

557660

Analysis Report for Preparation of 2000-2004 Culebra Potentiometric Surface Contour Maps

Task Number: 1.4.2.3

Report Date: 5/24/2012

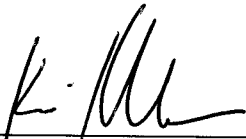



Author:	 _____ Kristopher L. Kuhlman, 6212 Repository Performance Department	<u>5/24/12</u> Date
Technical Review:	 _____ Kevin S. Barnhart, 6212 Repository Performance Department	<u>5/24/12</u> Date
QA Review:	 _____ Shelly R. Nielsen, 6210 Carlsbad Programs Group	<u>5-30-12</u> Date
Management Review:	 _____ Christi D. Leigh, 6212 Manager, Repository Performance Department	<u>5/29/12</u> Date

Table of Contents

1	Introduction	3
2	Scientific Approach	4
2.1	2000-2004 Freshwater Head and Density Data Review	4
2.2	Modeling Overview	7
2.3	Creating Average MODFLOW Simulation	8
2.4	Boundary Conditions.....	9
2.5	PEST Calibration of Averaged MODFLOW Model to Observations	10
2.6	Figures Generated from Averaged MODFLOW Model.....	11
3	2000 Results	12
3.1	2000 Freshwater Head Contours	12
3.2	2000 Particle Track.....	13
3.3	2000 Measured vs. Modeled Fit.....	13
4	2001 Results	17
4.1	2001 Freshwater Head Contours	17
4.2	2001 Particle Track.....	18
4.3	2001 Measured vs. Modeled Fit.....	18
5	2002 Results	21
5.1	2002 Freshwater Head Contours	21
5.2	2002 Particle Track.....	22
5.3	2002 Measured vs. Modeled Fit.....	22
6	2003 Results	26
6.1	2003 Freshwater Head Contours	26
6.2	2003 Particle Track.....	27
6.3	2003 Measured vs. Modeled Fit.....	27
7	2004 Results	30
7.1	2004 Freshwater Head Contours	30
7.2	2004 Particle Track.....	31
7.3	2004 Measured vs. Modeled Fit.....	31
8	Summary	34
9	References	35
10	Run Control Narrative	36
11	Appendix: Water Level and Density Data Listing	46
11.1	Input files for plotting water levels and densities	46
11.2	Listing of Water Level Plotting Script	48
11.3	Figures Generated by Python Water Level Script	63
12	Appendix: MODFLOW and Pest Files and Script Source Listings	105
12.1	Input File Listing.....	105
12.2	Output File Listing.....	106
12.3	Individual MODFLOW and Pest Script Listings.....	107

Information Only

1 Introduction

This report documents the preparation of five 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 the second half of Kuhlman (2012), the same analysis is performed on data from 2000-2004, rather than 2005-2007 data.

Historic data were taken from the Annual Site Environmental Reports (ASERs) to plot freshwater head and density¹ 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 2000-2004.

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 (2000-2004) 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.

¹ Density in units of g/cm^3 is numerically equivalent to specific gravity, the ratio of the density of any water to that of fresh water, since fresh water approximately has a density of $1 \text{ g}/\text{cm}^3$.

2 Scientific Approach

2.1 2000-2004 Freshwater Head and Density Data Review

As in Kuhlman (2012), data reported in past ASERs are plotted to ensure consistency and explain anomalies. Python scripts and resulting data plots at each well are listed in Sections 11.2 and 11.3. Table 1 summarizes freshwater heads, measurement dates, and Culebra groundwater densities for all five years included in the current analysis.

Water level and freshwater head values (and measurement dates) were obtained from Table F-8 in the 2001-2004 ASERs. Water level data were not reported in the 2000 ASER, but were instead obtained directly from WIPP personnel (Watterson, 2012). Culebra densities were estimated from historic Troll data for 2003 (Johnson, 2012a) and 2004 (Johnson, 2012b). No Troll data were available for the years 2000-2002. Culebra midpoint elevations were obtained from Johnson (2008) and SAND89-7068 (Cauffman et al., 1990). 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? 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 selected beginning time is prior to the beginning of this analysis and the chosen end time is in the future. 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. To increase consistency across years, all pre-2007 water level elevations were adjusted to use the newer reference point elevation.

The set of wells used for calibration targets in this report is different from the wells used in the 2005-2007 report (Kuhlman, 2012) because of the evolution of the Culebra groundwater monitoring network through time. AEC-7 was re-perforated at the Culebra in 2004 and did not have representative water levels until it was again re-perforated in 2008. Only one of the SNL-series wells (SNL-12) existed and had representative water levels during the 2000-2004 timeframe. Several wells were converted from Culebra to Magenta using bridge plugs in 2001 (with a permanent plugback in 2003), including H-14, H-15, H-18, and WIPP-18. DOE-1, WIPP-12, WIPP-21, WIPP-25, WIPP-26 and P-15 were plugged and abandoned without replacement wells. C-2737, IMC-461, and SNL-12 were drilled during the 2000-2004 timeframe and therefore do not have water levels for each year. On the H-07, H-09, and H-10 wellpads there were redundant Culebra monitoring wells, and the primary Culebra monitoring well was changed affecting the 2000-2004 timeframe. H-07b1 was a redundant well (quarterly measurements), but became the primary H-07 Culebra observation ASER well (monthly measurements) in 2005 when H-07b2 was plugged and abandoned. H-09c was a redundant Culebra well on the H-09 pad, but became the primary Culebra well when H-09a and H-09b were plugged in 2002. H-10c was converted to a Culebra

monitoring well by perforating the casing, to replace H-10b, which was plugged and abandoned at the same time in early 2002. At both the H-07 and H-09 well pads we used the same Culebra monitoring well through time, despite the historic shift in primary Culebra observation well at each pad. This was done to maintain as much consistency as possible (although earlier water level observations are only quarterly).

The data review revealed August to be the best contour month in 2004 because water levels were impacted in many wells due to a series of large precipitation events in September 2004. Wells in Nash Draw (e.g., WIPP-25 and WIPP-26) rose significantly due to this event. At AEC-7 we used a March water level because the well was re-perforated in April 2004, which resulted in abnormal water levels until 2008. At H-07b1 we used a September water level because prior to 2005 the water level in this well was only measured quarterly (H-07b2 was the primary well on the H-07 well pad until it was plugged and abandoned in May 2005). At IMC-461 and SNL-12 we used December water levels because they were both drilled new in late 2003, requiring several months to stabilize after drilling and well development activities. At WQSP-2 we used a July water level because of an anomalous water level, which might be due to sampling activities.

The data review revealed September to be a good month for contouring in 2003. At C-2737 we used a March water level because the well was configured for testing in the Magenta the remainder of 2003.

The data review revealed December 2002 to be the best contouring month in 2002 (the 2002 ASER also used December). At WQSP-2 and WQSP-3 we used September water levels because of drawdown due to sampling activities in October 2002.

The data review revealed December 2001 to be the best contouring month in 2001 (the 2001 ASER also used December). At WQSP-5, WQSP-6, WQSP-2 and WQSP-3 we again used later water levels because of drawdown due to sampling activities later in 2001.

The data review found December 2000 to be the best contouring month in 2000 (the 2000 ASER also used December). At ERDA-9 we used a September water level because of anomalously low water levels near the end of 2000. At H-09c we used a June water level because before 2002 only quarterly water levels were measured in this well (H-09B was the primary Culebra well on this pad until it was plugged in February 2002 during pressure-grouting activities in H-09a). At H-14 we used a September water level because. At P-15 we used a January water level because December was an anomalously low water level. At WIPP-30 we used a June water level because of low water levels following well-maintenance activities in October. At WQSP-2, WQSP-3, WQSP-5, and WQSP-6 we used water levels earlier in the year to avoid periods with residual drawdown from sampling events.

Information Only

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

	2004			2003			2002			2001			2000		
	date	FWH	SG	date	FWH	SG	date	FWH	SG	date	FWH	SG	date	FWH	SG
AEC-7	03/09/04	933.1	1.08	09/10/03	933.4	1.08	12/03/02	932.8	1.08	12/04/01	932.9	1.08	12/07/00	932.9	1.08
C-2737	08/11/04	919.4	1.01	03/12/03	920.1	1.00	12/04/02	920.0	1.00						
DOE-1				09/10/03	917.0	1.08	12/04/02	916.9	1.08	12/05/01	916.5	1.08	12/05/00	915.9	1.08
ERDA-9	08/11/04	923.8	1.06	09/10/03	923.9	1.06	12/02/02	923.8	1.06	12/05/01	923.4	1.06	09/11/00	923.3	1.06
H-02b2	08/11/04	926.6	1.00	09/09/03	926.9	1.00	12/04/02	926.8	1.00	12/05/01	926.4	1.00	12/06/00	926.1	1.00
H-03b2	08/11/04	918.0	1.04	09/10/03	918.2	1.04	12/02/02	918.1	1.04	12/05/01	917.8	1.04	12/06/00	917.4	1.04
H-04b	08/11/04	915.7	1.01	09/09/03	915.4	1.01	12/02/02	915.7	1.01	12/05/01	915.5	1.01	12/06/00	915.3	1.01
H-05b	08/10/04	937.3	1.10	09/10/03	937.4	1.10	12/03/02	937.1	1.10	12/04/01	937.0	1.10	12/07/00	936.9	1.10
H-06b	08/10/04	933.9	1.04	09/08/03	934.4	1.04	12/02/02	934.6	1.04	12/05/01	934.1	1.04	12/07/00	933.8	1.04
H-07b1	09/13/04	913.7	1.00	09/08/03	913.6	1.00	12/03/02	913.6	1.00	12/03/01	913.6	1.00	12/05/00	913.6	1.00
H-09c	08/10/04	912.9	1.00	09/08/03	911.7	1.00				12/04/01	912.0	1.00	06/06/00	911.5	1.00
H-10b										12/04/01	922.4	1.04	12/05/00	922.4	1.04
H-10c	08/10/04	922.1	1.00	09/09/03	922.1	1.00	12/03/02	922.3	1.00						
H-11b4	08/11/04	916.3	1.07	09/10/03	915.9	1.07	12/04/02	916.1	1.07	12/03/01	916.3	1.07	12/05/00	915.9	1.07
H-12				09/09/03	916.7	1.09	12/03/02	916.5	1.09	12/04/01	916.2	1.09	12/05/00	915.9	1.09
H-14													09/11/00	918.0	1.01
H-15													12/06/00	919.0	1.15
H-17	08/11/04	915.4	1.13	09/10/03	914.9	1.13	12/04/02	915.2	1.13	12/04/01	914.8	1.13	12/05/00	914.3	1.13
H-18													12/07/00	937.3	1.04
H-19b0	08/11/04	918.2	1.06	09/10/03	918.3	1.06	12/04/02	918.3	1.06	12/05/01	917.9	1.06	12/06/00	917.6	1.06
IMC-461	12/06/04	928.1	1.00												
P-15													01/19/00	919.0	1.01
P-17	08/11/04	914.4	1.07	09/10/03	914.1	1.07	12/04/02	914.3	1.07	12/03/01	913.9	1.07	12/05/00	914.0	1.07
SNL-12	12/07/04	915.2	1.00												
WIPP-12	08/11/04	935.7	1.10	09/09/03	936.3	1.10	12/02/02	936.4	1.10	12/05/01	936.0	1.10	12/06/00	935.8	1.10
WIPP-13	08/10/04	937.2	1.05	09/08/03	937.8	1.05	12/02/02	938.0	1.05	12/05/01	937.7	1.05	12/07/00	937.8	1.05
WIPP-18													12/06/00	935.9	1.10
WIPP-19	08/11/04	933.2	1.05	09/09/03	933.7	1.05	12/02/02	933.7	1.05	12/05/01	933.2	1.05	12/06/00	932.9	1.05
WIPP-21	08/11/04	926.8	1.07	09/09/03	927.0	1.07	12/02/02	927.0	1.07	12/05/01	926.6	1.07	12/06/00	926.3	1.07
WIPP-22	08/11/04	933.0	1.08	09/09/03	933.4	1.08	12/02/02	933.4	1.08	12/05/01	933.0	1.08	12/06/00	932.6	1.08
WIPP-25	08/09/04	933.4	1.01	09/08/03	933.9	1.01	12/03/02	934.3	1.01	12/04/01	933.8	1.01	12/07/00	932.8	1.01
WIPP-26	08/09/04	921.2	1.00	09/08/03	921.3	1.00	12/03/02	921.6	1.00	12/05/01	921.1	1.00	12/04/00	921.2	1.00
WIPP-30	08/10/04	937.4	1.01	09/09/03	937.9	1.01	12/03/02	937.8	1.01	12/03/01	937.0	1.01	06/05/00	936.5	1.01
WQSP-1	08/11/04	935.7	1.04	09/09/03	936.3	1.04	12/02/02	936.3	1.04	12/05/01	935.8	1.04	12/07/00	935.6	1.04
WQSP-2	07/08/04	938.4	1.04	09/09/03	939.1	1.04	09/09/02	939.1	1.04	09/06/01	938.8	1.04	09/12/00	938.8	1.04
WQSP-3	08/11/04	935.3	1.14	09/09/03	935.8	1.14	09/09/02	935.8	1.14	09/06/01	935.4	1.14	09/11/00	935.6	1.14
WQSP-4	08/11/04	918.3	1.07	09/09/03	918.4	1.07	12/02/02	918.5	1.07	12/05/01	918.1	1.07	12/05/00	917.8	1.07
WQSP-5	08/11/04	917.5	1.02	09/09/03	917.8	1.02	12/02/02	917.7	1.02	10/09/01	917.4	1.02	10/11/00	917.0	1.02
WQSP-6	08/11/04	921.0	1.01	09/09/03	921.0	1.01	12/02/02	920.6	1.01	10/10/01	920.5	1.01	11/09/00	920.3	1.01

C-2737	Drilled new in February 2001
DOE-1	November 2003, denser water began leaking into well (data not reliable)
H-09c	Culebra not accessible in 2002 due to bridge plug below Magenta for perforating and testing activities.
H-10b	Plugged and abandoned in February 2002
H-10c	Converted to Culebra in February 2002
H-12	Obstruction in well prevented monitoring in 2004
H-14	Bridge plug set below Magenta in March 2001
H-15	Configured for Magenta in February 2001
H-18	Bridge plug set below Magenta in March 2001
IMC-461	Drilled new in October 2003
P-15	Plugged and abandoned in February 2002
SNL-12	Drilled new in June 2003
WIPP-18	Bridge plug set below Magenta in March 2001

Information Only

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 2000-2004 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

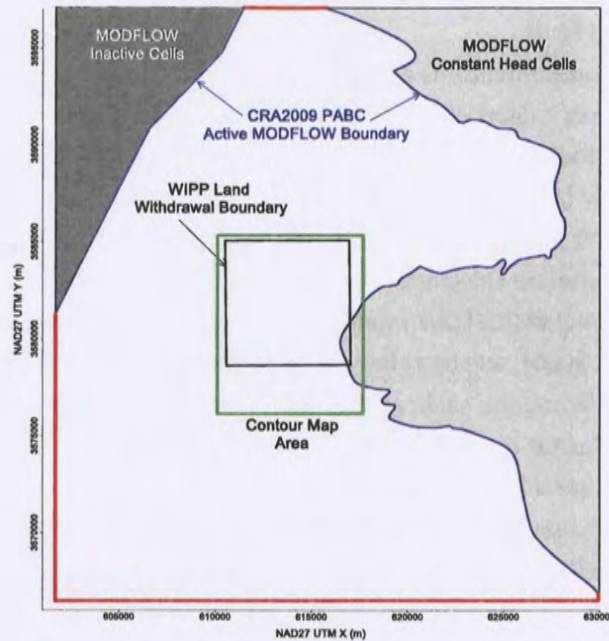


Figure 1. MODFLOW-2000 model domain, adjusted boundary conditions shown in red, contour area outlined in green.

