrect.

[1] Density in units of grams/cm^3 is numerically equivalent to specific gravity, the ratio of the density of any water to that of fresh water.

2 Scientific Approach

2.1 2005-2007 Freshwater Head and Density Data Review

In previous analysis reports based on the SP 9-9 procedure, recent data from the current ASER were used; plotting of the data was not done independently. Data reported in historic ASERs are being plotted to ensure consistency and explain anomalies. Python scripts and resulting data plots at each well are listed in Section 8.3. Table 1 summarizes freshwater heads, measurement dates, and Culebra groundwater densities for all three years.

Water level and freshwater head values (and measurement dates) were obtained from Table F-8 in the 2004-2010 ASERs. Estimates of Culebra densities from historic Troll data were done for 2005 (Johnson, 2012a) and 2006 (Johnson, 2012b). Culebra midpoint elevations were obtained from Johnson (2008). Historic events (pumping tests, purging events, drilling and plugging & abandonment) were tabulated from ASERs. All of these data were plotted together through time as part of the data review to determine two things. First, were there significant events in a well that warrant assigning a sudden change in fluid density? For example, H-10c was baled to remove fresh water in 2009 (see H-10c plot in Section 8.3). If there was a noticeable change in observed depth-to-water that occurred at the same time as a documented event, a different density is assigned to the periods before and after this event. Secondly, one or more representative densities were chosen for the well, and a start and end time was assigned to each density. If only one density is assigned, then the beginning time is just before the beginning of this analysis (1999), and the end is the current date. There is some variability in the measurements of density in wells. Especially with older data, the variability can become large, due to inaccuracies in recorded Troll installation depths. In January 2007 (beginning with the 2007 ASER) more accurate reference point elevations were used to compute water level and freshwater head elevations. For consistency across years, pre-2007 water level elevations were adjusted to use the newer reference point elevation.

The data review revealed June to be the best month in 2005 to contour (the 2005 ASER used December water levels), because water levels were impacted in many wells due to the large-scale SNL-14 pumping test in the summer of 2005. Some wells took several months to recover from the drawdown caused by this pumping test. June data are more representative of non-testing conditions. WIPP-30 used an August water level, because there was no water level reported in June 2005 at that well.

Table 1. Fresh Water Head (FWH) elevation AMSL and specific gravities used to compute FWH from depth to water observations. Depth to water in each well was measured on the FWH date.

	2005			2006			2007		
	FWH date	FWH	Specific gravity	FWH date	FWH	Specific gravity	FWH date	FWH	Specific gravity
C-2737	6/21/05	920.04	1.019	11/9/06	920.48	1.019	5/9/07	920.71	1.010
ERDA-9	6/20/05	924.18	1.067	11/9/06	924.78	1.067	5/9/07	924.68	1.067
H-02b2	6/20/05	927.24	1.000	11/9/06	928.24	1.000	5/9/07	928.34	1.000
H-03b2	6/21/05	918.53	1.042	11/9/06	918.31	1.042	5/9/07	918.65	1.042
H-04b	6/20/05	916.53	1.015	11/8/06	916.42	1.015	5/9/07	916.35	1.015
H-05b	6/16/05	938.30	1.095	11/6/06	938.96	1.095	5/10/07	939.15	1.095
H-06b	6/13/05	935.43	1.040	11/6/06	936.75	1.040	5/7/07	936.45	1.040
H-07b1	6/13/05	914.63	1.002	11/8/06	914.60	1.002	5/7/07	914.58	1.002
H-09c	6/20/05	913.53	1.001	11/8/06	912.58	1.001	5/8/07	912.78	1.001
H-10c	6/20/05	921.94	1.001	8/14/06	921.93	1.001	5/8/07	922.07	1.001
H-11b4	6/20/05	917.13	1.070	11/9/06	917.07	1.070	5/7/07	917.05	1.070
H-12	6/20/05	916.28	1.097	11/9/06	916.62	1.097	5/8/07	916.54	1.097
H-15	6/20/05	920.82	1.082				5/9/07	920.08	1.053
H-17	6/20/05	916.29	1.133	11/9/06	916.29	1.133	5/7/07	916.29	1.133
H-19b0	6/20/05	918.79	1.068	11/8/06	918.80	1.068	5/9/07	918.83	1.068
I-461	6/13/05	928.57	1.005	11/6/06	929.34	1.005	5/7/07	928.94	1.005
P-17	6/20/05	915.44	1.053						
SNL-01	6/16/05	939.24	1.033	11/6/06	941.47	1.033	5/8/07	941.85	1.033
SNL-02	6/13/05	937.02	1.012	11/6/06	938.35	1.012	5/7/07	937.66	1.012
SNL-03	6/16/05	937.85	1.023	11/6/06	939.47	1.023	5/8/07	939.77	1.023
SNL-05	6/13/05	937.01	1.010	11/6/06	938.61	1.010	5/7/07	938.59	1.010
SNL-08				11/6/06	930.52	1.052	5/7/07	930.01	1.052
SNL-09	6/13/05	931.48	1.024	11/6/06	932.50	1.024	5/7/07	932.03	1.024
SNL-10							5/7/07	931.57	1.011
SNL-12	6/20/05	915.52	1.005	11/6/06	915.22	1.005	5/7/07	915.24	1.005
SNL-13	6/21/05	917.55	1.027	11/6/06	918.00	1.027	5/7/07	918.20	1.027
SNL-14							11/14/07	916.37	1.048
SNL-16				11/8/06	918.43	1.010	9/17/07	918.17	1.010
SNL-17				11/6/06	916.75	1.006	5/7/07	916.78	1.006
SNL-18				11/6/06	939.86	1.028	5/8/07	939.90	1.028
SNL-19				11/6/06	937.92	1.003	5/7/07	937.58	1.003
WIPP-11	6/13/05	938.87	1.038	8/14/06	939.87	1.038	5/9/07	940.65	1.038
WIPP-13	6/13/05	938.33	1.053	11/8/06	939.86	1.053	5/9/07	939.84	1.053
WIPP-19	6/20/05	932.01	1.044	11/8/06	933.51	1.044	5/9/07	933.70	1.044
WIPP-25	6/13/05	935.73	1.011						
WIPP-30	8/17/05	938.35	1.000	11/6/06	939.29	1.000	5/8/07	939.06	1.000
WQSP-1	6/20/05	936.94	1.048	11/8/06	938.58	1.048	5/9/07	938.61	1.048
WQSP-2	6/20/05	939.45	1.048	11/8/06	941.14	1.048	5/9/07	941.20	1.048
WQSP-3	6/20/05	935.46	1.146	11/8/06	936.98	1.146	5/9/07	936.81	1.146
WQSP-4	6/20/05	918.90	1.075	11/8/06	918.97	1.075	5/9/07	918.96	1.075
WQSP-5	6/20/05	918.04	1.025	11/8/06	918.12	1.025	5/9/07	918.18	1.025
WQSP-6	6/20/05	921.54	1.014	11/8/06	921.95	1.014	5/9/07	921.88	1.014

The data review revealed November to be a good month in 2006 to contour (which was also used by the 2006 ASER). WIPP-11 used an August 2006 water level because anomalously high water levels were reported October-December, 2006. Similarly, H-10c used an August 2006 water level because of high water levels reported later in the year.

The data review revealed May 2007 to be the best month in 2007 to contour (the 2007 ASER used December). This month also coincides with the month used to pick data for the calibration of the Performance Assessment (PA) Culebra groundwater model. SNL-14 used a November 2007 water level because now water levels were measured January-October 2007 due to pumping and sampling activities in the well. SNL-16 used a September 2007 water level, because there was no May 2007 water recorded and previous to September, the well had anomalously high water levels.

2.2 Modeling Overview

Steady-state groundwater flow simulations are carried out using similar software as was used in the analysis report for AP-114 Task 7 (Hart et al., 2009), which was used to create the input calibrated fields. See Table 2 for a summary of all software used in this analysis. The MODFLOW parameter fields (transmissivity (T), anisotropy (A), and recharge (R)) used in this analysis are ensemble averages of the 100 sets of Culebra parameter fields used for WIPP PA for the 2009 Compliance Recertification Application (CRA-2009) PA baseline calculations (PABC). To clearly distinguish between the two MODFLOW models, the original MODFLOW model, which consists of 100 realizations of calibrated parameter fields (Hart et al., 2009), will be referred to as the "PA MODFLOW model." The model we derive from the PA MODFLOW model, calibrate using PEST, and use to construct the resulting contour map and particle track, is referred to as the "averaged MODFLOW model." The PA MODFLOW model T, A and R input fields are appropriately averaged across 100 realizations, producing a single averaged MODFLOW flow model. This averaged MODFLOW model is used to predict regional Culebra groundwater flow across the WIPP site.

For CRA 2009 PABC, PEST was used to construct 100 calibrated model realizations of the PA MODFLOW model by adjusting the spatial distribution of model parameters (T, A, and R); MODFLOW boundary conditions were fixed. The calibration targets for PEST in the PA MODFLOW model were both May 2007 freshwater heads and transient drawdown to large-scale pumping tests. Hart et al. (2009) describe the calibration effort and results that went into the CRA-2009 PABC. An analogous but much simpler process is used here for the averaged MODFLOW model. We use PEST to modify a subset of the MODFLOW boundary conditions (see red boundaries in Figure 1). The boundary conditions are modified, rather than the T, A, and R parameter fields for simplicity, because re-calibrating the 100 T, A, and R parameter fields would be a significant effort (thousands of hours of computer time). The PEST calibration targets for the averaged MODFLOW model are the 2005-2007 measured annual freshwater heads at Culebra monitoring wells. In the averaged MODFLOW model, boundary conditions are modified while holding model parameters T, A, and R constant. In contrast to this, the PA MODFLOW model used fixed boundary conditions and made adjustments to T, A, and R parameter fields.

Table 2. Software used

Software	Version	Description	Platform	Software QA status
MODFLOW-2000	1.6	Flow model	PA cluster	Acquired; qualified under NP 19-1 (Harbaugh et al., 2000)
PEST	9.11	Inverse model	PA cluster	Developed; qualified under NP 19-1 (Doherty, 2002)
DTRKMF	1.00	Particle tracker	PA cluster	Developed; qualified under NP 19-1
Python	2.3.4	Scripting language (file manipulation)	PA cluster	Commercial off the shelf
Enthought Python	7.2-2	Scripting language (plotting)	Mac desktop	Commercial off the shelf
Bash	3.00.15	Scripting language (file manipulation)	PA cluster	Commercial off the shelf

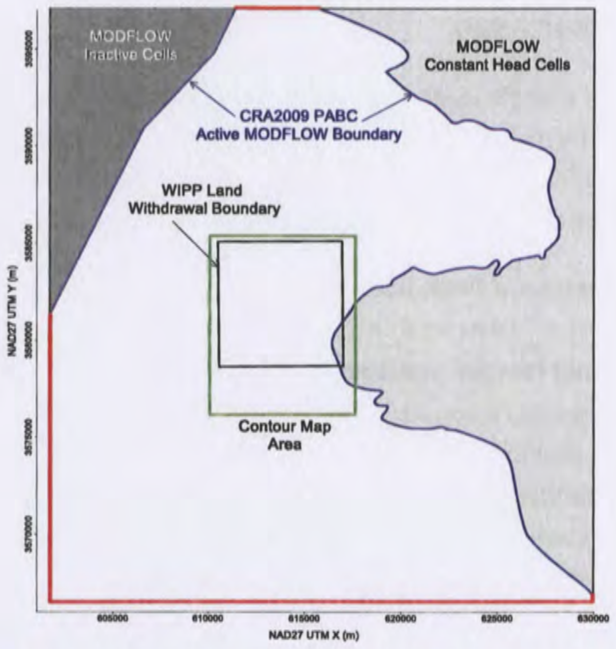


Figure 1. MODFLOW-2000 model domain, adjusted boundary conditions shown in red, contour area outlined in green.

The resulting heads from the PEST-calibrated averaged MODFLOW model are contoured over an area surrounding the WIPP site using matplotlib (a Python plotting library included in the Enthought Python Distribution (EPD)). The figure covers a subset of the complete MODFLOW model domain – see the green rectangle surrounding the WIPP LWB in Figure 1. We compute the path taken by a conservative (i.e., non-dispersive and non-reactive) particle to the WIPP LWB, initially released to the Culebra at the waste-handling shaft. The particle track is computed from the MODFLOW flow field using DTRKMF, these results are also plotted using matplotlib. Scatter plot statistics were computed using NumPy (an array-functionality Python library included in EPD), which summarize the quality of the fit between the averaged MODFLOW model and observed Culebra freshwater heads. MODFLOW, PEST, DTRKMF, and the Bash and Python scripts written for this work were executed on the PA Linux cluster (`alice.sandia.gov`), while the plotting and creation of figures was done using Python scripts on an Intel-Xeon-equipped desktop computer running Mac OS X, version 10.6.8.

2.3 Creating Average MODFLOW Simulation

An averaged MODFLOW model is used to compute the freshwater head and cell-by-cell flow vectors. The heads are contoured and the flow vectors are 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 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 groundwater flow equation). The alternative approach would be to averaging outputs from 100 models to produce a single average result, but the 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 100) that results in smoother freshwater head contours and faster particle tracks, compared to those predicted by the ensemble of fields in AP114 Task 7 (Hart et al., 2009).

The MODFLOW model grid is a single layer, comprised of 307 rows and 284 columns, each model cell being 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.

The calibrated T, A, and R parameter fields from the PA MODFLOW model were checked out of the PA repository using the `checkout_average_run_modflow.sh` script (scripts are listed completely in the Appendix; input and output files are available from the WIPP version control system in the repository `$CVSLIB/Analyses/SP9_9`). Model inputs can be divided into two groups. The first group includes model inputs that are the same 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 realization. The constant model inputs in the first group are used directly in the averaged MODFLOW model (checked out from the CVS repository), 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 arithmetically averaged in $\log_{10}$ space, since they vary over multiple orders of magnitude.

2.4 Boundary Conditions

The boundary conditions taken from the PA MODFLOW model are used as the initial condition from which PEST calibration proceeds. There are two types of boundary conditions in both MODFLOW models. 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 the 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 linear southern, southwestern, and northern boundaries that coincide with the rectangular frame surrounding the model domain (shown as heavy red lines in Figure 1). The value of specified head used along this second boundary type is adjusted in the PEST calibration process.

The Python script `boundary_types.py` is used to distinguish between the two different types of specified head boundary conditions based on the specified head value used in the PA MODFLOW model. All constant-head cells (specified by a value of -1 in the MODFLOW IBOUND array from the PA MODFLOW model) that have 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 the land surface. The remaining constant-head cells are distinguished by setting their IBOUND array value to -2 (which is still interpreted as a constant-head value by MODFLOW, but allows simpler discrimination between boundary conditions in scripts elsewhere).

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

2.5 PEST Calibration of Averaged MODFLOW Model to Observations

There are three major types of inputs to PEST. The first input type includes the observed freshwater head values, which are used as targets for the PEST calibration. The second input class includes the entire MODFLOW model setup derived from the PA MODFLOW model and described in the previous section, along with any pre- or post-processing scripts or programs needed. These files comprise the forward model that PEST runs repeatedly to estimate sensitivities of model outputs to model inputs. The third input type includes the PEST configuration files, which list parameter and observation groups, observation weights, and indicate which parameters in the MODFLOW model will be adjusted in the inverse simulation. Freshwater head values used as targets for the PEST calibration were taken from published ASERs (2005-2007) and are summarized in Table 1.

To minimize the number of estimable parameters, and to ensure a degree of smoothness in the 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:

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

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=928.0, B=8.0, C=1.2, D=1.0, α=0.5, E=1.0, and F=-1.0 are used with the above equation to assign the boundary conditions.

PEST was then used to estimate the values of parameters *A, B, C, D, E, F*, and α given the observed heads in Table 1. The Python script `surface_02_extrapolate.py` was used to compute the MODFLOW starting head input file (which is also used to specify the constant-head values) from the parameters *A-F* and α. Each forward run of the model corresponded to a call to the Bash script `run_02_model`. This script called the `surface_02_extrapolate.py` script, the MODFLOW-2000 executable, and the PEST utility `mod2obs.exe`, which is used to extract and interpolate model-predicted heads from the MODFLOW output files at observation well locations.

The PEST-specific input files (the third type of input) 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 model output), the template files (how to write the model input), and the PEST control file (listing the ranges and initial values for the estimable parameters and the weights associated with observations). The wells used in each year's PEST calibration were separated into three groups. Higher weights (2.5) were assigned to wells inside the LWB, and lower weights (0.4) were assigned to wells distant to the WIPP site, while wells in the middle were assigned an intermediate weight (1.0). Additional observations representing the average heads north of the LWB and south of the LWB were used to help prevent over-smoothing of the estimated results across the LWB. The additional observations and weights were assigned to improve the fit in the area of interest (inside the WIPP LWB), possibly at the expense of a somewhat poorer fit closer to the boundary conditions.

2.6 Figures Generated from Averaged MODFLOW Model

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

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

3 2005 Results

3.1 2005 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 in the eastern part of the figures corresponds to the portion of the Culebra that is located stratigraphically between halite in other members of the Rustler Formation (Tamarisk Member above and Los Medaños Member below). This region east of the "halite margin" has a high freshwater head but extremely low transmissivity, essentially serving as a no-flow boundary in this area.

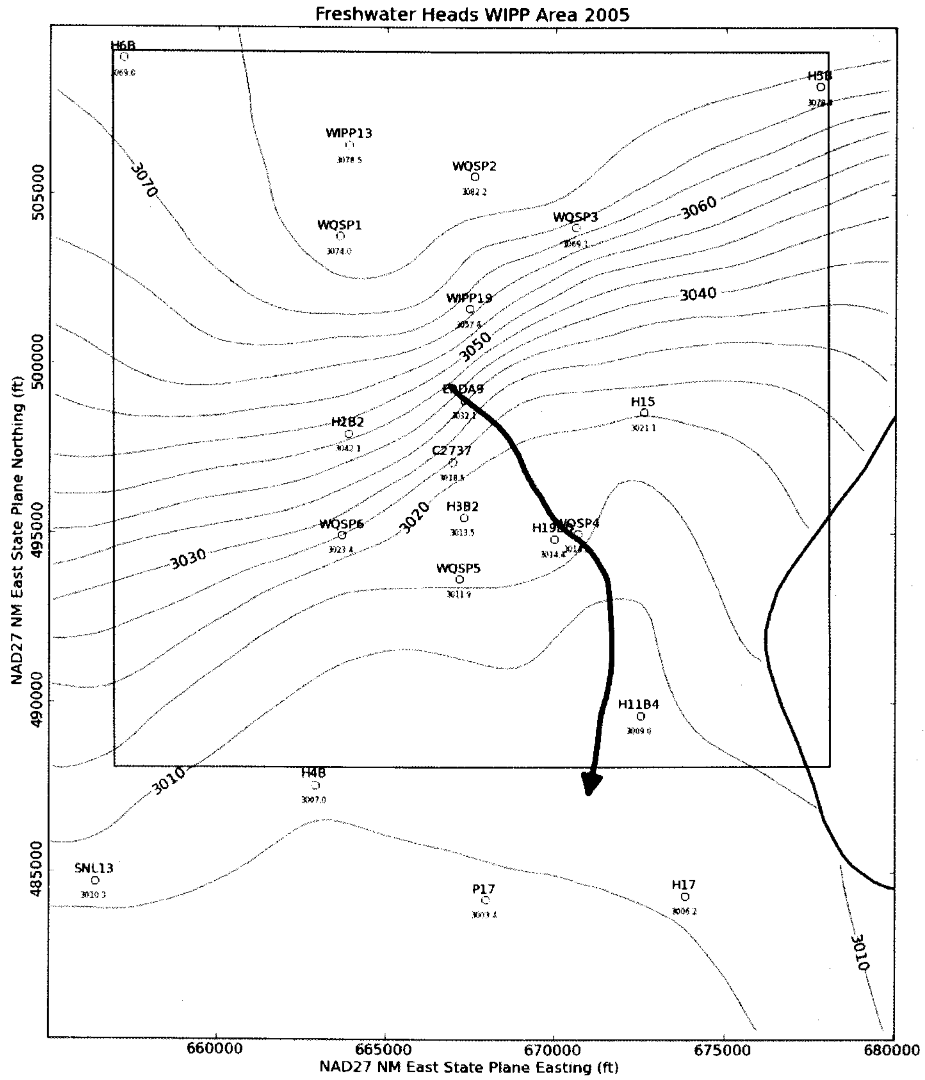


Figure 2. Model-generated June 2005 freshwater head contours with observed head listed at each well (5-foot contour interval) with blue water particle track from waste handling shaft to WIPP LWB

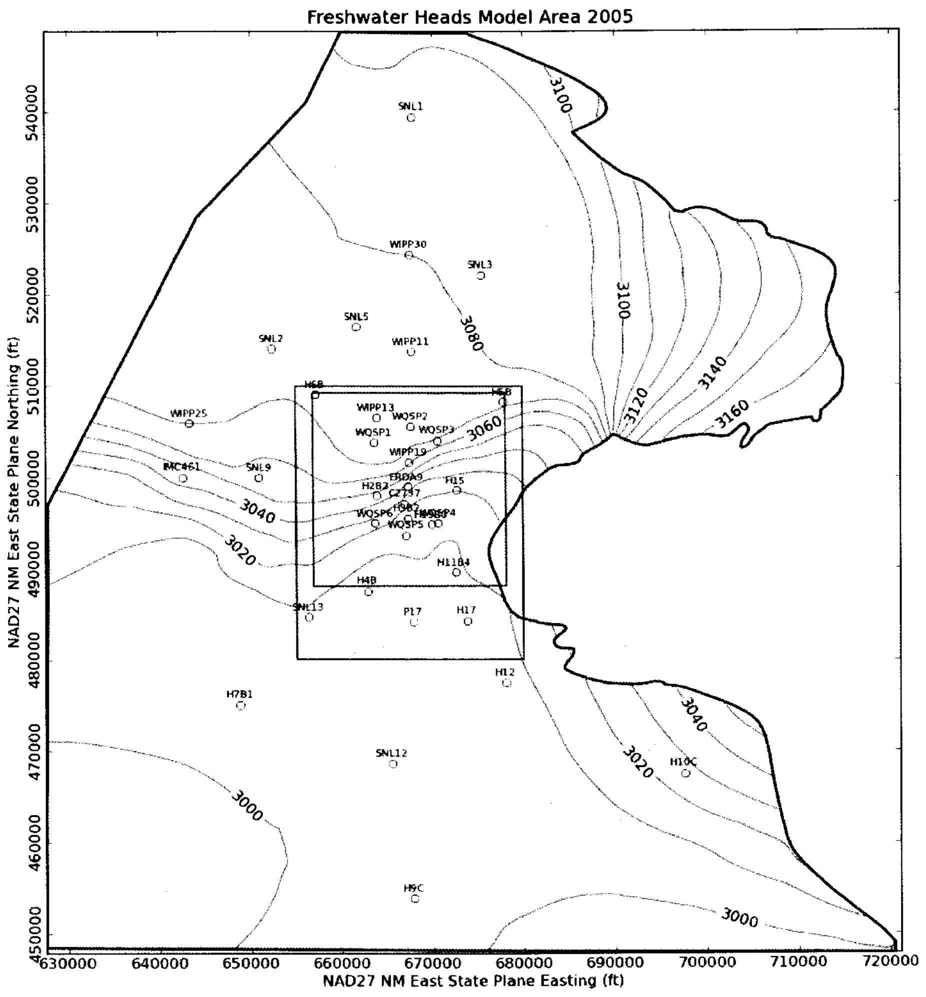


Figure 3. MODFLOW-modeled June 2005 heads for entire model domain (10-foot contour interval). Green rectangle indicates region contoured in Figure 2, black square is WIPP LWB.

3.2 2005 Particle Track

The blue arrow in Figure 2 shows the DTRKMF-calculated path a water particle would take through the Culebra from the coordinates corresponding to the WIPP waste handling shaft to the LWB (a path length of 4083 m). Assuming a 4-m thickness for the transmissive portion of the Culebra and a constant porosity of 16%, the travel time to the WIPP LWB is 6170 years (output from DTRKMF is adjusted from an original 7.75-m Culebra thickness). This is an average velocity of 0.66 m/yr.

3.3 2005 Measured vs. Modeled Fit

The scatter plot in Figure 4 shows measured and modeled freshwater heads at the observation locations used in the PEST calibration. The observations are divided into three groups, based on proximity to the WIPP site. Wells within the LWB are represented by red crosses, wells outside but within 3 km of the LWB are represented with green 'x's, and other wells within the MODFLOW model domain but distant from the WIPP site are given by a blue star. IMC-461 was given a high weight (2.5), treating it as if it was inside the WIPP LWB, to compensate the lack of SNL-16 in the 2005 network. The area at the north end of the constant head boundary, and the southern end of the no-flow boundary is strongly influenced by the assigned boundary conditions – in 2006 and later SNL-16 is located in this area.

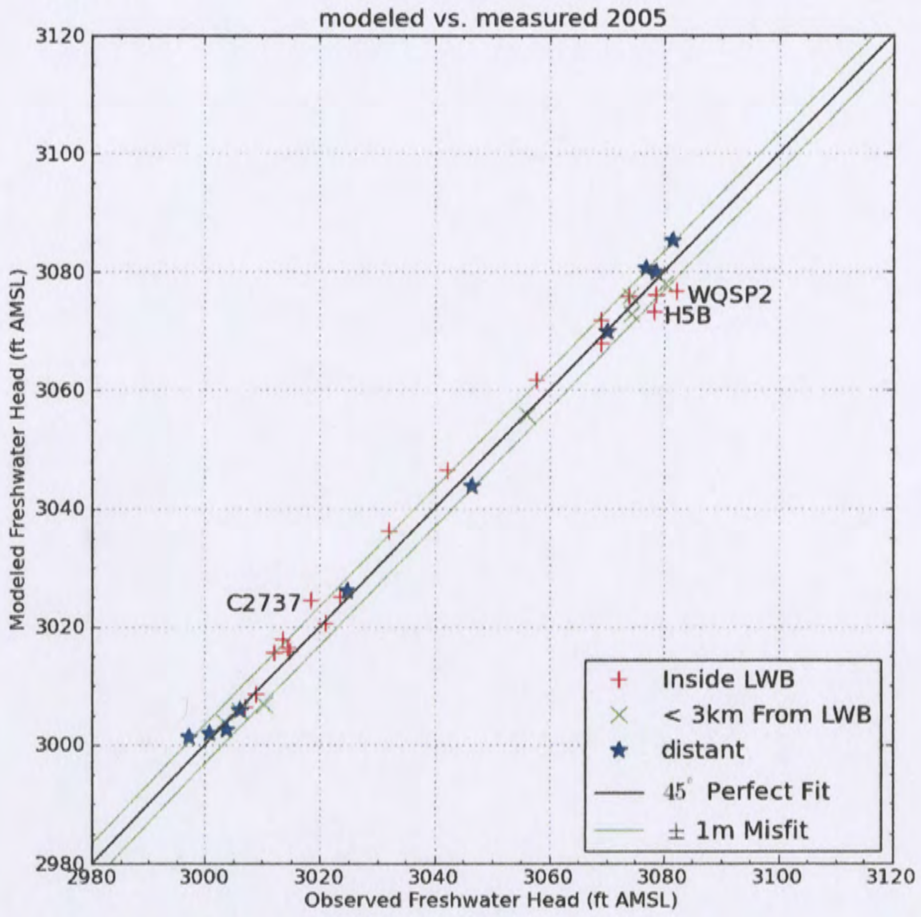


Figure 4. Measured vs. modeled scatter plot for averaged MODFLOW model generated heads and June 2005 observed freshwater heads

The central black diagonal line in Figure 4 represents a perfect model fit (1:1 or 45-degree slope); the two green lines on either side of this represent a 1-m misfit above or below the perfect fit. Wells more than 1.5 m from the 1:1 line are labeled. The calibrated parameters (for equation 1) were A=928.7, B=7.69, C=2.26, D=0.853, E=1.80, F=-0.894, and α=0.074. The parameter C, the weight associated with the overall x-direction term, had the largest relative change (88%) compared to the starting values. The parameter E, the weight associated with the cubic variation in the x-direction, also had a large relative change (-85%). The boundary conditions at the west edge of the model domain are impacted most by these parameters.

The squared correlation coefficient (R^2) for the measured vs. modeled data is listed in Table 3. Figure 5 and Figure 6 show the distribution of errors resulting from the PEST-adjusted model fit to observed data. The wells within and near the WIPP LWB have an R^2 of approximately 99%, and the calibration improved the R^2 value very slightly (third decimal place) inside the WIPP LWB. The distribution in Figure 5 does not have a strong bias.

Table 3. 2005 Measured vs. Modeled correlation coefficients

	dataset	measured vs. modeled R^2
Uncalibrated	wells inside WIPP LWB	0.988
	wells <3km from WIPP LWB	0.990
	all wells	0.982
Calibrated	wells inside WIPP LWB	0.989
	wells <3km from WIPP LWB	0.990
	all wells	0.982

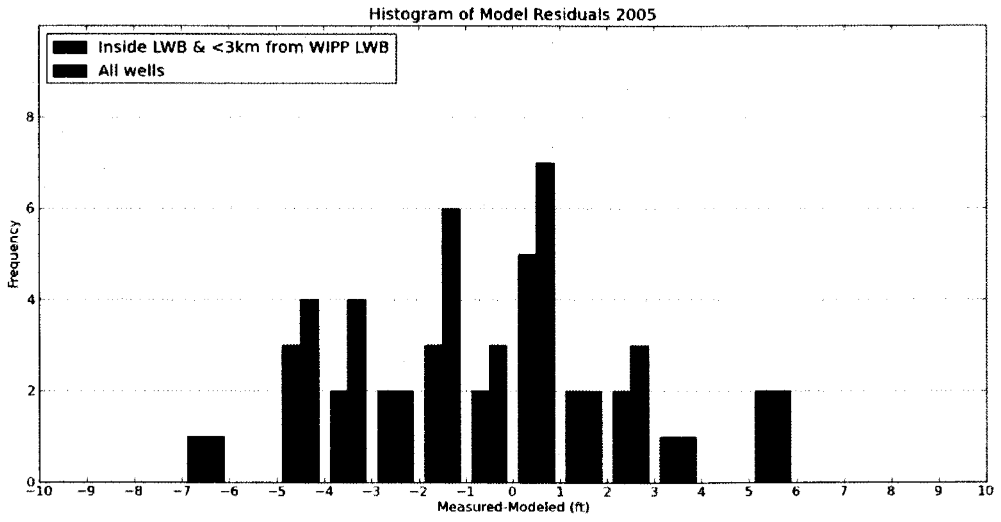


Figure 5. Histogram of Measured-Modeled errors for 2005

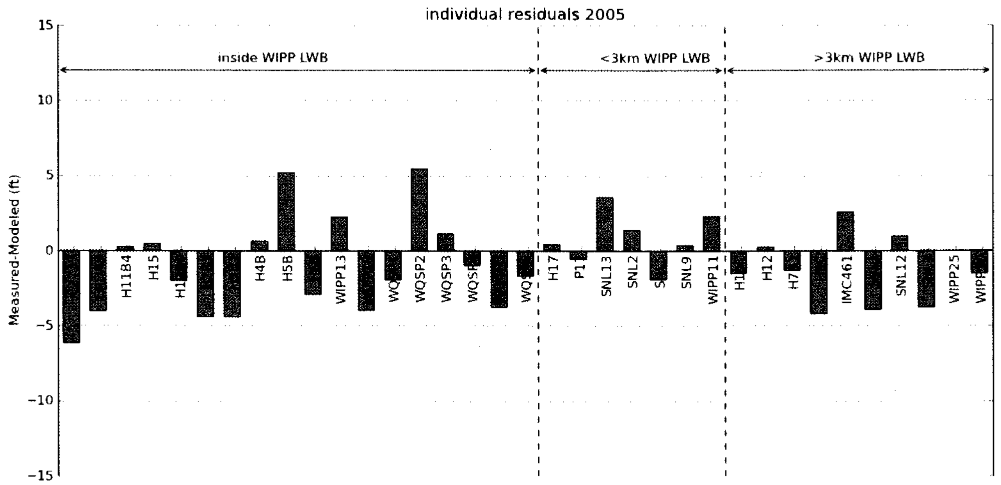


Figure 6. Measured-Modeled errors at each well location for 2005

The model fit to the June 2005 observations is very good. The averaged MODFLOW model captures the bulk Culebra flow behavior, while the PEST calibration improved the model fit to the specific June 2005 observations.

4 2006 Results

4.1 2006 Freshwater Head Contours

The model-generated freshwater head contours are given in Figure 7 and Figure 8. There is a roughly east-west trending band of steeper gradients, corresponding to lower Culebra transmissivity. The uncontoured region in the eastern part of the figures corresponds to the portion of the Culebra that is located stratigraphically between halite in other members of the Rustler Formation (Tamarisk Member above and Los Medaños Member below). This region east of the "halite margin" has high freshwater head but extremely low transmissivity, essentially serving as a no-flow boundary in this area.

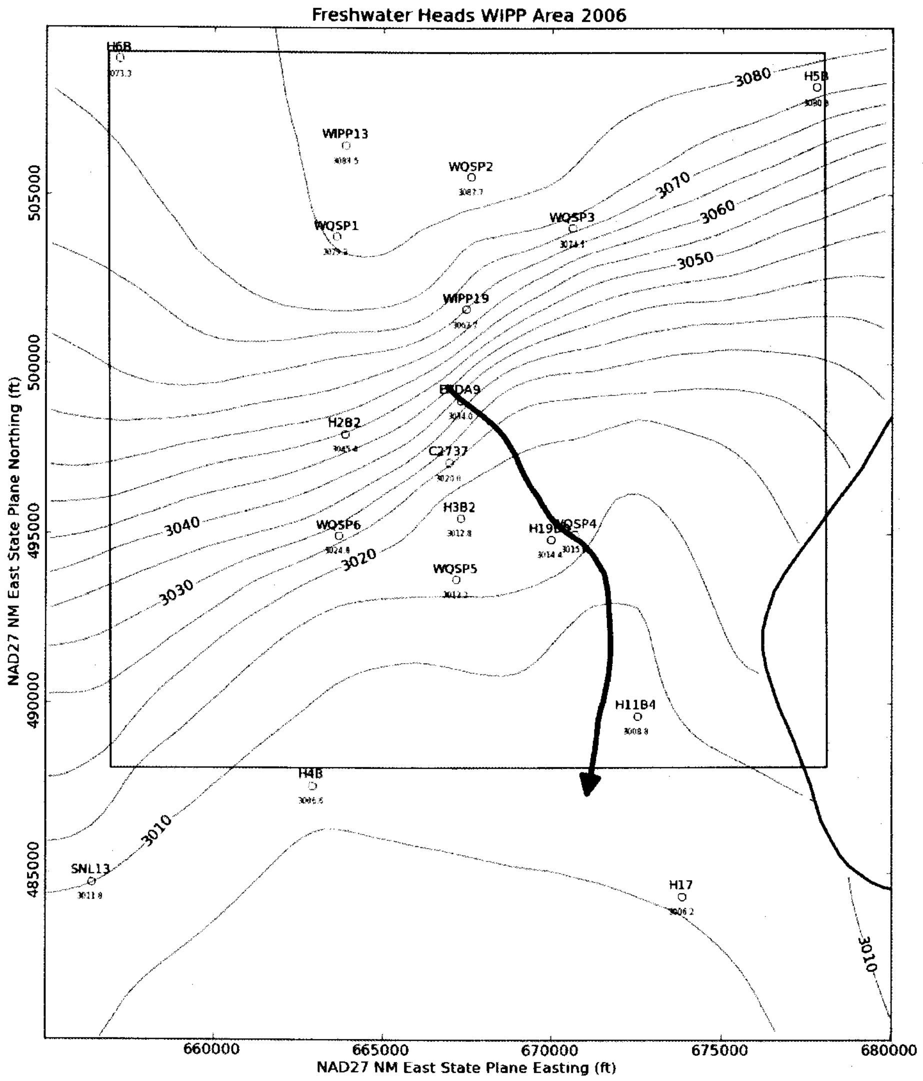


Figure 7. Model-generated November 2006 freshwater head contours with observed head listed at each well (5-foot contour interval) with blue water particle track from waste handling shaft to WIPP LWB

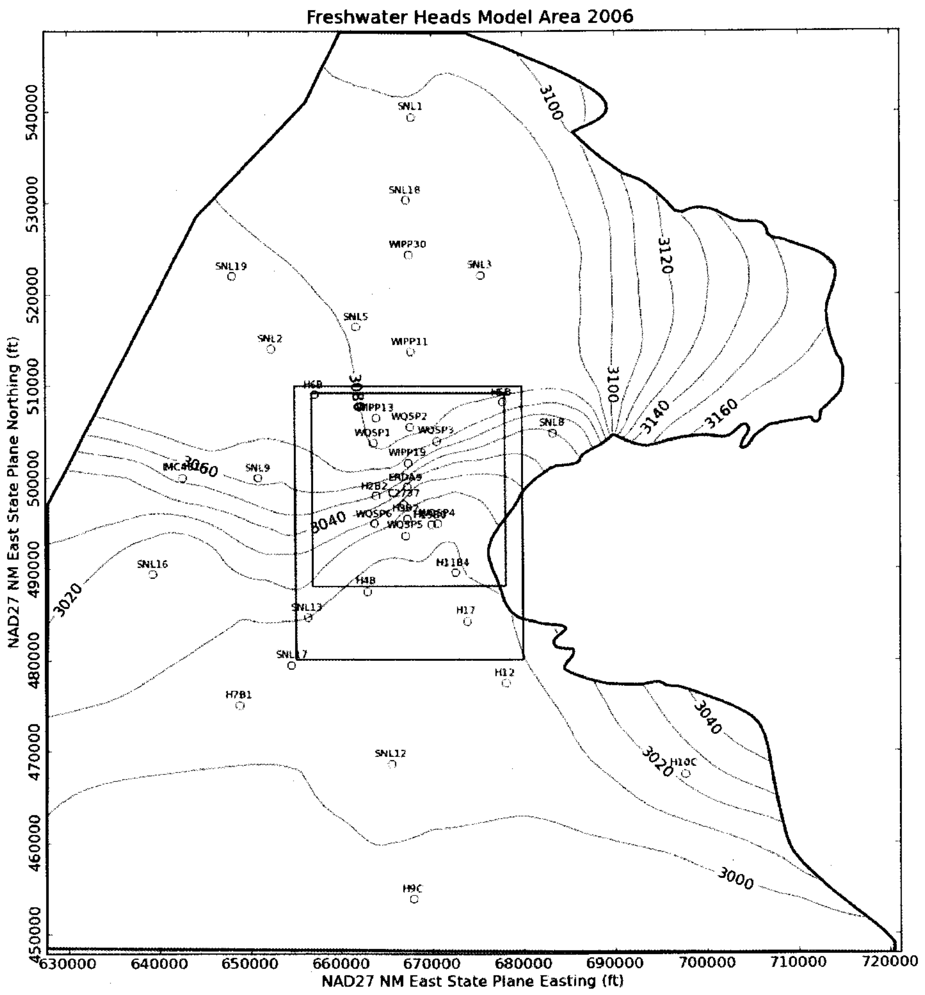


Figure 8. MODFLOW-modeled November 2006 heads for entire model domain (10-foot contour interval). Green rectangle indicates region contoured in Figure 7, black square is WIPP LWB.