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 (2001-2004) and Waterson (2012) for 2000 data; these water levels 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:

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

where $\text{abs}(y)$ is absolute value and $\text{sign}(y)$ is the function returning 1 for $y > 0$, -1 for $y < 0$ and 0 for $y = 0$ and x and y are coordinates scaled to the range $-1 \leq \{x, y\} \leq 1$. In Hart et al. (2009), the values $A=928.0$, $B=8.0$, $C=1.2$, $D=1.0$, $\alpha=0.5$, $E=1.0$, 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 area of interest, where there is a steep-sloping piezometric surface. 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.

In all the contour maps presented in this report (2000-2004), AEC-7 has a large misfit for two reasons. First, this well has historically had an anomalously low freshwater head elevation, lower than wells around it in all directions. Secondly, it did not have a May 2007 observation (due to ongoing well reconfiguration activities) and therefore was not included as a calibration target in the PA MODFLOW model calibration. The ensemble-average T, A, and R fields used here were not calibrated to accommodate this observation. This well is situated in a low-transmissivity region, and near the constant-head boundary associated with the halite margin, therefore PEST will not be able to improve this fit solely through adjustment of the boundary conditions indicated with red in Figure 1.

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 2000 Results

3.1 2000 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.

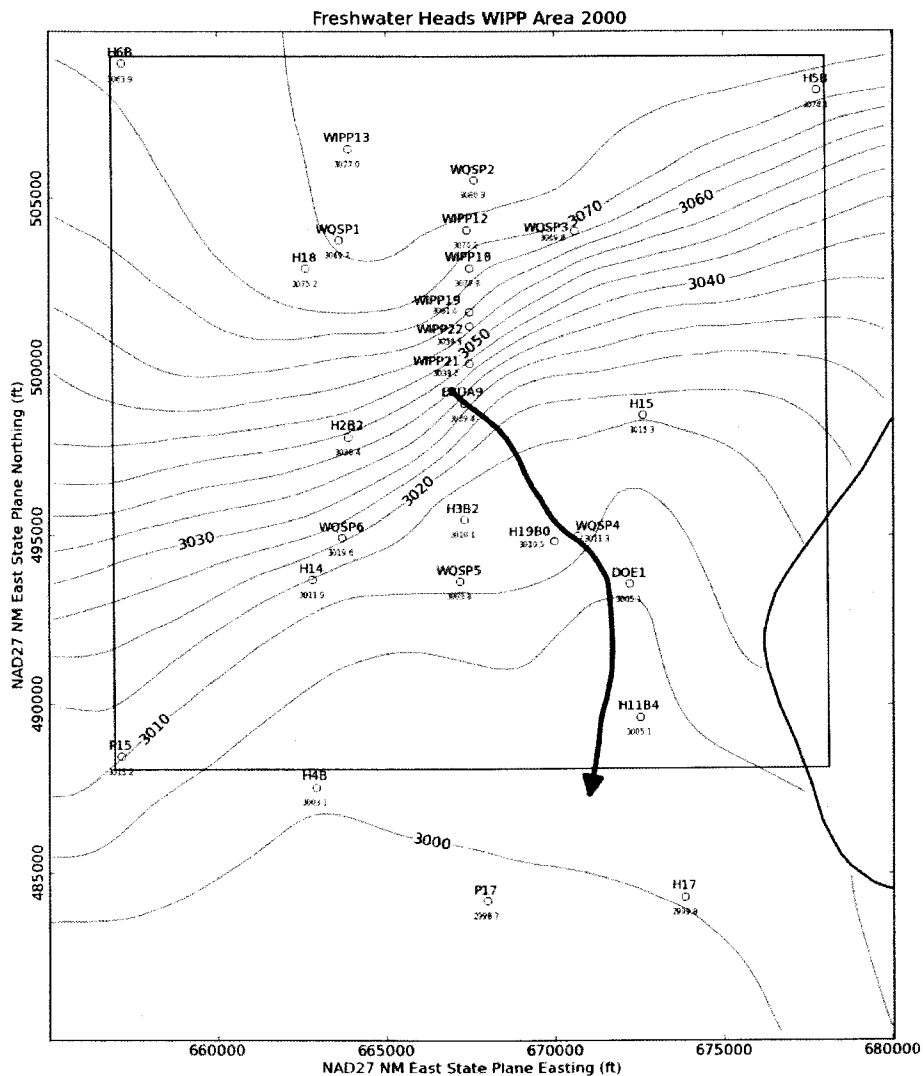


Figure 2. Model-generated December 2000 freshwater head contours with observed head listed at each well (5-foot contour interval) with blue water particle track from waste handling shaft to WIPP LWB. Purple curve is halite margin.

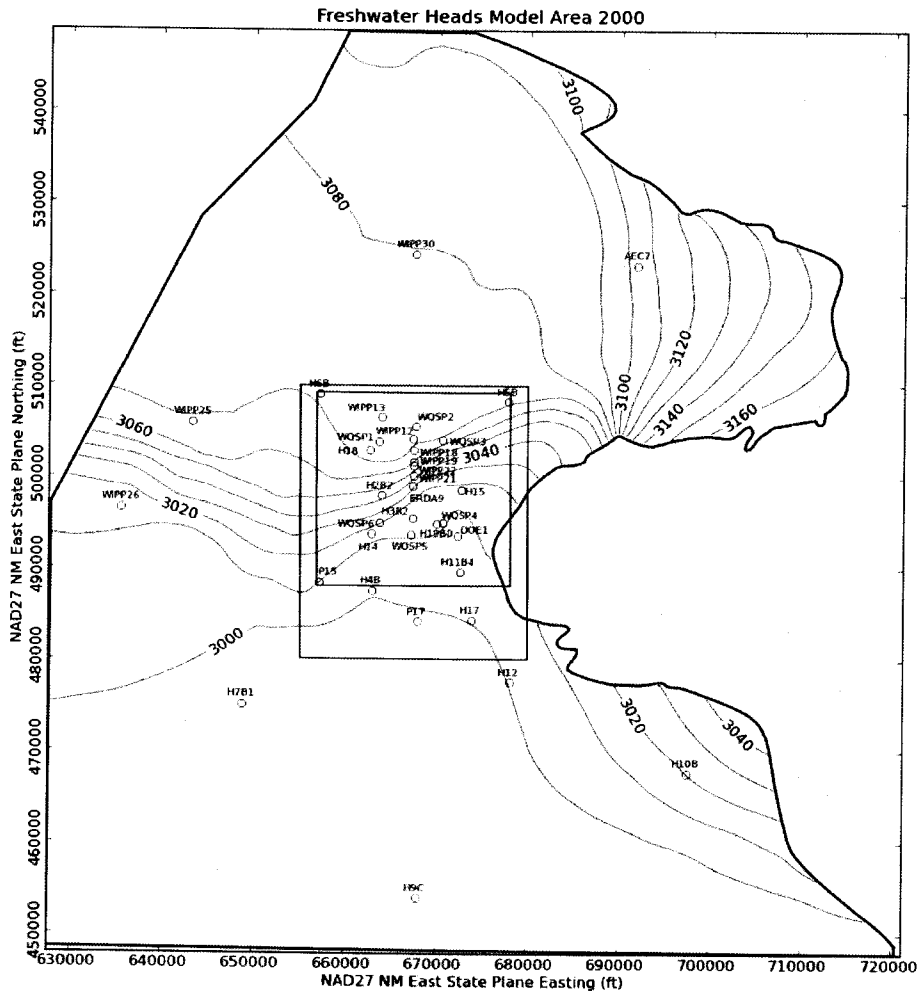


Figure 3. MODFLOW-modeled December 2000 heads for entire model domain (10-foot contour interval). Green rectangle indicates region contoured in Figure 2, black square is WIPP LWB.

3.2 2000 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 4086 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 5752 years (output from DTRKMF is adjusted from an original 7.75-m Culebra thickness). This is an average velocity of 0.71 m/yr.

3.3 2000 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. AEC-7 was given a low weight (0.01), to prevent its large residual from dominating the optimization. Additional observations representing the average heads north of the LWB and south of the LWB were used to help prevent over-smoothing of the estimated

results across the LWB. This allowed PEST to improve the fit of the model to observed heads inside the area contoured in Figure 2, at the expense of fitting wells closer to the boundary conditions.

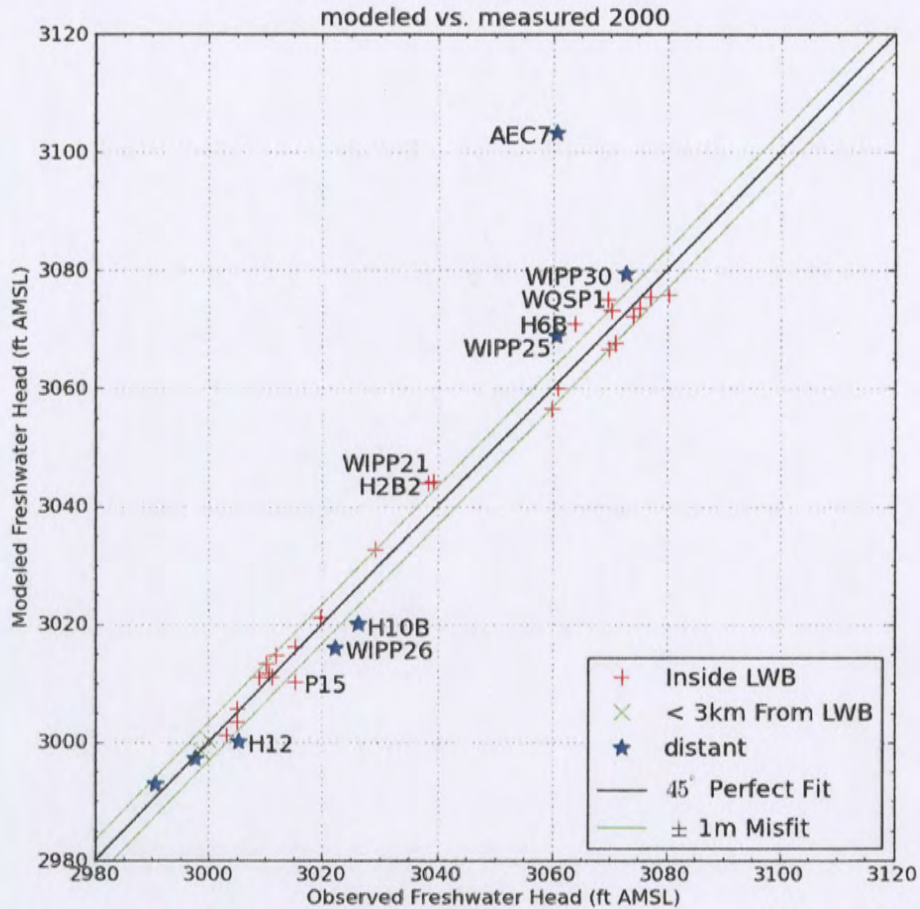


Figure 4. Measured vs. modeled scatter plot for averaged MODFLOW model generated heads and December 2000 observed freshwater heads

The central black diagonal line in Figure 4 represents a perfect model fit (1:1 or 45-degree slope); the two green lines on either side of this represent a 1-m misfit above or below the perfect fit. Wells more than 1.5 m from the 1:1 line are labeled. AEC-7 has a large misfit (42.6 ft), for the two reasons given in Section 2.4. The calibrated parameters (for equation 1) were $A=927.5$, $B=8.02$, $C=1.19$, $D=0.955$, $E=1.30$, $F=-1.16$, and $\alpha=0.0065$. The exponent α on the y-variation had the largest relative change (-99%) 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 (30%). The boundary conditions at the west edge of the model domain are impacted most by both 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 greater than 94%, and the calibration improved the R^2 value very slightly (third decimal place) inside the WIPP LWB, at the expense of the calibration of wells distant to the WIPP LWB. The distribution in Figure 5 does not have a strong bias.

Table 3. 2000 Measured vs. Modeled correlation coefficients

	dataset	measured vs. modeled R ²
Uncalibrated	wells inside WIPP LWB	0.987
	wells <3km from WIPP LWB	0.989
	all wells	0.941
Calibrated	wells inside WIPP LWB	0.989
	wells <3km from WIPP LWB	0.989
	all wells	0.940

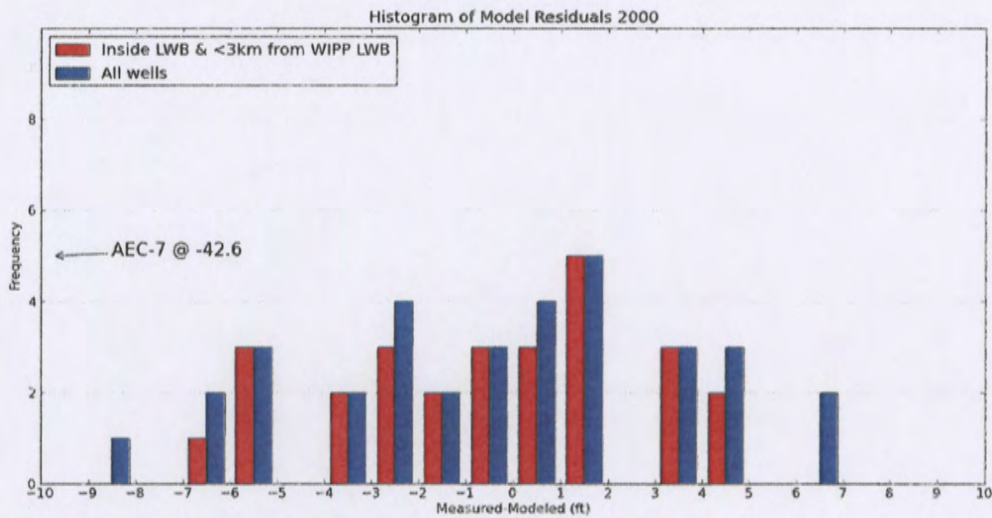


Figure 5. Histogram of Measured-Modeled errors for 2000

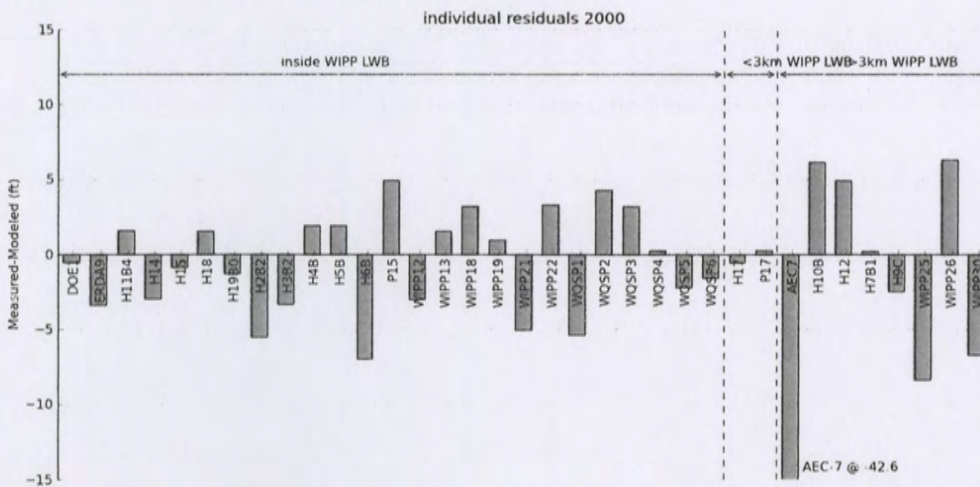


Figure 6. Measured-Modeled errors at each well location for 2000

The model fit to the December 2000 observations is good. The averaged MODFLOW model captures the bulk Culebra flow behavior, while the PEST calibration improved the model fit to the specific December

2000 observations. The results are good considering several wells (H-14, H-15, H-18, P-15, and WIPP-18) were used in the average model calibration that did not exist as Culebra monitoring wells after 2000, and therefore were not included in the steady-state T-Field calibration exercise for CRA-2009 PABC.

Information Only

4 2001 Results

4.1 2001 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.

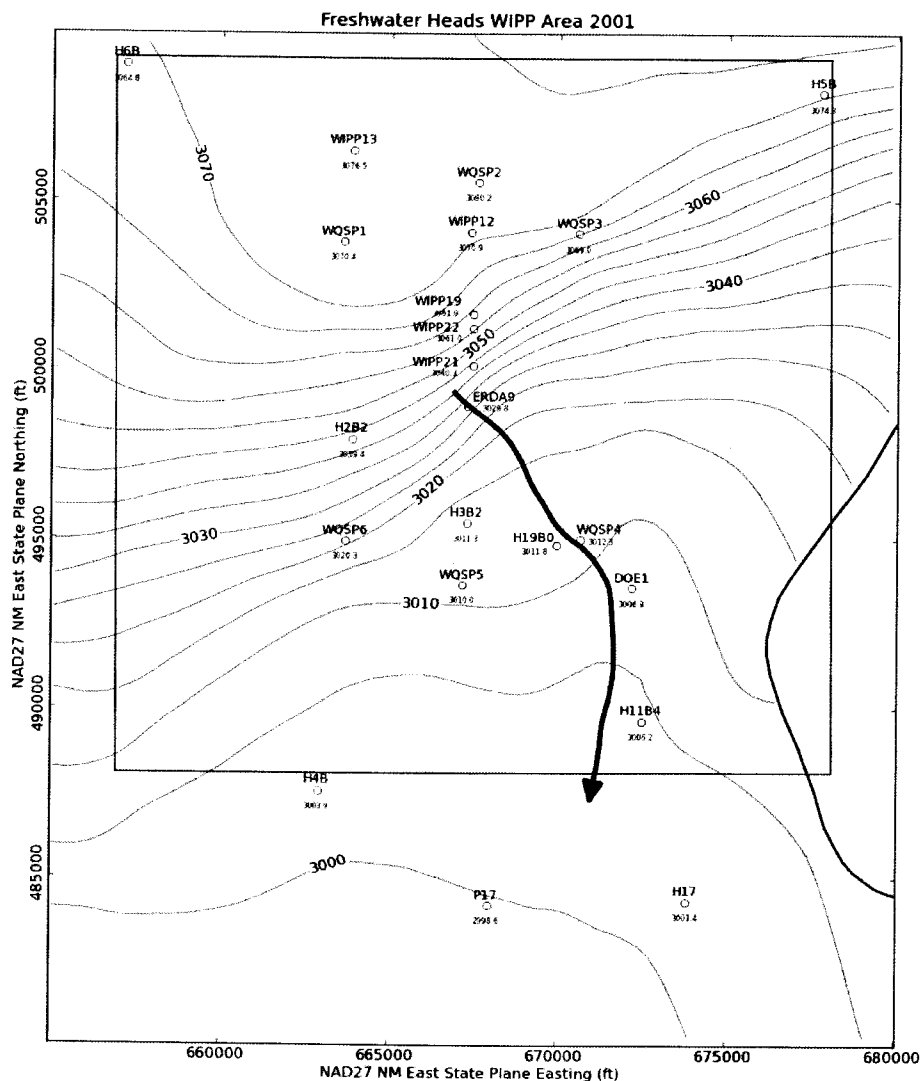


Figure 7. Model-generated December 2001 freshwater head contours with observed head listed at each well (5-foot contour interval) with blue water particle track from waste handling shaft to WIPP LWB

Information Only

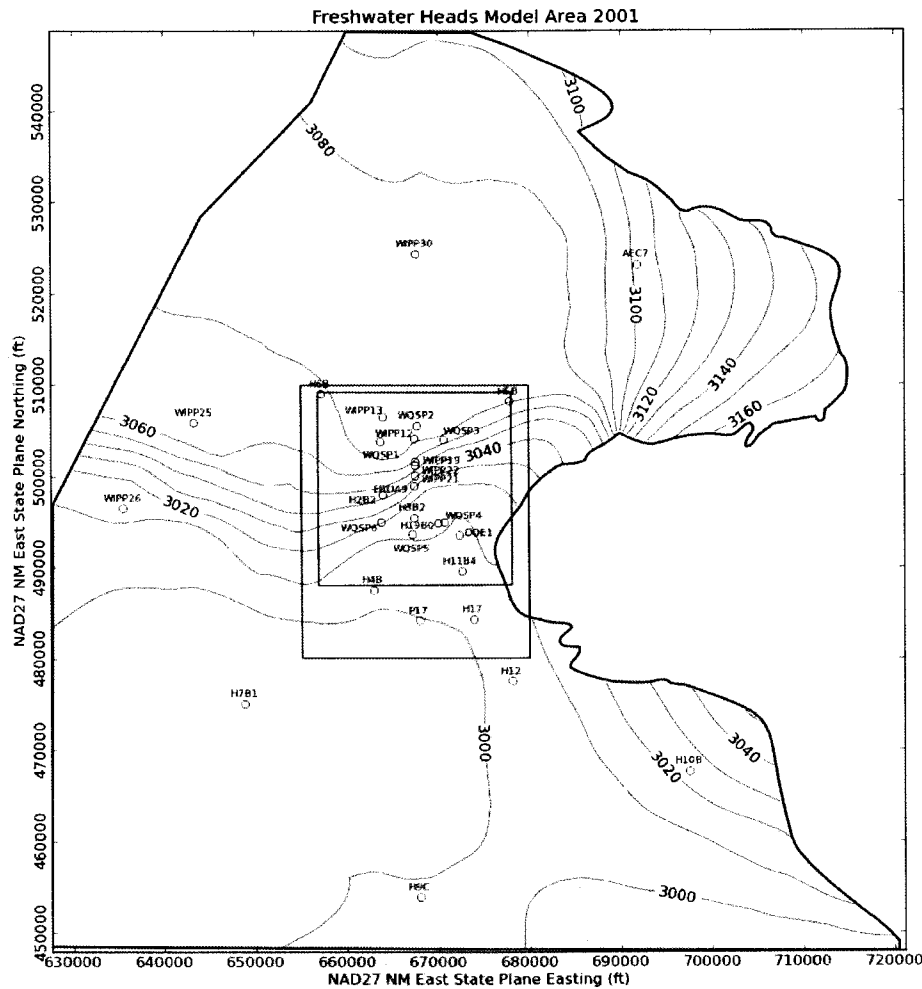


Figure 8. MODFLOW-modeled December 2001 heads for entire model domain (10-foot contour interval). Green rectangle indicates region contoured in Figure 7, black square is WIPP LWB.

4.2 2001 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 4080 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 6082 years (output from DTRKMF is adjusted from an original 7.75-m Culebra thickness). This is an average velocity of 0.67 m/yr.

4.3 2001 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. AEC-7 was given a low weight (0.01), to prevent its large residual from dominating the optimization. Additional observations representing the average heads north of the LWB and south of the LWB were used to help prevent over-smoothing of the estimated

results across the LWB. This allowed PEST to improve the fit of the model to observed heads inside the area contoured in Figure 7, at the expense of fitting wells closer to the boundary conditions.

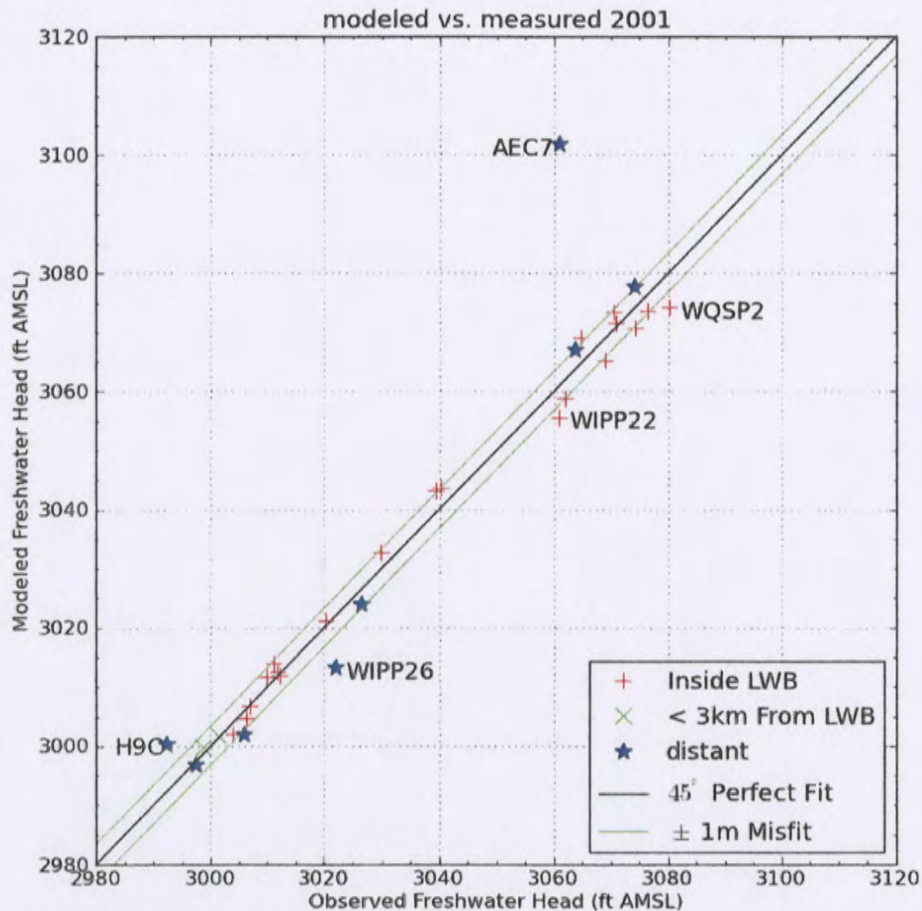


Figure 9. Measured vs. modeled scatter plot for averaged MODFLOW model generated heads and December 2001 observed freshwater heads

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. AEC-7 has a large misfit (41 ft), for the two reasons given in Section 2.4. The calibrated parameters (for equation 1) were $A=928.7$, $B=8.11$, $C=2.20$, $D=0.727$, $E=2.08$, $F=-1.82$, and $\alpha=0.065$. The parameter E, the coefficient controlling the cubic behavior in the x-direction, had the largest relative change (108%) compared to the starting values. The boundary conditions along the northern and southern edges of the model are impacted most by the x-axis coefficients.

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 93%, the calibration improved the R^2 fit inside and near the WIPP LWB, at the expense of the fit to wells distant to the WIPP LWB. The distribution in Figure 10 is roughly symmetric about zero, indicating there is not a strong bias.