4.2 2006 Particle Track

The blue arrow in Figure 7 shows the DTRKMF-calculated path a water particle would take through the Culebra from the coordinates corresponding to the WIPP waste handling shaft to the LWB (a path length of 4097 m). Assuming a 4-m thickness for the transmissive portion of the Culebra and a constant porosity of 16%, the travel time to the WIPP LWB is 5642 years (output from DTRKMF is adjusted from an original 7.75-m Culebra thickness). This is an average velocity of 0.73 m/yr.

4.3 2006 Measured vs. Modeled Fit

The scatter plot in Figure 9 shows measured and modeled freshwater heads at the observation locations used in the PEST calibration. The observations are divided into three groups, based on proximity to the WIPP site. Wells within the LWB are represented by red crosses, wells outside but within 3 km of the LWB are represented with green 'x's, and other wells within the MODFLOW model domain but distant from the WIPP site are given by a blue star.

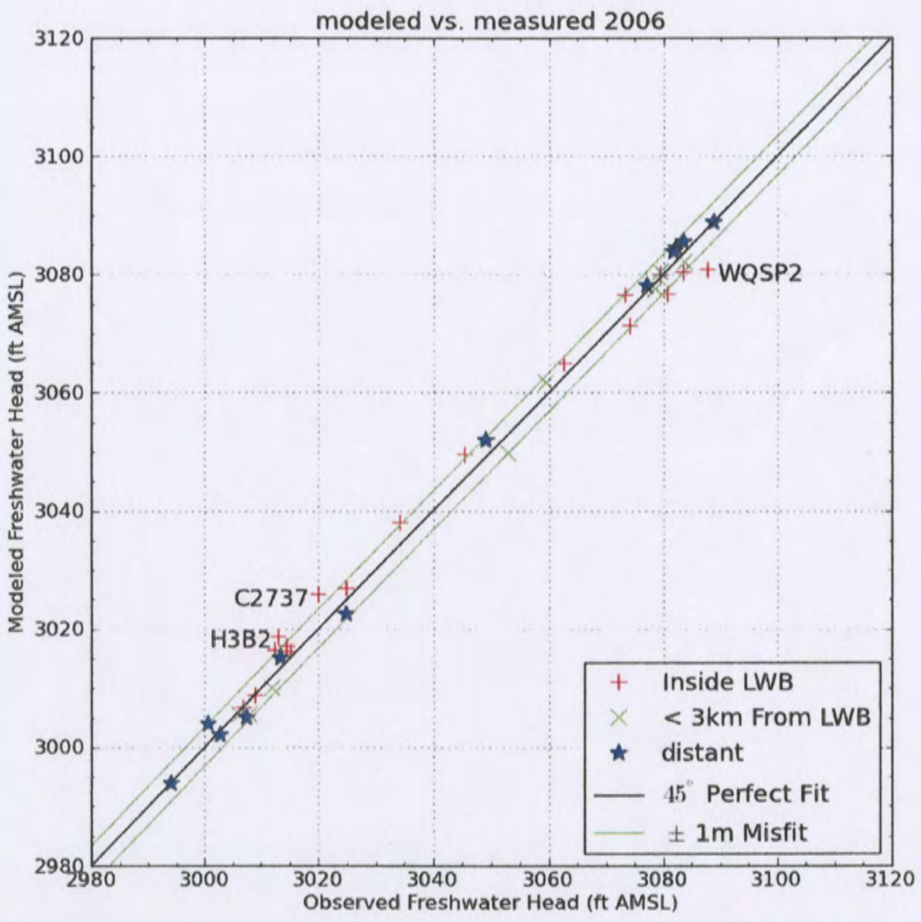


Figure 9. Measured vs. modeled scatter plot for averaged MODFLOW model generated heads and November 2006 observed freshwater heads

The central black diagonal line in Figure 9 represents a perfect model fit (1:1 or 45-degree slope); the two green lines on either side of this represent a 1-m misfit above or below the perfect fit. Wells more than 1.5 m from the 1:1 line are labeled. The calibrated parameters (for equation 1) were A=927.9, B=7.99, C=1.16, D=1.01, E=1.04, F=-1.01, and α=0.620. The parameter α, the y-direction exponent, had the largest relative change (24%) compared to the starting values. Overall, the difference between the parameters used in the original model and the calibrated model is slight. The boundary conditions along the northern and southern edges of the model are impacted most by the exponent.

The squared correlation coefficient (R^2) for the measured vs. modeled data is listed in Table 4. Figure 10 and Figure 11 show the distribution of errors resulting from the PEST-adjusted model fit to observed data. The wells within and near the WIPP LWB have an R^2 of greater than 99%, the calibration did not improve the R^2 fit within three significant digits, which can be expected because the original and calibrated model parameters are not dissimilar. The distribution in Figure 10 is roughly symmetric about 0, indicating there is not a strong bias.

Table 4. 2006 Measured vs. Modeled correlation coefficients

	dataset	measured vs. modeled R^2
Uncalibrated	wells inside WIPP LWB	0.991
	wells <3km from WIPP LWB	0.991
	all wells	0.993
Calibrated	wells inside WIPP LWB	0.991
	wells <3km from WIPP LWB	0.991
	all wells	0.993

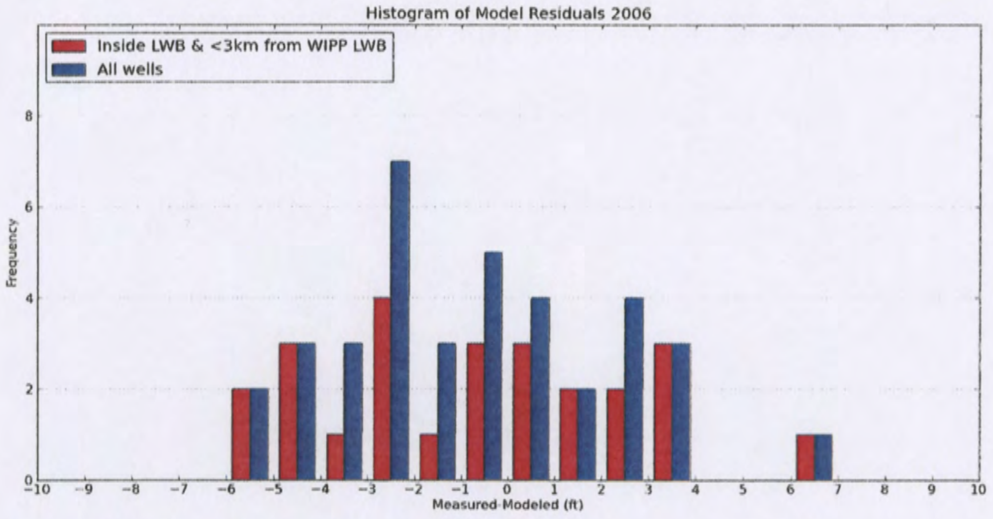


Figure 10. Histogram of Measured-Modeled errors for 2006

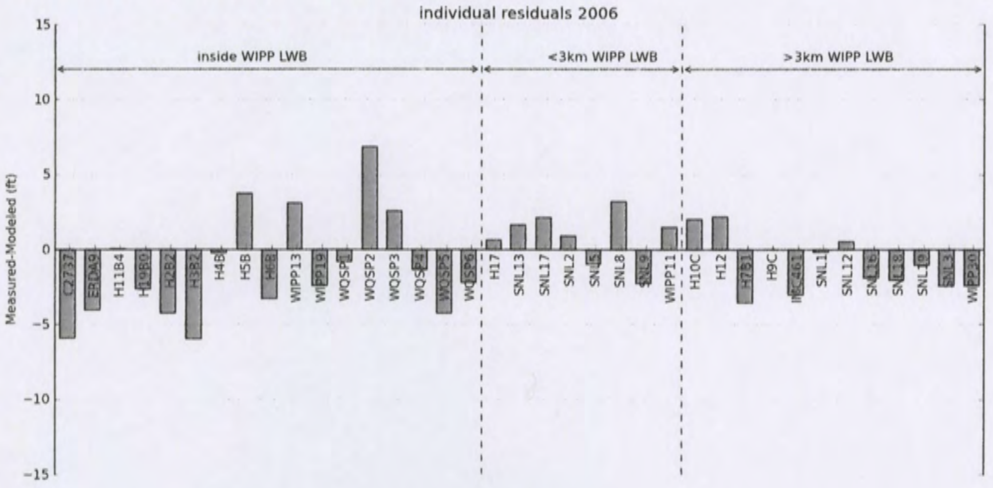


Figure 11. Measured-Modeled errors at each well for 2006

The model fit to the November 2006 observations is very good. The averaged MODFLOW model captures the bulk Culebra flow behavior, while the PEST calibration improved the model fit to the specific November 2006 observations.

5 2007 Results

5.1 2007 Freshwater Head Contours

The model-generated freshwater head contours are given in Figure 12 and Figure 13. There is a roughly east-west trending band of steeper gradients, corresponding to lower Culebra transmissivity. The uncontoured region in the eastern part of the figures corresponds to the portion of the Culebra that is located stratigraphically between halite in other members of the Rustler Formation (Tamarisk Member above and Los Medaños Member below). This region east of the "halite margin" has high freshwater head but extremely low transmissivity, essentially serving as a no-flow boundary in this area.

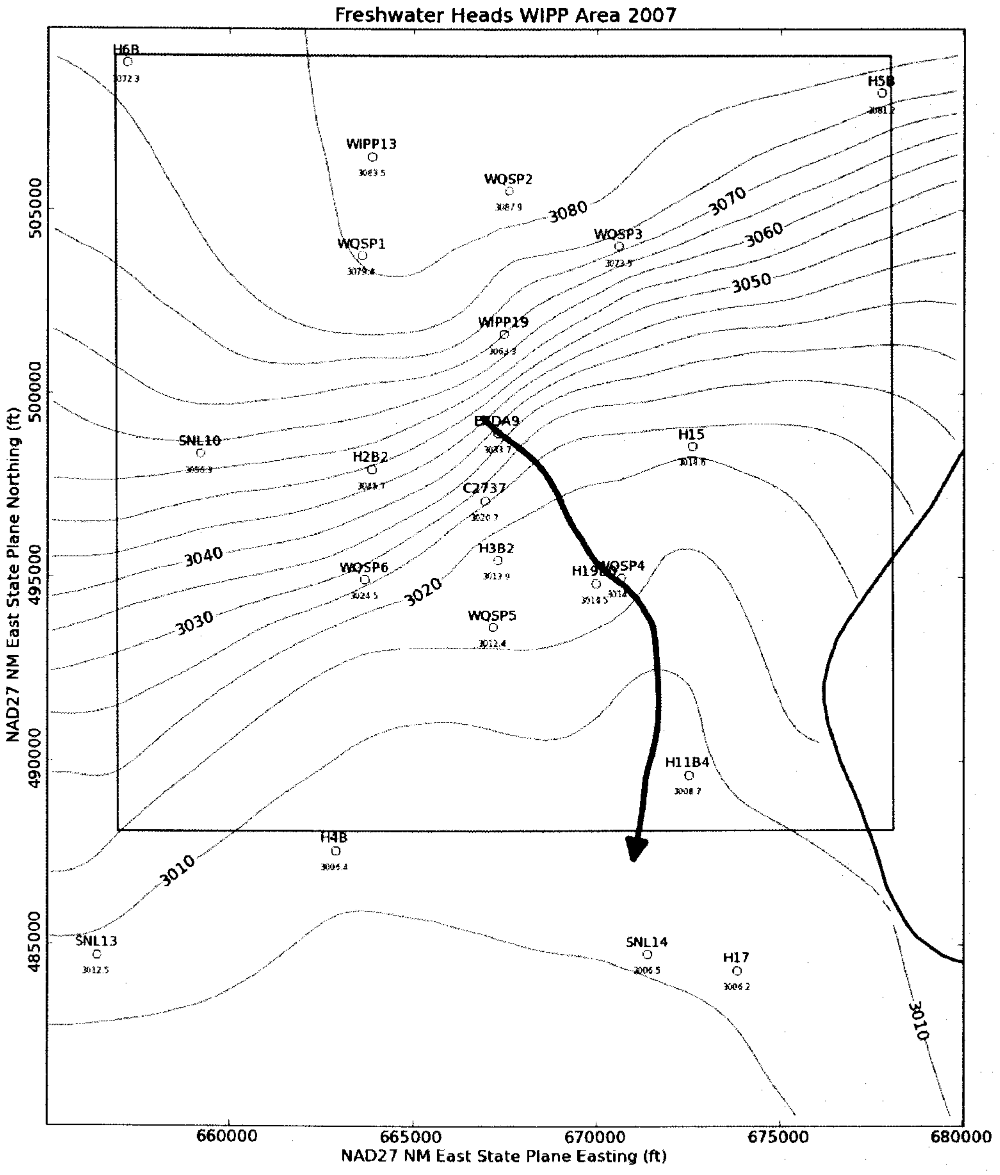


Figure 12. Model-generated May 2007 freshwater head contours with observed head listed at each well (5-foot contour interval) with blue water particle track from waste handling shaft to WIPP LWB

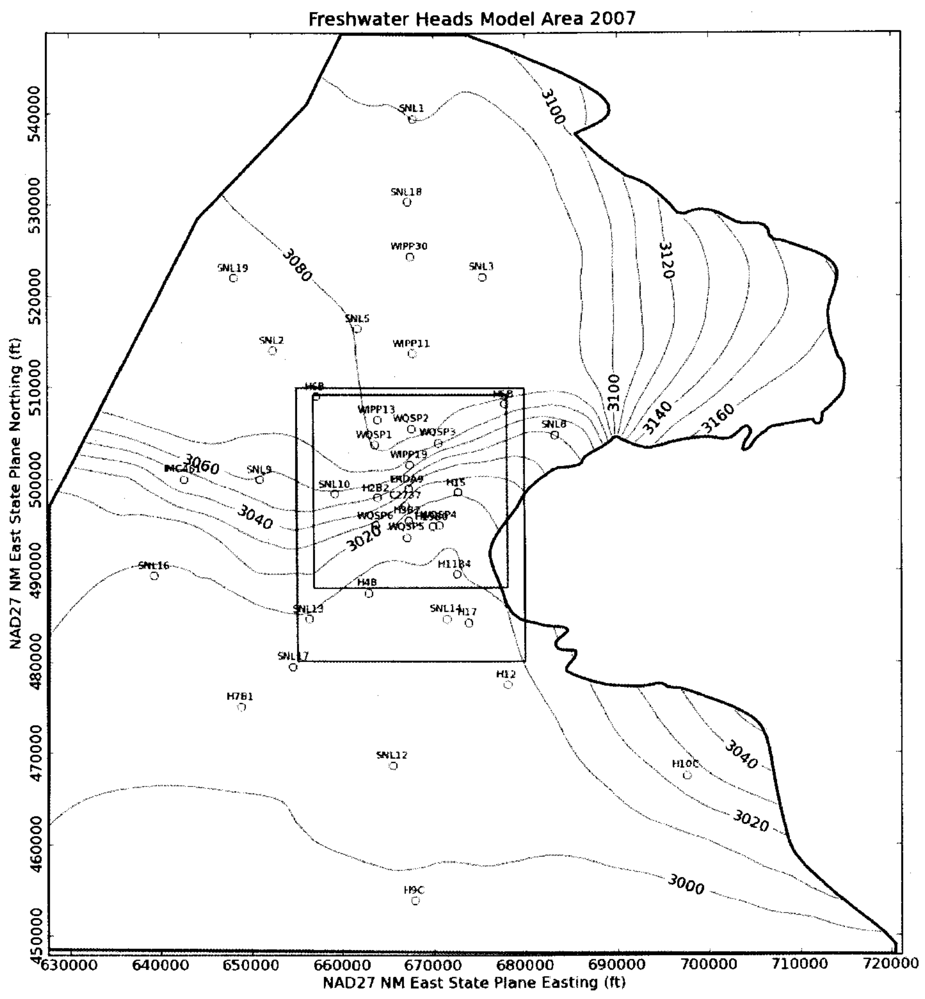


Figure 13. MODFLOW-modeled May 2007 heads for entire model domain (10-foot contour interval). Green rectangle indicates region contoured in Figure 12, black square is WIPP LWB.

5.2 2007 Particle Track

The blue arrow line in Figure 12 shows the DTRKMF-calculated path a water particle would take through the Culebra from the coordinates corresponding to the WIPP waste handling shaft to the LWB (a path length of 4084 m). Assuming a 4-m thickness for the transmissive portion of the Culebra and a constant porosity of 16%, the travel time to the WIPP LWB is 5845 years (output from DTRKMF is adjusted from an original 7.75-m Culebra thickness). This is an average velocity of 0.70 m/yr.

5.3 2007 Measured vs. Modeled Fit

The scatter plot in Figure 14 shows measured and modeled freshwater heads at the observation locations used in the PEST calibration. The observations are divided into three groups, based on proximity to the WIPP site. Wells within the LWB are represented by red crosses, wells outside but within 3 km of the LWB are represented with green 'x's, and other wells within the MODFLOW model domain but distant from the WIPP site are given by a blue star.

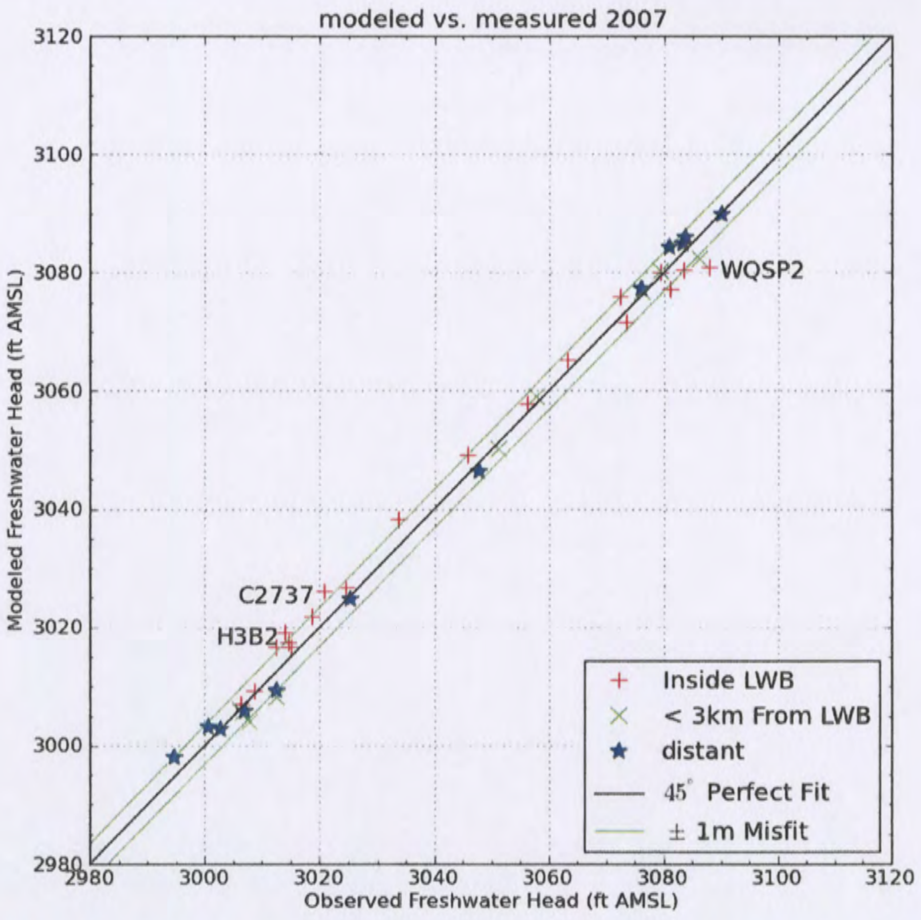


Figure 14. Measured vs. modeled scatter plot for averaged MODFLOW model generated heads and May 2007 observed freshwater heads

The black central diagonal line in Figure 14 represents a perfect model fit (1:1 or 45-degree slope); the two green lines on either side of this represent a 1-m misfit above or below the perfect fit. Wells more than 1.5 m from the 1:1 line are labeled. The calibrated parameters (for equation 1) were A=928.9 B=8.17, C=1.40, D=0.920, E=0.903, F=-0.75, and α=-0.052. The parameter α, the y-direction exponent, had the largest relative change (-110%) compared to the starting values. The parameter F, the weight associated with the second-order x-direction term, had the second largest relative change (-25%) compared to the starting values. The boundary conditions along the northern and southern edges of the model are impacted most by the exponent, while the boundary conditions along the western boundary is affected most by changes to the F coefficient.

The squared correlation coefficient (R^2) for the measured vs. modeled data is listed in Table 5. Figure 15 and Figure 16 show the distribution of errors resulting from the PEST-adjusted fit to observed data. The wells within and near the WIPP LWB have an R^2 of greater than 99%, with calibration providing a slight improvement inside the WIPP LWB, at the expense of the fit outside the WIPP LWB (in the third decimal place). The distribution in Figure 15 is roughly symmetric about 0, indicating there is not a strong bias.

Table 5. 2007 Measured vs. Modeled correlation coefficients

	dataset	measured vs. modeled R^2
Uncalibrated	wells inside WIPP LWB	0.991
	wells <3km from WIPP LWB	0.990
	all wells	0.993
Calibrated	wells inside WIPP LWB	0.992
	wells <3km from WIPP LWB	0.990
	all wells	0.993

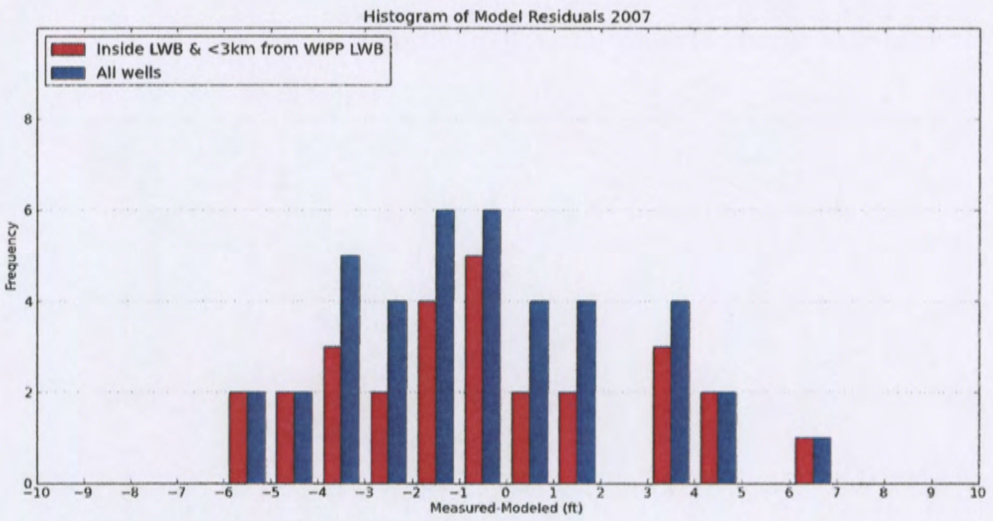


Figure 15. Histogram of Measured-Modeled errors for 2007

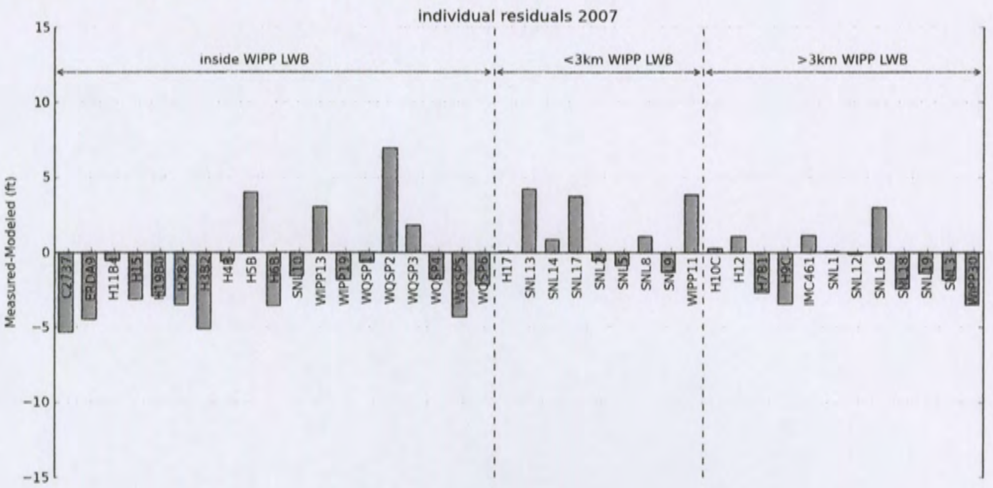


Figure 16. Measured-Modeled errors at each well for 2007

The model fit to the May 2007 observations is excellent, because these heads were the ones used to calibrate the PA MODFLOW model. The averaged MODFLOW model captures the bulk Culebra flow behavior, while the PEST calibration improved model fit to the May 2007 observations.

6 Summary

The development of the 2005-2007 historic Culebra contour maps in general followed quite closely to the procedure used in previous revisions of this report. The main deviations included additional work to gather, plot, and select appropriate freshwater head data for these years. The results of this additional work is presented in Section 8, and gives a unique view of the results presented in previous ASER reports, which is continuous across calendar years.

The average MODFLOW model calibration process resulted in similar (2006) or improved (2005 & 2007) model fits to the data selected for each year. The process of averaging the 100 realizations, and working with a single set of results from the average MODFLOW model creates a simpler result, which is still based upon the PA MODFLOW model.

This work began as part of an effort to create consistent Culebra contour maps for historic data already reported in the ASER. The results of this work (2005-2007) and upcoming work (pre-2005) will be consistency in maps across years and between regulators. The US Environmental Protection Agency and The NM Environment Department now receive compatible hydrology products (PA MODFLOW model and these contour maps) from the WIPP hydrology community.

7 References

Department of Energy. 2005. *WIPP Annual Site Environmental Report for 2004*. DOE/WIPP-05-2225.

Department of Energy. 2006. *WIPP Annual Site Environmental Report for 2005*. DOE/WIPP-06-2225.

Department of Energy. 2007. *WIPP Annual Site Environmental Report for 2006*. DOE/WIPP-07-2225.

Department of Energy. 2008. *WIPP Annual Site Environmental Report for 2007*. DOE/WIPP-08-2225.

Department of Energy. 2009. *WIPP Annual Site Environmental Report for 2008*. DOE/WIPP-09-2225.

Department of Energy. 2010. *WIPP Annual Site Environmental Report for 2009*. DOE/WIPP-10-2225.

Department of Energy. 2011. *WIPP Annual Site Environmental Report for 2010*. DOE/WIPP-11-2225.

Doherty, J. 2002. *PEST: Model Independent Parameter Estimation*. Watermark Numerical Computing, Brisbane, Australia.

Harbaugh, A.W., E.R. Banta, M.C. Hill, and M.G. McDonald. 2000. *MODFLOW-2000, the U.S. Geological Survey modular ground-water model – User guide to modularization concepts and the Ground-Water Flow Process*. U.S. Geological Survey Open-File Report 00-92.

Hart, D.B., S.A. McKenna, and R.L. Beauheim. 2009. *Analysis Report for Task 7 of AP-114: Calibration of Culebra Transmissivity Fields*. Carlsbad, NM, Sandia National Laboratories, ERMS 552391.

Johnson, P.B. 2008. *Potentiometric Surface, Adjusted to Equivalent Freshwater Heads, of the Culebra Dolomite Member of the Rustler Formation near the WIPP Site, May 2007 (AP-114 Task 6)*. Carlsabd, NM, Sandia National Laboratories, ERMS 548746.

Johnson, P.B. 2009. *Potentiometric Surface, Adjusted to Equivalent Freshwater Heads, of the Culebra Dolomite Member of the Rustler Formation near the WIPP Site, May 2007, Revision 2 (AP-114 Task 6)*. Carlsabd, NM, Sandia National Laboratories, ERMS 551116.

Johnson, P.B. 2012a. 2005 Calculated Densities, Sandia National Laboratories, Carlsbad, NM, ERMS 556883.

Johnson, P.B. 2012b. 2006 Calculated Densities, Sandia National Laboratories, Carlsbad, NM, ERMS 556887.

Kuhlman, K.L. 2011. Analysis Report for Preparation of 2010 Culebra Potentiometric Surface Contour Map, Rev 2, Sandia National Laboratories, Carlsbad, NM, ERMS 555318.

Kuhlman, K.L. 2009. Procedure SP 9-9, revision 0, Preparation of Culebra potentiometric surface contour maps. Carlsbad, NM, Sandia National Laboratories, ERMS 552306.

Moody, D.C. 2009. *Stipulated Final Order for Notice of Violation for Detection Monitoring Program*, Sandia National Laboratories, Carlsbad, NM. WIPP Records Center, ERMS 551713.

8 Run Control Narrative

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

Figure 17 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 for each of the 100 realizations of MODFLOW, listed in the file `keepers` (see lines 17-26 of script). Some of the realizations are inside the `Update` or `Update2` subdirectories in CVS, which complicates the directory structure. An equivalent list `keepers_short` is made from `keepers`, and the directories are moved to match the flat directory structure (lines 31-53). At this point, the directory structure has been modified but the MODFLOW input files checked out from CVS are unchanged.

Python script `average_realizations.py` (§A-4.2) is called, which first reads in the `keepers_short` list, then reads in each of the 400 input files and computes the arithmetic average of the base-10 logarithm of the value at each cell across the 100 realizations. The 400 input files are saved as a flattened 2D matrix, in row-major order. The exponentiated result is saved in 4 parameter fields, 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 6 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`.

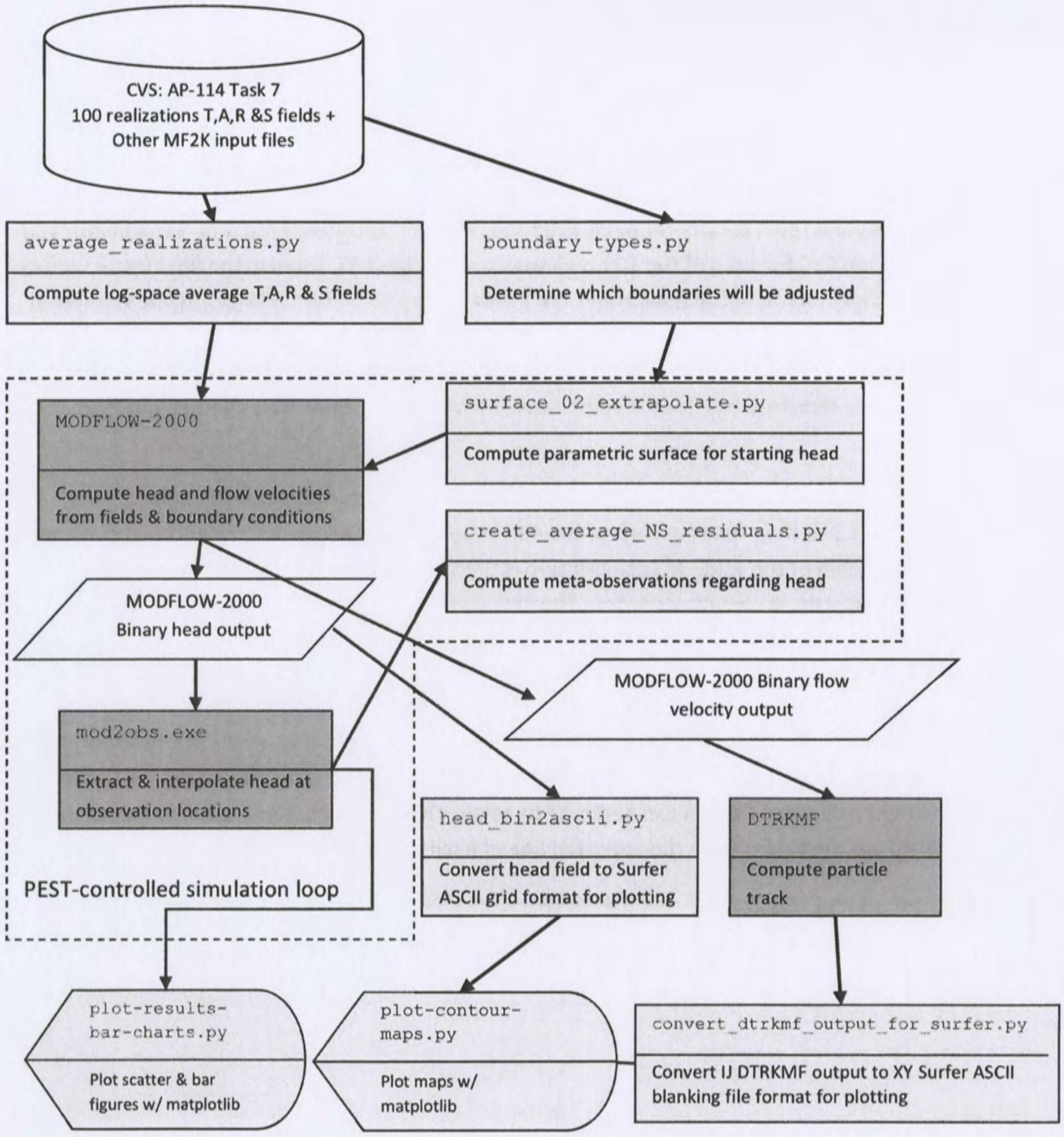

CVS: AP-114 Task 7
100 realizations T,A,R &S fields +
Other MF2K input files
average_realizations.py
Compute log-space average T,A,R & S fields
boundary_types.py
Determine which boundaries will be adjusted
MODFLOW-2000
Compute head and flow velocities from fields & boundary conditions
surface_02_extrapolate.py
Compute parametric surface for starting head
create_average_NS_residuals.py
Compute meta-observations regarding head
MODFLOW-2000
Binary head output
MODFLOW-2000 Binary flow velocity output
mod2obs.exe
Extract & interpolate head at observation locations
PEST-controlled simulation loop
head_bin2ascii.py
Convert head field to Surfer ASCII grid format for plotting
DTRKMF
Compute particle track
plot-results-bar-charts.py
Plot scatter & bar figures w/ matplotlib
plot-contour-maps.py
Plot maps w/ matplotlib
convert_dtrkmf_output_for_surfer.py
Convert IJ DTRKMF output to XY Surfer ASCII blanking file format for plotting

Figure 17. Process flowchart; dark gray indicates qualified programs, light gray are scripts written for this analysis

Table 6. 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 18 shows plots of the average log10 parameters, which compare with similar figures in Hart et al. (2009); inactive regions <1.0E-15 were reset to 1.0 to improve the plotted color scale. The rest of the calculations are done with these averaged fields.

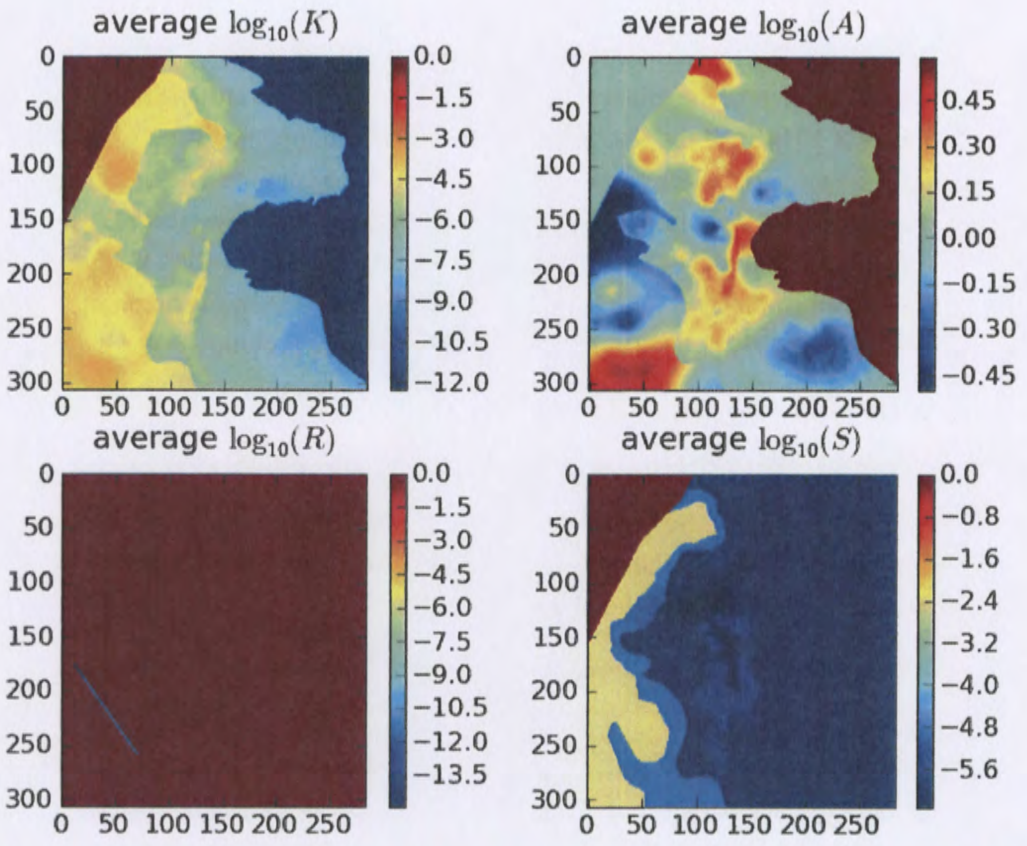


Figure 18. Plots of base-10 logarithms of average parameter fields; rows and columns are labeled on edges of figures.

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

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

The Python script `head_bin2ascii.py` (§A-4.7) converts the MODFLOW binary head file, which includes the steady-state head at every element in the flow model domain (307 rows × 284 columns) into a Surfer ASCII grid file format. This file is simply contoured in Python using matplotlib, no interpolation or gridding is needed. The Python script `convert_dtrkmf_output_for_surfer.py` (§A-4.9) reads the DTRKMF output file `dtrk.out` and does two things. First it converts the row, column format of this output file to an X,Y format suitable for plotting, and second it converts the effective thickness of the Culebra from 7.75m to 4m. 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 7) after the header are cumulative time (red), column (blue), and row (green). The three columns in the blanking file (second half of Table 7) after the header are UTM NAD27 X (blue), UTM NAD27 Y (green), and adjusted cumulative time (red, which is faster faster than the original cumulative travel time by the factor 7.75/4=1.9375). The conversion from row, column to X, Y is

$$\begin{aligned} X &= 601700.0 + 100.0 * column \\ Y &= 3597100.0 - 100.0 * row \end{aligned},$$

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 7. 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.exe` 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. The Python script `merge_observed_modeled_heads.py` (§A-4.9) simply puts the results from `mod2obs.exe` and the original observed heads in a single file together for easier plotting and later analysis.

A similar process to that described so far in this narrative 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 IBOUND array, where the two different types of boundary conditions are differentiated. This Python script uses the MODFLOW IBOUND array (`init_bnds_orig.inf` first ⅓ of Table 8) and the initial head array (`init_head_orig.mod` middle ⅓ of Table 8) as inputs, and writes a new MODFLOW IBOUND array (`init_bnds.inf` bottom ⅓ of Table 8) 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 8. Input IBOUND, starting head, and output IBOUND array data corresponding to first row of MODFLOW 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 8 shows the data corresponding to the northernmost row of the MODFLOW 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 treats any cells as constant head which have an IBOUND entry < 0, so both -2 and -1 are the same to MODFLOW, but allow distinguishing between them in the Python script which extrapolates the heads to the boundaries.

The required PEST input files are created by the Python script `create_pest_02_input.py` (§A-4.4). This script writes **1)** the PEST instruction file (`modeled_head.ins`), which shows PEST how to extract the model-predicted heads from the `mod2obs.exe` output; **2)** the PEST template file (`surface_par_params.ptf`), which shows PEST how to write the input file for the surface extrapolation script; **3)** the PEST parameter file (`surface_par_params.par`), which lists the starting parameter values to use when checking the PEST input; **4)** the PEST control file (`bc_adjust_2009ASER.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_2009ASER.smp`), and the weights are read in from the input file (`obs_loc_2009ASER.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.exe` and creates a meta-observation that consists of the average residual between measured and model-prediction, only averaged across the northern or southern WIPP wells (the wells in the center of the WIPP site are not included in either group). This was done to minimize cancelation of the errors north (where the model tended to underestimate heads) and south (where the model tended to overestimate heads) of the WIPP. The results of this script are read directly by PEST and incorporated as four additional observations (mean and median errors, both north and south of WIPP).

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

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

These two plotting scripts (`plot-contour-maps.py` and `plot-results-bar-charts.py`) are included in the appendix for completeness, but only draw the figures included in this report, and passed on to WRES for the ASER. These two scripts automate the plotting process and take the place of the Microsoft Excel, USACE Corpscon, and Golden Software Surfer input files that were previously used.

9 Appendix: Water Level and Density Data Listing

9.1 Input files for plotting water levels and densities

bytes	description	file name
2.3K	Culebra midpoint elevations	`culebra-midpoint-elevations.csv`
4.8K	UTM X and Y coordinates for wells	`well-coordinates.csv`
0.9K	reference point change for pre-2007 elevations	`reference-point-change-2007.dat`
43K	data from Table-F.8 of 2004 ASER	`ASER-2004-waterlevel-data.csv`
40K	data from Table-F.8 of 2005 ASER	`ASER-2005-waterlevel-data.csv`
41K	data from Table-F.8 of 2006 ASER	`ASER-2006-waterlevel-data.csv`
42K	data from Table-F.8 of 2007 ASER	`ASER-2007-waterlevel-data.csv`
43K	data from Table-F.8 of 2008 ASER	`ASER-2008-waterlevel-data.csv`
43K	data from Table-F.8 of 2009 ASER	`ASER-2009-waterlevel-data.csv`
42K	data from Table-F.8 of 2010 ASER	`ASER-2010-waterlevel-data.csv`
15K	summary of events in wells from ASERs	`well-events.csv`
23K	data from Table-6.3 of ASERs	`reported-density-values.csv`
2.9K	designated densities to use at wells	`densities-to-use.csv`

9.1.1 `densities-to-use.csv` input file

well	begin date	end date	density	source
C-2737	1/1/1998	5/12/2003	1.000	ASER
	11/19/2003	2/1/2007	1.019	TROLL2005
	2/1/2007	6/28/2008	1.010	JOHNSON2009
	6/28/2008	1/1/2012	1.029	TROLL2008
ERDA-9	1/1/1998	1/1/2012	1.067	JOHNSON2009
H-02B2	1/1/1998	4/12/2005	1.014	JOHNSON2009
	4/12/2005	2/24/2009	1.000	SNL notebooks
	2/24/2009	1/1/2012	1.011	TROLL2010
H-03B2	1/1/1998	1/1/2012	1.042	JOHNSON2009
H-04B	1/1/1998	6/14/2009	1.015	JOHNSON2009
H-05B	1/1/1998	6/11/2005	1.104	ASER
	6/11/2005	1/1/2012	1.095	JOHNSON2009
H-06B	1/1/1998	2/19/2008	1.040	JOHNSON2009
H-07B1	1/1/1998	1/1/2012	1.002	JOHNSON2009
H-09C	1/1/1998	1/1/2012	1.001	JOHNSON2009
H-10C	2/19/2002	7/12/2009	1.001	JOHNSON2009
	7/12/2009	1/1/2012	1.089	TROLL2009
H-11B4	1/1/1998	1/1/2012	1.070	JOHNSON2009
H-12	1/1/1998	12/1/2003	1.083	SNL notebooks
	4/12/2005	11/24/2008	1.097	JOHNSON2009
	11/24/2008	1/1/2012	1.096	TROLL2008
H-15	1/1/1998	2/1/2001	1.154	SAND89-7068/2
	11/18/2003	4/10/2006	1.082	TROLL2005
	4/10/2006	3/5/2008	1.053	JOHNSON2009
H-17	1/1/1998	1/1/2012	1.133	JOHNSON2009
H-19B0	1/1/1998	1/1/2012	1.068	JOHNSON2009
I-461	10/15/2003	1/1/2012	1.005	JOHNSON2009
P-17	1/1/1998	6/17/2005	1.070	PDS 2004 2005 avg
	6/17/2005	8/18/2006	1.053	SGR 2006 SNL SURVEY
SNL-01	3/25/2004	1/1/2012	1.033	JOHNSON2009
SNL-02	4/28/2003	1/1/2012	1.012	JOHNSON2009
SNL-03	8/14/2003	1/1/2012	1.023	JOHNSON2009
SNL-05	4/26/2004	1/1/2012	1.010	JOHNSON2009
SNL-08	6/14/2005	8/28/2007	1.052	JOHNSON2009
	8/28/2007	1/1/2012	1.103	TROLL2007
SNL-09	5/17/2003	1/1/2012	1.024	JOHNSON2009
SNL-10	5/31/2006	1/1/2012	1.011	JOHNSON2009
SNL-12	6/25/2003	1/1/2012	1.005	JOHNSON2009
SNL-13	4/11/2005	1/1/2012	1.027	JOHNSON2009
SNL-14	5/3/2005	1/1/2012	1.0480	JOHNSON2009
SNL-16	4/11/2006	1/1/2012	1.010	JOHNSON2009
SNL-17	4/25/2006	1/1/2012	1.006	JOHNSON2009
SNL-18	6/20/2006	1/1/2012	1.028	JOHNSON2009
SNL-19	5/5/2006	1/1/2012	1.003	JOHNSON2009
WIPP-11	9/7/2004	1/1/2012	1.038	JOHNSON2009
WIPP-13	1/1/1998	1/1/2012	1.053	JOHNSON2009
WIPP-19	1/1/1998	1/1/2012	1.044	JOHNSON2009
WIPP-21	1/1/1998	1/1/2012	1.071	ASER
WIPP-22	1/1/1998	1/1/2012	1.087	ASER
WIPP-25	1/1/1998	6/12/2009	1.011	JOHNSON2009
WIPP-29	1/1/1998	1/1/2012	1.180	ASER
WIPP-30	1/1/1998	6/22/2005	1.018	SAND89-7068/2
	6/22/2005	3/1/2008	1.000	JOHNSON2009
WQSP-1	1/1/1998	1/1/2012	1.048	SGR ROUNDS 23 to 25
WQSP-2	1/1/1998	1/1/2012	1.048	SGR ROUNDS 23 to 25
WQSP-3	1/1/1998	1/1/2012	1.146	SGR ROUNDS 23 to 25
WQSP-4	1/1/1998	1/1/2012	1.075	SGR ROUNDS 23 to 25
WQSP-5	1/1/1998	1/1/2012	1.025	SGR ROUNDS 23 to 25
WQSP-6	1/1/1998	1/1/2012	1.014	SGR ROUNDS 23 to 25

9.2 Listing of Water Level Plotting Script

9.2.1 Python script `plot-waterlevels.py`

This script is not run on the QA linux cluster, `alice.sandia.gov`. This script is run on a desktop PC. It is only used to create figures for the selection of sampling dates and proper density values.

```
# this python script plots water level and density data for WIPP wells
# for the purpose of choosing freshwater heads for historic contouring
# using data taken from the Annual Site Environmental Reports (ASER).
#
# by Kris Kuhlman (6212)
# September 2011 through January 2012
#

import matplotlib  # set plotting backend
matplotlib.use('Agg')

import numpy as np  # array library
import numpy.core.defchararray as npchar  # functions for character arrays

import matplotlib.pyplot as plt  # plotting library
import matplotlib.mlab as mlab
from matplotlib.dates import MonthLocator, YearLocator, DateFormatter
from matplotlib.ticker import NullFormatter
from matplotlib.font_manager import FontProperties

import datetime
import re # regular expressions

# for making small text in figures
AnnFontP = FontProperties()
AnnFontP.set_size('xx-small') # for annotations
LegFontP = FontProperties()
LegFontP.set_size('small') # for legend

# read in corrections to pre-2007 data
with open('reference-point-change-2007.dat','r') as fh:
    lines = fh.readlines()

rpcorr = {}
for line in lines:
    well,delta = line.split()
    rpcorr[well] = float(delta)


# read in freshwater heads used to create past contour maps
# for years 2008, 2009 and 2010.
# $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
aserfwh = {}
for year in [2008,2009,2010]:
    with open('meas_head_%iASER.smp' % (year,),'r') as fh:
        lines = fh.readlines()

    aserfwh[year] = {}
    for line in lines:
        fields = line.rstrip().split()

        # regexp for splitting wellnames up into parts
        # non-numbers, numbers, and non-numbers (last group is sometimes empty)
        m = re.search(r"([^0-9]+)([0-9]+)([^0-9]*)",fields[0])
        w = [m.group(1),m.group(2),m.group(3)]

        # handle mapping between well names used in pest input files
        # and names used in more complete database
        if w[0] == 'SNL' or w[0] == 'H':
            wstr = '%s-%2.2i%s' % (w[0],int(w[1]),w[2].upper())
        elif w[0] == 'IMC':
            wstr = 'I-461'
        else:
            wstr = '%s-%s%s' % (w[0],w[1],w[2].upper())
        aserfwh[year][wstr.strip()] = {'fwh':float(fields[3]),'name':fields[0]}


# read in "best" densities proposed to use for contour maps
# $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
with open('densities-to-use.csv','r') as fh:
    f = fh.read()
    lines = f.split('\r')[1:]

findendtype = np.dtype([('well','S56'),('dt0','O8'),('dt1','O8'),
                        ('den','f8'),('src','S13')])
finden = []
for line in lines:
    r = [x.strip() for x in line.split(',')]
    dt0 = datetime.datetime.strptime(r[1],'%m/%d/%Y')
    dt1 = datetime.datetime.strptime(r[2],'%m/%d/%Y')
    finden.append((r[0],dt0,dt1,float(r[3]),r[4].upper()))

finden = np.array(finden,dtype=findendtype)
finden = np.sort(finden,order=('well','dt0','src'))


# read in Culebra midpoint elevations (mostly from Johnson, 2009
# developed and checked for model calibration)
# $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
with open('culebra-midpoint-elevations.csv','r')  as fh:
    f = fh.read()
    lines = f.split('\r')[1:]

midpt = {}
for line in lines:
    r = [x.strip() for x in line.split(',')]
    # elevation is in feet AMSL, convert to meters
    midpt[r[0].upper()] = float(r[1])*0.3048


# read in well XY coordinates
# $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
with open('well-coordinates.csv','r')  as fh:
    f = fh.read()
    lines = f.split('\r')[1:]

xyz = {}
for line in lines:
    r = [x.strip() for x in line.split(',')]
    # XY is UTM NAD27 ZONE 13 (m)
    xyz[r[0].upper()] = {'x':float(r[2]),'y':float(r[1])}


# read in table of published density values from ASER/SAND reports and
# from 2007-2010 the tables of TROLL-derived densities from SNL.
# $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
with open('reported-density-values.csv','r') as fh:
    f = fh.read()
    lines = f.split('\r')[1:]

dendtype = np.dtype([('well','S56'),('dt','O8'),('dt2','O8'),
                     ('den','f8'),('src','S13')])
den = []
```

```
for line in lines:
    r = [x.strip() for x in line.split(',')]
    dt = datetime.datetime.strptime(r[1],'%m/%d/%Y')

    # only troll densities have a date range
    if r[2] == "":
        dt2 = datetime.datetime(1941,12,7) # bogus datetime
    else:
        dt2 = datetime.datetime.strptime(r[2],'%m/%d/%Y')
    den.append((r[0].upper(),dt,dt2,float(r[3]),r[4].upper()))

den = np.array(den,dtype=dendtype)
den = np.sort(den,order=('well','dt','src'))
srcsymb = {'PDS':'.', 'SGR':'o','TROLL':'x','SAND89-7068/2':'<'}
srccolor = {'PDS':'gray','SGR':'black','TROLL':'red','SAND89-7068/2':'orange'}

# earliest date that should be plotted
# densities before this date are plotted here as an arrow pointing left
firstplot = datetime.datetime(2004,12,1)


# read in well event log: includes drilling, P&A, tests and
# well maintainence activities that might effect WL or densities
# $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
with open('well-events.csv','r') as fh:
    f = fh.read()
    lines = f.split('\r')[1:]

eventstype = np.dtype([('well','S56'),('dt0','O8'),('dt1','O8'),('s','S50'),('c','S10')])
events = []
for line in lines:
    r = [x.strip() for x in line.split(',')]
    if r[0] == '':
        # empty line at end?
        continue
    dt0 = datetime.datetime.strptime(r[1],'%m/%d/%y')
    if len(r[2]) > 0:
        dt1 = datetime.datetime.strptime(r[2],'%m/%d/%y')
    else:
        dt1 = None

    # try to categorize events based on color
    if 'samp' in r[3]:
        # water quality sampling
        c = 'gray'
    elif ('drill' in r[3] or 'perf' in r[3] or
          ('recomp' in r[3] and 'ulebra' in r[3])):
        # new or replacement well drilling
        c = 'green'
    elif 'PIP' in r[3] or 'packer' in r[3]:
        # added, removed, or reset packers
        c = 'blue'
    elif 'plug' in r[3] or 'recomp' in r[3]:
        # plugged back, plug & abandoned, or recompleted
        c = 'red'
    elif ('test' in r[3] and not 'well integrity' in r[3] and
          not 'configured' in r[3]):
        # slug or pumping test
        c = 'cyan'
    elif 'bail' in r[3] or 'swab' in r[3]:
        # bailed or swabbed well/tubing
        c = 'magenta'
    else:
        # something else
```

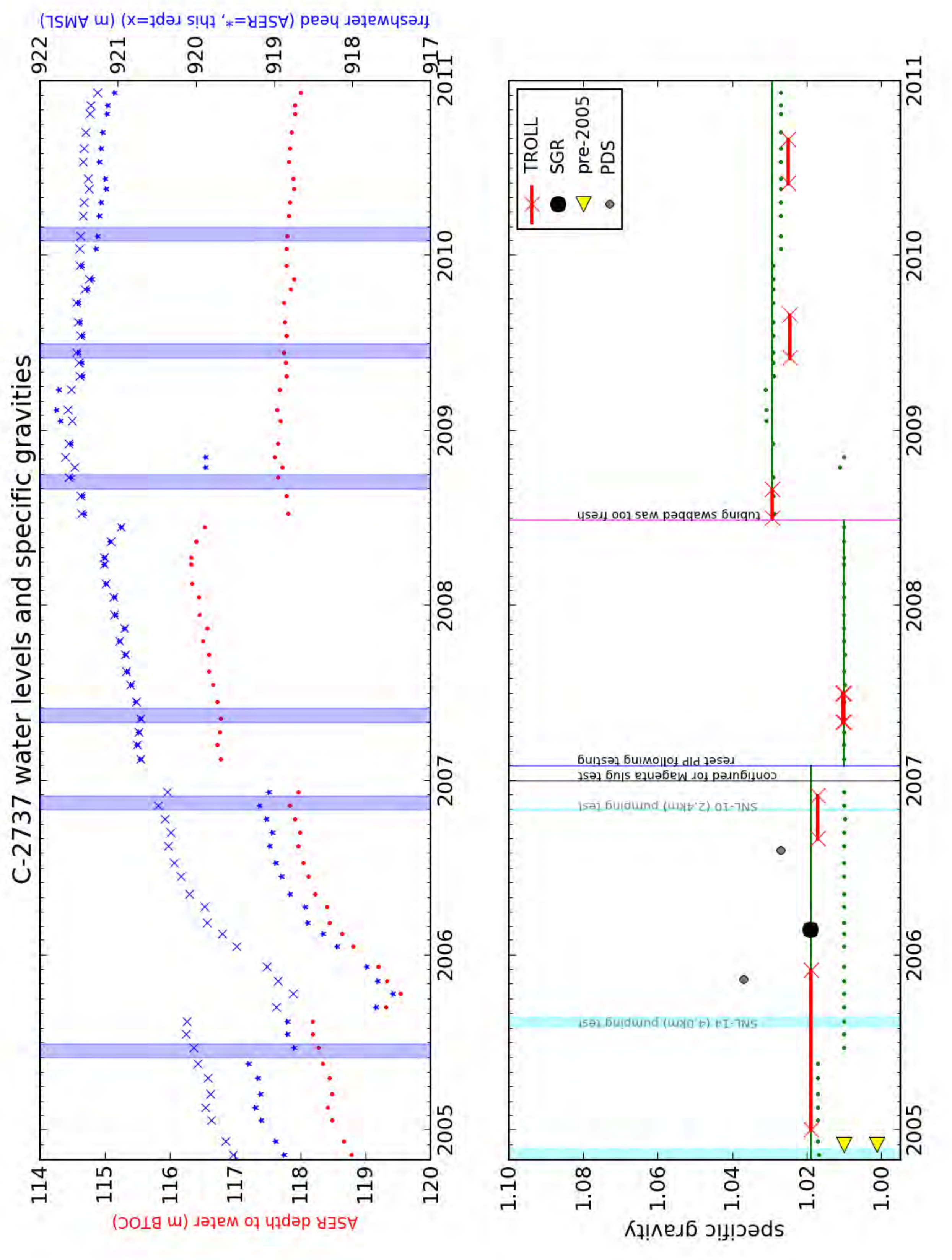
```
            c = 'black'

        events.append((r[0].upper(),dt0,dt1,r[3],c))

events = np.array(events,dtype=eventstype)
evwells = list(set(events[:]['well']))


# read in water level data from 2001-2010 ASER tables
# $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

# one header row
# mostly common column format
# A0 : well
# B1 : zone (CUL,MAG,etc.)
# C2 : date (no time)
# D3 : adjusted depth below top of casing (ft)
# E4 : adjusted depth below top of casing (m)
# F5 : water level elevation (ft amsl)
# G6 : water level elevation (m amsl)
# H7 : adjusted freshwater head (ft amsl)

# read in yearly files with csv reader

# make all well names uppercase, strip off anything in parenthesis
# strip "/" out of zone names

# convert dates to python date objects

# save (E) depth to water (meters)
# save (G) water level elevation (meters)
# save (H) freshwater head (feet) -> convert to meters

wldtype = np.dtype([('well','S56'),('zone','S4'),('dt','O8'),
                     ('dtwm','f8'),('wlem','f8'),('cwlem','f8'),('fwhm','f8')])
data = []

for year in range(2001,2010+1):

    earliest = datetime.datetime(2100,12,31)
    latest = datetime.datetime(1900,1,1)

    fn = 'ASER-%i-waterlevel-data.csv' % year
    print fn,
    with open(fn,'r') as fh:
        f = fh.read()
        lines = f.split('\r')  # Mac line endings

    print '# values',len(lines)
    for line in lines[1:]:

        r = line.split(',')
        dt = datetime.datetime.strptime(r[2],'%m/%d/%y')

        if dt < earliest:
            earliest = dt
        elif dt > latest:
            latest = dt

        # clean up and simplify well names to be consistent with HANALYST
        well = r[0].upper().partition('(')[0].partition('/')[0].strip()
        zone = r[1].upper().replace('/','').strip()

        if 'SNL' in well:
            # some SNL wells are not zero padded some years
            # therefore they appear as different wells
            num = well.split('-')[1]
            if len(num) == 1:
                well = 'SNL-0' + num

        if zone == 'SRD':
            zone = 'SRDL'
        elif 'RUSS' in zone:
            zone = 'RS'

        if len(r) < 8 or r[7].strip() == "":
            fwh = -999.0
        else:
            fwh = float(r[7])*0.3048

        wlem = float(r[6])
        if year < 2007 and well in rpcorr:
            cwlem = wlem + rpcorr[well]
        else:
            cwlem = wlem

        data.append((well,zone,dt,float(r[4]),wlem,cwlem,fwh))

data = np.array(data,dtype=wldtype)
data = np.sort(data,order=('dt','zone','well'))


# wells reported by WRES (doesn't include Gnome wells)
wells = list(set(data[:]['well']))
wells.sort()

# can plot all wells reported by WRES
zones = list(set(data[:]['zone']))
for zone in zones:
    zmask = data['zone'] == zone
    zwells = list(set(data[zmask]['well']))
    zwells.sort()

# months used for creating ASER contour maps (no apparent ranges for 2003 & 2004)
# NB: 2003, 2004, 2005 & 2006 are my choice, not necessarily what used in ASER
cmonths = {2001:(datetime.datetime(2001,12,1),datetime.datetime(2001,12,31)),
           2002:(datetime.datetime(2002,12,1),datetime.datetime(2002,12,31)),
           2003:(datetime.datetime(2003,9,1),datetime.datetime(2003,9,30)),
           2004:(datetime.datetime(2004,9,1),datetime.datetime(2004,9,30)),
           2005:(datetime.datetime(2005,6,1),datetime.datetime(2005,6,30)),
           2005:(datetime.datetime(2005,6,1),datetime.datetime(2005,6,30)),
           2006:(datetime.datetime(2006,11,1),datetime.datetime(2006,11,30)),
           2007:(datetime.datetime(2007,5,1),datetime.datetime(2007,5,31)),
           2008:(datetime.datetime(2008,9,1),datetime.datetime(2008,9,30)),
           2009:(datetime.datetime(2009,6,1),datetime.datetime(2009,6,30)),
           2010:(datetime.datetime(2010,2,1),datetime.datetime(2010,2,28))}

# exceptions to the above rules, based on looking closer at data
cexceptions = {'WIPP-11':{2006:(datetime.datetime(2006,8,1),
                                datetime.datetime(2006,8,31))},
               'H-10C':{2006:(datetime.datetime(2006,8,1),
                              datetime.datetime(2006,8,31))},
               'SNL-14':{2005:(None,None),
                         2007:(datetime.datetime(2007,11,1),
                               datetime.datetime(2007,11,30))},
               'SNL-16':{2007:(datetime.datetime(2007,9,1),
                               datetime.datetime(2007,9,30))},
               'WIPP-26':{2005:(None,None)},
                 'WIPP-30':{2005:(datetime.datetime(2005,8,1),
                                  datetime.datetime(2005,8,31))}}


newfwhfh = {}
newfwhfh[2005] = open('meas_head_2005ASER.smp','w')
newfwhfh[2006] = open('meas_head_2006ASER.smp','w')
newfwhfh[2007] = open('meas_head_2007ASER.smp','w')

# minimum density range for plot
denmax = 1.100
denmin = 1.000
denrange = denmax - denmin

#@#fhrpchange = open('reference-point-change-2007.dat','w')

# %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
# cycle over all Culebra wells, plotting figures for each
zone = 'CUL'
zmask = data['zone'] == zone
zdat = data[zmask]
for well in wells:
    wm = zdat['well'] == well
    if wm.sum() == 0:
        continue # no data for this well
    else:
        print 'processing',zone,well

    fig = plt.figure(1,figsize=(11,8.5))
    ax1 = fig.add_subplot(211)

    # plot reported (adjusted) WRES depth-to-water measurements (small red circles)
    ax1.plot_date(zdat[wm]['dt'],zdat[wm]['dtwm'],'ro',markersize=2,markeredgecolor='red')

    # invert depth-to-water axis (bigger numbers on bottom)
    ymin,ymax = ax1.get_ylim()
    ax1.set_ylim([ymax,ymin])

    pmask = zdat[wm]['fwhm'] > 0.0
    pdata = (zdat[wm])[pmask]

    # uncomment below and run once on data to generate corrections file
    #@#pre2007mask = pdata['dt'] < datetime.datetime(2007,1,1)
    #@#post2007mask = np.logical_not(pre2007mask)
    #@## only compute correction to reference point elevation if data straddles January/1/2007
    #@#if pre2007mask.sum() > 0 and post2007mask.sum() > 0:
    #@#    oldrpelev = (pdata[pre2007mask]['dtwm'] + pdata[pre2007mask]['wlem']).mean()
    #@#    newrpelev = (pdata[post2007mask]['dtwm'] + pdata[post2007mask]['wlem']).mean()
    #@#    earlyrpcorr = newrpelev - oldrpelev
    #@#else:
    #@#    earlyrpcorr = 0.0
    #@#fhrpchange.write('%s\t%.2f\n' % (well,earlyrpcorr))

    # plot freshwater heads on second y-axis
    ax2 = ax1.twinx()

    # plot freshwater head data as reported in ASER (small blue stars)
    ax2.plot_date(pdata[:]['dt'],pdata[:]['fwhm'],'b*',markersize=3,markeredgecolor='blue')

    # correct FWH for densities chosen as "correct" densities over a given time range
    fdmask = finden[:]['well'] == well
    fdata = finden[fdmask]

    for dval in fdata:
        # mask off ASER values in the date range associated with this density value
        datemask = np.logical_and(pdata['dt'] >= dval['dt0'],
                                  pdata['dt'] <= dval['dt1'])

        newfwhdates = pdata[datemask]['dt']

        # compute new fwh using consistent density
        newfwh = (pdata[datemask]['cwlem']-midpt[well])*dval['den'] + midpt[well]

        # plot consistent freshwater heads (large blue x's)
        ax2.plot_date(newfwhdates,newfwh,'bx',markersize=5)

        # save 2005, 2006, and 2007 fwh to file for use by PEST
        for yr in [2005,2006,2007]:
            exception = False
            skipwell = False
            if well in cexceptions:
                if yr in cexceptions[well]:
                    exception = True
                    if cexceptions[well][yr][0] == None:
                        # skip this well this year, even though there is data
                        skipwell = True
                    else:
                        # use a different month than the one selected for all other wells
                        maskyrfwh = np.logical_and(newfwhdates >= cexceptions[well][yr][0],
                                                   newfwhdates <= cexceptions[well][yr][1])
            if not exception:
                # use the standard month selected for all other wells
                maskyrfwh = np.logical_and(newfwhdates >= cmonths[yr][0],
                                           newfwhdates <= cmonths[yr][1])

            if maskyrfwh.sum() > 0 and not skipwell:
                # use the same name if this well was used before,
                # otherwise strip hyphen and go for it
                if well.upper() in aserfwh[2010]:
                    wname = aserfwh[2010][well]['name']
                elif well.upper() in aserfwh[2009]:
                    wname = aserfwh[2009][well]['name']
                elif well.upper() in aserfwh[2008]:
                    wname = aserfwh[2008][well]['name']
                else:
                    wname = well.replace('-0','-') # remove leading zero
                    wname = wname.replace('-','') # remove hyphen

                # if there are more than one fwh during that month, select the first one
                newfwhfh[yr].write('%s\t%s\t12:00:00\t%.3f\t%.4f\n' %
                                 (wname,newfwhdates[maskyrfwh][0].strftime("%m/%d/%Y"),
                                  newfwh[maskyrfwh][0],dval['den']))

    ax1.xaxis.set_major_formatter( NullFormatter() )
    ax1.set_title('%s water levels and specific gravities' % well)
    ax2.xaxis.set_major_locator( YearLocator() )
    ax2.xaxis.set_minor_locator( MonthLocator() )
    ax2.xaxis.set_major_formatter( DateFormatter('%Y') )
    ax1.set_ylabel('ASER depth to water (m BTOC)',color='red',fontsize=9)

    ax2.set_ylabel('freshwater head (ASER=*, this rept=x) (m AMSL)',
                   color='blue',fontsize=9)

    # add date ranges used for contouring
    for yr in cmonths.keys():
        exception = False
        if well in cexceptions:
            if yr in cexceptions[well]:
                exception = True
                if cexceptions[well][yr][0] == None:
                    # don't use this well this year
                    continue
                else:
                    # use a different month for this well this year
                    ax2.axvspan(cexceptions[well][yr][0],cexceptions[well][yr][1],
                                alpha=0.25,color='green')
        if not exception:
            # use the standard month for this well and year
            ax2.axvspan(cmonths[yr][0],cmonths[yr][1],alpha=0.25,color='blue')
    axd = fig.add_subplot(212)

    # aparent specific gravity is (fwh_el - midpt_el)/(wl_el - midpt_el)
    # computed from wl elevation and freshwater head  reported by WRERS +
    # Culebra midpoint elevations (small green circles)
    specgrav = (((zdat[wm])[pmask]['fwhm'] - midpt[well])/
                ((zdat[wm])[pmask]['wlem'] - midpt[well]))
    axd.plot_date((zdat[wm])[pmask]['dt'],specgrav,'go',
                  markersize=2,markeredgecolor='green')

    fdmask = finden[:]['well'] == well
    fdata = finden[fdmask]
    for dval in fdata:
        # plot the "final" gravity as a line across the figure (solid green line)
        axd.plot_date([dval['dt0'],dval['dt1']],[dval['den'],dval['den']],'g-')

    # plot reported density values; different symbols
    dm = den['well'] == well
    ddat = den[dm]
    setsrc = set(ddat[:]['src'])
    sources = list(setsrc)

    # change TROLLYYYY to just TROLL
    for src in sources:
        if 'TROLL' in src:
            setsrc.remove(src)
            setsrc.add('TROLL')
        if 'PDS' in src:
            setsrc.remove(src)
            setsrc.add('PDS')
        if 'SGR' in src:
            setsrc.remove(src)
            setsrc.add('SGR')

    sources = list(setsrc)
    for src in sources:
        # densities from this sources (all TROLLYYY just count as TROLL, etc.)
        mddat = npchar.find(ddat[:]['src'],src) >= 0 # find returns -1 if not found
        sddat = ddat[mddat]

        # densities since cutoffdate
        msddat = sddat['dt'] > firstplot
        dsddat = sddat[msddat]
        nrecent = msddat.sum()

        if src == 'TROLL':
            if nrecent > 0:
                tmpden = sddat[msddat]['den']
                # move any troll-computed densities below 1.0 to fresh water (1.0)
                tmpden[tmpden < 1.0] = 1.0
                # troll densities plot as a date range
                axd.plot_date([sddat[msddat]['dt'],sddat[msddat]['dt2']],[tmpden,tmpden],
                              srcsymb[src],linestyle='solid',color=srccolor[src],
                              linewidth=2.0,label=src,markersize=9)

        else:
            if nrecent > 0:
                tmpden = sddat[msddat]['den']
                # move any densities below 1.0 to fresh water (1.0)
                tmpden[tmpden < 1.0] = 1.0
                # non-troll densities plot as a single date
                axd.plot_date(sddat[msddat]['dt'],tmpden,srcsymb[src],
                              color=srccolor[src],label=src,markersize=9)

        # densities before cutoff date (plot on edge with left-pointing triangle)
        msddat = sddat['dt'] < firstplot
        nold = msddat.sum()

        if nold > 0:
            axd.plot_date([firstplot]*nold,sddat[msddat]['den'],'<',
                          color='yellow',label='pre-2005',markersize=9)

    # make range of density plots at least a minimum range
    ymin,ymax = axd.get_ylim()
    if ymax < denmax:
        ymax = denmax

    if ymin > denmin:
        ymin = denmin

    axd.set_ylim((ymin,ymax))

    # add pumping,drilling,and pluggin events located at the current well
    em = events[:]['well'] == well
    if em.sum() > 0:
        ee = events[em]
        ymin,ymax = axd.get_ylim()
        yann = ymax - (ymax - ymin)/5.0
        for ev in ee:
            if ev['dt1'] == None:
                # no ending date
                axd.axvline(ev['dt0'],alpha=0.5,color=ev['c'])
                axd.annotate(ev['s'],(ev['dt0'],yann),
                             rotation='vertical',fontproperties=AnnFontP)
            else:
                axd.axvspan(ev['dt0'],ev['dt1'],alpha=0.25,color=ev['c'])
                axd.annotate(ev['s'],(ev['dt0']+(ev['dt1']-ev['dt0'])//2,yann),
                             rotation='vertical',fontproperties=AnnFontP)


    # add drilling and P&A events for wells within 500 m (i.e., same pad)
    for other in evwells:
        if not other == well:
            em = events[:]['well'] == other
            ee = events[em]
            om = np.logical_or(ee[:]['c'] == 'red',ee[:]['c'] == 'green')
            dist  = np.sqrt((xyz[well]['x']-xyz[other]['x'])**2 +
                            (xyz[well]['y']-xyz[other]['y'])**2)
            # does other well have drilling or p&a activities?
            if om.sum() > 0:
                if dist < 500:
                    for ev in ee[om]:
                        if ev['dt1'] == None:
                            # no ending date
                            axd.axvline(ev['dt0'],alpha=0.5,
                                        linestyle='dashed',color=ev['c'])
                        else:
                            axd.axvspan(ev['dt0'],ev['dt1'],alpha=0.25,
                                        linestyle='dashed',color=ev['c'])
                        axd.annotate('%s (%im) %s' % (ev['well'],dist,ev['s']),(ev['dt0'],yann)
                                     rotation='vertical',fontproperties=AnnFontP,color='gray')

            # nearby pumping tests (not slug tests) within 5.0 km
            # npchar.find() returns -1 for not found
            om = np.logical_and(ee[:]['c'] == 'cyan',
                                npchar.find(ee[:]['s'],'slug') == -1)
            if om.sum() > 0:
                if dist < 5000:
                    for ev in ee[om]:
                        if ev['dt1'] == None:
                            # no ending date
                            axd.axvline(ev['dt0'],alpha=0.5,
                                        linestyle='dashed',color=ev['c'])
                        else:
                            axd.axvspan(ev['dt0'],ev['dt1'],alpha=0.25,
                                        linestyle='dashed',color=ev['c'])
                        axd.annotate('%s (%.1fkm) %s' % (ev['well'],dist/1000.0,ev['s']),
                                     (ev['dt0'],yann),rotation='vertical',
                                     fontproperties=AnnFontP,color='gray')


        ax2.set_xlim(left=datetime.datetime(2004,11,1))
        ax2.set_xlim(right=datetime.datetime(2011,1,1))

        # force subplots to have same data range
        axd.set_xlim(ax2.get_xlim())
        axd.set_ylabel('specific gravity')

        # legend for type of density measurement
        # while removing duplicate entries from legend
        handles,labels = axd.get_legend_handles_labels()
        newhandles = []
        newlabels = []
        if len(handles) > 0:
            for h,l in zip(handles,labels):
                if not l in newlabels:
                    newhandles.append(h)
                    newlabels.append(l)

        leg = axd.legend(newhandles,newlabels,loc=0,prop=LegFontP,numpoints=1,scatterpoints=1)

        axd.xaxis.set_major_locator( YearLocator() )
        axd.xaxis.set_minor_locator( MonthLocator() )
        axd.xaxis.set_major_formatter( DateFormatter('%Y') )
    plt.savefig('%s-%s-ASER-waterlevels.png' % (zone,well),dpi=150)
    plt.close(1)

newfwhfh[2005].close()
newfwhfh[2006].close()
#@#rhrpchange.close()
```