Table 4. 2001 Measured vs. Modeled correlation coefficients

	dataset	measured vs. modeled R ²
Uncalibrated	wells inside WIPP LWB	0.987
	wells <3km from WIPP LWB	0.989
	all wells	0.938
Calibrated	wells inside WIPP LWB	0.989
	wells <3km from WIPP LWB	0.991
	all wells	0.932

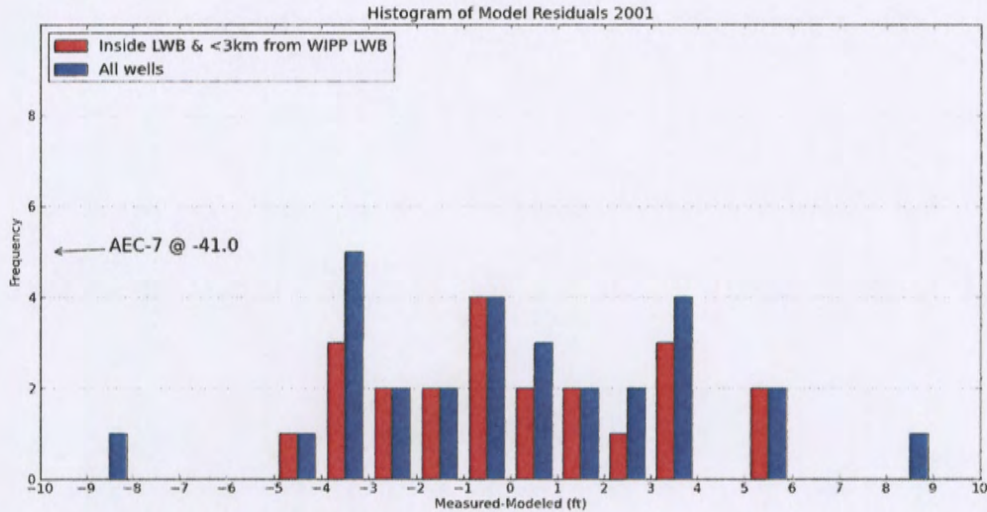


Figure 10. Histogram of Measured-Modeled errors for 2001

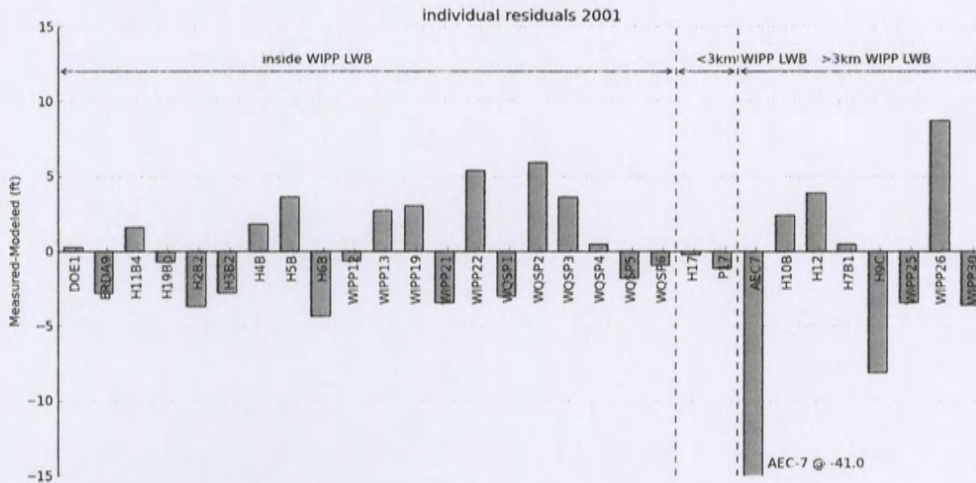


Figure 11. Measured-Modeled errors at each well for 2001

Aside from AEC-7 and to a lesser extent two more distant wells (H-09C far south of WIPP and WIPP-26 in Nash Draw), the model fit to the December 2001 observations is good. The averaged MODFLOW model captures the bulk Culebra flow behavior, while the PEST calibration improved the model fit to the specific December 2001 observations.

5 2002 Results

5.1 2002 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 uncountoured 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.

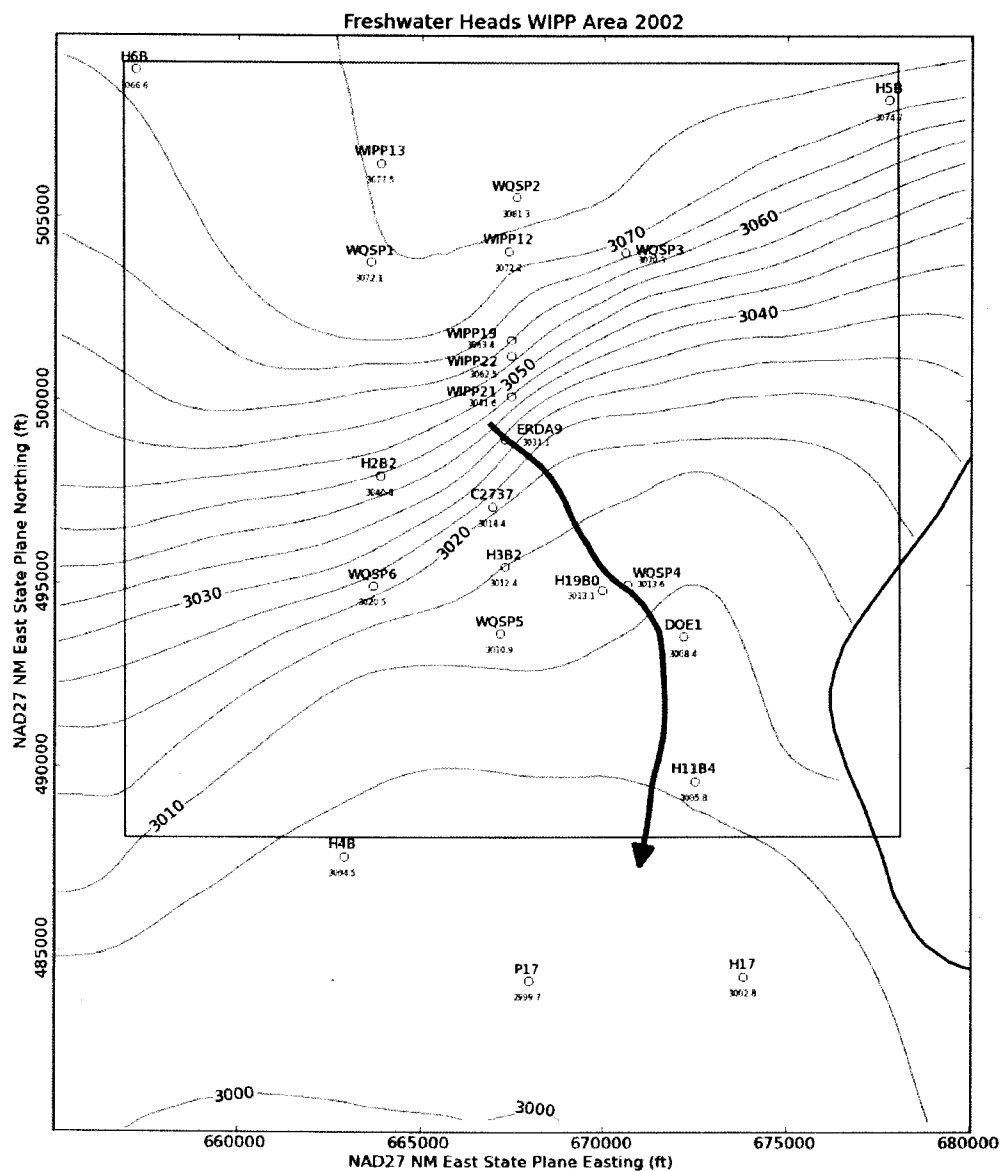


Figure 12. Model-generated December 2002 freshwater head contours with observed head listed at each well (5-foot contour interval) with blue water particle track from waste handling shaft to WIPP LWB

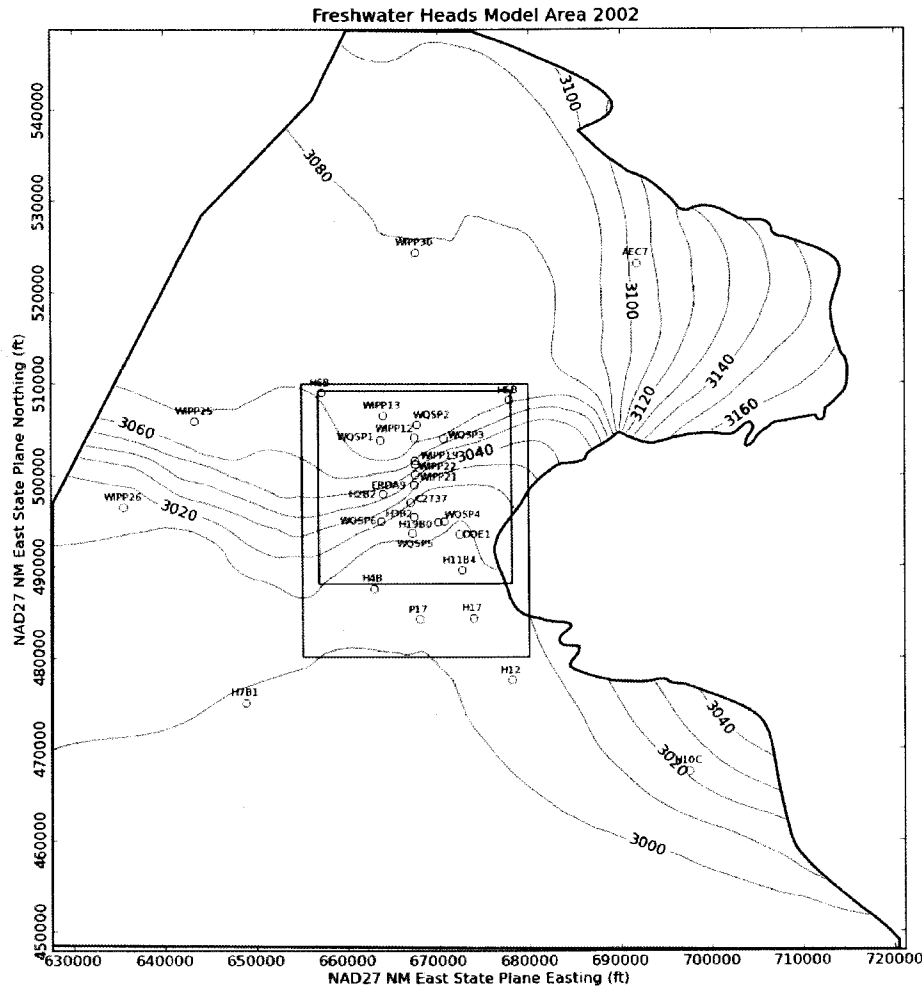


Figure 13. MODFLOW-modeled December 2002 heads for entire model domain (10-foot contour interval). Green rectangle indicates region contoured in Figure 12, black square is WIPP LWB.

5.2 2002 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 4086 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 5942 years (output from DTRKMF is adjusted from an original 7.75-m Culebra thickness). This is an average velocity of 0.69 m/yr.

5.3 2002 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. AEC-7 was given a low weight (0.01), to prevent its large residual from dominating the optimization. Additional observations representing the average heads north of the LWB and south of the LWB were used to help prevent over-smoothing of the

estimated results across the LWB. This allowed PEST to improve the fit of the model to observed heads inside the area contoured in Figure 12, at the expense of fitting wells closer to the boundary conditions.

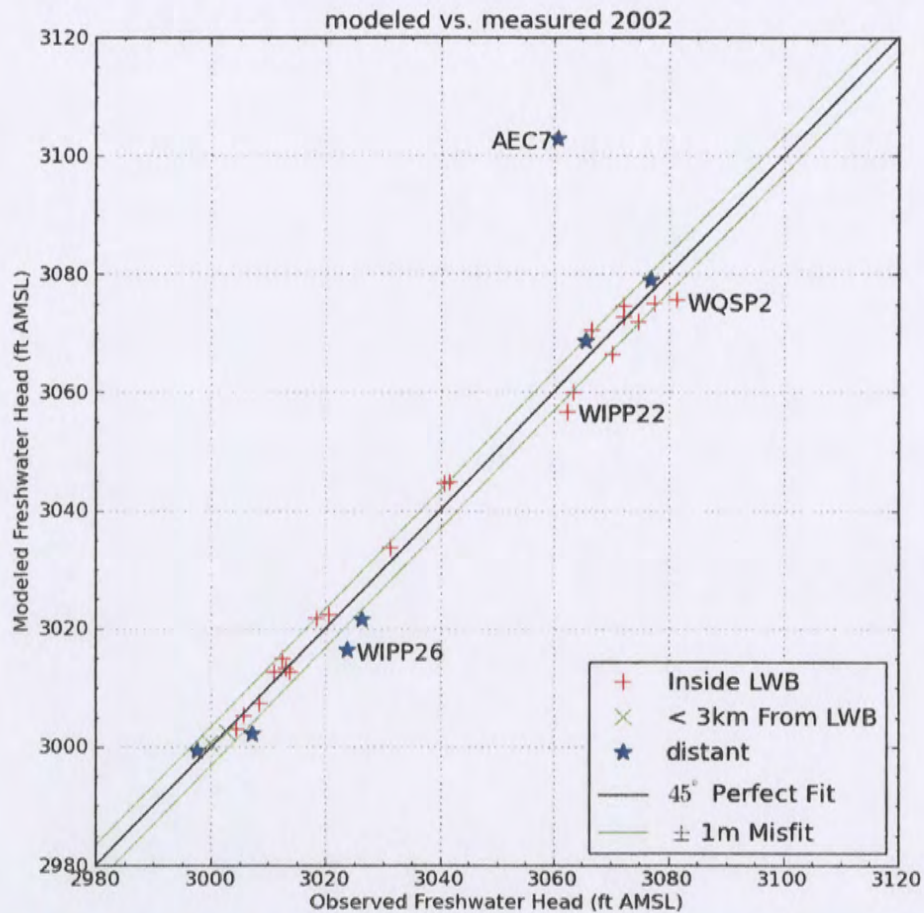


Figure 14. Measured vs. modeled scatter plot for averaged MODFLOW model generated heads and December 2002 observed freshwater heads

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. AEC-7 has a large misfit (42.3 ft), for the two reasons given in Section 2.4. 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 $\alpha=-0.052$. 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 zero, indicating there is not a strong bias.