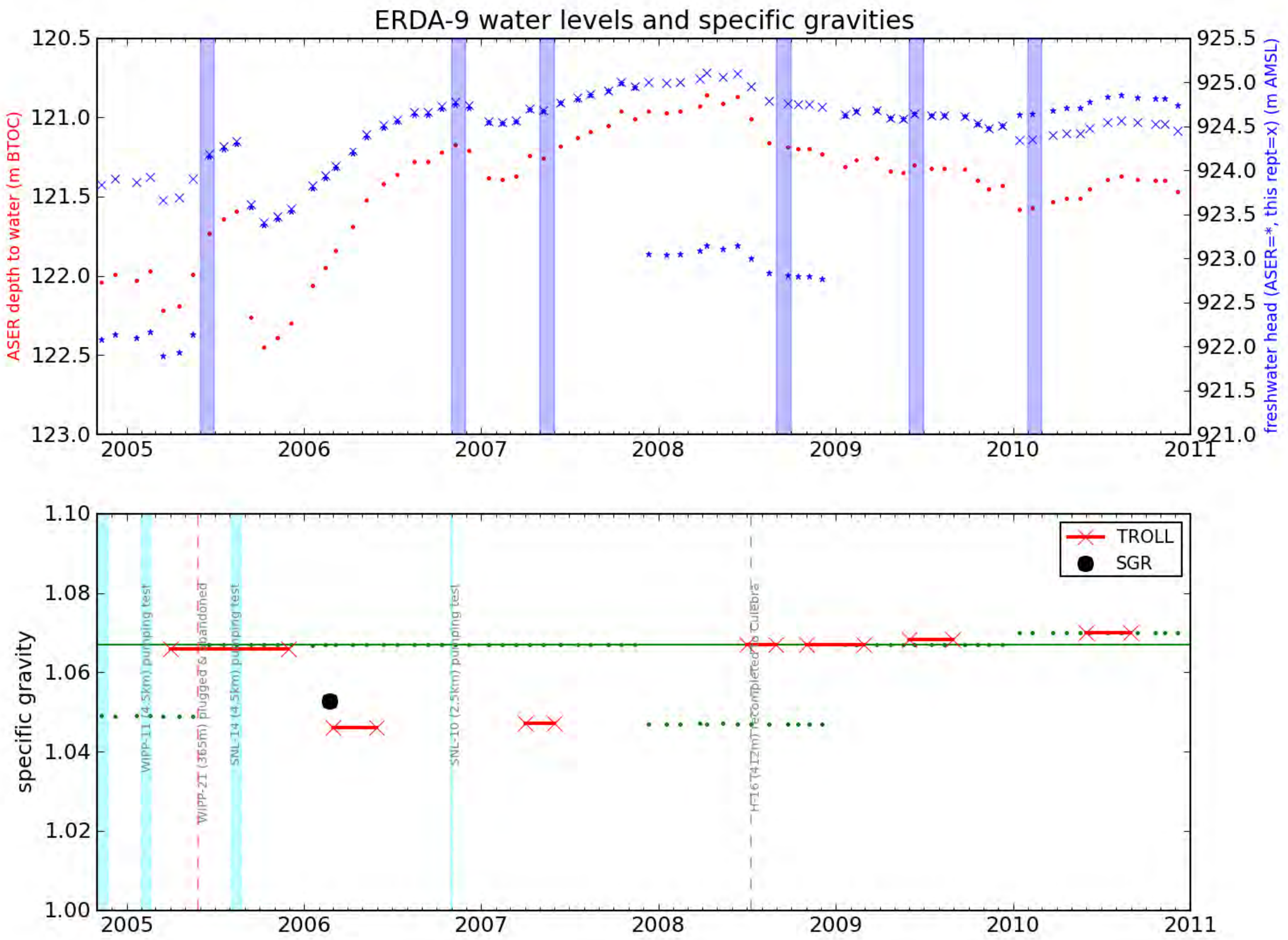
9.3 Figures Generated by Python Water Level Script

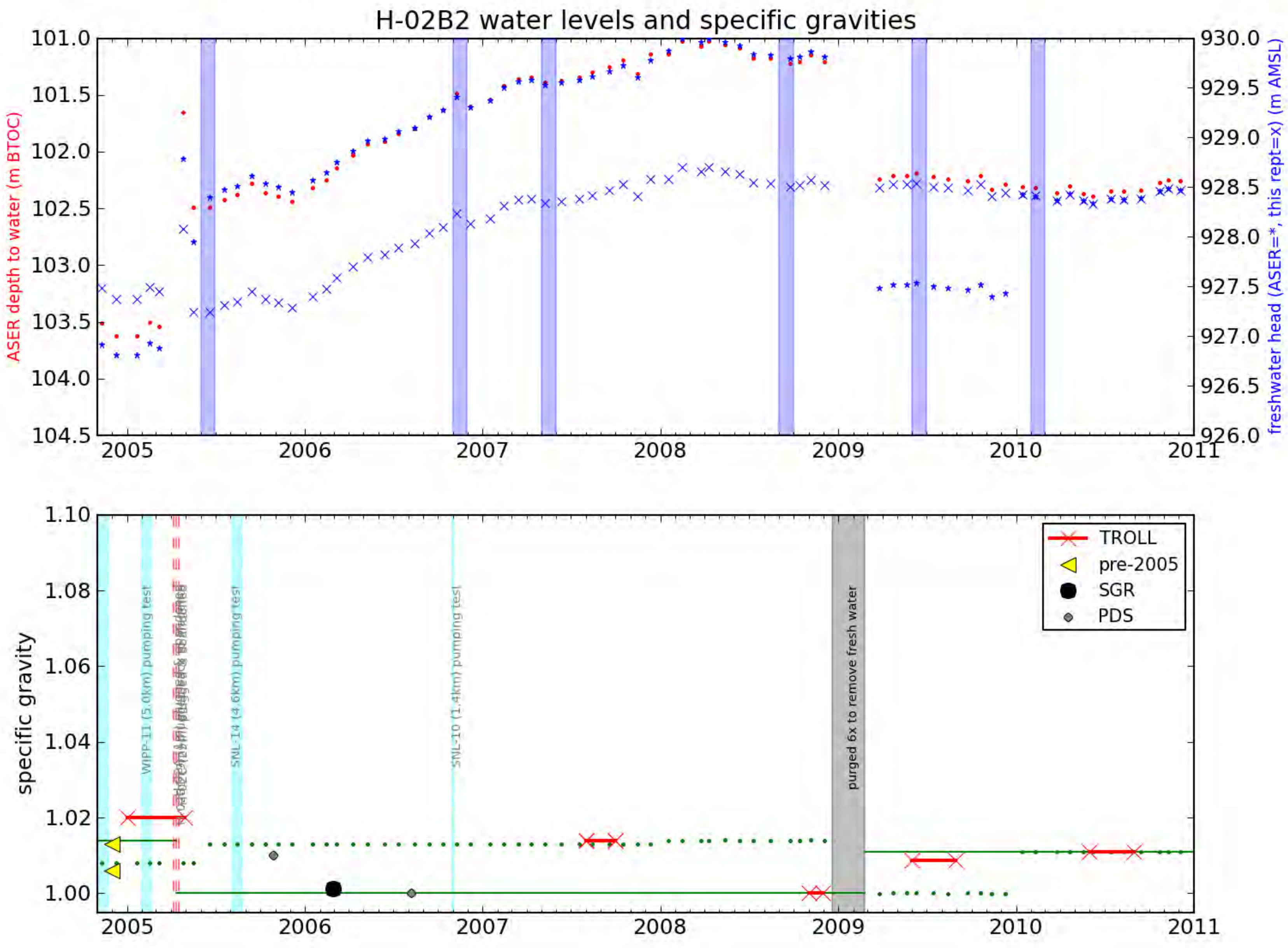
The following figures were generated by the Python script `plot-waterlevels.py` and represent the water level, density (aka specific gravity), and well-event data listed in the 2004-2010 ASERs.

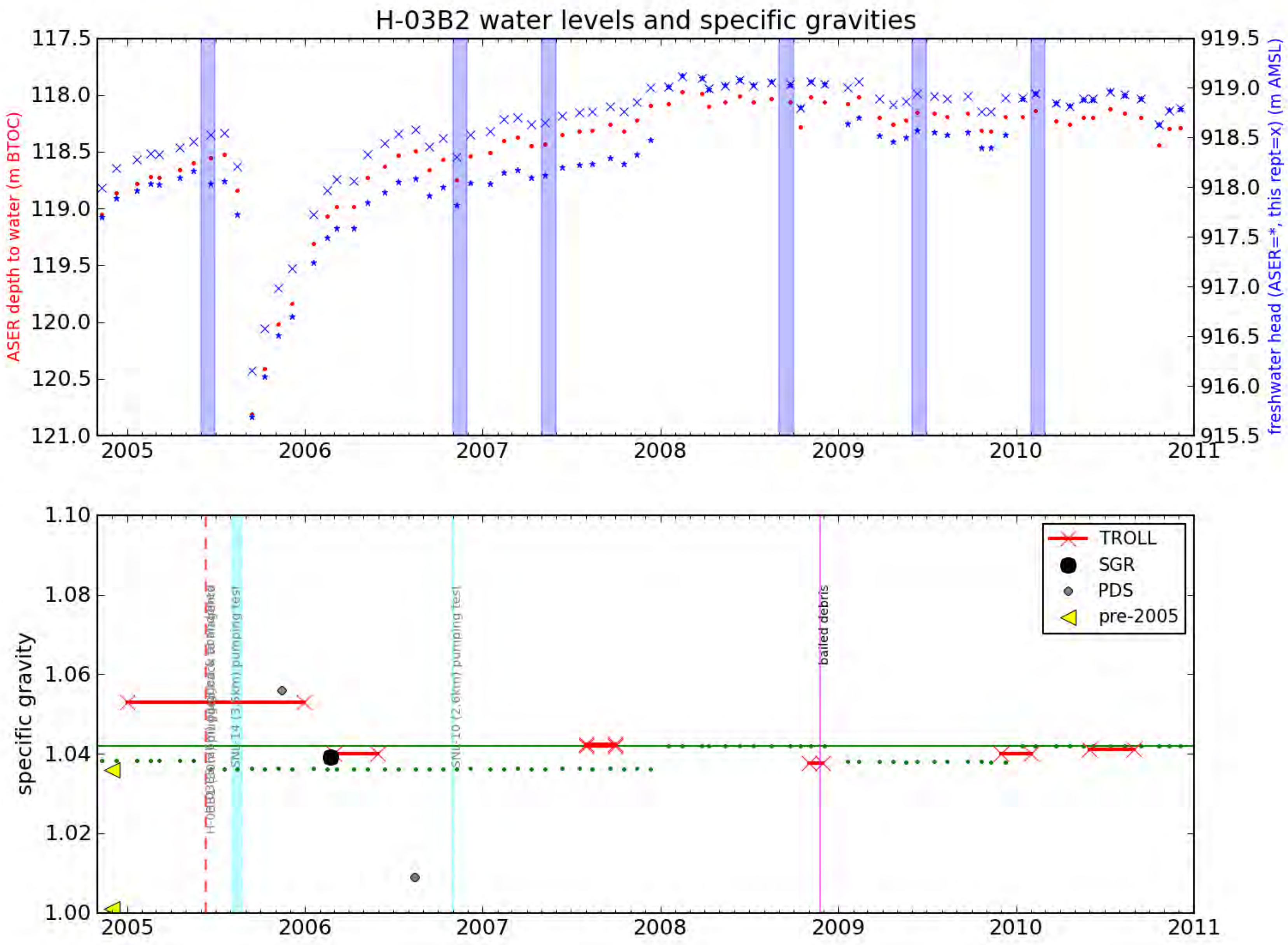
Each page represents the 2005-current ASER data for a given Culebra well. In the water level plots (top), filled red circles are reported depths to water (meters below top of casing (BTOC)), filled blue stars are ASER-reported freshwater head elevations (meters above mean sea level (AMSL)), and the blue $\times$'s are the freshwater head elevations (AMSL) computed using the density values recommended in the file `densities-to-use.csv`. Adjustments to pre-2007 water level elevations to use better-surveyed modern reference point elevations are reflected in the re-computed freshwater heads (blue $\times$'s), but not in the ASER-reported freshwater heads (blue stars). Vertical bands indicate the months that were used for contouring heads.

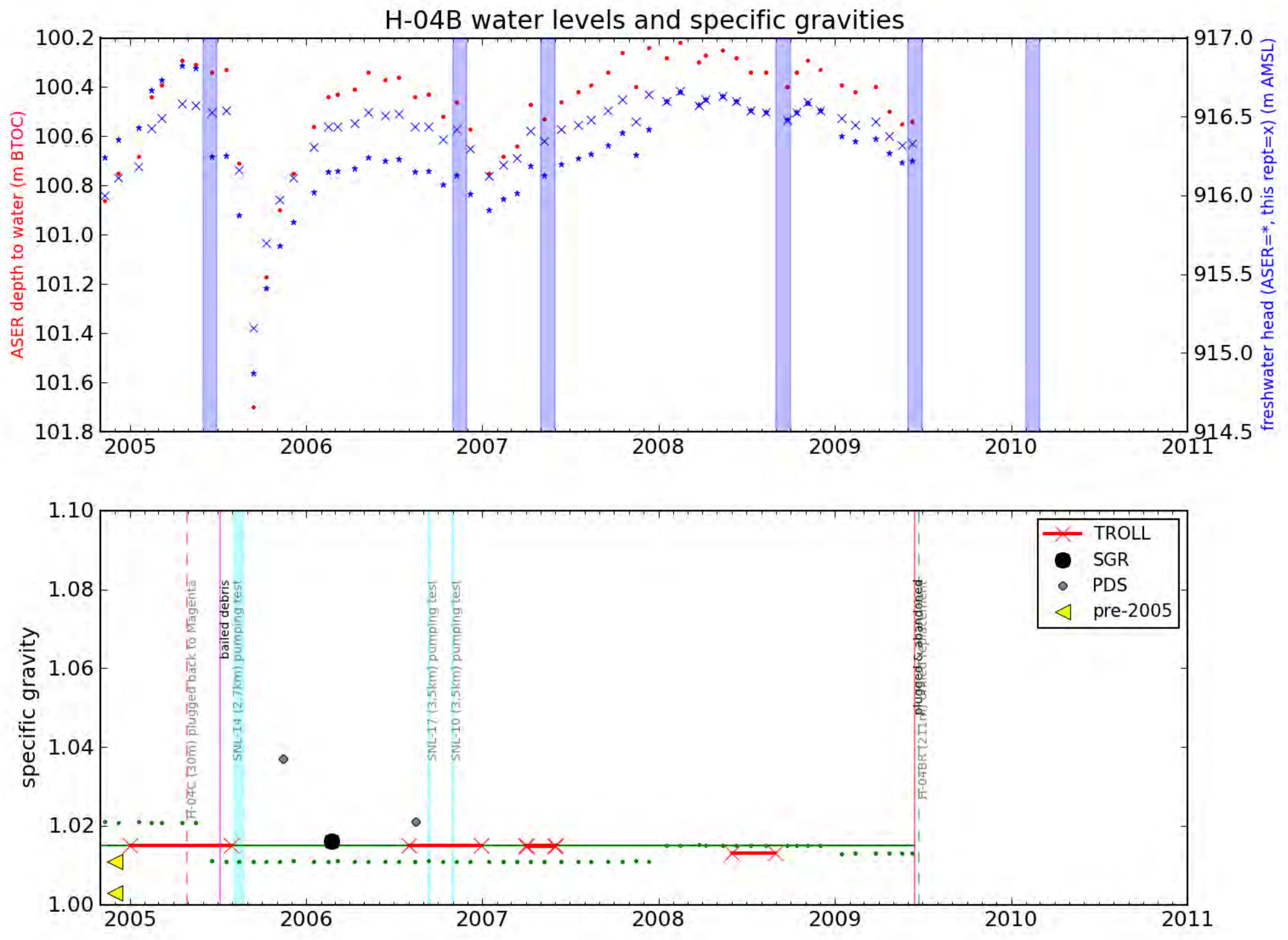
In the density plots (bottom), horizontal green lines indicate the density/specific gravity values chosen to be used at a given time (`densities-to-use.csv`), vertical lines are events in current or nearby wells (nearby wells are gray text and indicate the distance between the wells, while the current well is in black text). Red $\times$'s with connecting red lines are density values computed from Troll data (representing the date range used to compute the density), gray circles indicate reported pressure-density surveys (PDS), large black circles are field specific gravity readings (SGR), and small green dots are densities back-calculated from the freshwater head elevations and water level elevations reported in the ASERs.

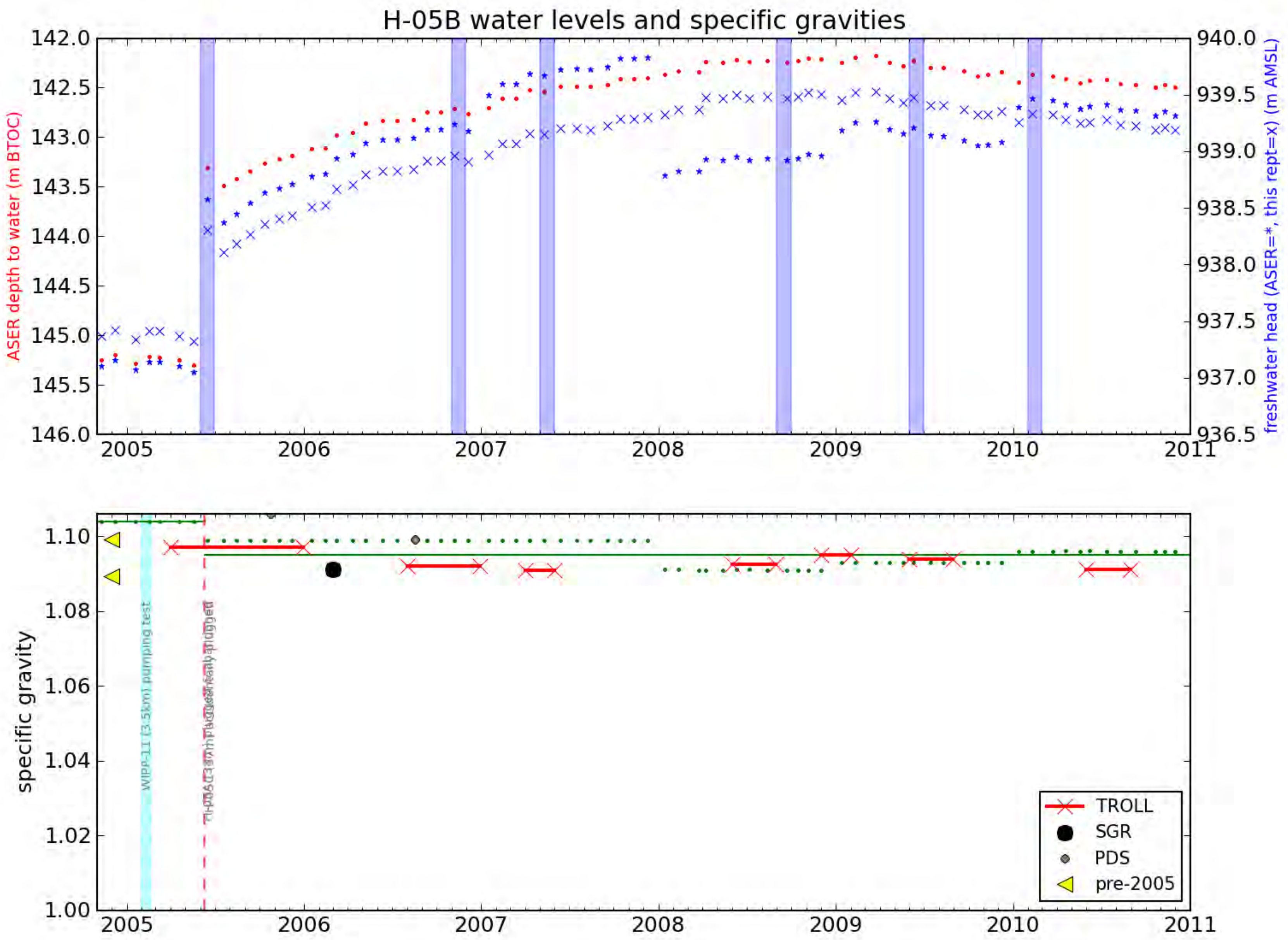

C-2737 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
specific gravity
TROLL
SGR
pre-2005
PDS
tubing swabbed was too fresh
reset PIP following testing
configured for Magenta slug test
SNL-10 (2.4km) pumping test
SNL-14 (4.0km) pumping test
2005
2006
2007
2008
2009
2010
2011
114
115
116
117
118
119
120
922
921
920
919
918
917
1.10
1.08
1.06
1.04
1.02
1.00

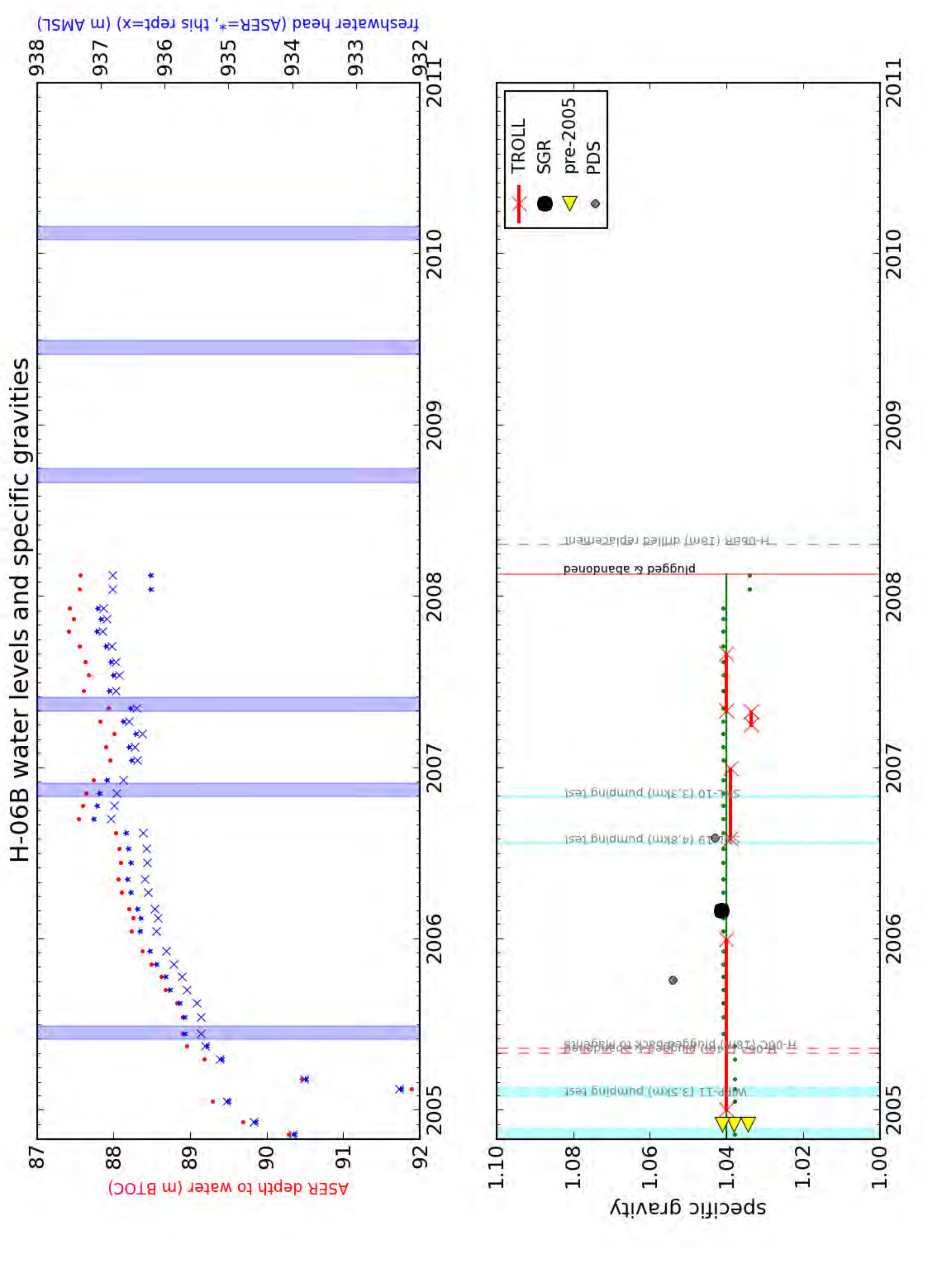

ERDA-9 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
120.5
121.0
121.5
122.0
122.5
123.0
925.5
925.0
924.5
924.0
923.5
923.0
922.5
922.0
921.5
921.0
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
TROLL
SGR
WIPP-11 (4.5km) pumping test
WIPP-21 (365m) plugged & abandoned
SNL-14 (4.5km) pumping test
SNL-10 (2.5km) pumping test
H-16 (4.12m) recompleted to Culebra

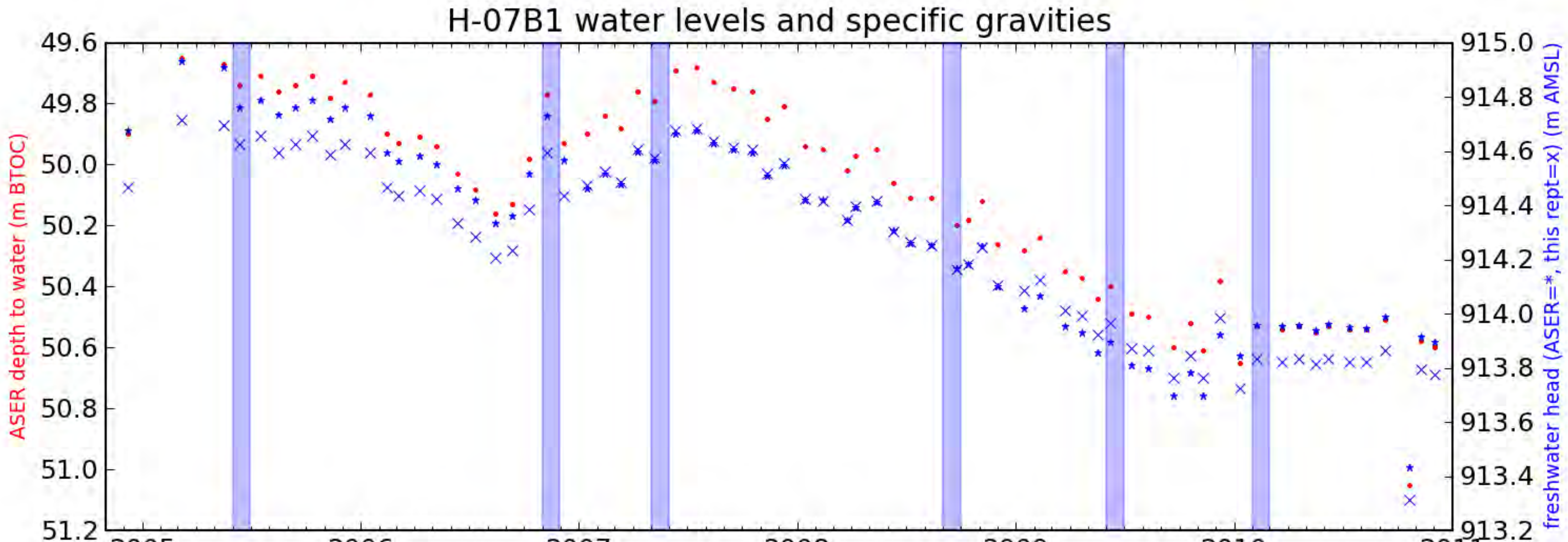

H-02B2 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
101.0
101.5
102.0
102.5
103.0
103.5
104.0
104.5
930.0
929.5
929.0
928.5
928.0
927.5
927.0
926.5
926.0
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
WIPP-11 (5.0km) pumping test
SNL-14 (4.6km) pumping test
SNL-10 (1.4km) pumping test
purged 6x to remove fresh water
TROLL
pre-2005
SGR
PDS

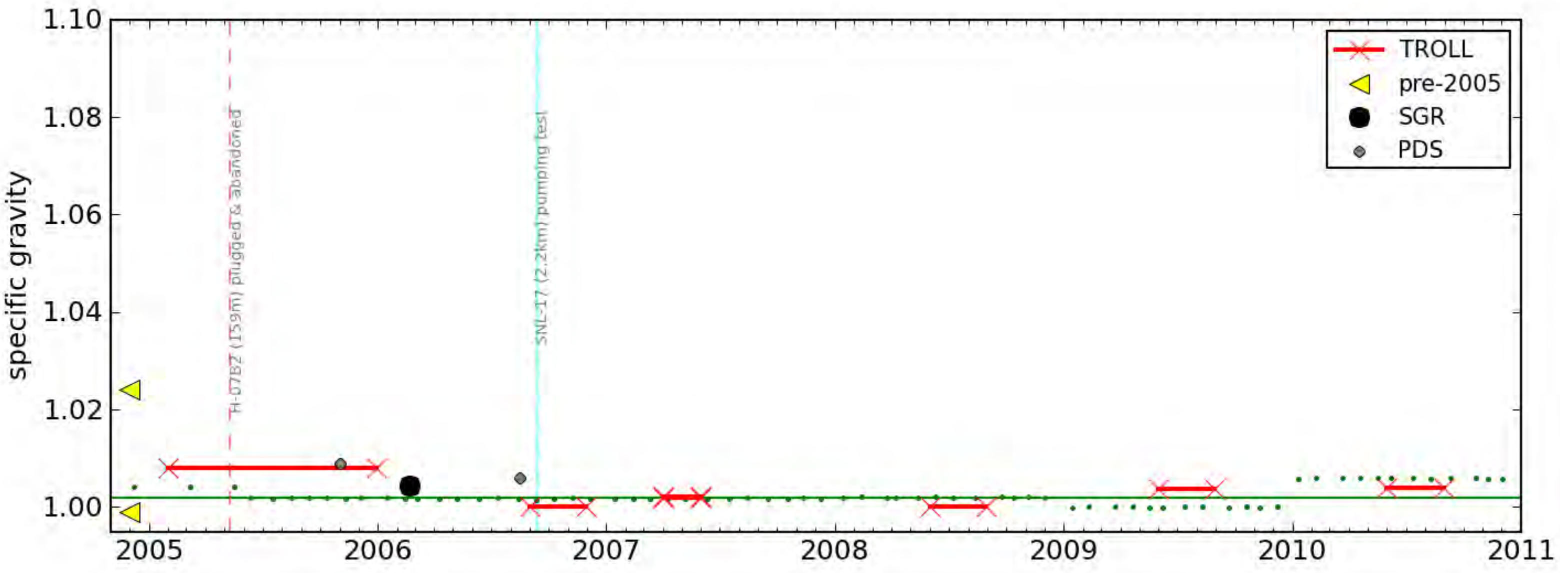

H-03B2 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
117.5
118.0
118.5
119.0
119.5
120.0
120.5
121.0
919.5
919.0
918.5
918.0
917.5
917.0
916.5
916.0
915.5
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
SNL-14 (3.5km) pumping test
SNL-10 (2.6km) pumping test
bailed debris
TROLL
SGR
PDS
pre-2005

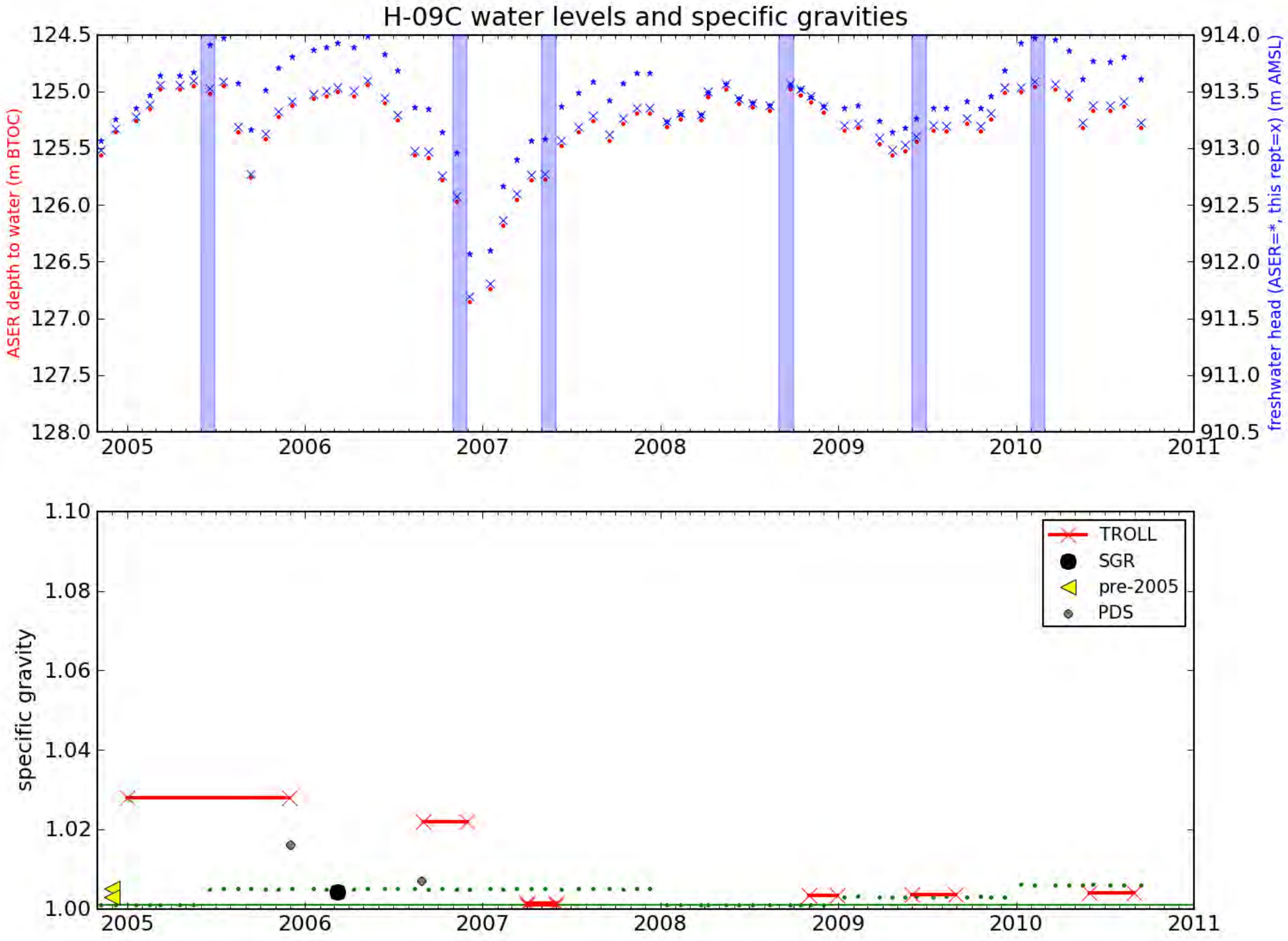

H-04B water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
100.2
100.4
100.6
100.8
101.0
101.2
101.4
101.6
101.8
917.0
916.5
916.0
915.5
915.0
914.5
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
H-04C (30m) plugged back to Magenta
bailed debris
SNL-14 (2.7km) pumping test
SNL-17 (3.5km) pumping test
SNL-10 (3.5km) pumping test
plugged & abandoned
H-04BR (211m) drilled replacement
TROLL
SGR
PDS
pre-2005