Table 5. 2002 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	0.931
Calibrated	wells inside WIPP LWB	0.989
	wells <3km from WIPP LWB	0.990
	all wells	0.929

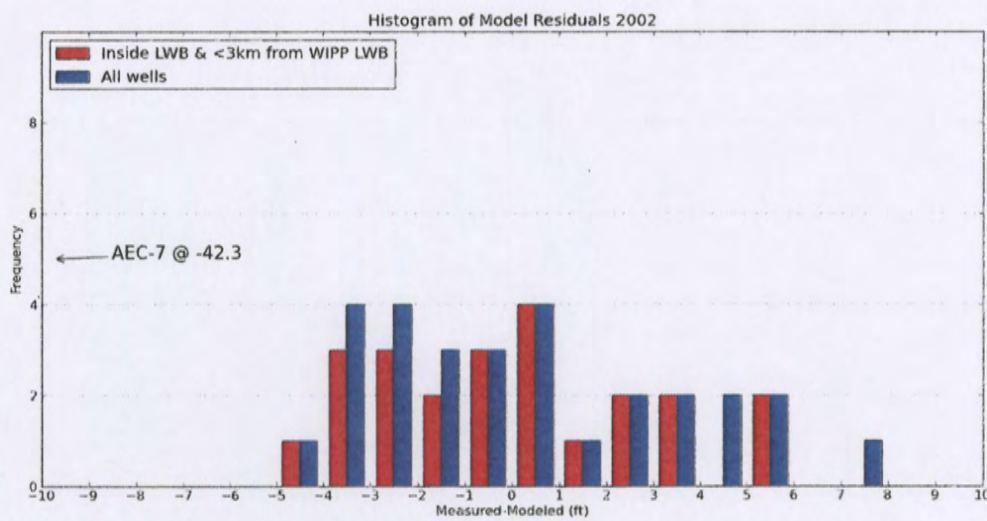


Figure 15. Histogram of Measured-Modeled errors for 2002

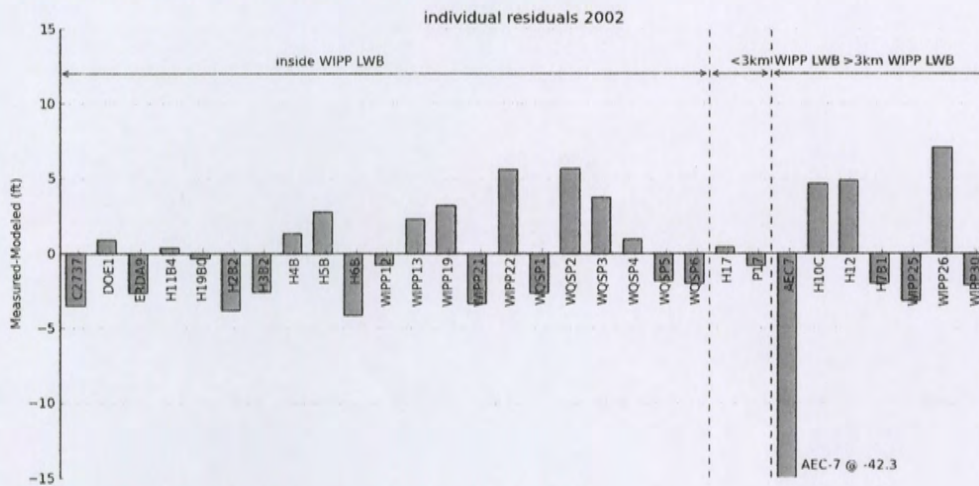


Figure 16. Measured-Modeled errors at each well for 2002

Aside from AEC-7, and to a lesser extent WIPP-26 (which is in Nash Draw), the model fit to the December 2002 observations is very good. The averaged MODFLOW model captures the bulk Culebra flow behavior, while the PEST calibration improved model fit to the December 2002 observations.

Information Only

6 2003 Results

6.1 2003 Freshwater Head Contours

The model-generated freshwater head contours are given in Figure 17 and Figure 18. 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.

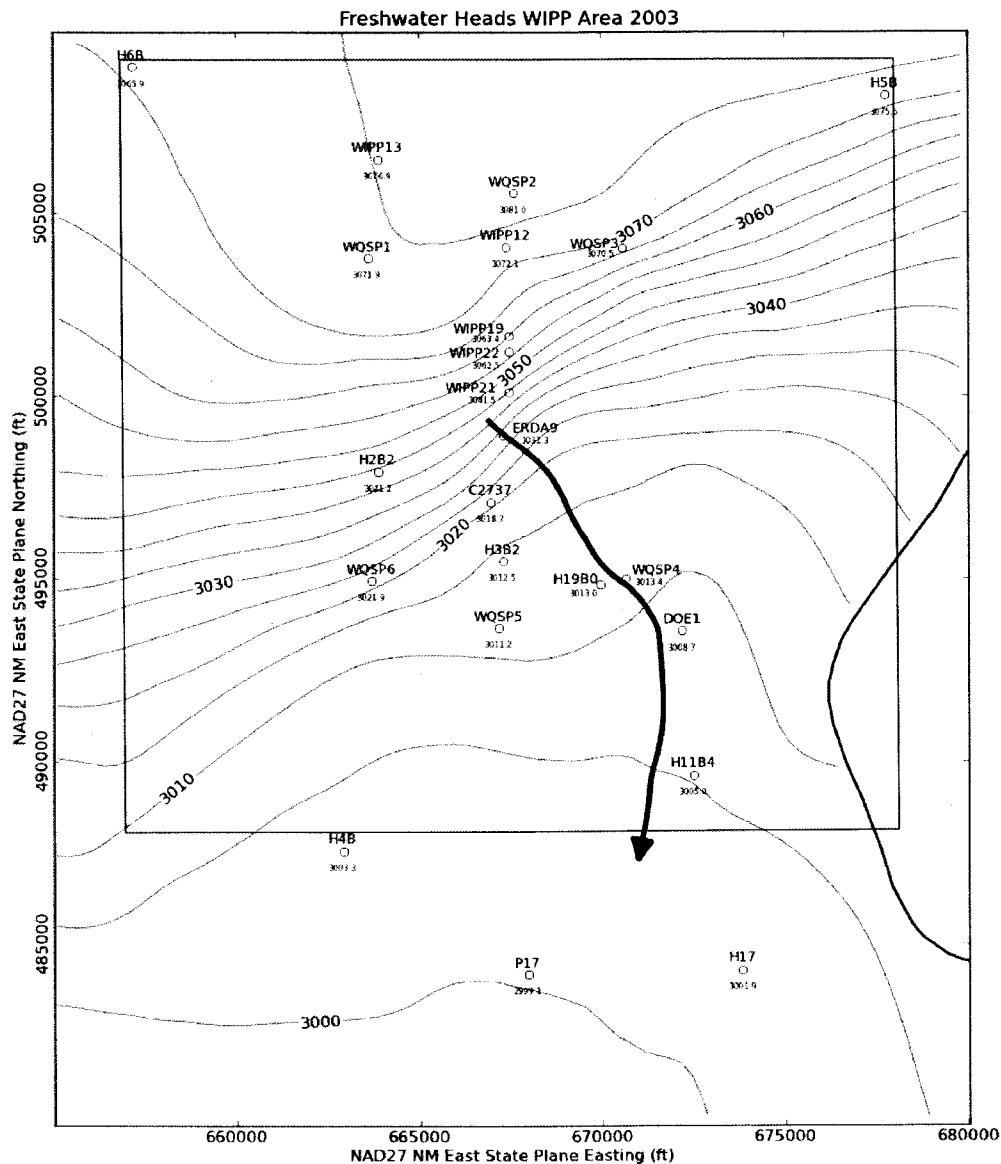


Figure 17. Model-generated September 2003 freshwater head contours with observed head listed at each well (5-foot contour interval) with blue water particle track from waste handling shaft to WIPP LWB

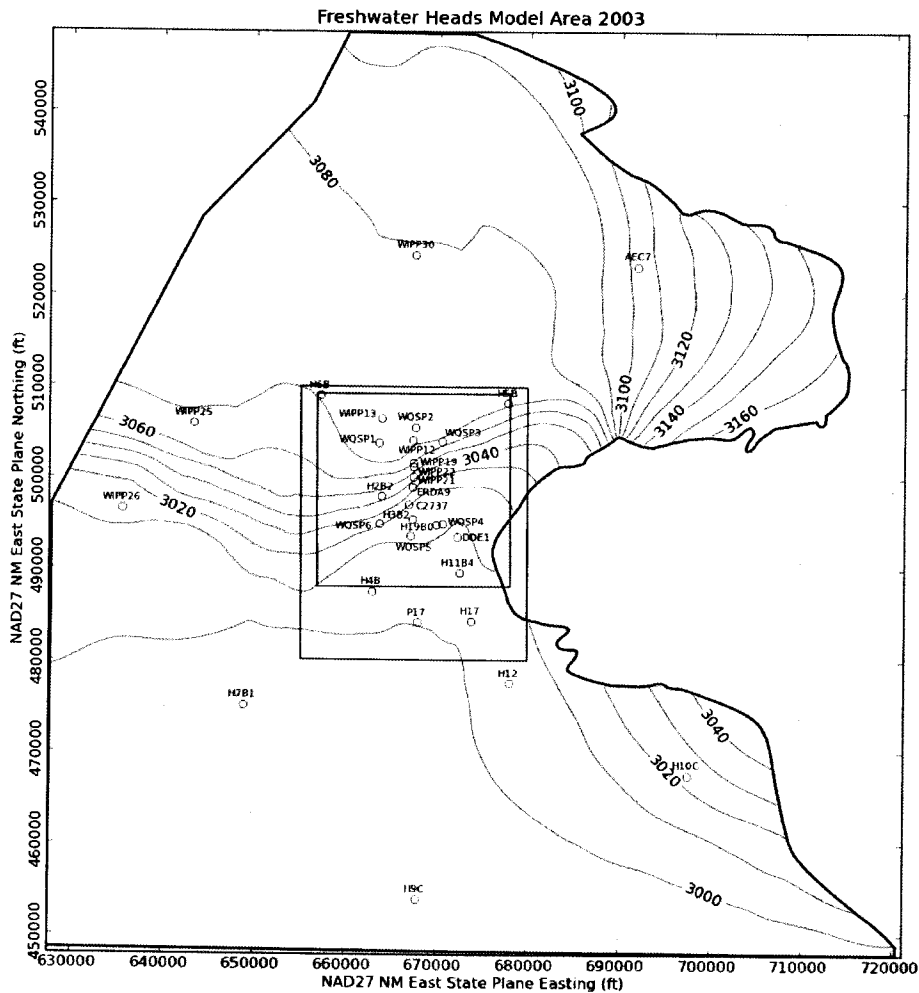


Figure 18. MODFLOW-modeled September 2003 heads for entire model domain (10-foot contour interval). Green rectangle indicates region contoured in Figure 17, black square is WIPP LWB.

6.2 2003 Particle Track

The blue arrow line in Figure 17 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 4082 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 5984 years (output from DTRKMF is adjusted from an original 7.75-m Culebra thickness). This is an average velocity of 0.68 m/yr.

6.3 2003 Measured vs. Modeled Fit

The scatter plot in Figure 19 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. AEC-7 was given a low weight (0.01), to prevent its large residual from dominating the optimization. Additional observations representing the average heads north of the LWB and south of the LWB were used to help prevent over-smoothing of the

estimated results across the LWB. This allowed PEST to improve the fit of the model to observed heads inside the area contoured in Figure 17, at the expense of fitting wells closer to the boundary conditions.

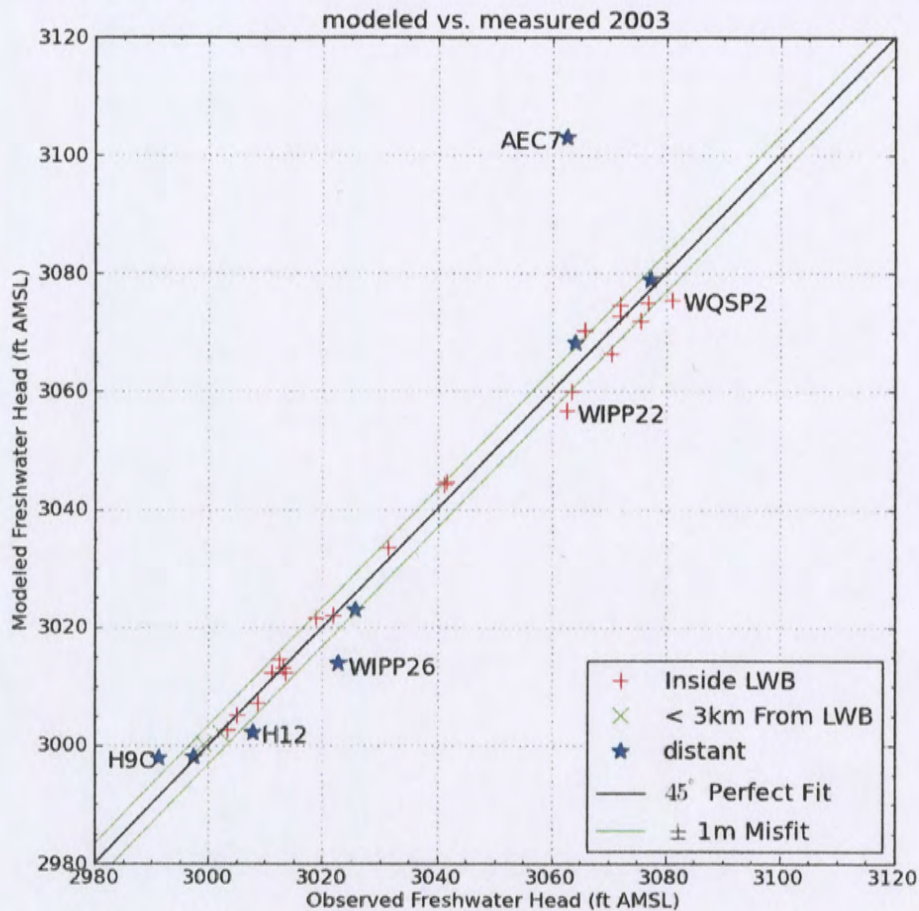


Figure 19. Measured vs. modeled scatter plot for averaged MODFLOW model generated heads and September 2003 observed freshwater heads

The black central diagonal line in Figure 19 represents a perfect model fit (1:1 or 45-degree slope); the two green lines on either side of this represent a 1-m misfit above or below the perfect fit. Wells more than 1.5 m from the 1:1 line are labeled. AEC-7 has a large misfit (40.4 ft), for the two reasons given in Section 2.4. The calibrated parameters (for equation 1) were $A=928.1$, $B=7.58$, $C=1.84$, $D=0.959$, $E=1.64$, $F=-1.21$, and $\alpha=-0.037$. The parameter α , the y-direction exponent, had the largest relative change (-107%) compared to the starting values. 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 Figure 6. Figure 20 and Figure 21 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 94%, with calibration providing no improvement within the 3 decimal places given here (there was a improvement in the 4th decimal place). The distribution in Figure 20 is roughly symmetric about zero, indicating there is not a strong bias.

Information Only

Table 6. 2003 Measured vs. Modeled correlation coefficients

	dataset	measured vs. modeled R ²
Uncalibrated	wells inside WIPP LWB	0.990
	wells <3km from WIPP LWB	0.991
	all wells	0.940
Calibrated	wells inside WIPP LWB	0.990
	wells <3km from WIPP LWB	0.991
	all wells	0.936

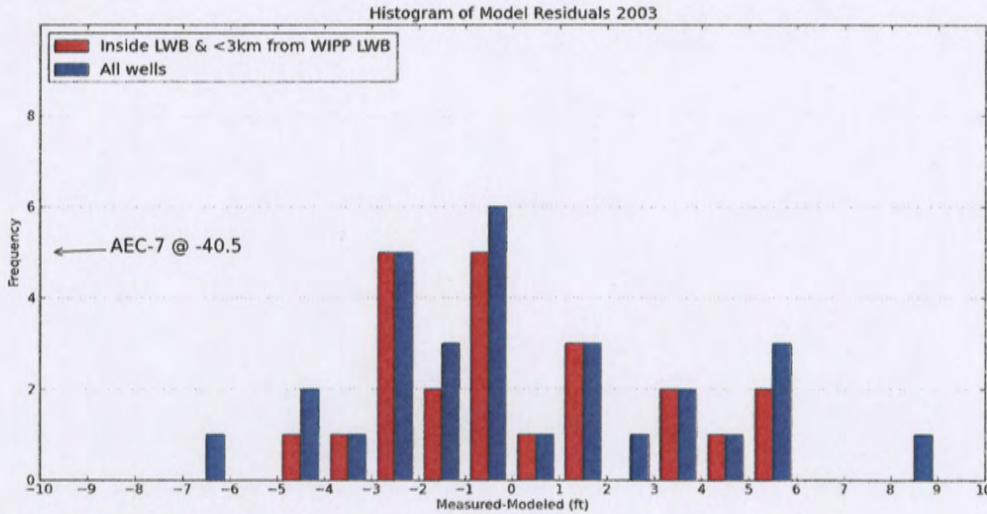


Figure 20. Histogram of Measured-Modeled errors for 2003

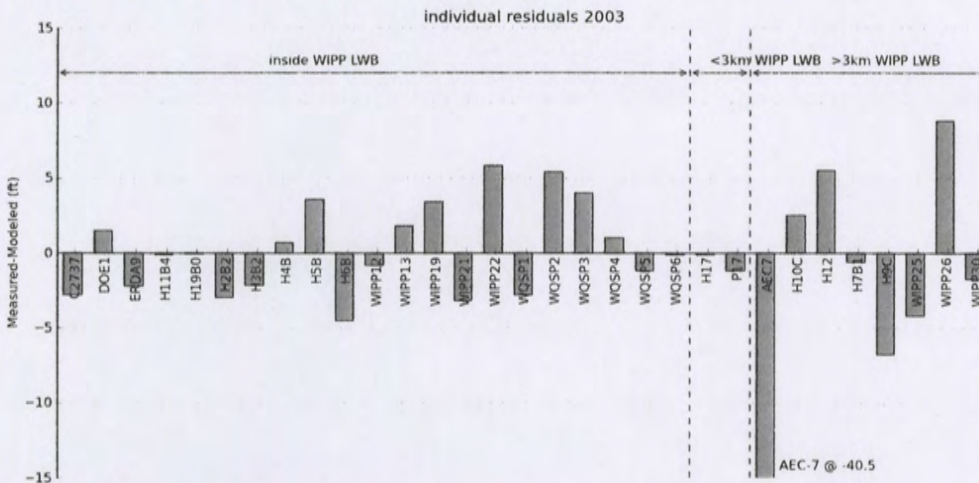


Figure 21. Measured-Modeled errors at each well for 2003

Aside from AEC-7, and to a lesser extent WIPP-26 (in Nash Draw), the model fit to the September 2003 observations is good. The averaged MODFLOW model captures the bulk Culebra flow behavior, while the PEST calibration improved model fit to the September 2003 observations.

7 2004 Results

7.1 2004 Freshwater Head Contours

The model-generated freshwater head contours are given in Figure 22 and Figure 23. 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.

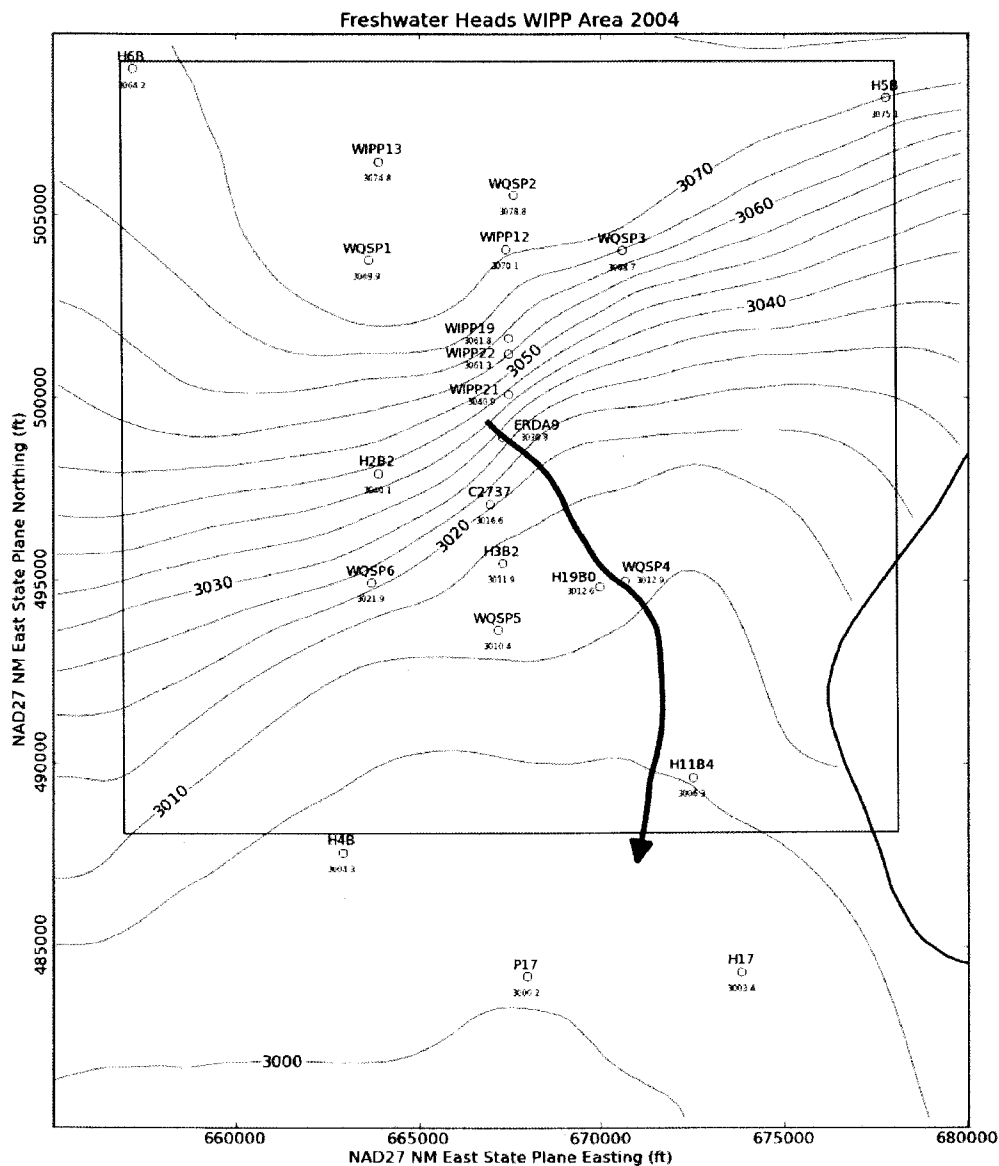


Figure 22. Model-generated August 2004 freshwater head contours with observed head listed at each well (5-foot contour interval) with blue water particle track from waste handling shaft to WIPP LWB

Information Only

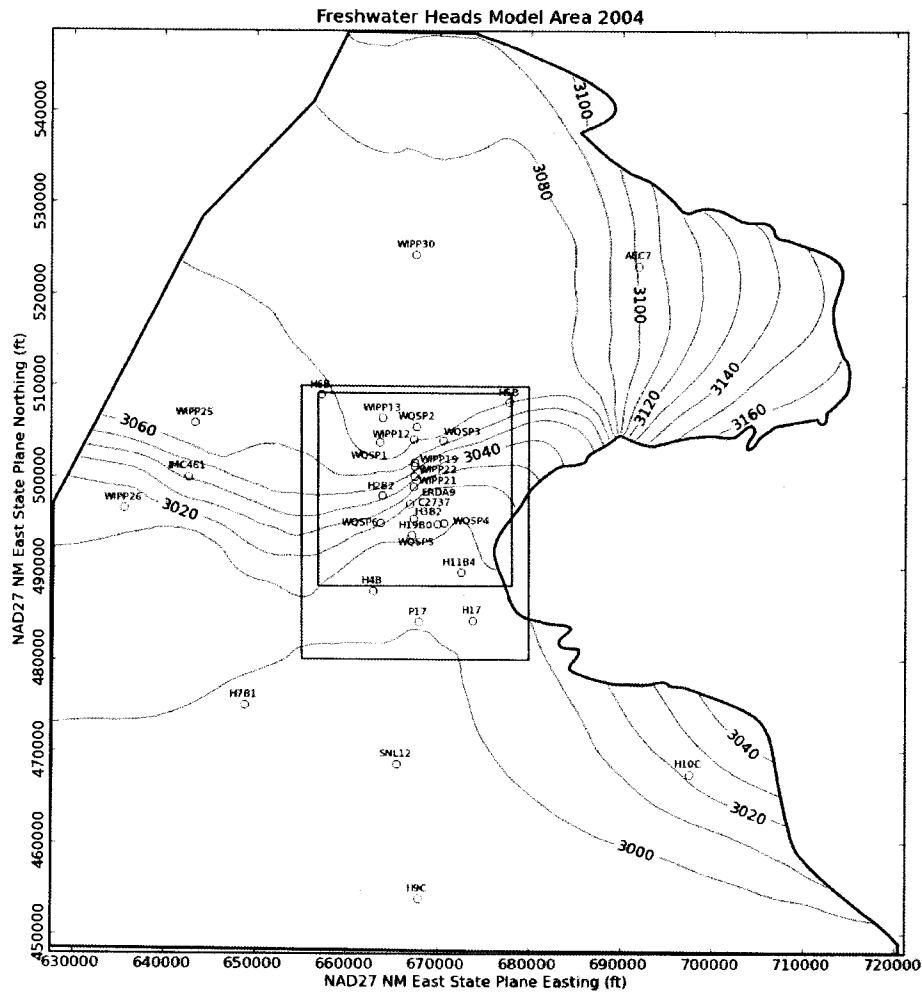


Figure 23. MODFLOW-modeled August 2004 heads for entire model domain (10-foot contour interval). Green rectangle indicates region contoured in Figure 22, black square is WIPP LWB.

7.2 2004 Particle Track

The blue arrow line in Figure 22 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 4085 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 6105 years (output from DTRKMF is adjusted from an original 7.75-m Culebra thickness). This is an average velocity of 0.67 m/yr.

7.3 2004 Measured vs. Modeled Fit

The scatter plot in Figure 24 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. AEC-7 was given a low weight (0.01), to prevent its large residual from dominating the optimization. Additional observations representing the average heads north of the LWB and south of the LWB were used to help prevent over-smoothing of the

estimated results across the LWB. This allowed PEST to improve the fit of the model to observed heads inside the area contoured in Figure 22, at the expense of fitting wells closer to the boundary conditions.

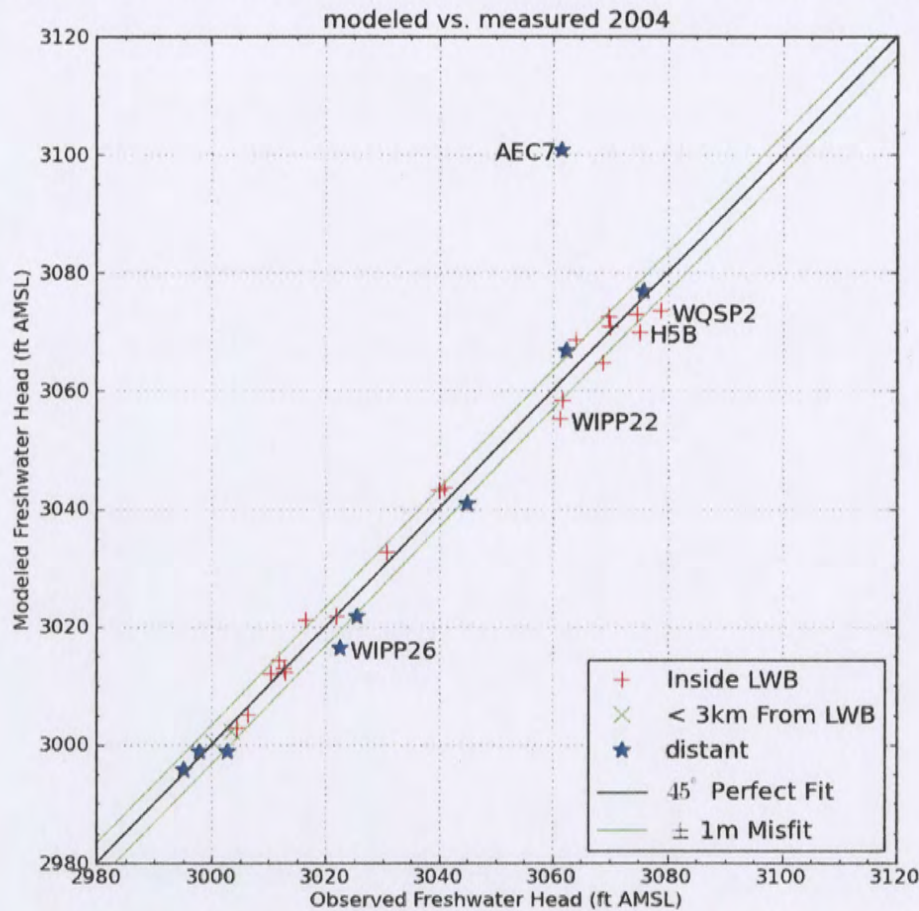


Figure 24. Measured vs. modeled scatter plot for averaged MODFLOW model generated heads and August 2004 observed freshwater heads

The black central diagonal line in Figure 24 represents a perfect model fit (1:1 or 45-degree slope); the two green lines on either side of this represent a 1-m misfit above or below the perfect fit. Wells more than 1.5 m from the 1:1 line are labeled. AEC-7 has a large misfit (39.4 ft), for the two reasons given in Section 2.4. The calibrated parameters (for equation 1) were A=927.5 B=7.90, C=1.51, D=0.858, E=1.12, F=-1.37, and $\alpha=-0.0029$. The parameter α , the y-direction exponent, had the largest relative change (-101%) compared to the starting values. 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 7. Figure 25 and Figure 26 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 93%, 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 25 is roughly symmetric about zero, indicating there is not a strong bias.