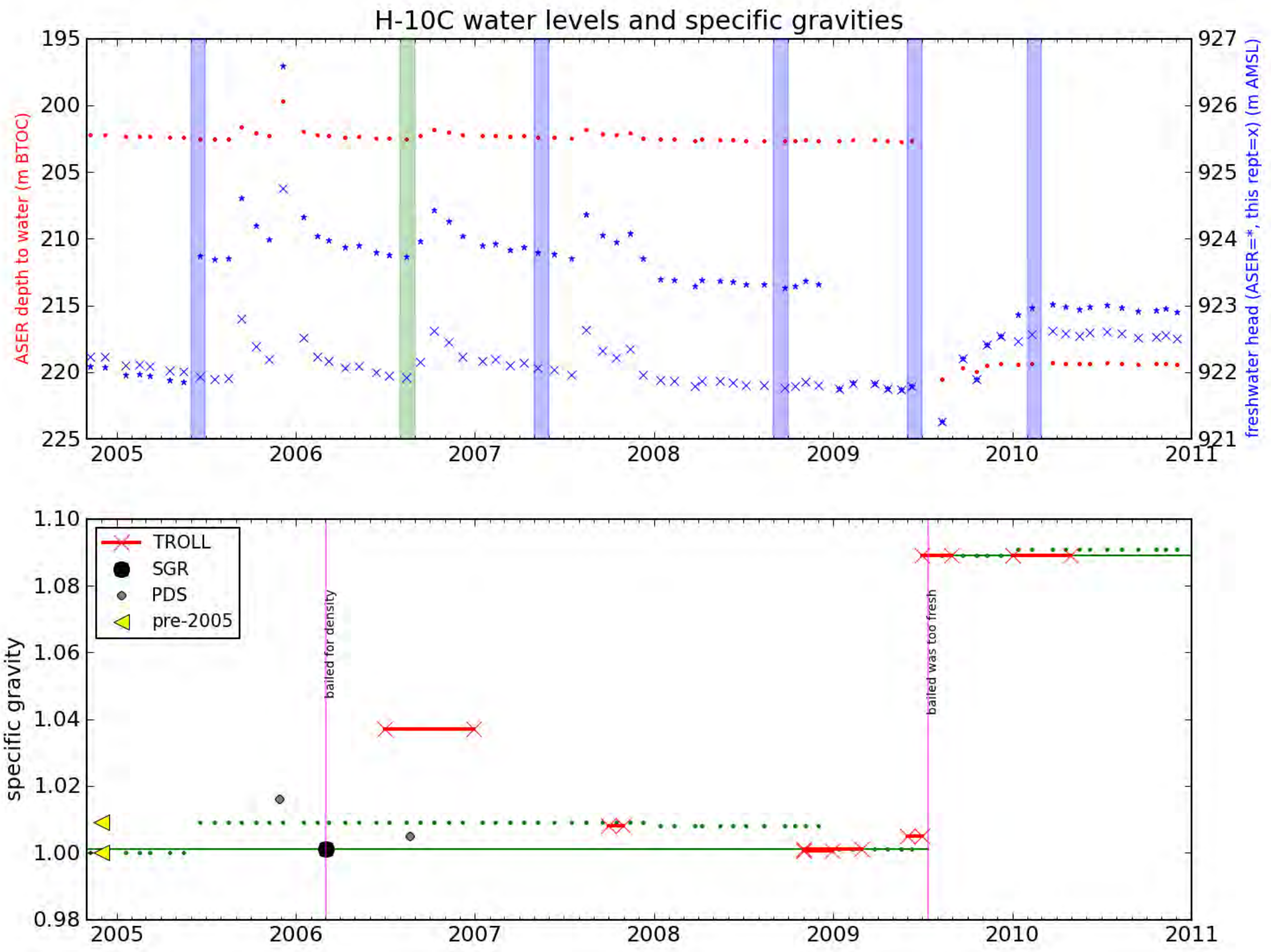

H-05B water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
142.0
142.5
143.0
143.5
144.0
144.5
145.0
145.5
146.0
940.0
939.5
939.0
938.5
938.0
937.5
937.0
936.5
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
WIPP-11 (3.5km) pumping test
H-05C3 ... plugged
TROLL
SGR
PDS
pre-2005

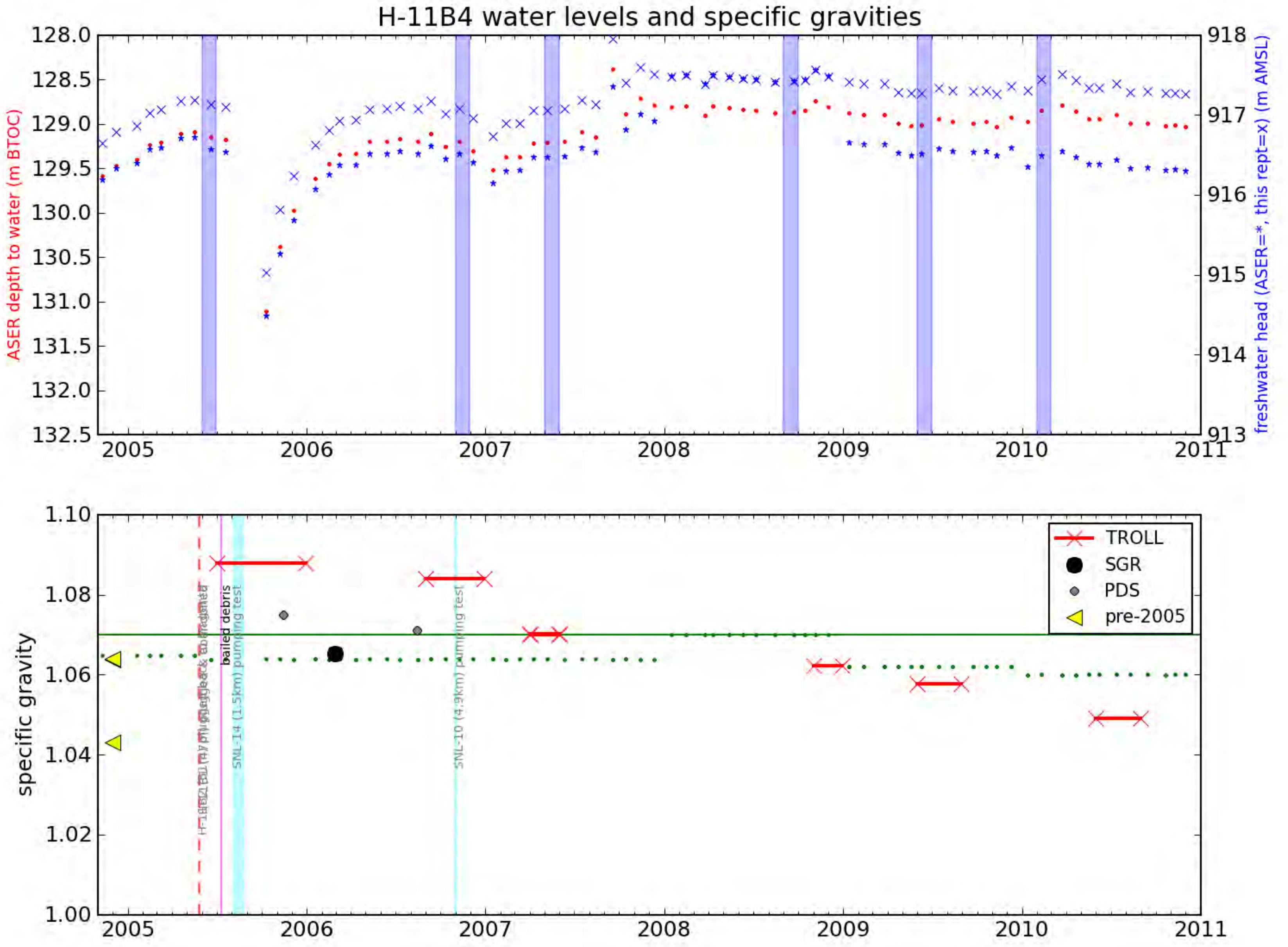

H-06B water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
specific gravity
TROLL
SGR
pre-2005
PDS
plugged & abandoned
H-06bR (18m) drilled replacement
SNL-10 (3.3km) pumping test
SNL-19 (4.8km) pumping test
H-06a (?m) plugged & abandoned
H-06c (18m) plugged back to Magenta
WIPP-11 (3.5km) pumping test
2005
2006
2007
2008
2009
2010
2011

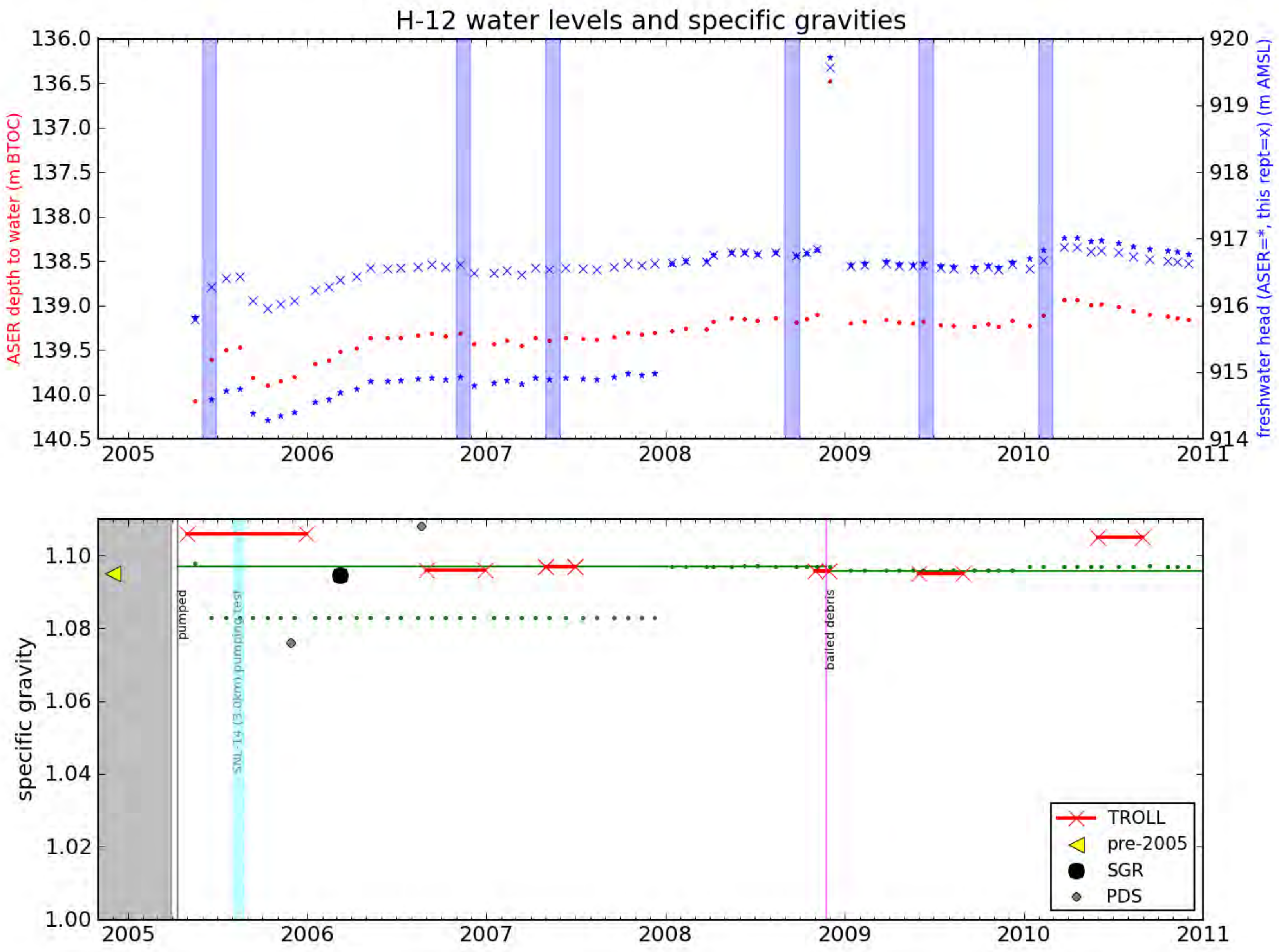

H-07B1 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
49.6
49.8
50.0
50.2
50.4
50.6
50.8
51.0
51.2
915.0
914.8
914.6
914.4
914.2
914.0
913.8
913.6
913.4
913.2
2005
2006
2007
2008
2009
2010
2011


specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
2005
2006
2007
2008
2009
2010
2011
H-07B2 (159m) plugged & abandoned
SNL-17 (2.2km) pumping test
TROLL
pre-2005
SGR
PDS

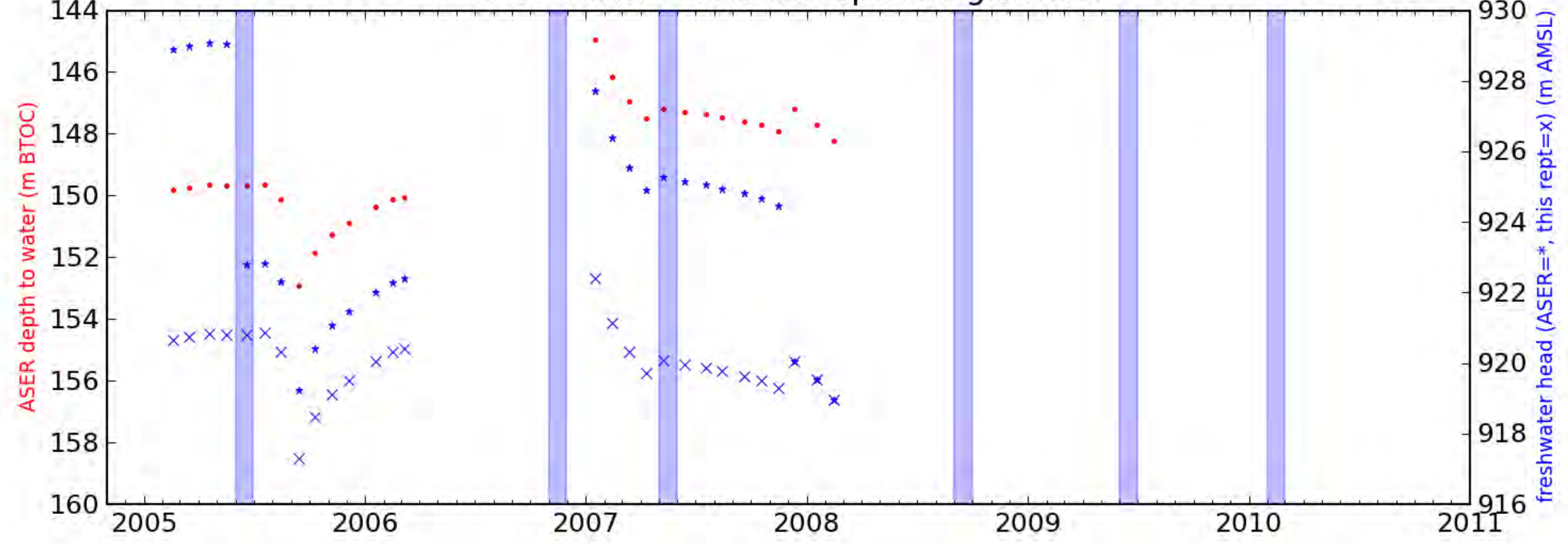

H-09C water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
124.5
125.0
125.5
126.0
126.5
127.0
127.5
128.0
914.0
913.5
913.0
912.5
912.0
911.5
911.0
910.5
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
TROLL
SGR
pre-2005
PDS

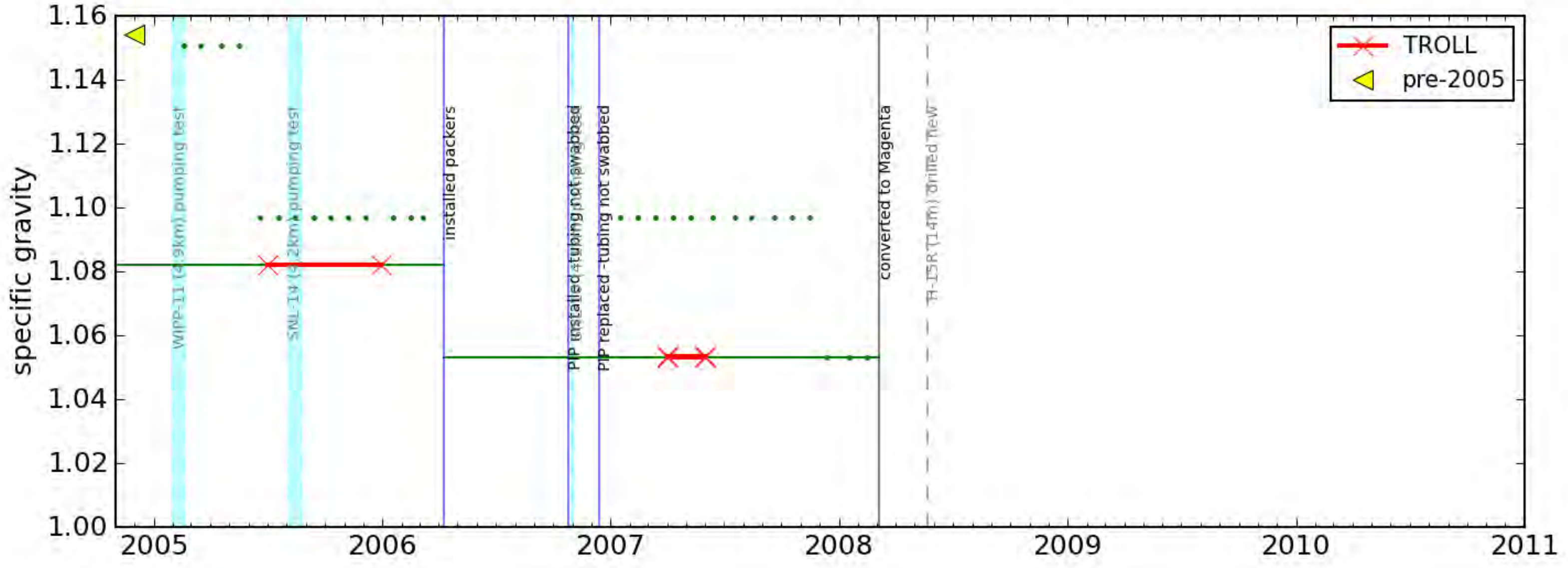

H-10C water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
195
200
205
210
215
220
225
927
926
925
924
923
922
921
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
0.98
TROLL
SGR
PDS
pre-2005
bailed for density
bailed was too fresh

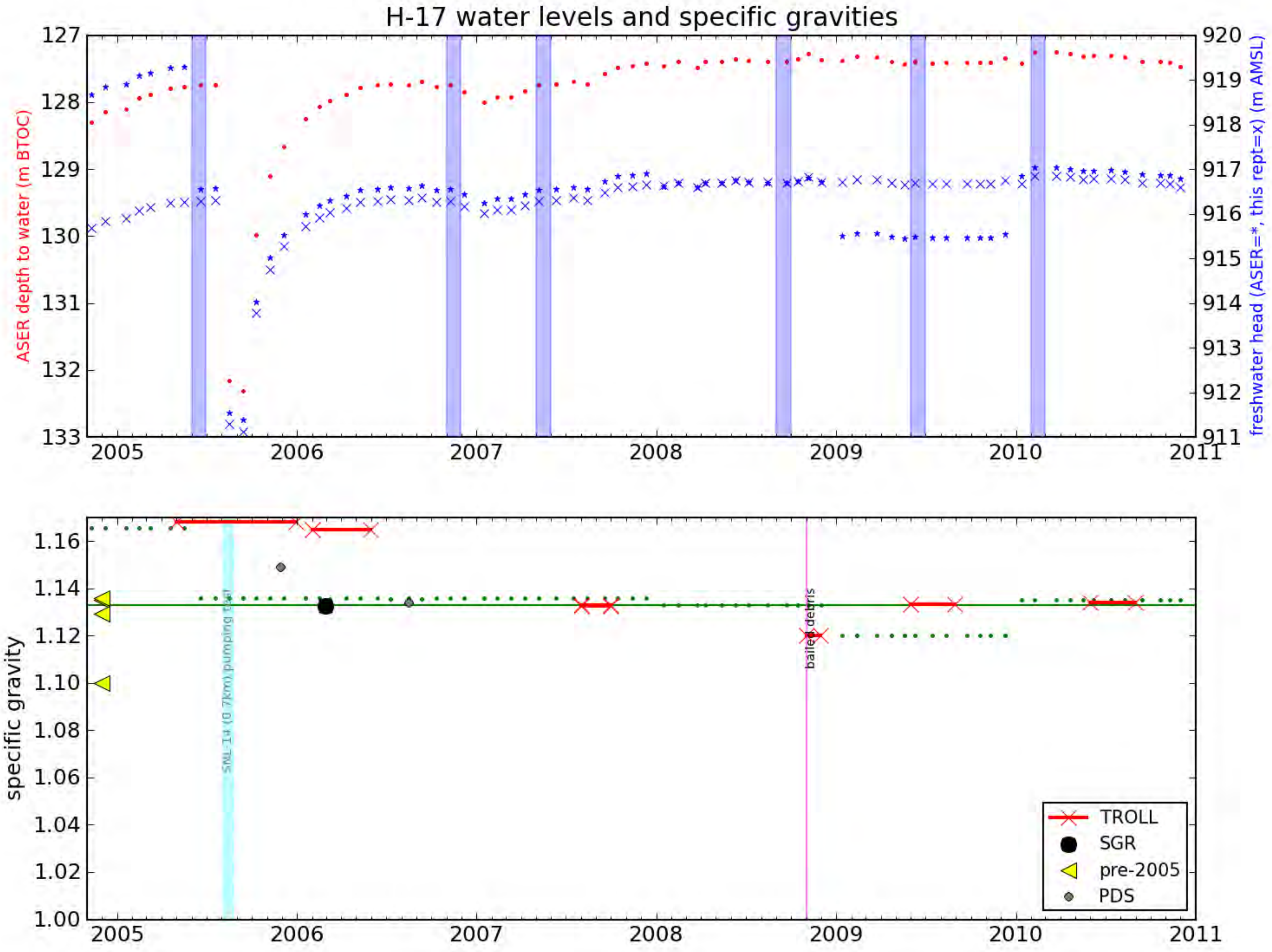

H-11B4 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
128.0
128.5
129.0
129.5
130.0
130.5
131.0
131.5
132.0
132.5
918
917
916
915
914
913
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
TROLL
SGR
PDS
pre-2005
bailed debris
SNL-14 (1.5km) pumping test
SNL-10 (4.9km) pumping test

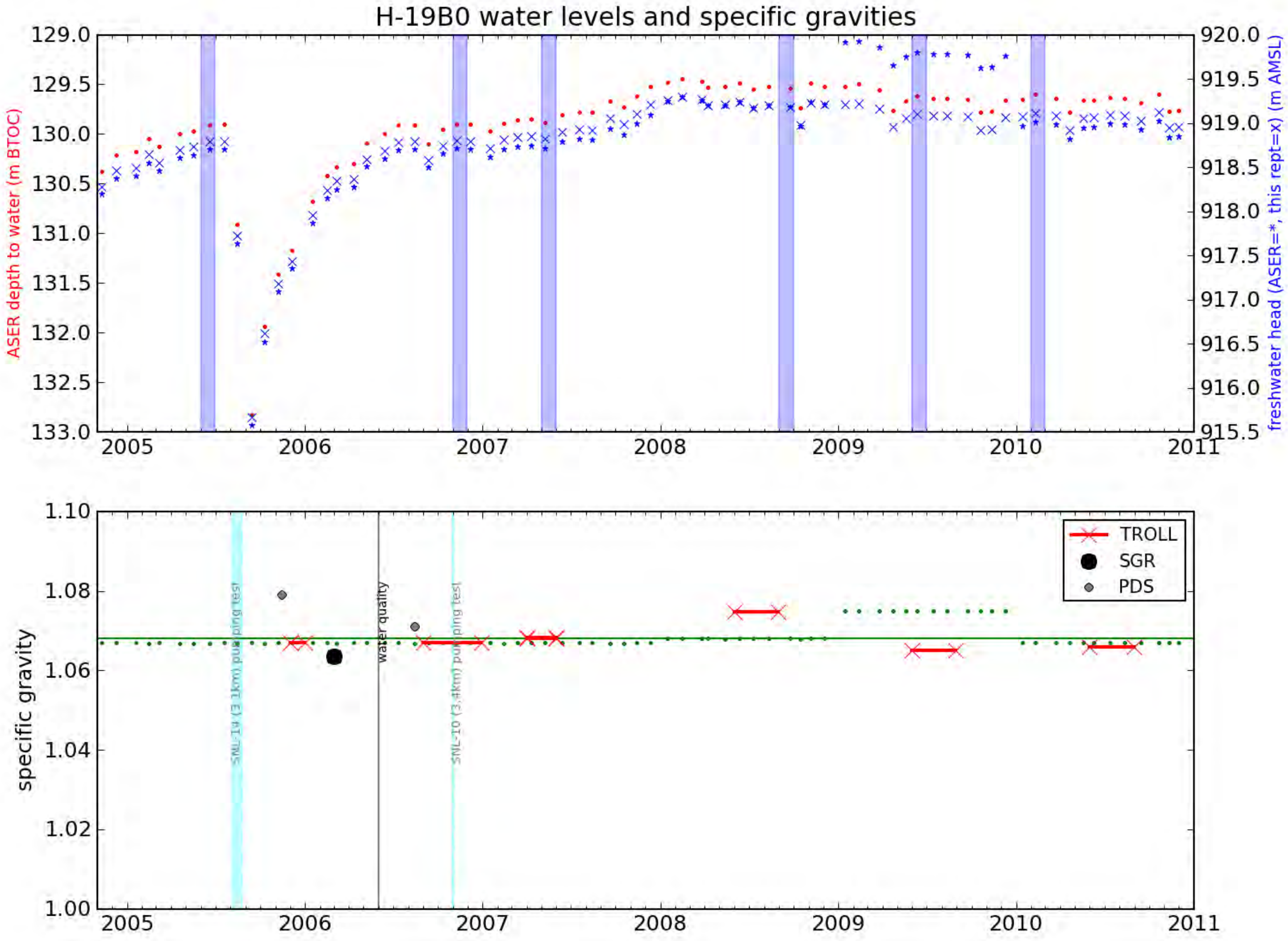

H-12 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
136.0
136.5
137.0
137.5
138.0
138.5
139.0
139.5
140.0
140.5
920
919
918
917
916
915
914
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
pumped
SNL-14 (3.0km) pumping test
bailed debris
TROLL
pre-2005
SGR
PDS

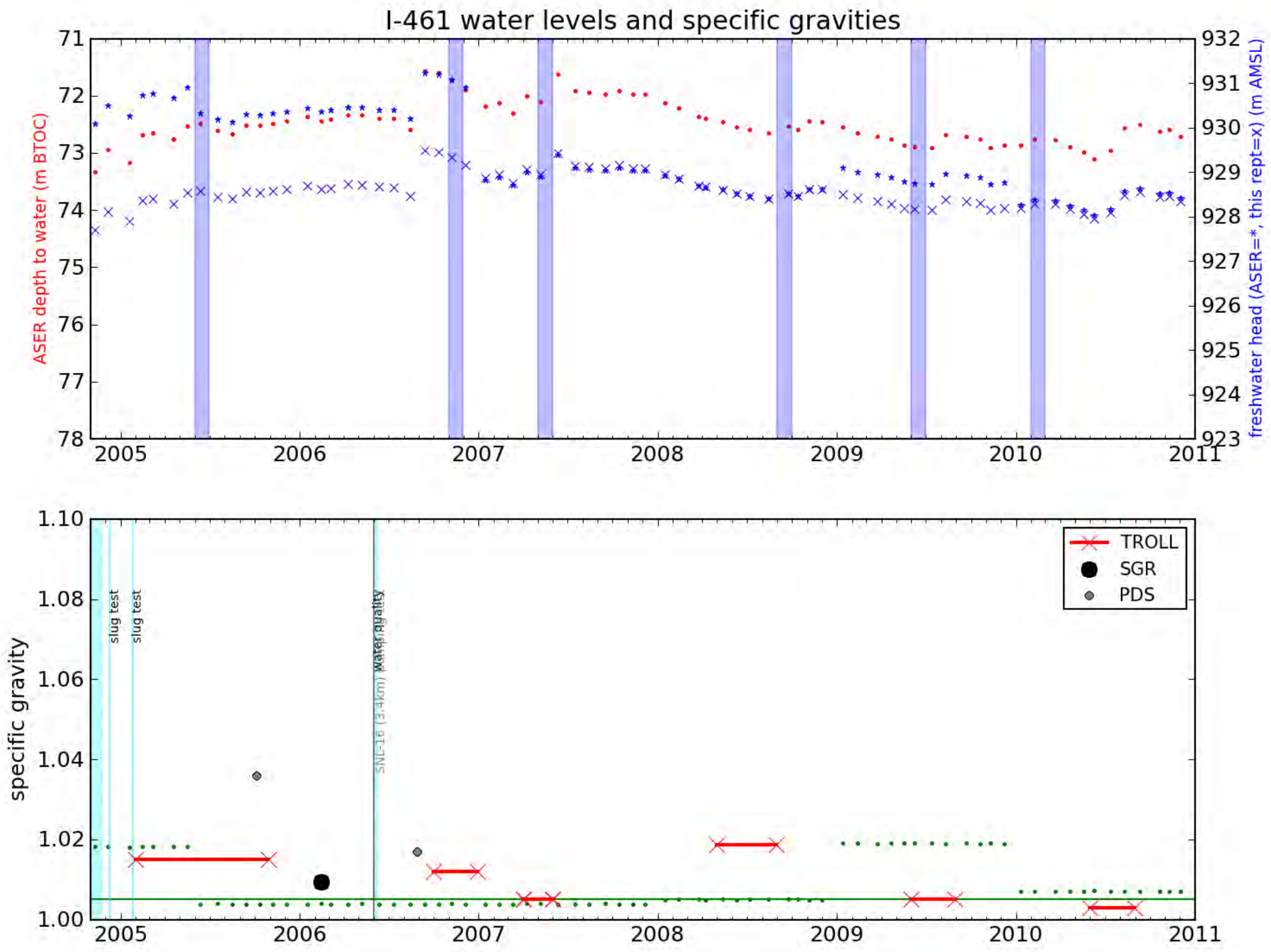
H-15 water levels and specific gravities

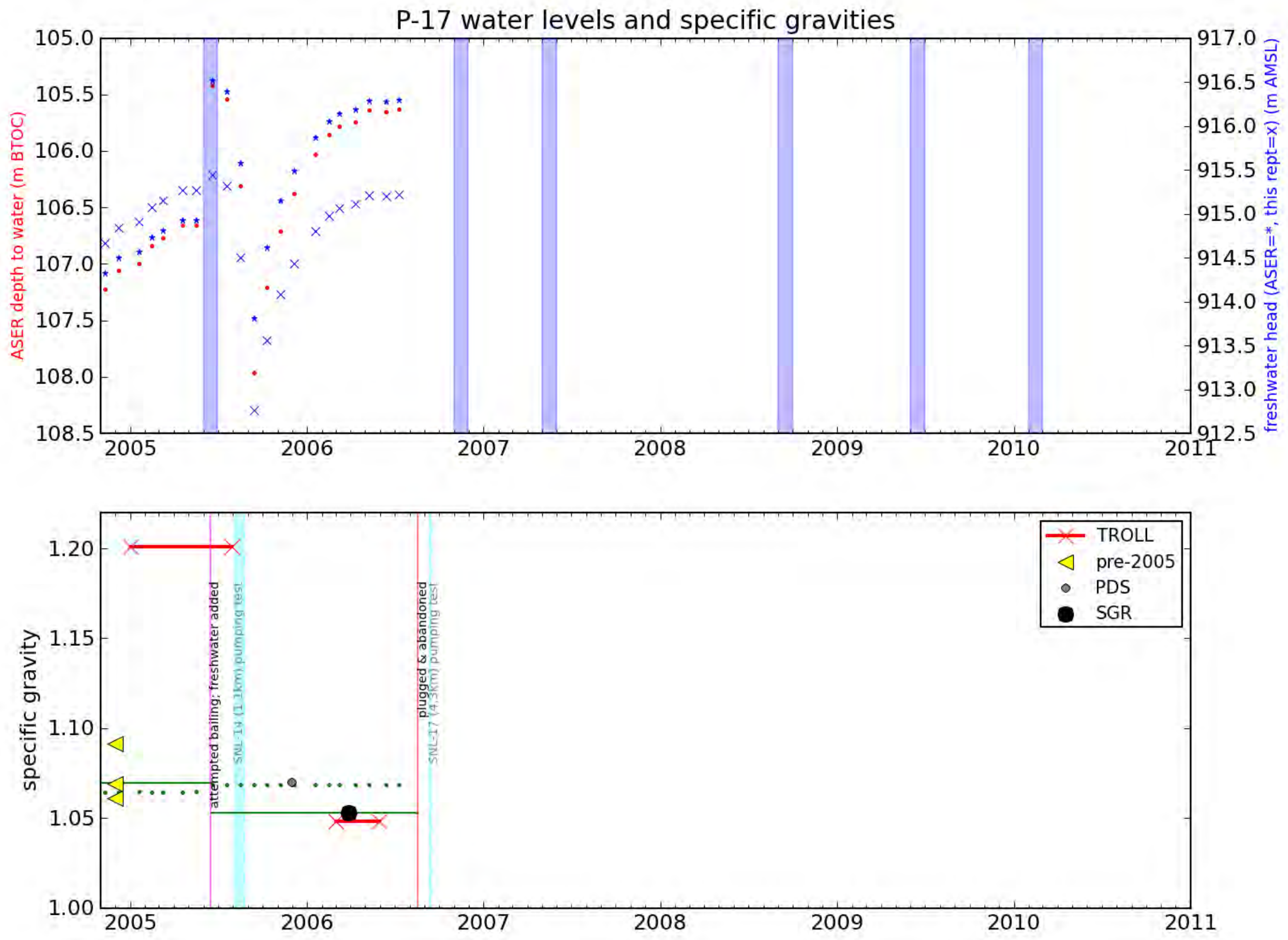


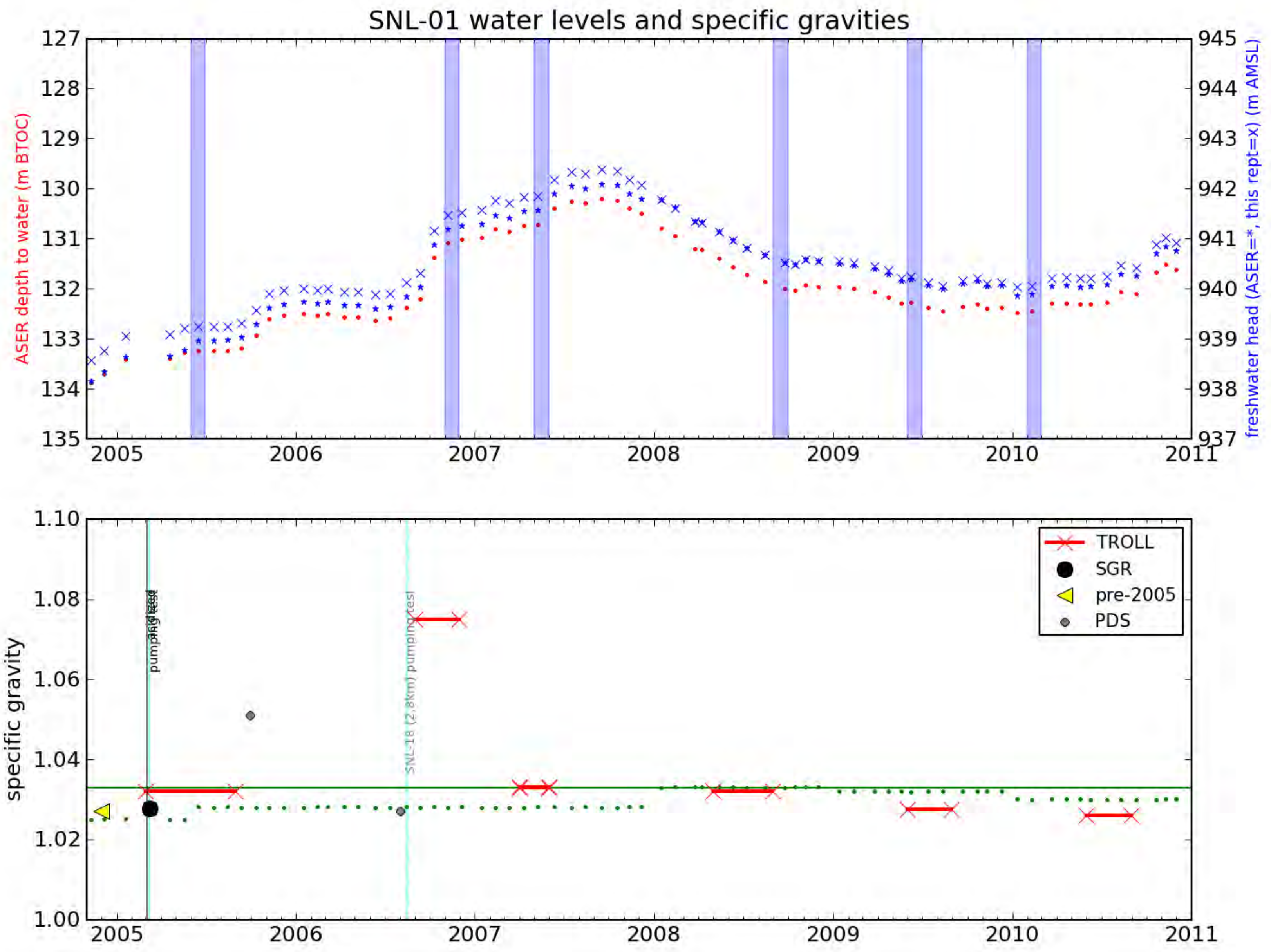


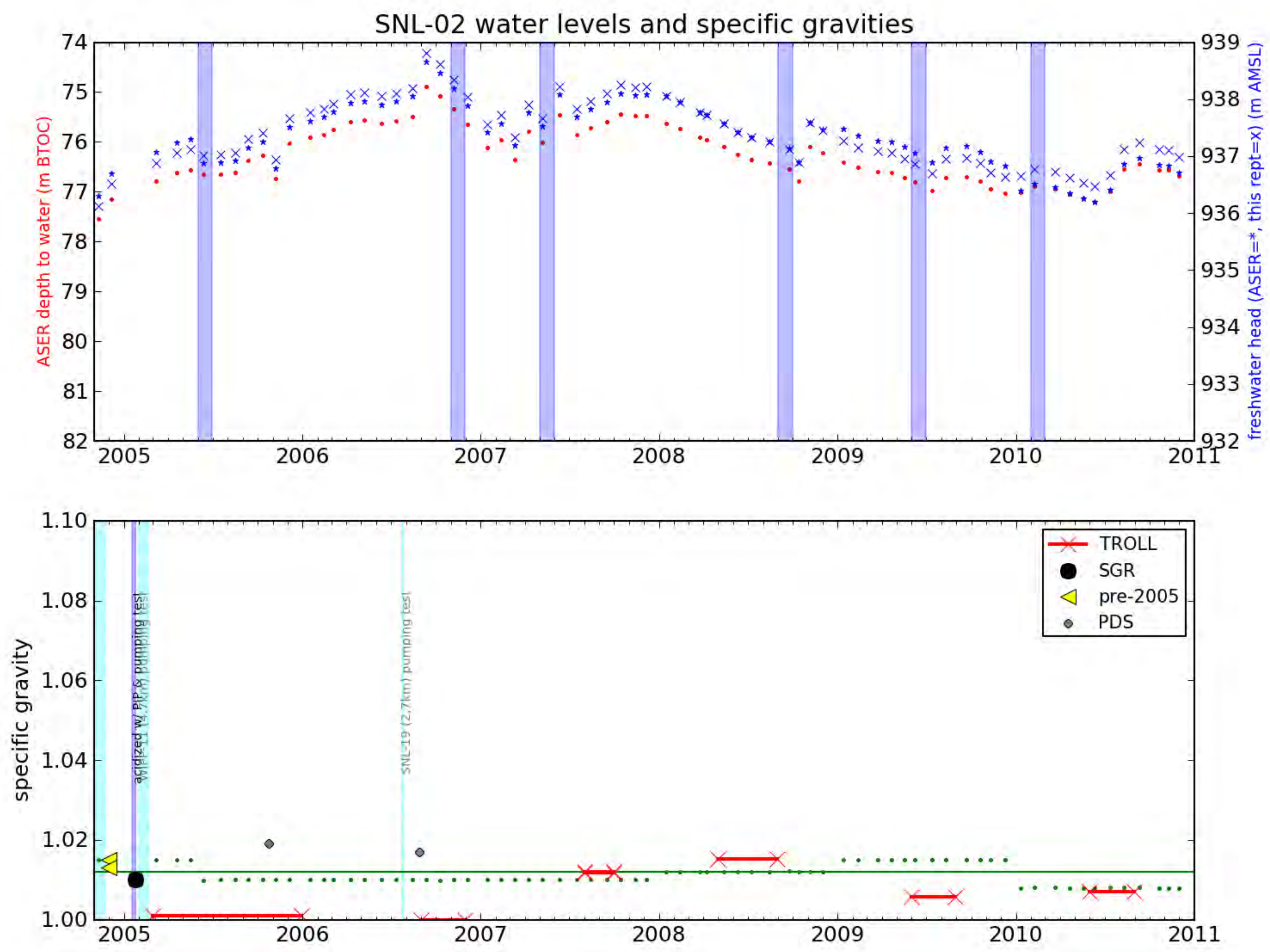

H-17 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
127
128
129
130
131
132
133
920
919
918
917
916
915
914
913
912
911
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.16
1.14
1.12
1.10
1.08
1.06
1.04
1.02
1.00
SNL-14 (0.7km) pumping test
bailed debris
TROLL
SGR
pre-2005
PDS

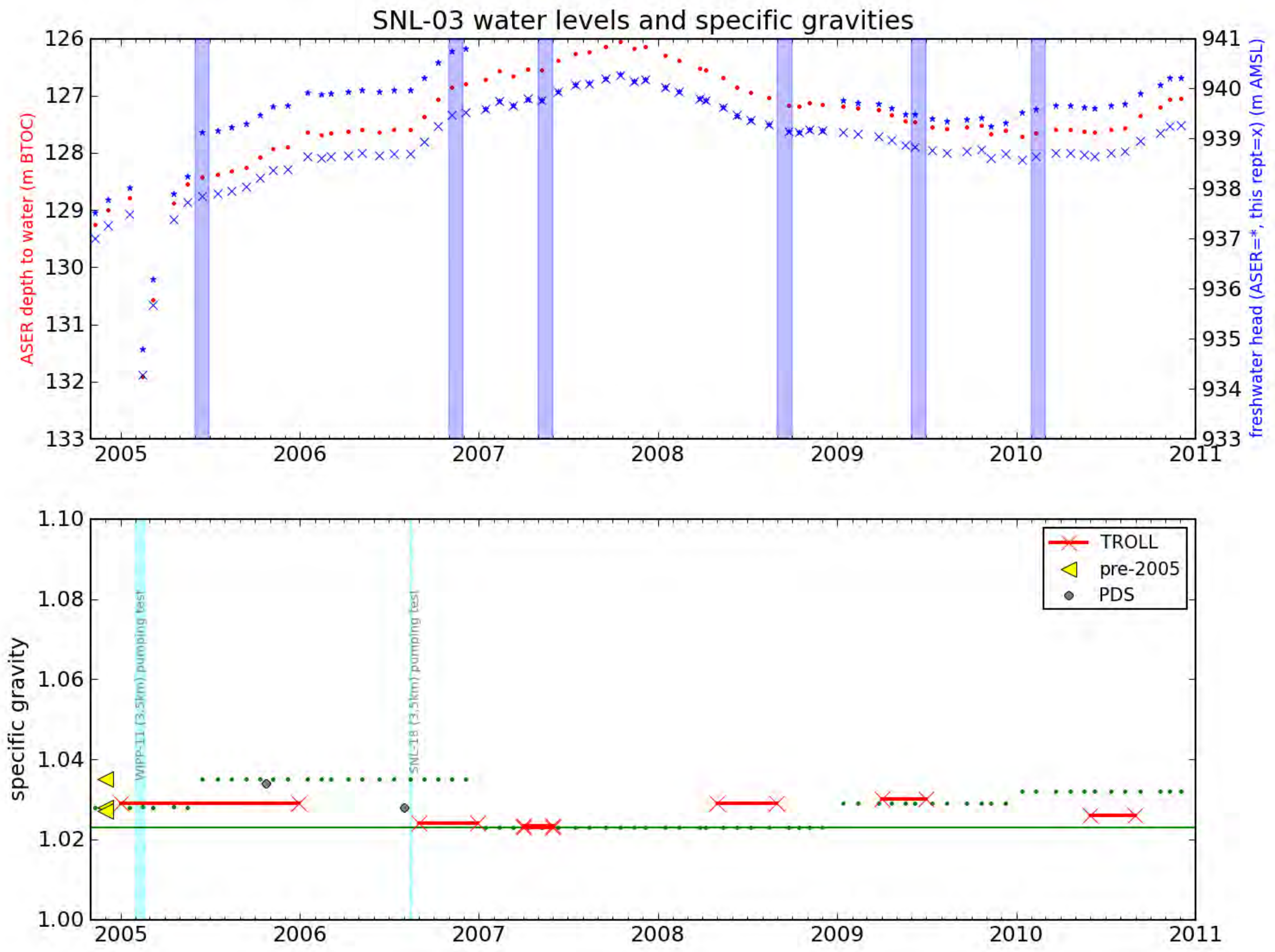

H-19B0 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
specific gravity
SNL-14 (3.1km) pumping test
water quality
SNL-10 (3.4km) pumping test
TROLL
SGR
PDS

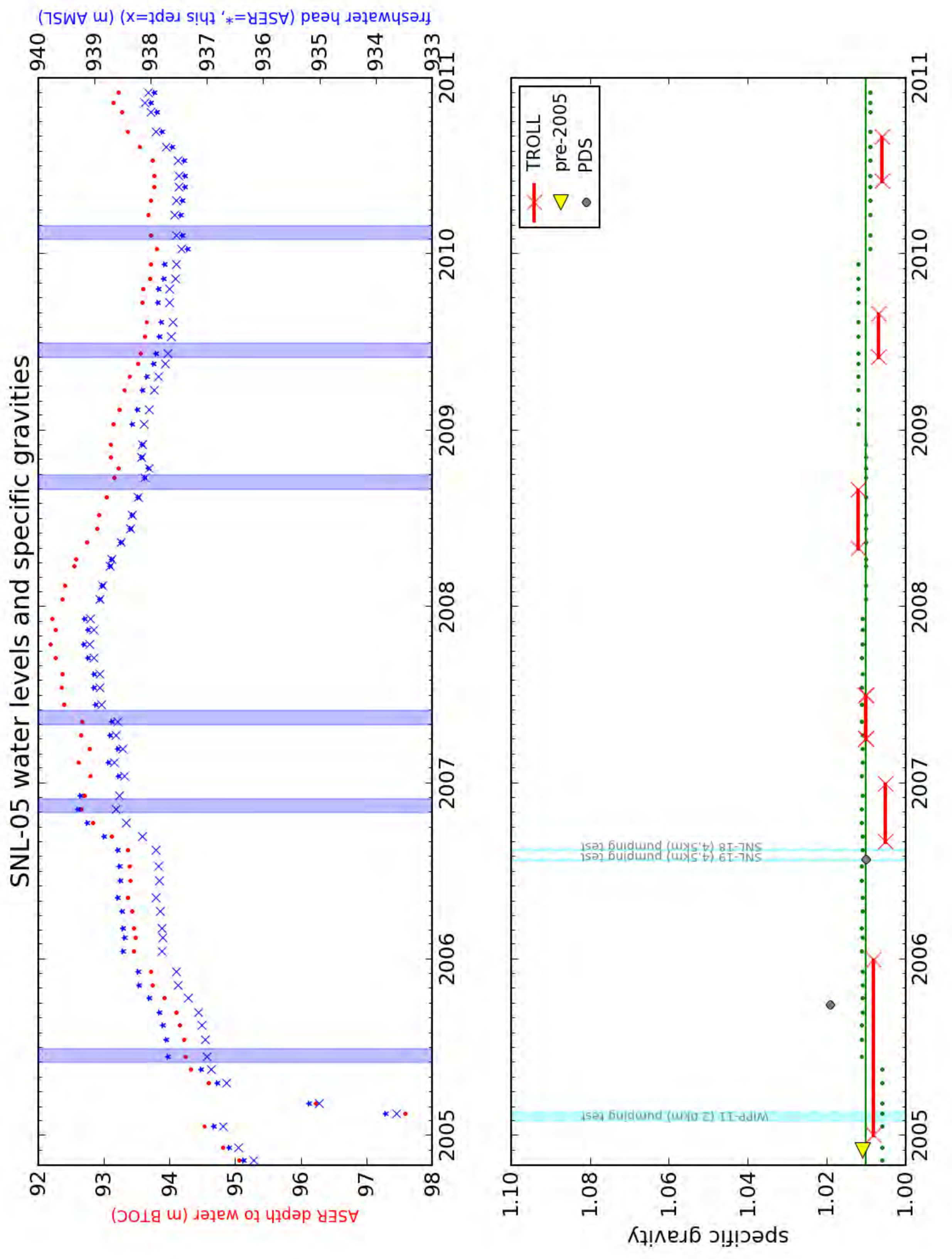

I-461 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
71
72
73
74
75
76
77
78
932
931
930
929
928
927
926
925
924
923
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
slug test
slug test
water quality
SNL-16 (3.4km) pumping
TROLL
SGR
PDS


P-17 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
105.0
105.5
106.0
106.5
107.0
107.5
108.0
108.5
917.0
916.5
916.0
915.5
915.0
914.5
914.0
913.5
913.0
912.5
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.20
1.15
1.10
1.05
1.00
attempted bailing: freshwater added
SNL-14 (1.1km) pumping test
plugged & abandoned
SNL-17 (4.3km) pumping test
TROLL
pre-2005
PDS
SGR

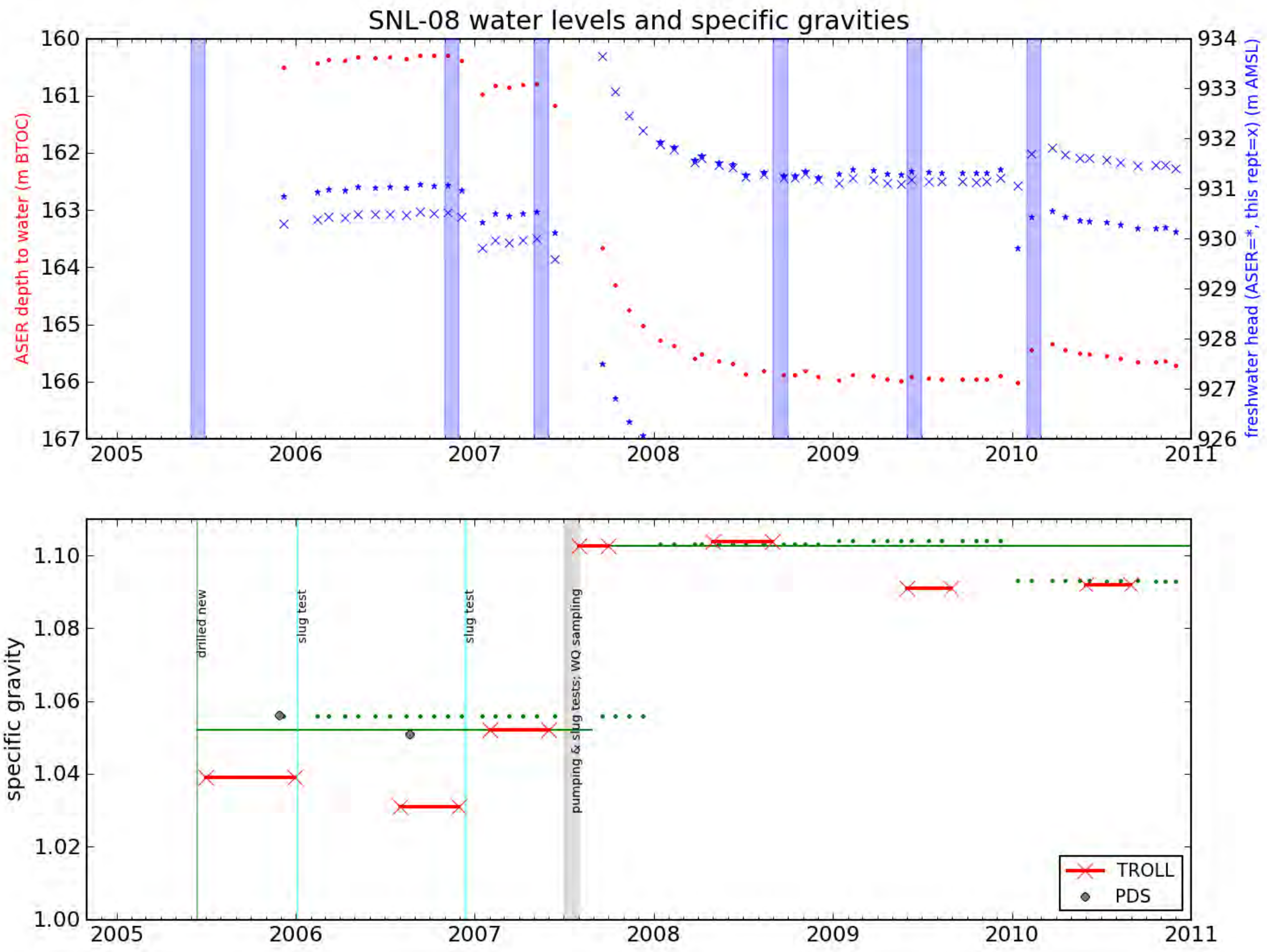

SNL-01 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
127
128
129
130
131
132
133
134
135
945
944
943
942
941
940
939
938
937
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
pumping test
SNL-18 (2.8km) pumping test
TROLL
SGR
pre-2005
PDS

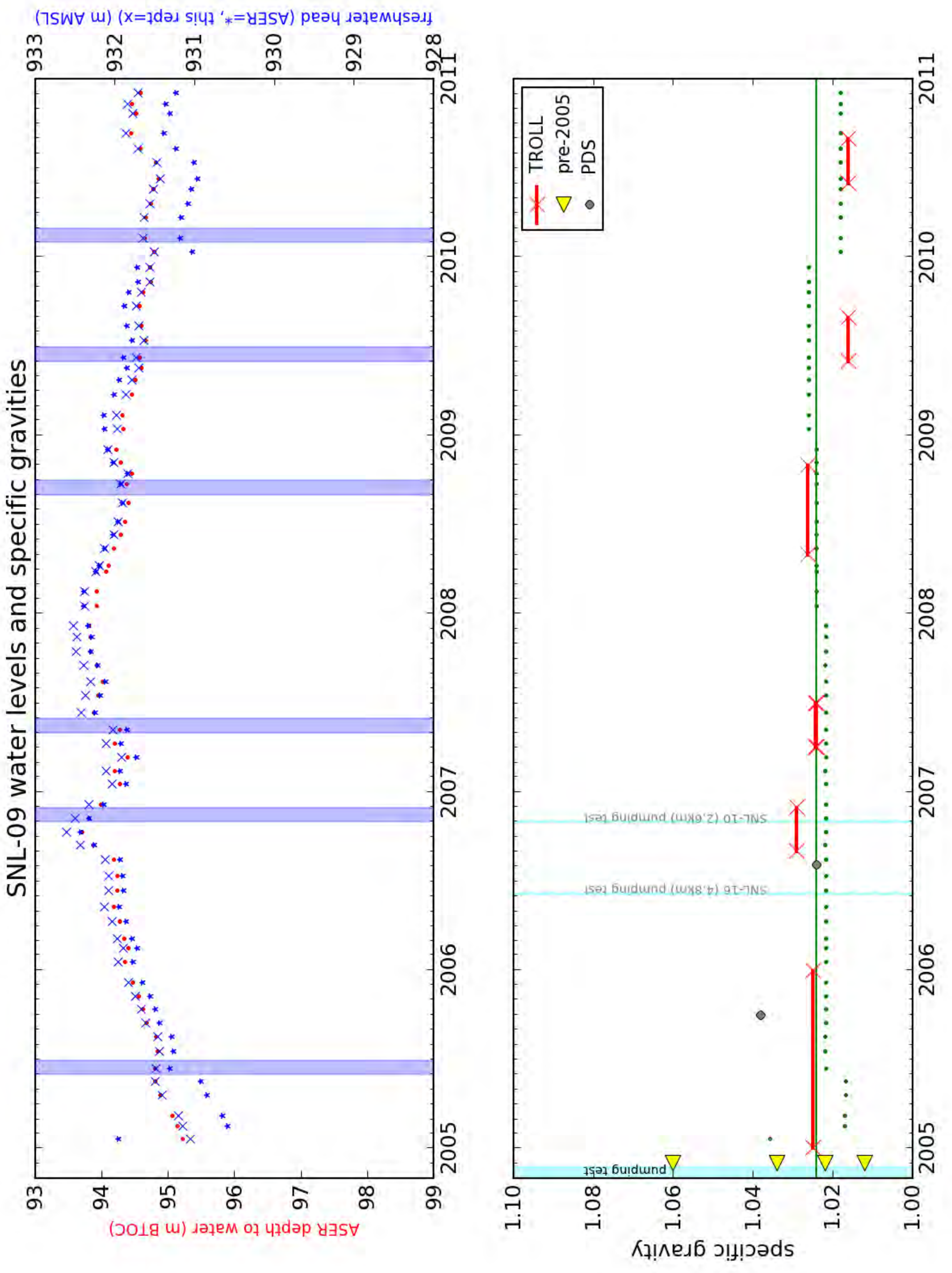

SNL-02 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
74
75
76
77
78
79
80
81
82
939
938
937
936
935
934
933
932
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
acidized w/ HCl & pumping test
SNL-19 (2.7km) pumping test
TROLL
SGR
pre-2005
PDS

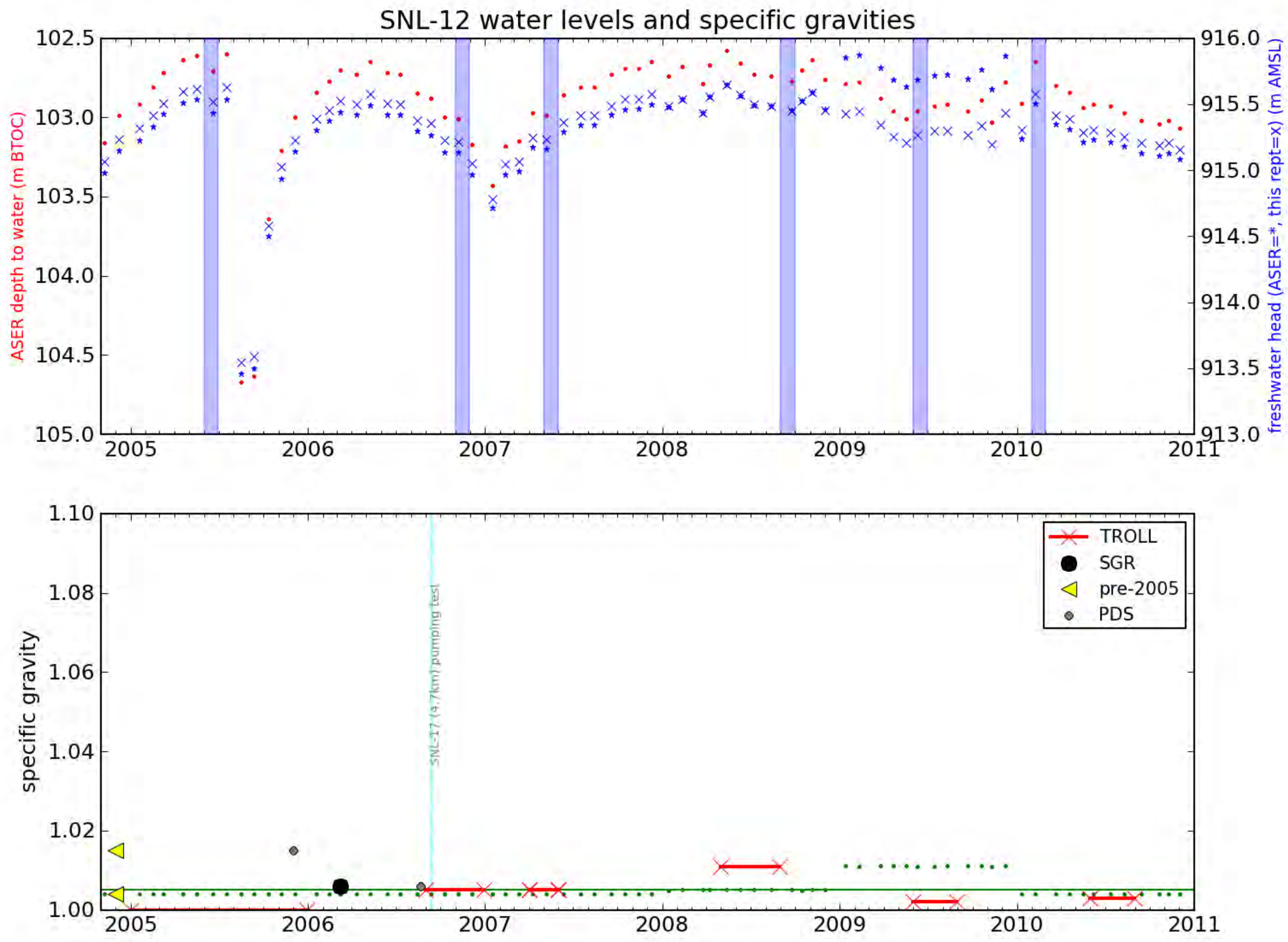

SNL-03 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
126
127
128
129
130
131
132
133
941
940
939
938
937
936
935
934
933
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
WIPP-11 (3.5km) pumping test
SNL-18 (3.5km) pumping test
TROLL
pre-2005
PDS



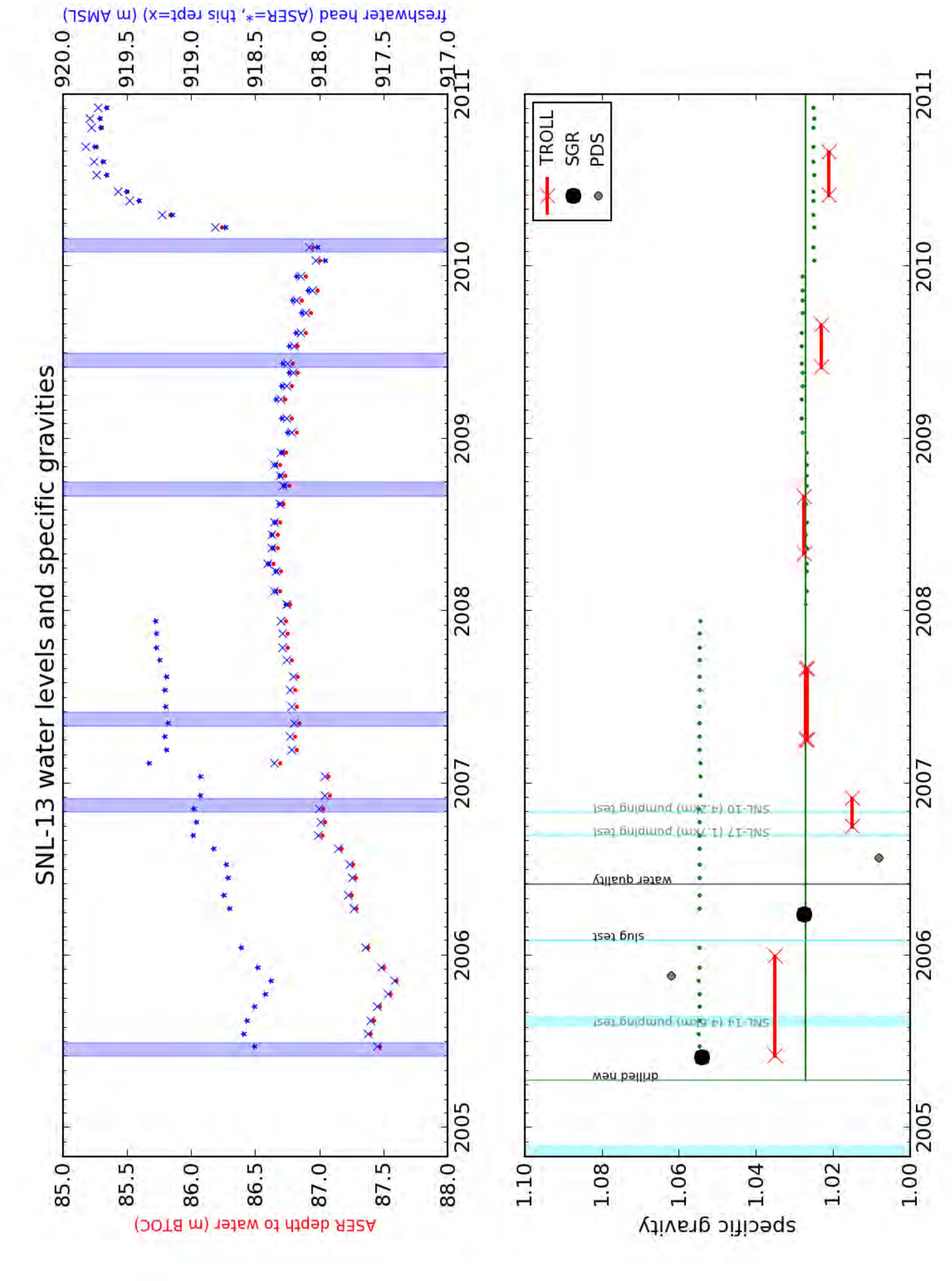
SNL-05 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
92
93
94
95
96
97
98
940
939
938
937
936
935
934
933
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.00
1.02
1.04
1.06
1.08
1.10
TROLL
pre-2005
PDS
WIPP-11 (2.0km) pumping test
SNL-19 (4.5km) pumping test
SNL-18 (4.5km) pumping test

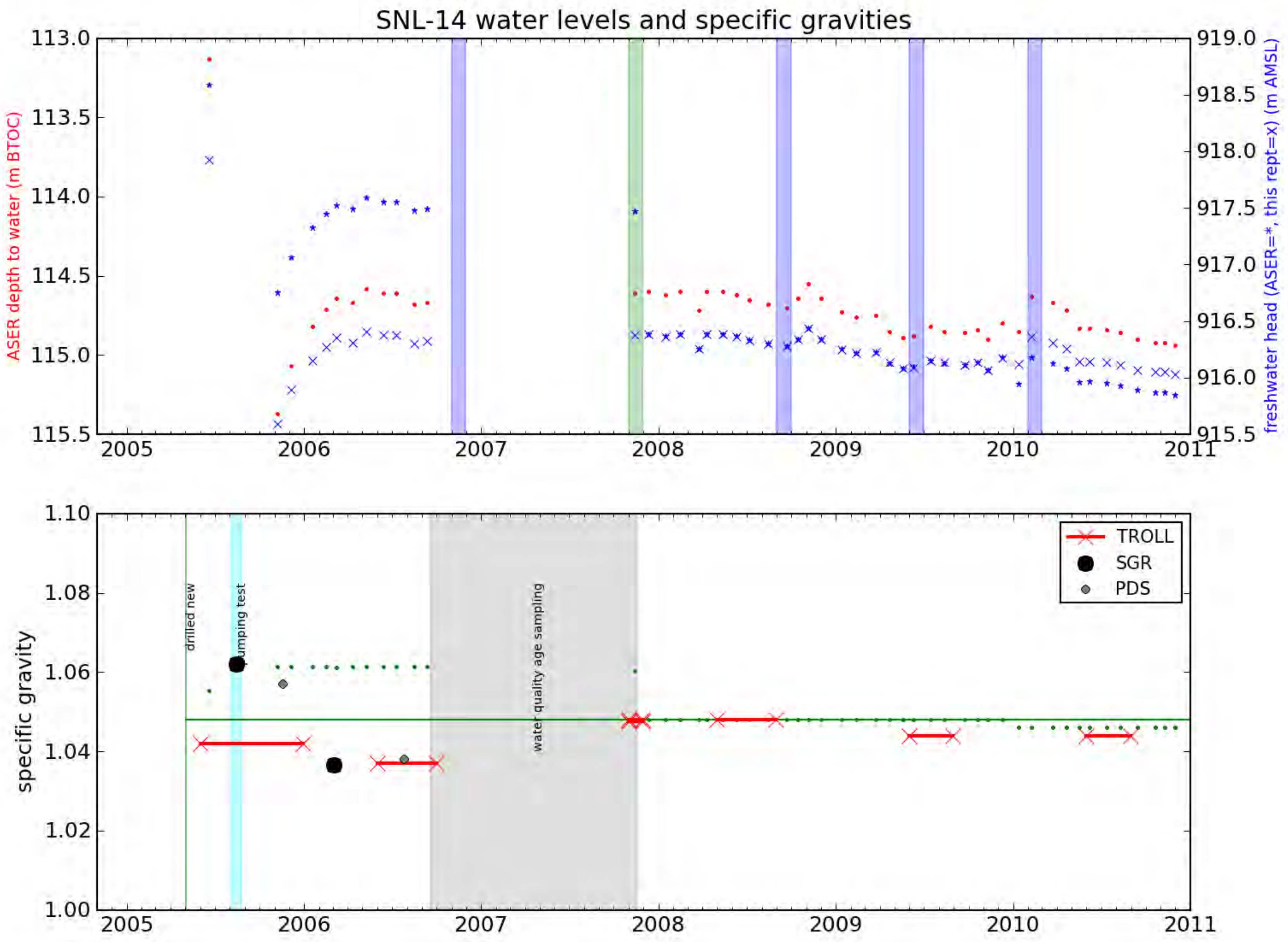

SNL-08 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
specific gravity
drilled new
slug test
slug test
pumping & slug tests; WQ sampling
TROLL
PDS
2005
2006
2007
2008
2009
2010
2011

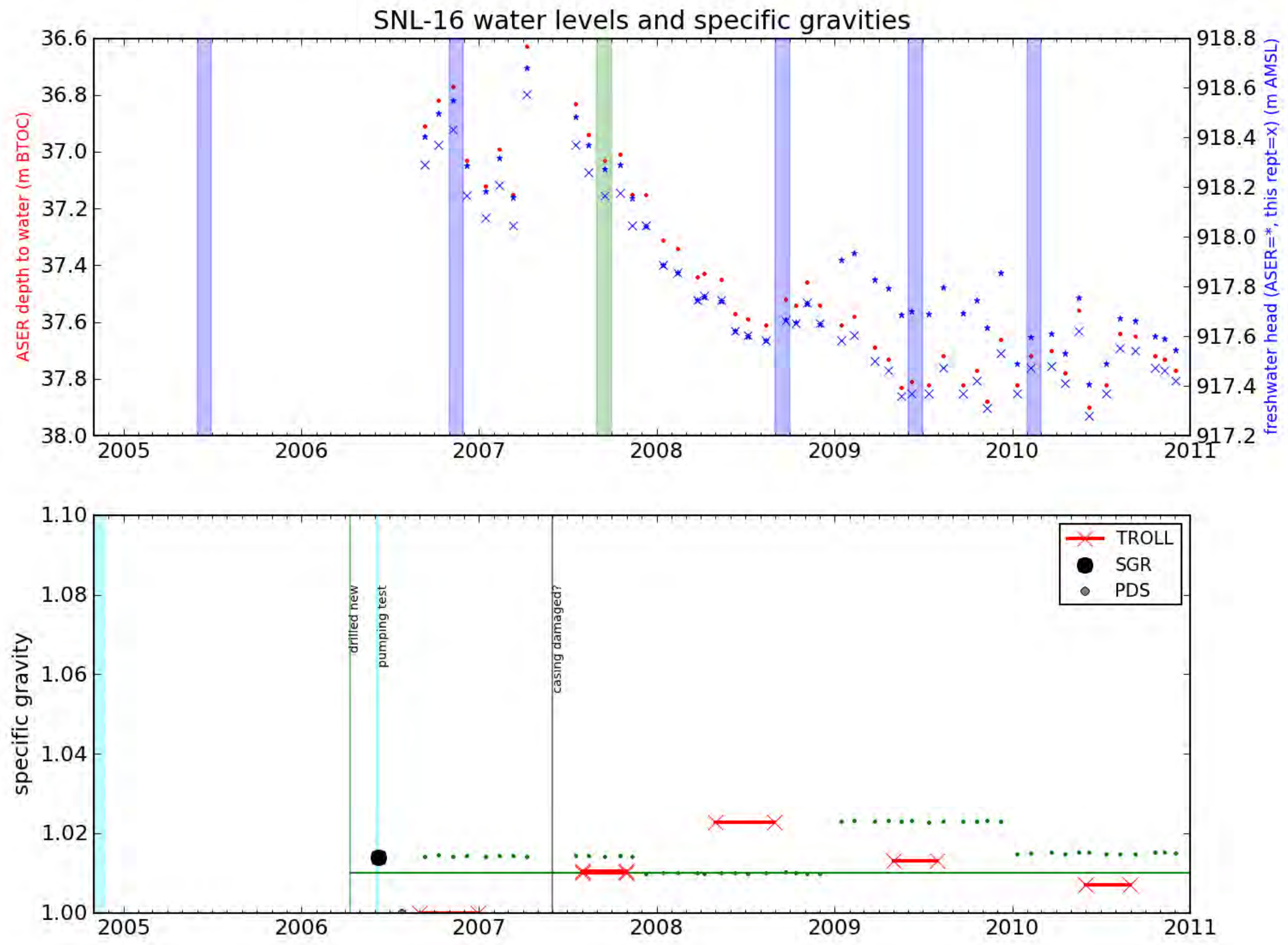

SNL-09 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
specific gravity
TROLL
pre-2005
PDS
SNL-16 (4.8km) pumping test
SNL-10 (2.6km) pumping test
pumping test