Table 7. 2004 Measured vs. Modeled correlation coefficients

	dataset	measured vs. modeled R ²
Uncalibrated	wells inside WIPP LWB	0.986
	wells <3km from WIPP LWB	0.988
	all wells	0.939
Calibrated	wells inside WIPP LWB	0.987
	wells <3km from WIPP LWB	0.989
	all wells	0.935

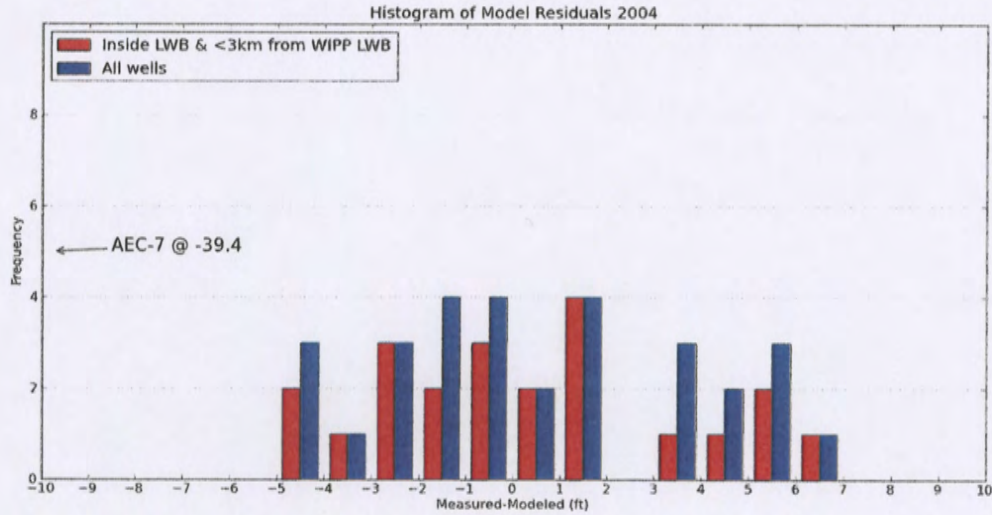


Figure 25. Histogram of Measured-Modeled errors for 2004

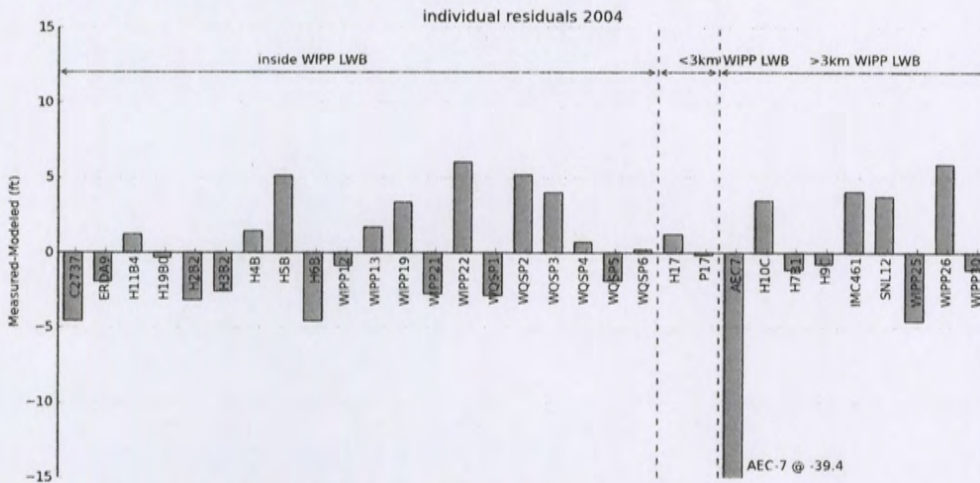


Figure 26. Measured-Modeled errors at each well for 2004

Aside from AEC-7, the model fit to the August 2004 observations is very good. The averaged MODFLOW model captures the bulk Culebra flow behavior, while the PEST calibration improved model fit to the August 2004 observations.

8 Summary

The development of the 2000-2004 historic Culebra contour maps in general followed quite closely to the procedure used in the first half of this report (Kuhlman, 2012) that dealt with the 2005-2007 historic Culebra contour maps.

The average MODFLOW model calibration process resulted in improved 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 combination of results of previous work (2005-2007) and this work (2000-2004) provides consistent maps through time 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.

Information Only

9 References

- Cauffman, T.L., A.M. LaVenue, and J.P. McCord. 1990. *Ground-Water Flow Modeling of the Culebra Dolomite, Volume II: Data Base*. Intera Inc., Austin TX. SAND89-7068/2.
- Department of Energy. 2001. *WIPP Annual Site Environmental Report for 2000*. DOE/WIPP-01-2225.
- Department of Energy. 2002. *WIPP Annual Site Environmental Report for 2001*. DOE/WIPP-02-2225.
- Department of Energy. 2003. *WIPP Annual Site Environmental Report for 2002*. DOE/WIPP-03-2225.
- Department of Energy. 2004. *WIPP Annual Site Environmental Report for 2003*. DOE/WIPP-04-2225.
- Department of Energy. 2005. *WIPP Annual Site Environmental Report for 2004*. DOE/WIPP-05-2225.
- 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)*. Carlsbad, 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)*. Carlsbad, NM, Sandia National Laboratories, ERMS 551116.
- Johnson, P.B. 2012a. *2003 Calculated Densities*, Sandia National Laboratories, Carlsbad, NM, ERMS 557402.
- Johnson, P.B. 2012b. *2004 Calculated Densities*, Sandia National Laboratories, Carlsbad, NM, ERMS 557405.
- Kuhlman, K.L. 2012. *Analysis Report for Preparation of 2005-2007 Culebra Potentiometric Surface Contour Maps, Revision 1*, Sandia National Laboratories, Carlsbad, NM, ERMS 556988.
- 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.
- Watterson, D. 2012. *2000 & 1999 ASER, [Transmittal of 2000 Culebra Water Level Data]*, Washington TRU Solutions, Carlsbad, NM. WIPP Records Center, ERMS 557523.

10 Run Control Narrative

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

Figure 27 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 8 can be found by taking the geometric average of the values in the text file, which are the values from the indicated line of each of the 100 realizations.

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

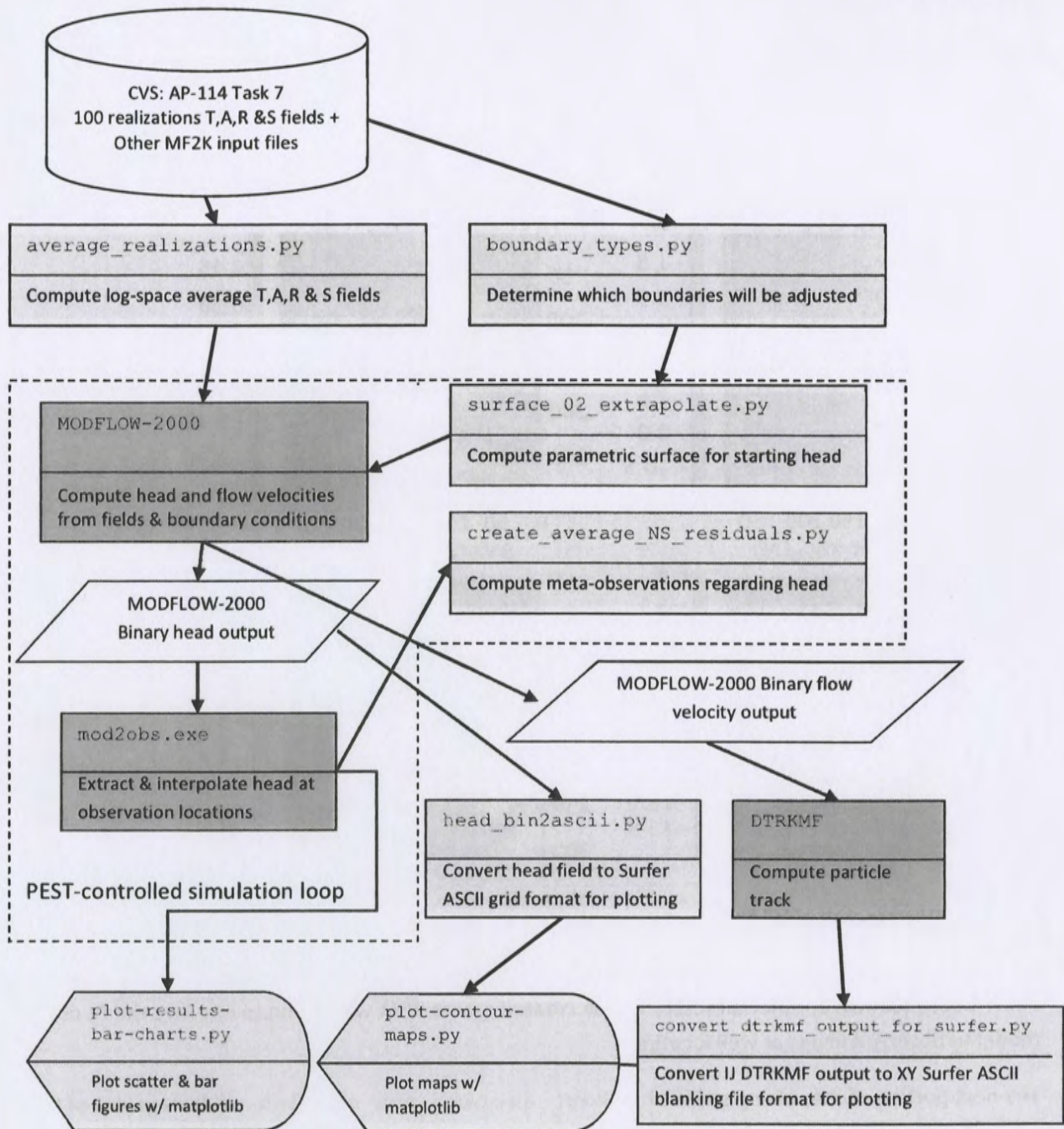


Figure 27. Process flowchart; dark gray indicates qualified programs, light gray are scripts written for this analysis

Table 8. 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

Information Only

Figure 28 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.

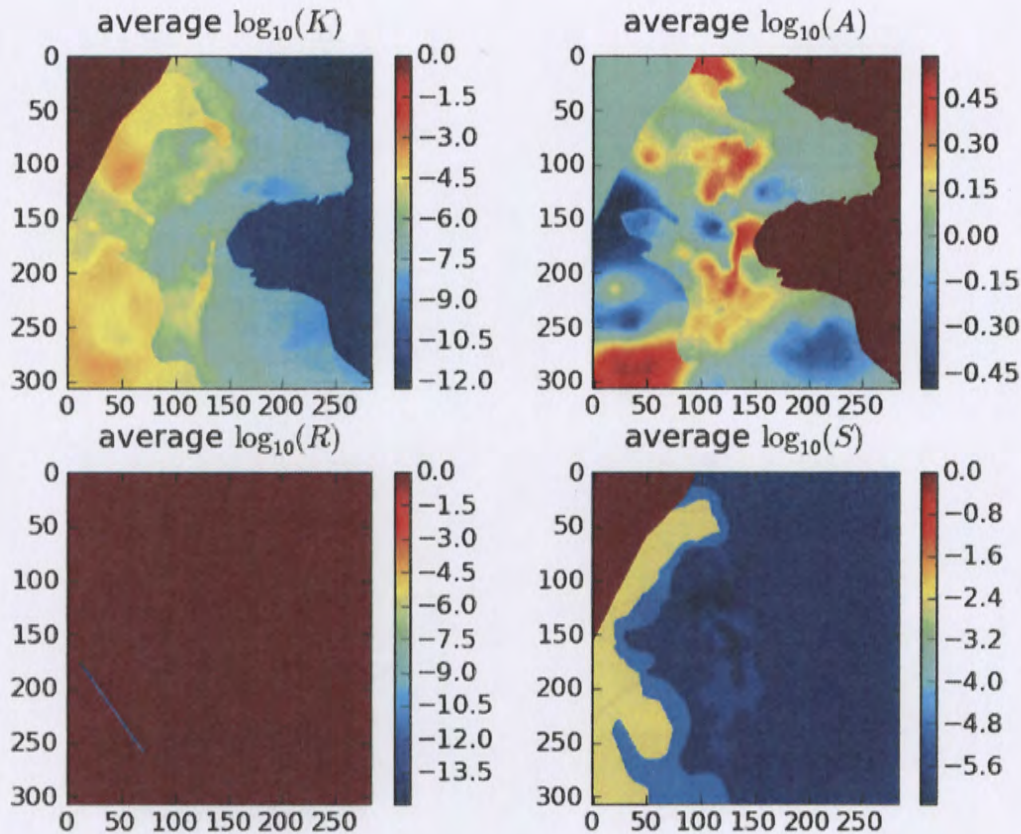


Figure 28. Plots of base-10 logarithms of average parameter fields; rows and columns are labeled on edges of figures.