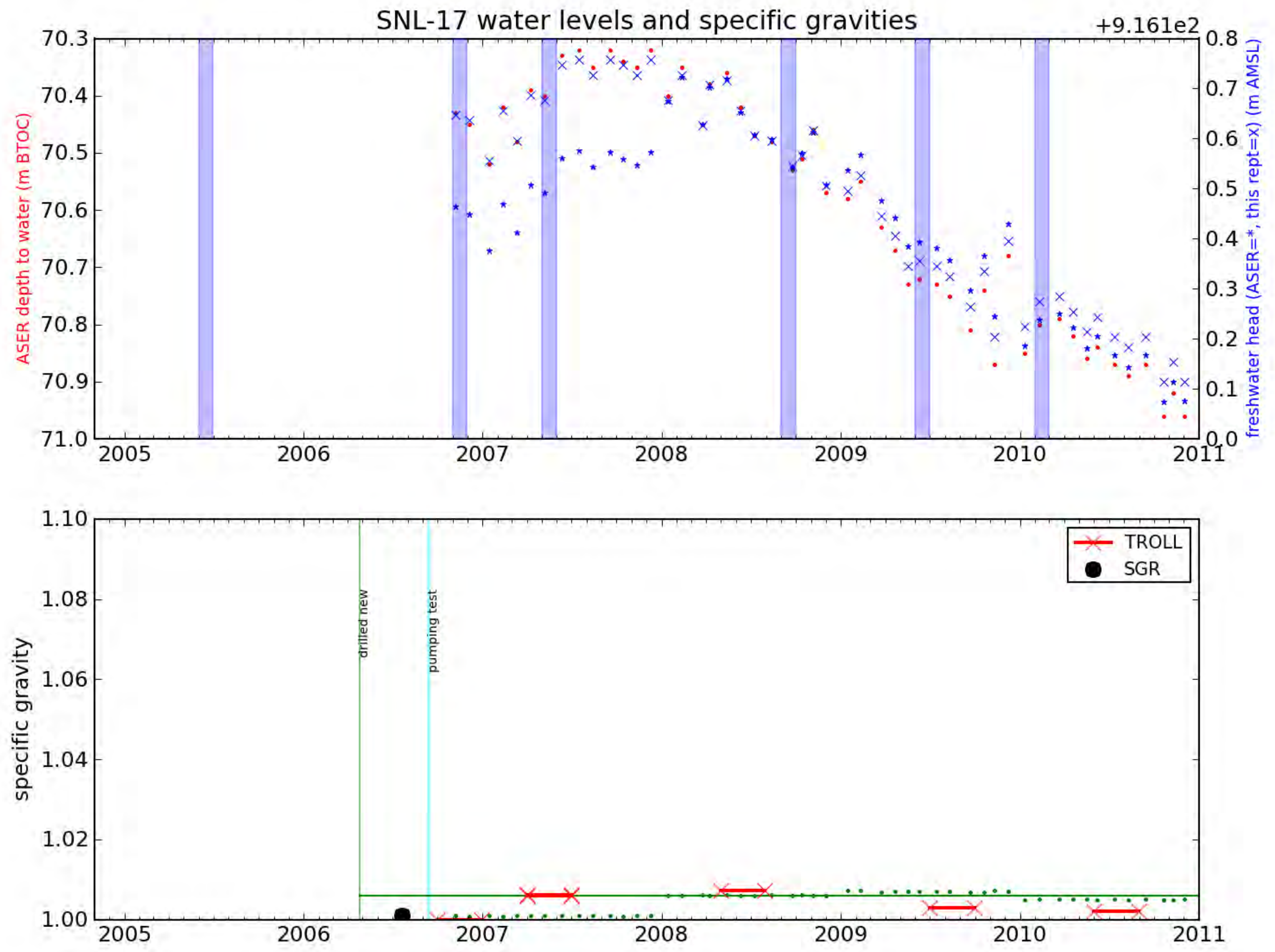


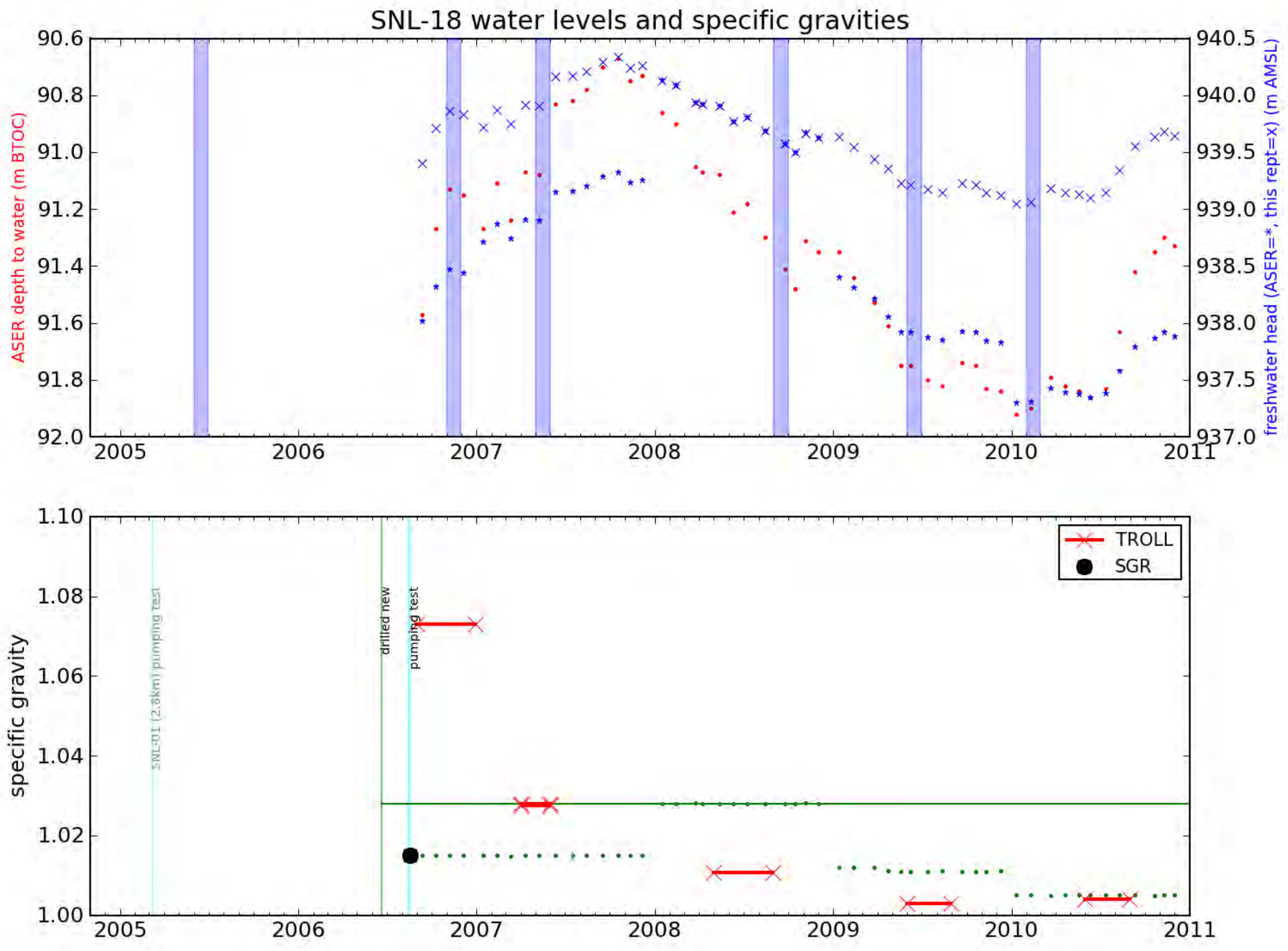
SNL-12 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
102.5
103.0
103.5
104.0
104.5
105.0
916.0
915.5
915.0
914.5
914.0
913.5
913.0
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
SNL-17 (4.7km) pumping test
TROLL
SGR
pre-2005
PDS

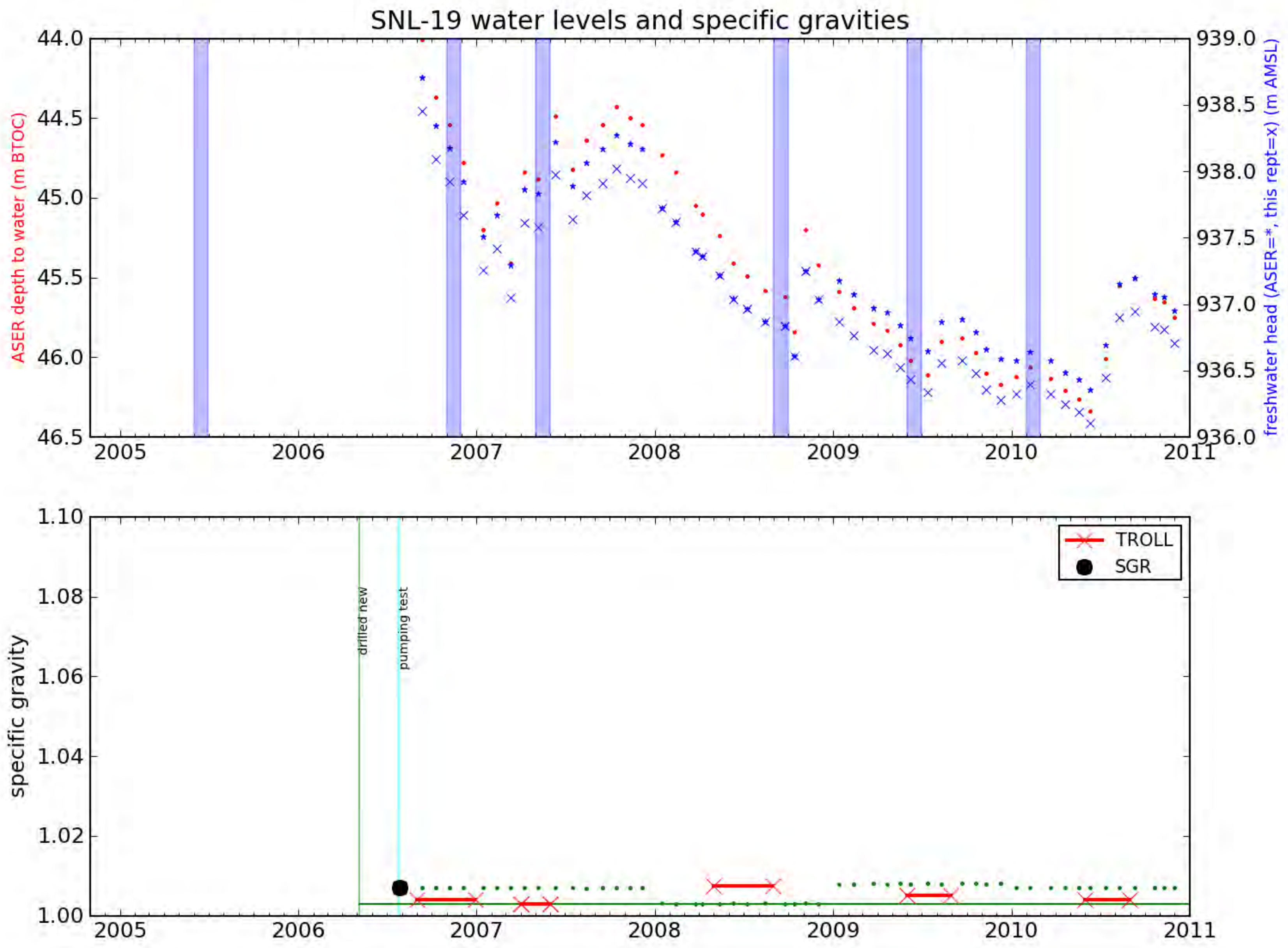

SNL-13 water levels and specific gravities
ASER depth to water (m BTOC)
85.0
85.5
86.0
86.5
87.0
87.5
88.0
freshwater head (ASER=*, this rept=x) (m AMSL)
920.0
919.5
919.0
918.5
918.0
917.5
917.0
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
TROLL
SGR
PDS
drilled new
SNL-14 (4.6km) pumping test
slug test
water quality
SNL-17 (1.7km) pumping test
SNL-10 (4.2km) pumping test

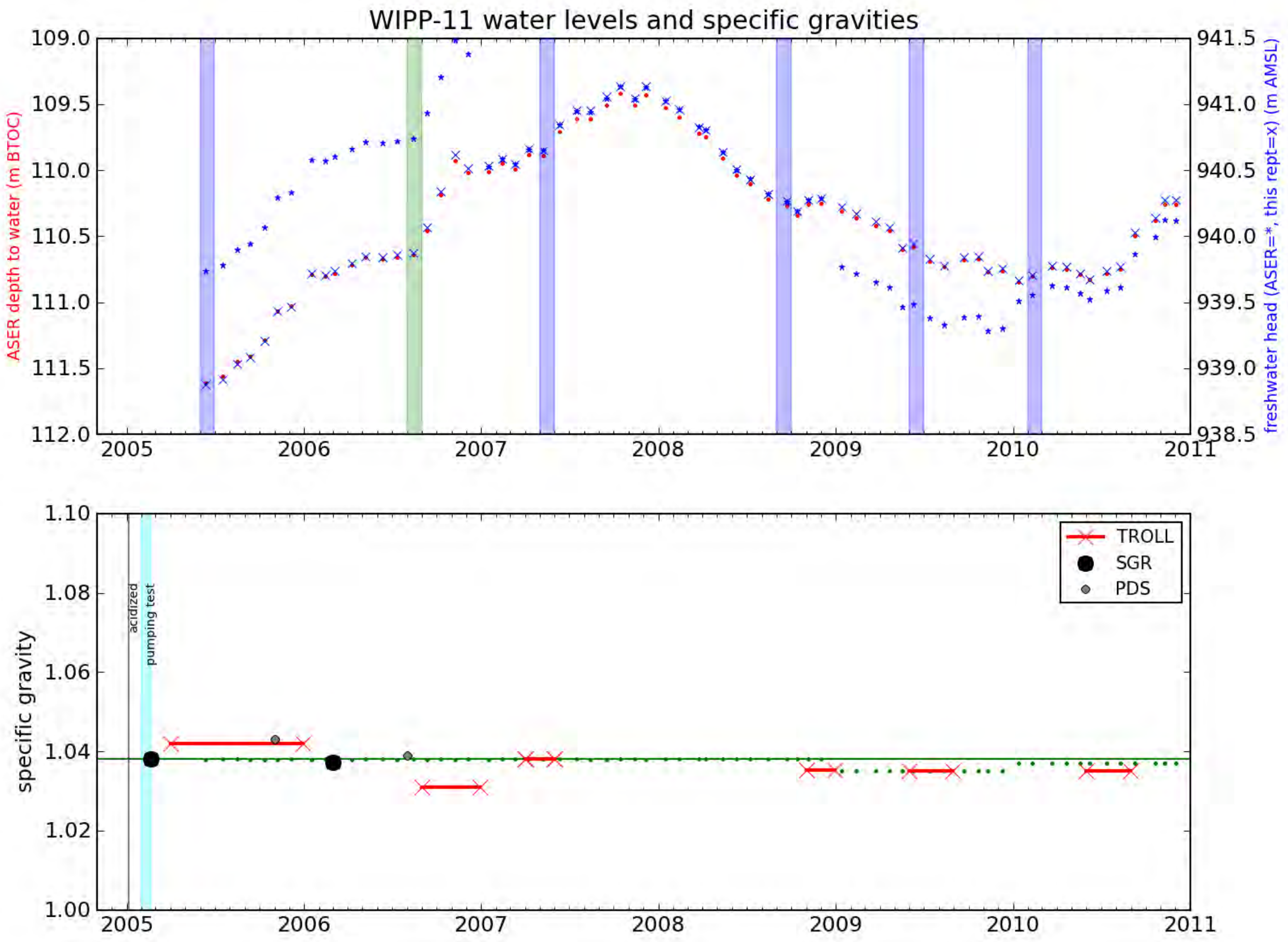

SNL-14 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
113.0
113.5
114.0
114.5
115.0
115.5
919.0
918.5
918.0
917.5
917.0
916.5
916.0
915.5
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
drilled new
pumping test
water quality age sampling
TROLL
SGR
PDS

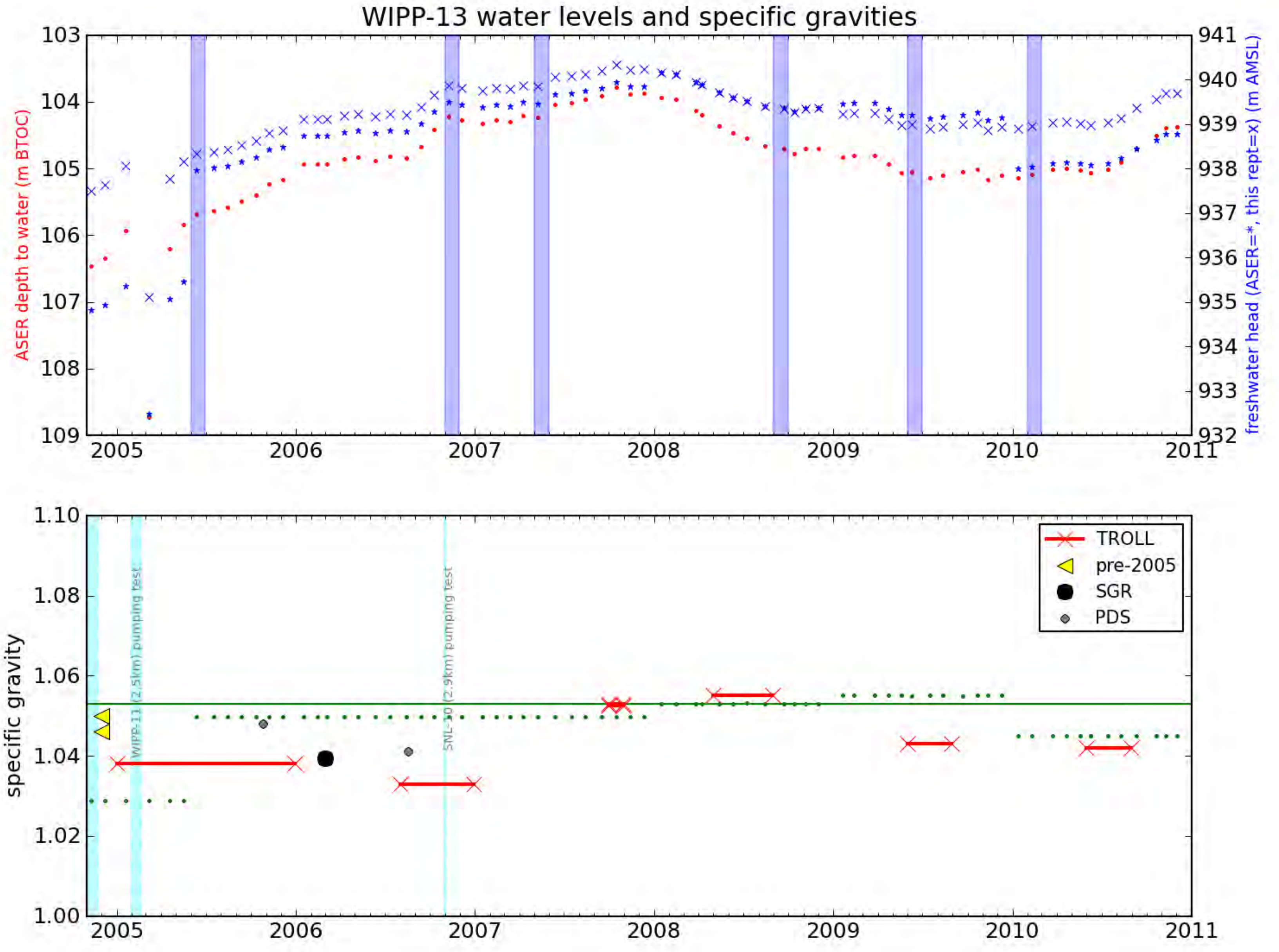

SNL-16 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
36.6
36.8
37.0
37.2
37.4
37.6
37.8
38.0
918.8
918.6
918.4
918.2
918.0
917.8
917.6
917.4
917.2
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
drilled new
pumping test
casing damaged?
TROLL
SGR
PDS

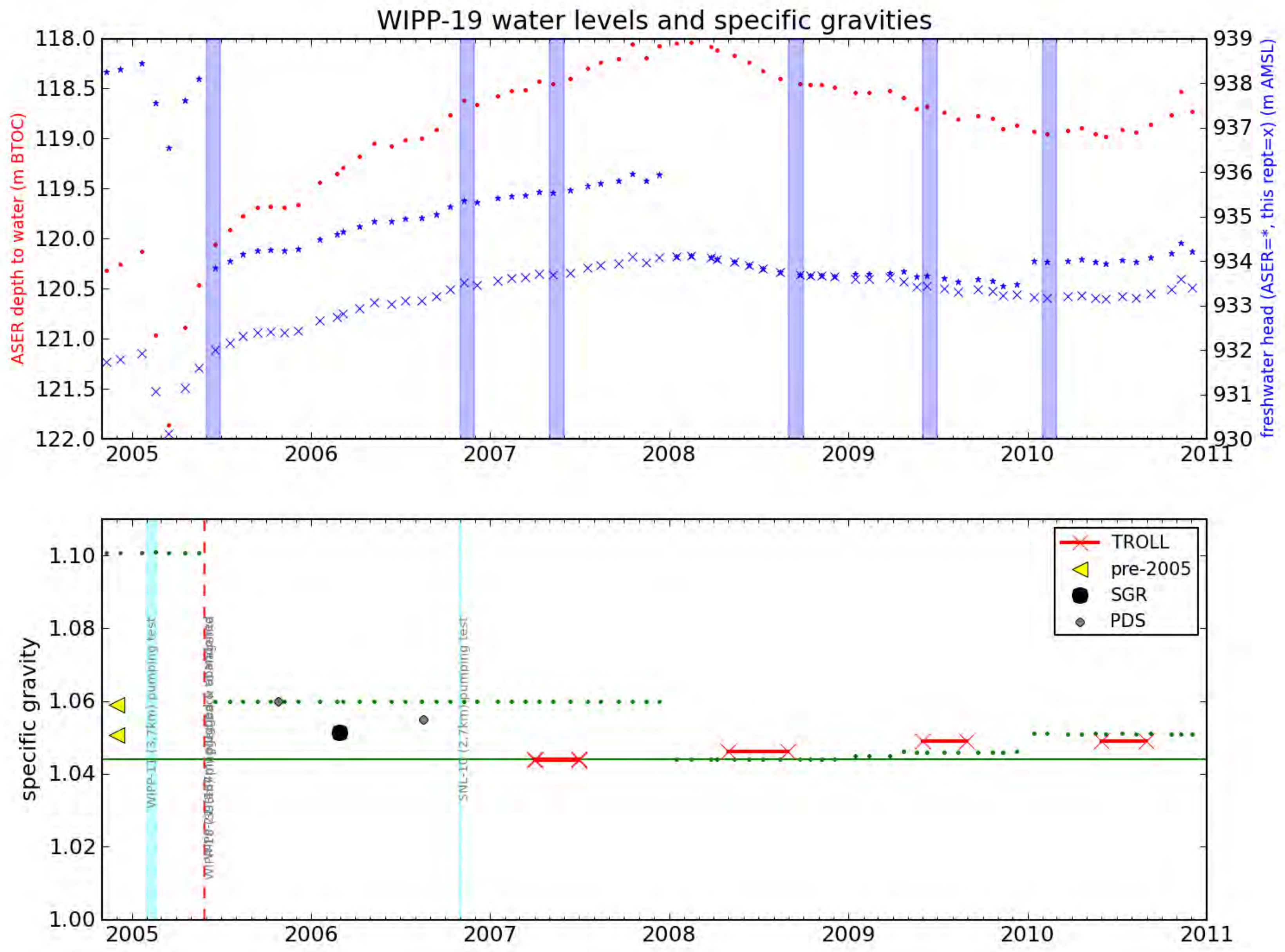

SNL-17 water levels and specific gravities
+9.161e2
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
70.3
70.4
70.5
70.6
70.7
70.8
70.9
71.0
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
drilled new
pumping test
TROLL
SGR

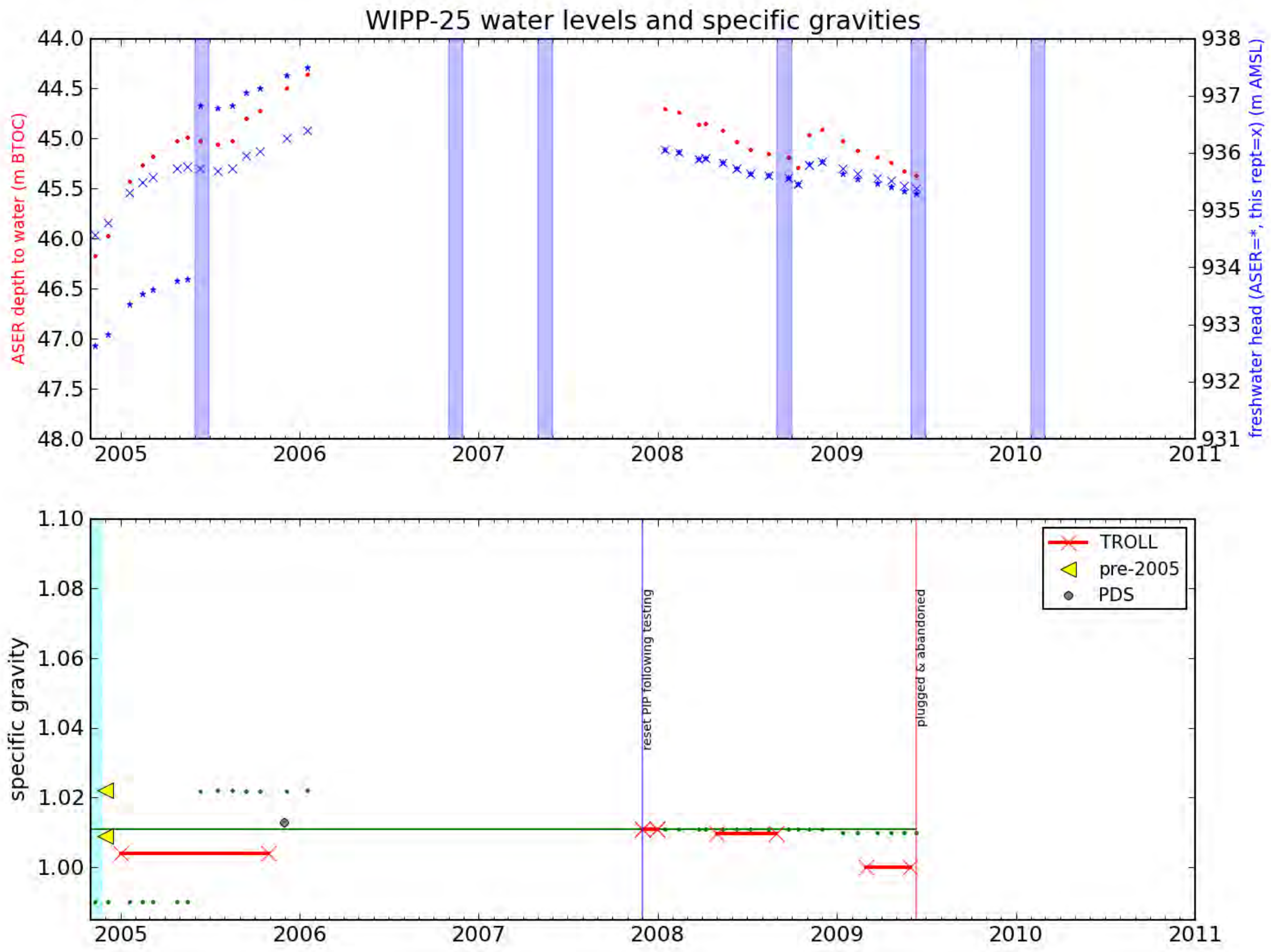

SNL-18 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
90.6
90.8
91.0
91.2
91.4
91.6
91.8
92.0
940.5
940.0
939.5
939.0
938.5
938.0
937.5
937.0
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
SNL-01 (2.8km) pumping test
drilled new
pumping test
TROLL
SGR


SNL-19 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
44.0
44.5
45.0
45.5
46.0
46.5
939.0
938.5
938.0
937.5
937.0
936.5
936.0
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
drilled new
pumping test
TROLL
SGR

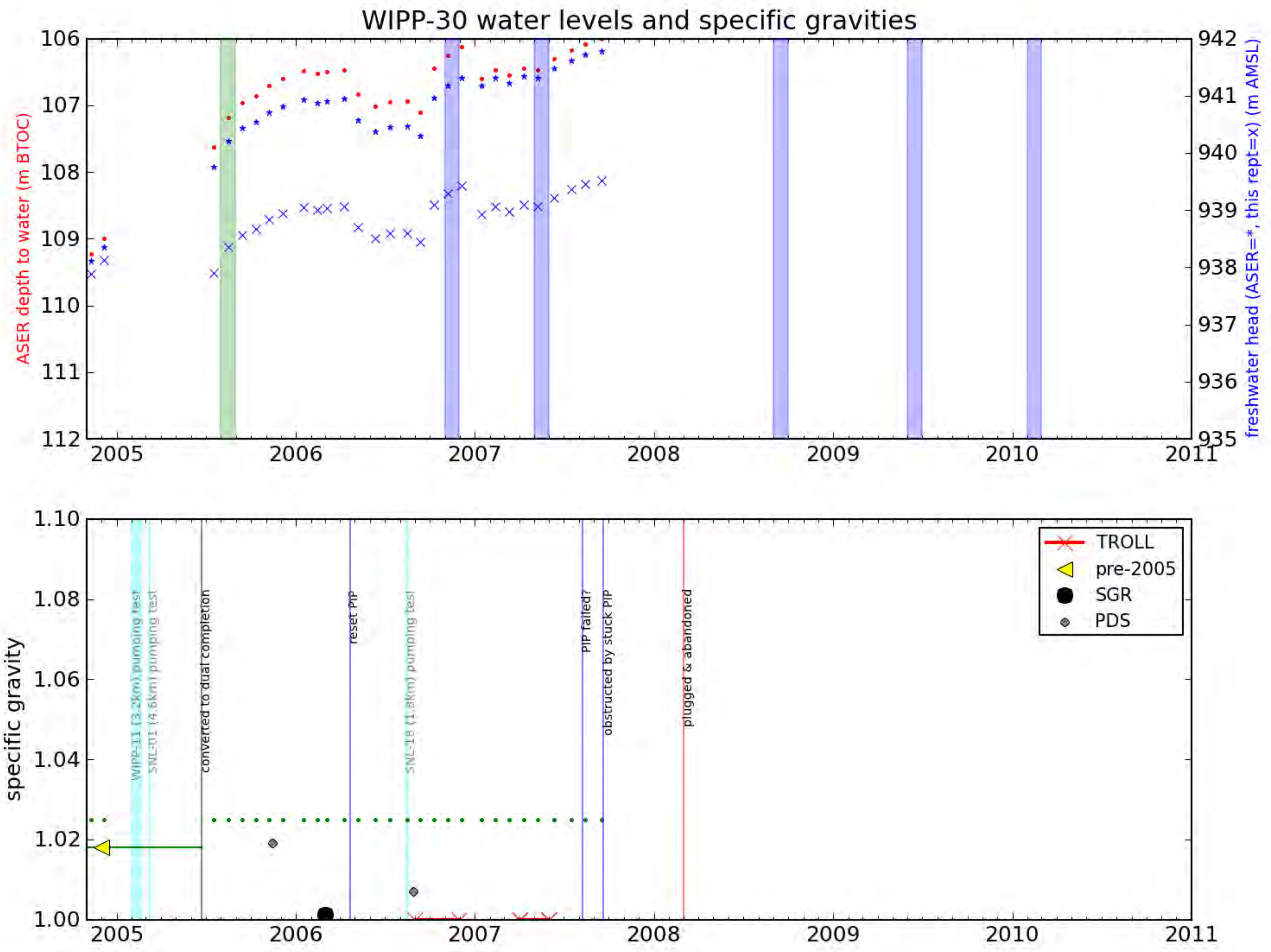

WIPP-11 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
109.0
109.5
110.0
110.5
111.0
111.5
112.0
941.5
941.0
940.5
940.0
939.5
939.0
938.5
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
acidized
pumping test
TROLL
SGR
PDS

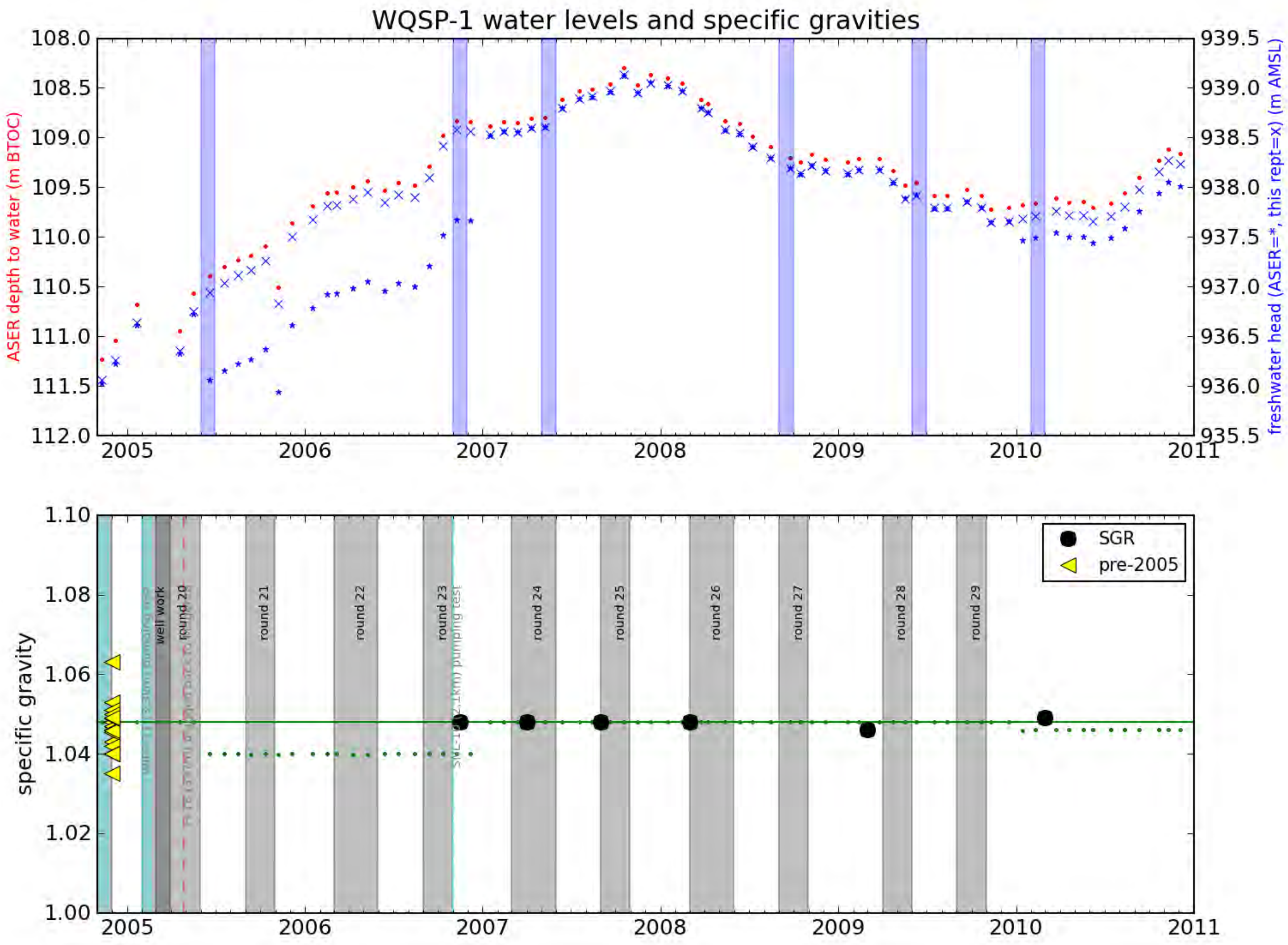

WIPP-13 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
specific gravity
WIPP-11 (2.5km) pumping test
SNL-10 (2.9km) pumping test
TROLL
pre-2005
SGR
PDS
2005
2006
2007
2008
2009
2010
2011

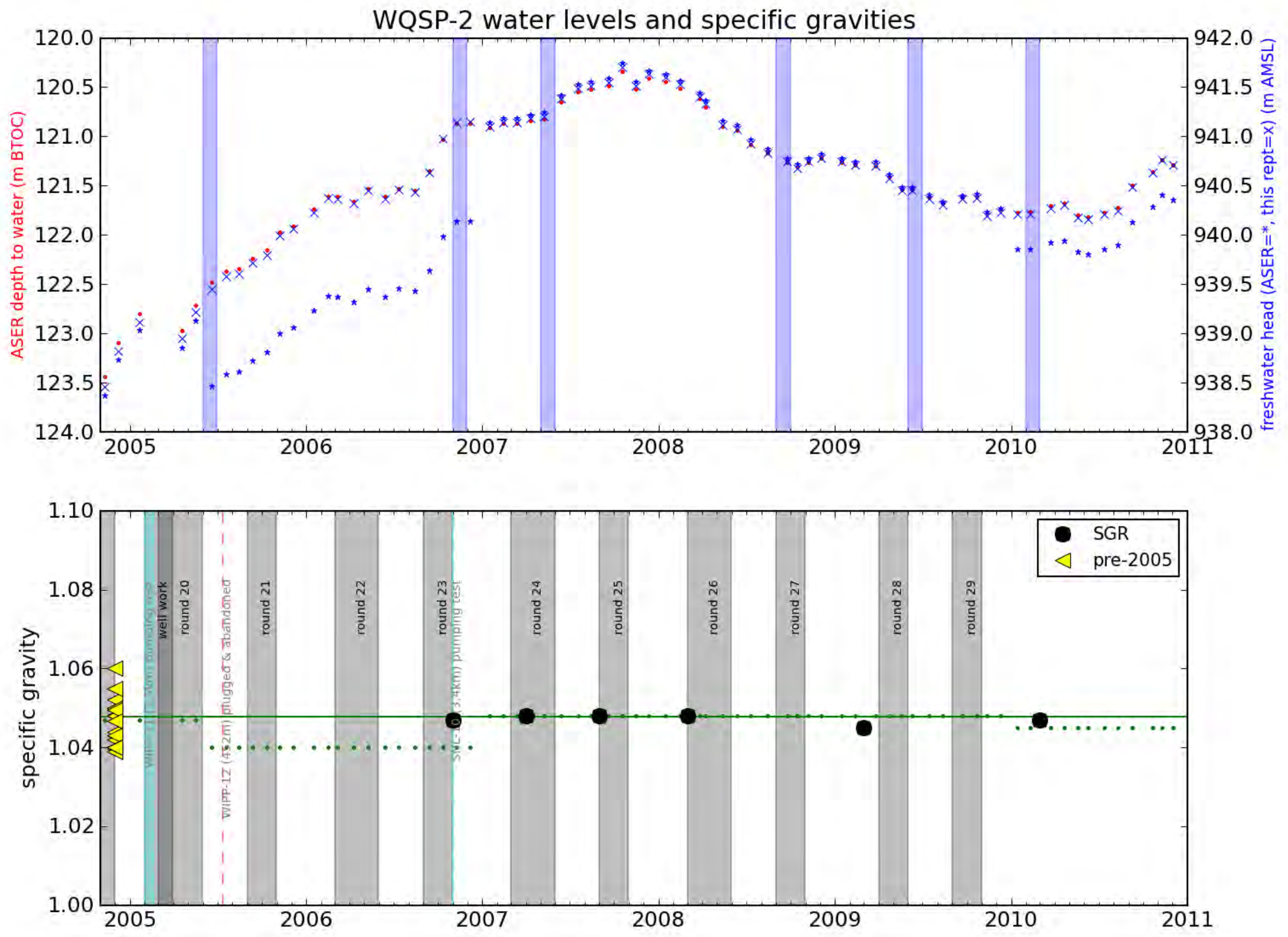

WIPP-19 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
118.0
118.5
119.0
119.5
120.0
120.5
121.0
121.5
122.0
939
938
937
936
935
934
933
932
931
930
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
WIPP-11 (3.7km) pumping test
SNL-10 (2.7km) pumping test
TROLL
pre-2005
SGR
PDS

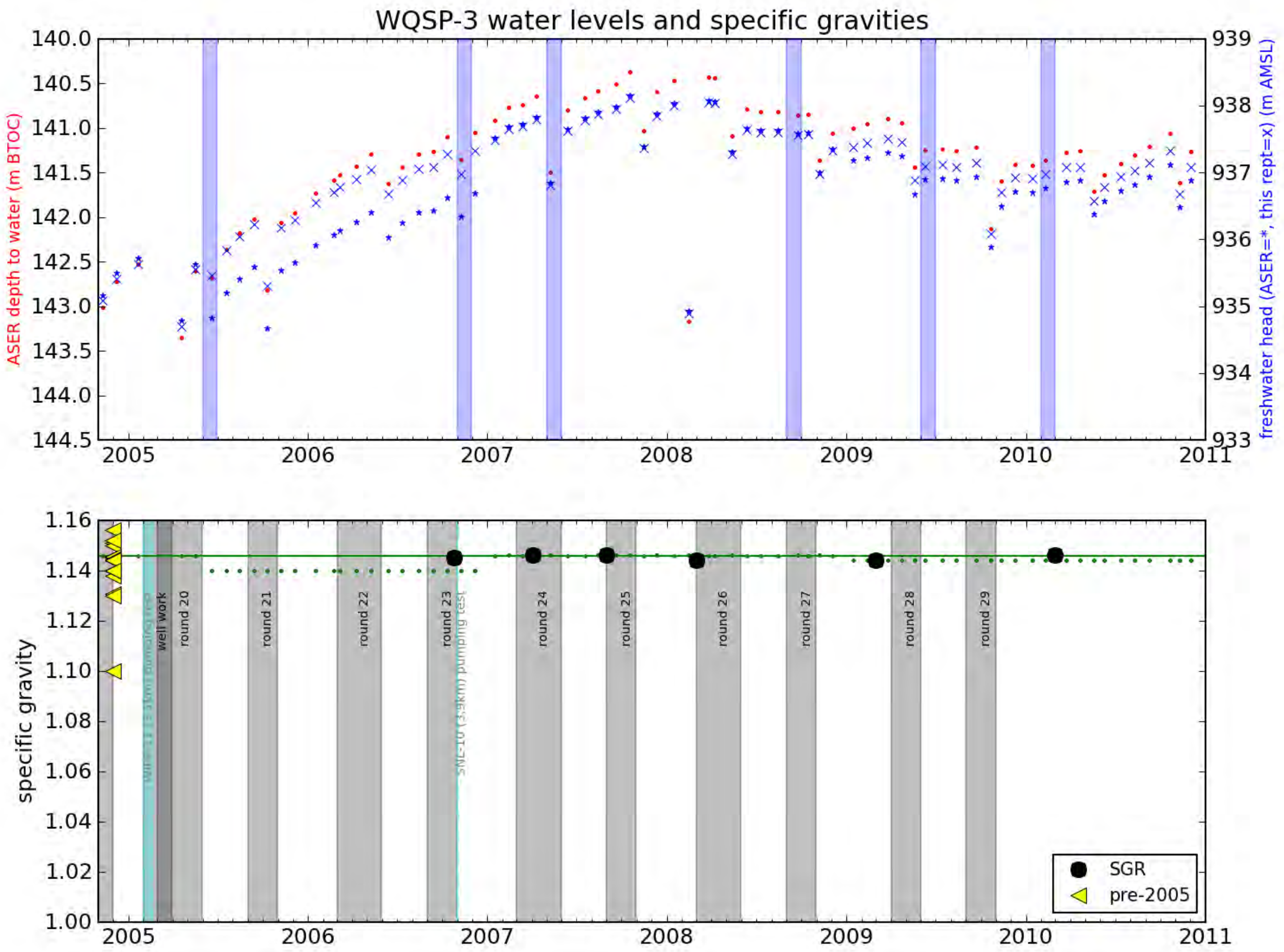


WIPP-25 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
44.0
44.5
45.0
45.5
46.0
46.5
47.0
47.5
48.0
938
937
936
935
934
933
932
931
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
reset PIP following testing
plugged & abandoned
TROLL
pre-2005
PDS

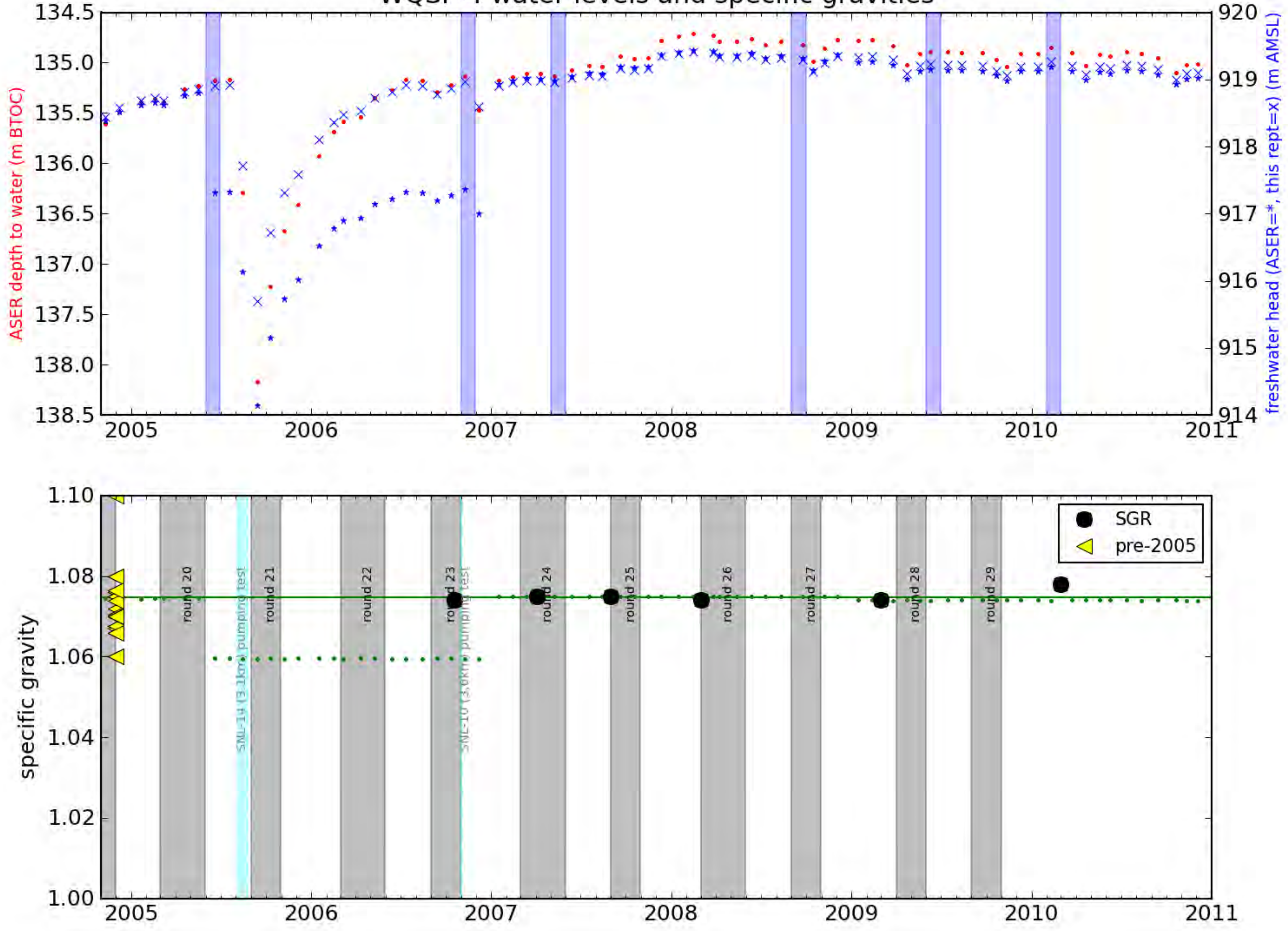

WIPP-30 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
106
107
108
109
110
111
112
942
941
940
939
938
937
936
935
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
WIPP-11 (3.2km) pumping test
SNL-01 (4.6km) pumping test
converted to dual completion
reset PIP
SNL-18 (1.8km) pumping test
PIP failed?
obstructed by stuck PIP
plugged & abandoned
TROLL
pre-2005
SGR
PDS

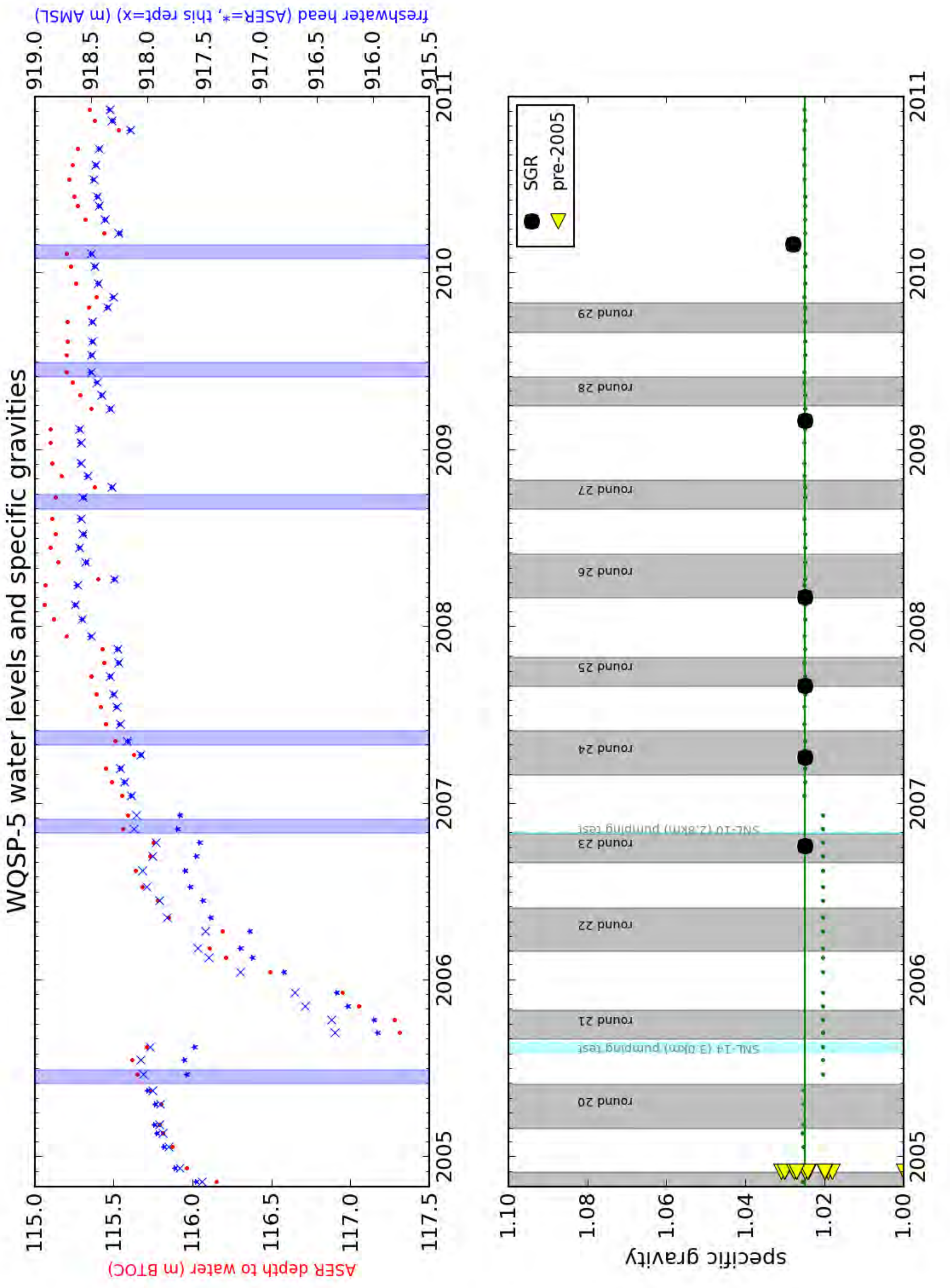

WQSP-1 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
108.0
108.5
109.0
109.5
110.0
110.5
111.0
111.5
112.0
939.5
939.0
938.5
938.0
937.5
937.0
936.5
936.0
935.5
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
well work
round 20
round 21
round 22
round 23
SNL-12 (2.1km) pumping test
round 24
round 25
round 26
round 27
round 28
round 29
SGR
pre-2005

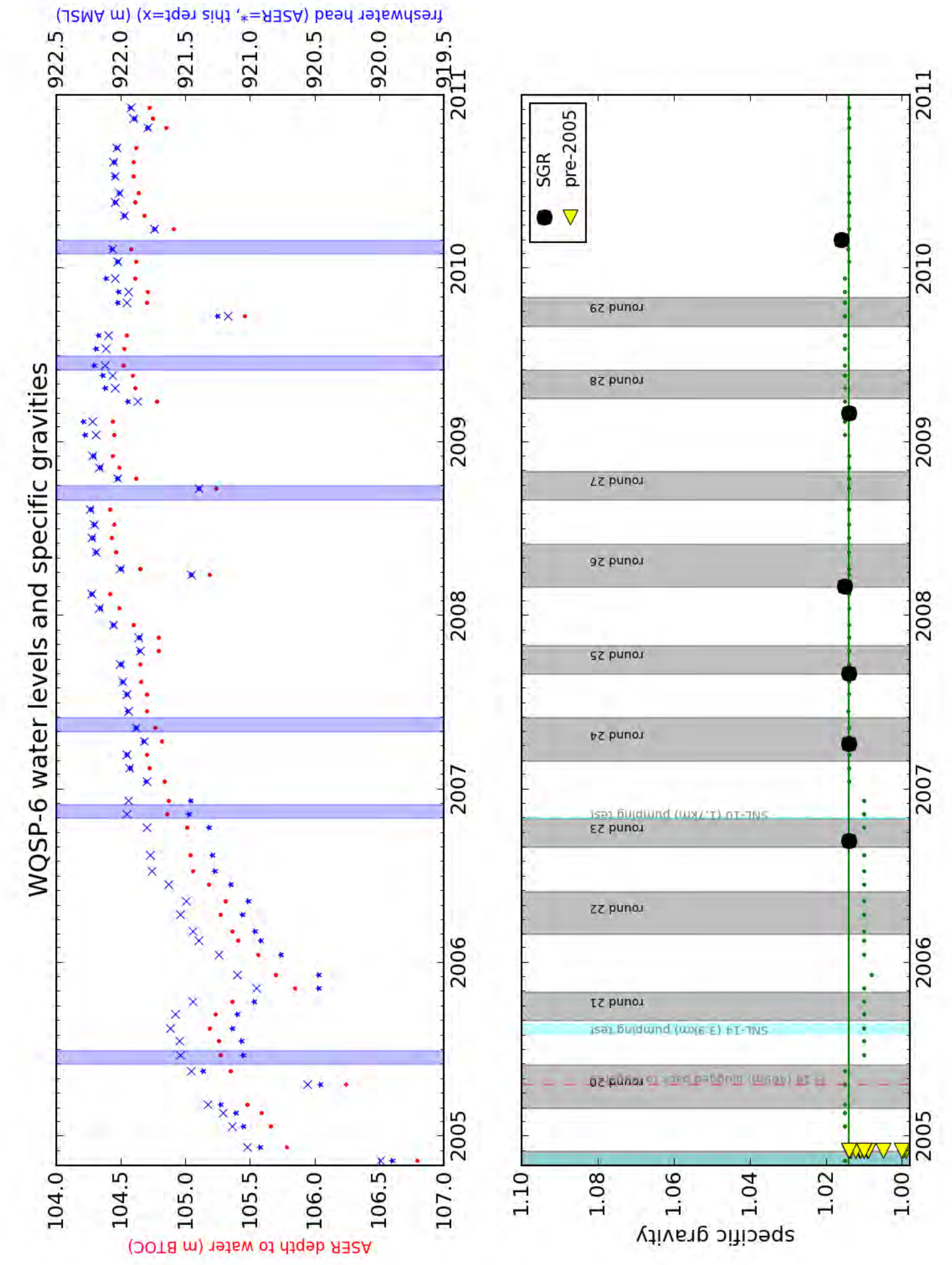

WQSP-2 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
120.0
120.5
121.0
121.5
122.0
122.5
123.0
123.5
124.0
942.0
941.5
941.0
940.5
940.0
939.5
939.0
938.5
938.0
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
SGR
pre-2005
well work
round 20
WIPP-12 (452m) plugged & abandoned
round 21
round 22
round 23
SNL-1 (3.4km) pumping test
round 24
round 25
round 26
round 27
round 28
round 29


WQSP-3 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
140.0
140.5
141.0
141.5
142.0
142.5
143.0
143.5
144.0
144.5
939
938
937
936
935
934
933
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.16
1.14
1.12
1.10
1.08
1.06
1.04
1.02
1.00
WIPP-11 (3.5km) pumping test
well work
round 20
round 21
round 22
round 23
SNL-10 (3.9km) pumping test
round 24
round 25
round 26
round 27
round 28
round 29
SGR
pre-2005

WQSP-4 water levels and specific gravities




WQSP-5 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
115.0
115.5
116.0
116.5
117.0
117.5
919.0
918.5
918.0
917.5
917.0
916.5
916.0
915.5
2005
2006
2007
2008
2009
2010
2011
specific gravity
1.10
1.08
1.06
1.04
1.02
1.00
SGR
pre-2005
round 20
SNL-14 (3.0km) pumping test
round 21
round 22
round 23
SNL-10 (2.8km) pumping test
round 24
round 25
round 26
round 27
round 28
round 29


WQSP-6 water levels and specific gravities
ASER depth to water (m BTOC)
freshwater head (ASER=*, this rept=x) (m AMSL)
specific gravity
SGR
pre-2005
round 20
round 21
round 22
round 23
round 24
round 25
round 26
round 27
round 28
round 29
SNL-14 (3.9km) pumping test
SNL-10 (1.7km) pumping test
2005
2006
2007
2008
2009
2010
2011

10 Appendix: MODFLOW and Pest Files and Script Source Listings

10.1 Input File Listing

The following table lists the input files for the 2005 contour map. The 2006 and 2007 contour maps have the same files with analogous names.

bytes	file type	description	file name
2.1K	Python script	average 100 realizations	`average_realizations.py`
2.3K	Python script	distinguish different BC types	`boundary_types.py`
6.6K	Bash script	main routine: checkout files,run MODFLOW run PEST, call Python scripts	`checkout_average_run_modflow.sh`
809	Python script	convert DTRKMF IJ output to Surfer X,Y blanking format	`convert_dtrkmf_output_for_surfer.py`
3.2K	Python script	create PEST input files from observed data	`create_pest_02_input.py`
48	input listing	responses to DTRKMF prompts	`dtrkmf.in`
4.2K	Python script	convert MODFLOW binary output to Surfer ASCII grid format	`head_bin2ascii.py`
1.1K	input	listing of 100 realizations from CVS	`keepers`
1.4K	input	observed heads in `mod2obs.exe` bore sample file format	`meas_head_2010ASER.smp`
1.2K	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.exe`	`mod2obs_files.dat`
138	input listing	responses to `mod2obs.exe` prompts	`mod2obs_head.in`
372	file listing	files needed to run MODFLOW	`modflow_files.dat`
401	input	listing of wells and geographic groupings	`obs_loc_2010ASER.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`
389	Bash script	PEST model: execute MODFLOW and do pre- and post-processing	`run_02_model`
26	input	`mod2obs.exe` input file	`settings.fig`
47	input	`mod2obs.exe` input file	`spec_domain.spc`
1.7K	input	`mod2obs.exe` input file	`spec_wells.crd`
2.7K	Python script	compute starting head from parameter and coordinate inputs	`surface_02_extrapolate.py`
506	input	DTRKMF input file	`wippctrl.inp`
5.6K	Python script	plot contour map figures	`plot-contour-maps.py`
6.7K	Python script	plot bar and scatter figures	`plot-results-bar-charts.py`
90	plotting data	UTM coordinates of ASER map area	`ASER_boundary.csv`
9.2K	plotting data	UTM coordinates of MODFLOW model area	`total_boundary.dat`
6.7K	plotting data	UTM coordinates of WIPP LWB	`wipp_boundary.csv`

Table 1: Listing of Input Files

10.2 Output File Listing

The following table lists the input files for the 2005 contour map. The 2006 and 2007 contour maps have the same files with analogous names.

bytes	file type	description	file name
19K	DTRKMF output	particle track results	`dtrk.out`
16K	DTRKMF output	particle track debug	`dtrk.dbg`
2.0K	script output	heads at well locations	`modeled_vs_observed_head_pest_02.txt`
1.1M	script output	formatted MODFLOW heads	`modeled_head_pest_02.grd`
5.3K	script output	formatted DTRKMF particle	`dtrk_output_pest_02.bln`
16K	PEST output	matrix condition numbers	`bc_adjust_2010ASER.cnd`
2.7K	PEST output	binary intermediate file	`bc_adjust_2010ASER.drf`
7.4K	PEST output	binary intermediate file	`bc_adjust_2010ASER.jac`
7.5K	PEST output	binary intermediate file	`bc_adjust_2010ASER.jco`
9.9K	PEST output	binary intermediate file	`bc_adjust_2010ASER.jst`
3.8K	PEST output	parameter statistical matrices	`bc_adjust_2010ASER.mtt`
477	PEST output	parameter file	`bc_adjust_2010ASER.par`
62K	PEST output	optimization record	`bc_adjust_2010ASER.rec`
4.6K	PEST output	model outputs for last iteration	`bc_adjust_2010ASER.rei`
8.4K	PEST output	summary of residuals	`bc_adjust_2010ASER.res`
28	PEST output	binary restart file	`bc_adjust_2010ASER.rst`
24K	PEST output	relative parameter sensitivities	`bc_adjust_2010ASER.sen`
4.0K	PEST output	absolute parameter sensitivities	`bc_adjust_2010ASER.seo`
213K	png image	matplotlib plot (Fig. 2)	`aser-area-contour-map.png`
223K	png image	matplotlib plot (Fig. 3)	`large-area-contour-map.png`
33K	png image	matplotlib plot (Fig. 5)	`model-error-histogram.png`
55K	png image	matplotlib plot (Fig. 6)	`model-error-residuals.png`
93K	png image	matplotlib plot (Fig. 4)	`scatter_pest_02.png`

Table 2: Listing of Output Files

10.3 Individual MODFLOW and Pest Script Listings

10.3.1 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 files - INPUTS)
#                \---- original_average  (foward sim using average fields)
#                 \--- bin      (MODFLOW and DTRKMF binaries)
#                  \- pest_0?  (pest-adjusted results)

##set -o xtrace # loads of verbose debugging info

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 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 compound 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

# ------------------------------------------------------------
# the averaging was a slow step, when done in python
```

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

python average_realizations.py

# checkout MODFLOW / DTRKMF executables
cvs -d /nfs/data/CVSLIB/MODFLOW2K checkout bin/mf2k/mf2k_1.6.release
cvs -d /nfs/data/CVSLIB/MODFLOW2K checkout bin/dtrkmf/dtrkmf_v0100

# check out pest and obs2mod binaries
cd bin
cvs -d /nfs/data/CVSLIB/PEST checkout Builds/Linux/pest.exe
cvs -d /nfs/data/CVSLIB/PEST checkout Builds/Linux/mod2obs.exe
cd ..

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

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 and DTRKMF and export results for plotting"
echo "^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^"

# run MODFLOW, producing average head and CCF
../bin/mf2k/mf2k_1.6.release mf2k_head.nam

# run DTRKMF, producing particle track (from ccf)
../bin/dtrkmf/dtrkmf_v0100 <dtrkmf.in

# convert binary MODFLOW 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_2005ASER.smp .
ln -sf ../obs_loc_2005ASER.dat .
../bin/Builds/Linux/mod2obs.exe <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
      echo ${p}" directory exists: removing and re-creating"
      rm -rf ${p}
  fi

  mkdir ${p}
  cd ${p}
  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

  # link to mod2obs files (needed for pest)
  for file in `cat ../mod2obs_files.dat`
    do
    ln -sf ${file} .
  done

  # link to pest files
  for file in `cat ../${p}_files.dat`
```

```
    do
    ln -s ${file} .
  done

  # rename 'original' versions of files to be modified by pest
  rm init_head.mod
  ln -sf ../Inputs/data/init_head.mod ./init_head_orig.mod
  rm init_bnds.inf
  ln -sf ../Inputs/data/init_bnds.inf ./init_bnds_orig.inf

  # create new ibound array for easier modification during PEST
  # optimization iterations
  python boundary_types.py

  # create the necessary input files from observations
  python create_${p}_input.py

  # run pest
  ../bin/Builds/Linux/pest.exe bc_adjust_2005ASER

  # last output files should be best run
  # extract all the stuff from that output
  ########################################

  ln -sf elev_top.mod fort.33
  ln -sf elev_bot.mod fort.34

  ../bin/dtrkmf/dtrkmf_v0100 <dtrkmf.in

  ln -sf ../head_bin2ascii.py .
  python head_bin2ascii.py
  mv modeled_head_asciihed.grd modeled_head_${p}.grd

  ln -sf ../convert_dtrkmf_output_for_surfer.py .
  python convert_dtrkmf_output_for_surfer.py
  mv dtrk_output.bln dtrk_output_${p}.bln

  for file in `cat ../mod2obs_files.dat`
  do
    ln -sf ${file} .
  done

  ../bin/Builds/Linux/mod2obs.exe <mod2obs_head.in
  ln -sf ../merge_observed_modeled_heads.py
  python merge_observed_modeled_heads.py
  mv both_heads.smp modeled_vs_observed_head_${p}.txt

  cd ..
done
```

10.3.2 Python script average_realizations.py

```python
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
for i in xrange(nel):
    if i%10000 == 0:
        print i
    for h,d in zip(fh,fields):
        h.write('%18.11e\n' % pow(10.0,sum(map(log10,d[i]))/nfr) )

for h in fh:
    h.close()
```

10.3.3 Python script boundary_types.py

```python
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 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):
        # is this constant head and is starting head less than 1000m ?
        if ibound[i][j] == -1 and h[i][j] < 1000.0:
            ibound[i][j] = -2

# save new IBOUND array that allows easy discrimination between types in python script during
# PEST optimization runs, and is still handled the same by MODFLOW
# since all ibound values < 0 are treated as constant head.
intsave('init_bnds.inf',ibound)
```

10.3.4 Python script create_pest_02_input.py

```
prefix = '2005ASER'

##################################################
## 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))
fpar.close()


####################
## pest control file

f = open('bc_adjust_%s.pst' % prefix,'w')

f.write("""pcf
* control data
restart estimation
%i %i 1 0 2
1 2 double point 1 0 0
5.0 2.0 0.4 0.001 10
3.0 3.0 1.0E-3
0.1
30 0.001 4 4 0.0001 4
1 1 1
* parameter groups
bc relative 0.005 0.0001 switch 2.0 parabolic
""" % (len(params),len(wells)+4))

f.write('* parameter data\n')
for n,p in zip(pnames,params):
        if p > 0:
                f.write('%s  none  relative  %.3f  %.3f  %.3f  bc  1.0  0.0  1\n' %
                        (n, p, -2.0*p, 3.0*p))
        else:
                f.write('%s  none  relative  %.3f  %.3f  %.3f  bc  1.0  0.0  1\n' %
                        (n, p, 3.0*p, -2.0*p))

f.write("""* observation groups
ss_head
avg_head
* observation data
""")

## read in observation weighting group definitions
fin = open('obs_loc_%s.dat' % prefix,'r')
location = [line.rstrip().split()[1] for line in fin]
fin.close()

weights = []

for l in location:
    # inside LWB
    if l == '0':
        weights.append(2.5)
    # near LWB
    if l == '1':
        weights.append(1.0)
    # distant to LWB
    if l == '2':
        weights.append(0.4)
    if l == '99':
        weights.append(0.01) # AEC-7


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

# one fewer N observation (WIPP-25 removed), there were 13
# there are 12 N observations in the average and 11 S, therefore
# split the weight between the mean and median
f.write("""medianN   0.0   18.0   avg_head
medianS   0.0   16.5   avg_head
meanN     0.0   18.0   avg_head
meanS     0.0   16.5   avg_head
""")

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()
```

10.3.5 Python script surface_02_extrapolate.py

```python
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 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()
```

10.3.6 Bash shell script `run_02_model`

```
#!/bin/bash

#set -o xtrace

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

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

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

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

10.3.7 Python script head_bin2ascii.py

```python
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)
        if len(check) != 4:
            raise IOError('Didn''t read enough data')
        if struct.unpack(self.ENDIAN+'i',check)[0] != l:
            raise IOError('Error reading record from data file')
        return data_str

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

```

def floatmatsave(filehandle,m):
    """Writes array to open filehandle, format '568%e12.5'.
    Outer list is rows, inner lists are columns."""

    for row in m:
        f.write(''.join([' %12.5e' % col for col in row]) + '\n')

# open file and set endian-ness
try:
    infn,outfn = argv[1:3]
except:
    print '2 command-line arguments not given, using default in/out filenames'
    infn = 'modeled_head.bin'
    outfn = 'modeled_head_asciihed.grd'

ff = FortranFile(infn)

# currently this assumes a single-layer MODFLOW model (or at least only one layer of output)

# format of MODFLOW header in binary layer array
fmt = '<2i2f16s3i'
# little endian, 2 integers, 2 floats,
#    16-character string (4 element array of 4-byte strings), 3 integers

while True:
    try:
        # read in header
        h = ff.readRecord()

    except IOError:
        # exit while loop
        break

    else:
        # 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\n%i %i\n%.1f %.1f\n%.1f %.1f\n%.8e %.8e" %
        (ncol,nrow,xmin,xmax,ymin,ymax,hmin,hmax) )
hmat = reshapev2m(h,ncol,nrow)

# MODFLOW 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()
```

10.3.8 Python script merge_observed_modeled_heads.py

```
fobs = open('meas_head_2005ASER.smp','r') # measured head
fmod = open('modeled_head.smp','r')       # modeled head
fwgt = open('obs_loc_2005ASER.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()
```

10.3.9 Python script `convert_dtrkmf_output_for_surfer.py`

```python

# grid origin for dtrkmf cell -> x,y conversion
x0 = 601700.0
y0 = 3597100.0

dx = 100.0
dy = 100.0

fout = open('dtrk_output.bln','w')

# read in all results for saving particle tracks
fin = open('dtrk.out','r')
results = [l.split() for l in fin.readlines()[1:]]
fin.close()

npts = len(results)

# write Surfer blanking file header
fout.write('%i,1\n' % npts)

# write x,y location and time
for pt in results:
    x = float(pt[1])*dx + x0
    y = y0 - float(pt[2])*dy
    t = float(pt[0])/7.75*4.0   # convert to 4m Culebra thickness
    fout.write('%.1f,%.1f,%.8e\n' % (x,y,t))

fout.close()
```

10.3.10 Python script `plot-results-bar-charts.py`

This script is not run on the QA linux cluster, `alice.sandia.gov`. This script is run on a desktop PC, but is only used to create figures for the analysis report. This script is only included here for completeness.

```
import numpy as np
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt

fprefix = 'pest_02/'
mprefix = '../../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 = '2005'

# 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')

## 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
ax1.plot(buff[:,0],buff[:,1],'g--') # WIPP LWB+3km
for xv,yv,n,w in zip(x,y,names,weights):
    plt.annotate('%s %i'%(n,w),xy=(xv,yv),fontsize=8)
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 in zip(x,y,names,weights):
    plt.annotate('%s %i'%(n,w),xy=(xv,yv),fontsize=8)
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 = np.arange(-10,11)
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=(-10.0,10.0),
        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.savefig('model-error-histogram-'+year+'.png')
plt.close(1)

# create bar chart plot of individual residual for ASER
# ****************************************

# separate wells into groups
resin =    res[weights==0,2]
resnear = res[weights==1,2]
resfar =  res[weights==2,2]

nin = resin.size
nnear = resnear.size
nfar = resfar.size

# separate names into groups
namin =    names[weights==0]
namnear = names[weights==1]
namfar =  names[weights==2]

# get indices that sort vectors
ordin = np.argsort(namin)
ordnear = np.argsort(namnear)
ordfar = np.argsort(namfar)

# put vectors back together (groups adjacent and sorted inside each group)
resagg = np.concatenate((resin[ordin],resnear[ordnear],resfar[ordfar]),axis=0)
namagg = np.concatenate((namin[ordin],namnear[ordnear],namfar[ordfar]),axis=0)

fig = plt.figure(1,figsize=rectfig)
ax = fig.add_subplot(111)

wid = 0.6
shift = 0.5 - wid/2.0
ab = np.arange(res.shape[0])

ax.bar(left=ab+shift,height=resagg,width=0.6,bottom=0.0,color='gray')
ax.set_ylim([-15.0,15.0])
ax.spines['bottom'].set_position('zero')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
plt.xticks(ab+wid,namagg,rotation=90)
# vertical lines dividing groups
ax.axvline(x=nin,color='black',linestyle='dashed')
ax.axvline(x=nin+nnear,color='black',linestyle='dashed')
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=(0.0,12.0),xycoords='data',
             xytext=(nin,12.0),textcoords='data',
             arrowprops={'arrowstyle':'<->'})
plt.annotate('inside WIPP LWB',xy=(nin/3.0,12.5),xycoords='data')

plt.annotate('',xy=(nin,12.0),xycoords='data',
             xytext=(nin+nnear,12.0),textcoords='data',
             arrowprops={'arrowstyle':'<->'})
plt.annotate('<3km WIPP LWB',xy=(nin+nnear/3.0,12.5),xycoords='data')

plt.annotate('',xy=(nin+nnear,12.0),xycoords='data',
             xytext=(nin+nnear+nfar,12.0),textcoords='data',
             arrowprops={'arrowstyle':'<->'})
plt.annotate('>3km WIPP LWB',xy=(nin+nnear+nfar/3.0,12.5),xycoords='data')

ax.set_ylabel('Measured-Modeled (ft)')
ax.set_title('individual residuals '+year)
plt.savefig('model-error-residuals-'+year+'.png')
plt.close(1)


# create scatter plot of measured vs. modeled
# ****************************************
m = 1.0/M2FT
sr = [2980,3120]

print 'modeled-vs-measured correlation coefficients'
print 'all data: %.4f'  % np.corrcoef(res[:,0],res[:,1])[1,0]**2
print 'inside WIPP: %.4f' % np.corrcoef(res[weights==0,0],res[weights==0,1])[1,0]**2
print 'inside 3km: %.4f' % np.corrcoef(res[weights<2,0], res[weights<2,1])[1,0]**2

print 'uncalibrated model'
print 'all data: %.4f' % np.corrcoef(ores[:,0],ores[:,1])[1,0]**2
print 'inside WIPP: %.4f ' % np.corrcoef(ores[weights==0,0],ores[weights==0,1])[1,0]**2
print 'inside 3km: %.4f' % np.corrcoef(ores[weights<2,0], ores[weights<2,1])[1,0]**2


fig = plt.figure(1,figsize=squarefig)
ax = fig.add_subplot(111)
ax.plot(res[weights==0,0],res[weights==0,1],color='red',markersize=10,
        marker='+',linestyle='none',label='Inside LWB')
ax.plot(res[weights==1,0],res[weights==1,1],color='green',markersize=10,
        marker='x',linestyle='none',label='< 3km From LWB')
ax.plot(res[weights==2,0],res[weights==2,1],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],8))
ax.set_yticks(np.linspace(sr[0],sr[1],8))
ax.set_xlim(sr)
ax.set_ylim(sr)
plt.minorticks_on()
plt.legend(loc='lower right',scatterpoints=1,numpoints=1)
plt.grid()
for j,lab in enumerate(names):
    if res[j,2] < -1.5*m:
        # plot labels to left of value far above 45-degree line
        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
        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)
plt.savefig('scatter_pest_02_'+year+'.png')
```

10.3.11 Python script `plot-contour-maps.py`

This script is not run on the QA linux cluster, `alice.sandia.gov`. This script is run on a desktop PC, but is only used to create figures for the analysis report. This script is only included here for completeness.

```python
import numpy as np
#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/31013/
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 = '2005'
fprefix = 'pest_02/'
mprefix = '../../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'

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
wellx,welly = transform(*np.loadtxt(wfname,skiprows=1,usecols=(1,2),unpack=True))
names = np.loadtxt(wfname,skiprows=1,usecols=(0,),dtype='|S6')

# 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 model, WIPP LWB & ASER contour domain (UTM X & Y)
# %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
modx,mody =   transform(*np.loadtxt(mprefix+'total_boundary.dat',
                                      unpack=True))
```

```
wippx,wippy = transform(*np.loadtxt(mprefix+'wipp_boundary.dat',
                                    usecols=(0,1),unpack=True))
aserx,asery = transform(*np.loadtxt(mprefix+'ASER_boundary.csv',
                                    delimiter=',',usecols=(1,2),unpack=True))

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')
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='o',
        markeredgecolor='green',markerfacecolor='none')
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)

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


### >>>>>>>manually fix labels>>>>
##for lab in a:
##    lab.draggable()
##plt.show()
### <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

plt.savefig('large-area-contour-map'+year+'.png')
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.plot(wippx,wippy,'k-')
ax.plot(modx,mody,'-',color='purple',linewidth=2)
ax.plot(wellx,welly,linestyle='none',marker='o',
        markeredgecolor='green',markerfacecolor='none')
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.clabel(CS,lev[::2],fmt='%i',inline_spacing=2)
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():
        # name above
        a.append(plt.annotate(n,xy=(x,y),xytext=(0,5),
                     textcoords='offset points',
                     horizontalalignment='center',
                     fontsize=10))
        # observed FW head below
        a.append(plt.annotate('%.1f'%res[j,0],xy=(x,y),xytext=(0,-15),
                     textcoords='offset points',
                     horizontalalignment='center',
                     fontsize=6))
ax.set_title('Freshwater Heads WIPP Area '+ year)
ax.set_xlabel('NAD27 NM East State Plane Easting (ft)')
ax.set_ylabel('NAD27 NM East State Plane Northing (ft)')

### >>>>>>>manually fix labels>>>>
##for lab in a:
##    lab.draggable()
##plt.show()
### <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

plt.savefig('aser-area-contour-map'+year+'.png')
plt.close(1)
```