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.blm`. The first three columns of `dtrk.out` (top half of Table 9) after the header are cumulative time (red), column (blue), and row (green). The three columns in the blanking file (second half of Table 9) after the header are UTM NAD27 X (blue), UTM NAD27 Y (green), and adjusted cumulative time (red, which is faster than the original cumulative travel time by the factor $7.75/4=1.9375$). The conversion from row, column to X, Y is

$$X = 601700.0 + 100.0 * column$$

$$Y = 3597100.0 - 100.0 * row$$

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

Table 9. 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`

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_XXXXASER.pst`, where `XXXX` is 2000, 2001, 2002, 2003 or 2004), 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_XXXXASER.smp`, where `XXXX` is the year), and the weights are read in from the input file (`obs_loc_XXXXASER.dat`, where `XXXX` is the year).

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.

Information Only

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.

Information Only

11 Appendix: Water Level and Density Data Listing

11.1 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
37K	data from Table-F.8 of 1999 ASER	ASER-1999-waterlevel-data.csv
31K	data from Watterson (2012)	ASER-2000-waterlevel-data.csv
40K	data from Table-F.8 of 2001 ASER	ASER-2001-waterlevel-data.csv
37K	data from Table-F.8 of 2002 ASER	ASER-2002-waterlevel-data.csv
44K	data from Table-F.8 of 2003 ASER	ASER-2003-waterlevel-data.csv
42K	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

11.1.1 densities-to-use.csv input file

Missing begin or end dates indicate the date did not fall in the interval 2000-2010.

well	begin date	end date	density	source
AEC-7		04/06/2004	1.0890	PDS-2006ASER
AEC-7	06/18/2008		1.0780	TROLL2008
C-2737	02/14/2001	05/12/2003	1.0000	ASER
C-2737	11/19/2003	02/01/2007	1.0190	TROLL2005
C-2737	02/01/2007	06/28/2008	1.0100	JOHNSON2009
C-2737	06/28/2008		1.0293	TROLL2008
DOE-1		11/01/2003	1.0880	SAND89-7068
ERDA-9			1.0670	JOHNSON2009
H-02B2		04/12/2005	1.0060	SAND89-7068
H-02B2	04/12/2005	02/24/2009	1.0000	SNL notebooks
H-02B2	02/24/2009		1.0110	TROLL2010
H-03B2			1.0420	JOHNSON2009
H-04B		06/14/2009	1.0150	JOHNSON2009
H-05B		06/11/2005	1.1040	ASER
H-05B	06/11/2005		1.0950	JOHNSON2009
H-06B		02/19/2008	1.0400	JOHNSON2009
H-07B1			1.0020	JOHNSON2009
H-07B2		05/10/2005	0.9990	SAND89-7068
H-09B		02/11/2002	1.0010	ASER
H-09C			1.0010	JOHNSON2009
H-10B		02/01/2002	1.0470	SAND89-7068
H-10C	02/19/2002	07/12/2009	1.0010	JOHNSON2009
H-10C	07/12/2009		1.0890	TROLL2009
H-11B4			1.0700	JOHNSON2009
H-12		12/01/2003	1.0950	SAND89-7068
H-12	04/12/2005	11/24/2008	1.0970	JOHNSON2009
H-12	11/24/2008		1.0957	TROLL2008
H-14		04/21/2005	1.0100	SAND89-7068
H-15		02/01/2001	1.1540	SAND89-7068
H-15	11/18/2003	04/10/2006	1.0820	TROLL2005
H-15	04/10/2006	03/05/2008	1.0530	JOHNSON2009
H-17			1.1330	JOHNSON2009
H-18		03/01/2001	1.0450	ASER
H-19B7			1.0813	TROLL2007
I-461	10/15/2003		1.0050	JOHNSON2009
P-14		08/27/1999	1.0180	SAND89-7068
P-15		02/11/2002	1.0150	SAND89-7068
P-17		06/17/2005	1.0695	PDS-2004-2005-avg
P-17	06/17/2005	08/18/2006	1.0529	SGR-2006-SNL-SURVEY
P-18		02/25/2002	1.1190	SAND89-7068
SNL-12	06/25/2003		1.0050	JOHNSON2009
WIPP-11	09/07/2004		1.0380	JOHNSON2009
WIPP-12		07/12/2005	1.1000	SAND89-7068
WIPP-13			1.0530	JOHNSON2009
WIPP-18		03/01/2001	1.1000	ASER
WIPP-19		05/29/2005	1.0590	SAND89-7068
WIPP-19	05/29/2005		1.0440	JOHNSON2009
WIPP-21		05/28/2005	1.0710	ASER
WIPP-22		05/29/2005	1.0865	ASER
WIPP-25		06/12/2009	1.0110	JOHNSON2009
WIPP-26		10/12/2006	1.0090	SAND89-7068
WIPP-30		06/22/2005	1.0180	SAND89-7068
WIPP-30	06/22/2005	03/01/2008	1.0000	JOHNSON2009
WQSP-1			1.0480	SGR-ROUNDS-23-to-25
WQSP-2			1.0477	SGR-ROUNDS-23-to-25
WQSP-3			1.1457	SGR-ROUNDS-23-to-25
WQSP-4			1.0747	SGR-ROUNDS-23-to-25
WQSP-5			1.0250	SGR-ROUNDS-23-to-25
WQSP-6			1.0140	SGR-ROUNDS-23-to-25

11.2 Listing of Water Level Plotting Script

11.2.1 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.

```
1 # this python script plots water level and density data for WIPP wells
2 # for the purpose of choosing freshwater heads for historic contouring
3 # using data taken from the Annual Site Environmental Reports (ASER).
4 #
5 # by Kris Kuhlman (6212)
6 # September 2011 through 2012
7 #
8
9 import matplotlib # set plotting backend
10 matplotlib.use('Agg')
11
12 import numpy as np # array library
13 from mpl_toolkits.mplot3d import axes3d
14 import numpy.core.defchararray as npchar # functions for character arrays
15
16 import matplotlib.pyplot as plt # plotting library
17 import matplotlib.mlab as mlab
18 from matplotlib.dates import MonthLocator, YearLocator, DateFormatter
19 from matplotlib.ticker import NullFormatter
20 from matplotlib.font_manager import FontProperties
21
22 import datetime
23 import re # regular expressions
24
25 plot_density_contours = True # also boxplots of densities
26 plot_timeseries = False
27
28 # for making small text in figures
29 AnnFontP = FontProperties()
30 AnnFontP.set_size('xx-small') # for annotations
31 LegFontP = FontProperties()
32 LegFontP.set_size('small') # for legend
33
34 # read in corrections to pre-2007 data
35 with open('reference-point-change-2007.dat','r') as fh:
36     lines = fh.readlines()
37
38 rpcorr = {}
39 for line in lines:
40     well,delta = line.split()
41     rpcorr[well] = float(delta)
42
43
44 # read in freshwater heads used to create past contour maps
45 # for years 2008, 2009 and 2010.
46 # $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
47 aserfwh = {}
48 for year in [2008,2009,2010]:
49     with open('meas_head_%iASER.smp' % (year,),'r') as fh:
50         lines = fh.readlines()
51
52     aserfwh[year] = {}
53     for line in lines:
54         fields = line.rstrip().split()
55
56         # regexp for splitting wellnames up into parts
57         # non-numbers, numbers, and non-numbers (last group is sometimes empty)
58         m = re.search(r"([\^0-9]+)([0-9]+)([\^0-9]*)",fields[0])
59         w = [m.group(1),m.group(2),m.group(3)]
```

```

60
61 # handle mapping between well names used in pest input files
62 # and names used in more complete database
63 if w[0] == 'SNL' or w[0] == 'H':
64     wstr = '%s-%2.2i%s' % (w[0], int(w[1]), w[2].upper())
65 elif w[0] == 'IMC':
66     wstr = 'I-461'
67 else:
68     wstr = '%s-%s%s' % (w[0], w[1], w[2].upper())
69     aserfwh[year][wstr.strip()] = {'fwh': float(fields[3]), 'name': fields[0]}
70
71
72 # read in "best" densities proposed to use for contour maps
73 # $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
74 with open('densities-to-use.csv', 'r') as fh:
75     f = fh.read()
76     lines = f.split('\r')[1:]
77
78 findendtype = np.dtype([( 'well', 'S56'), ('dt0', '08'), ('dt1', '08'),
79                          ('den', 'f8'), ('src', 'S13')])
80 finden = []
81 for line in lines:
82     r = [x.strip() for x in line.split(',')]
83
84 # handle empty or blank records gracefully
85 if len(r) == 1:
86     continue
87 if r[0] == '':
88     continue
89
90 # when no start time, it is now blank
91 # (was 1/1/1998 previously, which is fragile)
92 if r[1] == '':
93     # bogus early datetime when none provided
94     dt0 = datetime.datetime(1941,12,7)
95 else:
96     dt0 = datetime.datetime.strptime(r[1], '%m/%d/%Y')
97
98 # when no end time, it is now blank
99 # (was 1/1/2012 previously, which is fragile)
100 if r[2] == '':
101     # bogus future datetime when none provided
102     dt1 = datetime.datetime(2015,9,24)
103 else:
104     dt1 = datetime.datetime.strptime(r[2], '%m/%d/%Y')
105
106 # assemble record datatype
107 finden.append((r[0], dt0, dt1, float(r[3]), r[4].upper()))
108
109 finden = np.array(finden, dtype=findendtype)
110 finden = np.sort(finden, order=('well', 'dt0', 'src'))
111
112
113 # read in Culebra midpoint elevations (mostly from Johnson, 2009
114 # developed and checked for model calibration)
115 # $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
116 with open('culebra-midpoint-elevations.csv', 'r') as fh:
117     f = fh.read()
118     lines = f.split('\r')[1:]
119
120 midpt = {}
121 for line in lines:
122     r = [x.strip() for x in line.split(',')]
123     # elevation is in feet AMSL, convert to meters

```

```

124     midpt[r[0].upper()] = float(r[1])*0.3048
125
126
127 # read in well XY coordinates
128 # $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
129 with open('well-coordinates.csv','r') as fh:
130     f = fh.read()
131     lines = f.split('\r')[1:]
132
133 xyz = {}
134 for line in lines:
135     r = [x.strip() for x in line.split(',') ]
136     # XY is UTM NAD27 ZONE 13 (m)
137     xyz[r[0].upper()] = {'x':float(r[2]), 'y':float(r[1])}
138
139
140 # read in table of published density values from ASER/SAND reports and
141 # from 2007-2010 the tables of TROLL-derived densities from SNL.
142 # $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
143 with open('reported-density-values.csv','r') as fh:
144     f = fh.read()
145     lines = f.split('\r')[1:]
146
147 dendtype = np.dtype([( 'well', 'S56'), ('dt', '08'), ('dt2', '08'),
148                      ('den', 'f8'), ('src', 'S13'), ('type', 'S13')])
149 den = []
150 for line in lines:
151     r = [x.strip() for x in line.split(',') ]
152     dt = datetime.datetime.strptime(r[1], '%m/%d/%Y')
153
154     # only troll densities have a date range
155     if r[2] == "":
156         dt2 = datetime.datetime(1941,12,7) # bogus datetime
157     else:
158         dt2 = datetime.datetime.strptime(r[2], '%m/%d/%Y')
159     den.append((r[0].upper(), dt, dt2, float(r[3]), r[4].upper(), r[5].upper()))
160
161 den = np.array(den, dtype=dendtype)
162 den = np.sort(den, order=( 'well', 'dt', 'type' ))
163 typesymb = { 'PDS': '.', 'SGR': 'o', 'TROLL': 'x', 'PITZER': 's', 'AVG': '*' }
164 typecolor = { 'PDS': 'gray', 'SGR': 'black', 'TROLL': 'red', 'PITZER': 'orange', 'AVG': 'orange' }
165
166 # earliest date that should be plotted
167 # densities before this date are plotted here as an arrow pointing left
168 firstplot = datetime.datetime(1998,3,1)
169
170
171 # read in well event log: includes drilling, P&A, tests and
172 # well maintenance activities that might effect WL or densities
173 # $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
174 with open('well-events.csv','r') as fh:
175     f = fh.read()
176     lines = f.split('\r')[1:]
177
178 eventstype = np.dtype([( 'well', 'S56'), ('dt0', '08'), ('dt1', '08'), ('s', 'S50'), ('c', 'S10')])
179 events = []
180 for line in lines:
181     r = [x.strip().strip(' ') for x in line.split(',') ]
182     if r[0] == '':
183         # empty line at end?
184         continue
185     dt0 = datetime.datetime.strptime(r[1], '%m/%d/%y')
186     if len(r[2]) > 0:
187         dt1 = datetime.datetime.strptime(r[2], '%m/%d/%y')

```

```

188 else:
189     dt1 = None
190
191 # try to categorize events based on color
192 if 'samp' in r[3]:
193     # water quality sampling
194     c = 'gray'
195 elif ('drill' in r[3] or 'perf' in r[3] or
196       ('recomp' in r[3] and 'ulebra' in r[3])):
197     # new or replacement well drilling
198     c = 'green'
199 elif 'PIP' in r[3] or 'packer' in r[3]:
200     # added, removed, or reset packers
201     c = 'blue'
202 elif 'plug' in r[3] or 'recomp' in r[3]:
203     # plugged back, plug & abandoned, or recompleted
204     c = 'red'
205 elif ('test' in r[3] and not 'well integrity' in r[3] and
206       not 'configured' in r[3]):
207     # slug or pumping test
208     c = 'cyan'
209 elif 'bail' in r[3] or 'swab' in r[3]:
210     # bailed or swabbed well/tubing
211     c = 'magenta'
212 else:
213     # something else
214     c = 'black'
215
216 events.append((r[0].upper(), dt0, dt1, r[3], c))
217
218 events = np.array(events, dtype=eventstype)
219 evwells = list(set(events[:, 'well']))
220
221 if plot_density_contours:
222     # %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
223     # $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
224     # using XY coordinates and reported densities,
225     # plot contour map of densities for checking / visualization
226     scale = 1000.0
227     datadir = 'wipp-polyline-data/'
228
229     # subset of wells with a "chosen" density value
230     denwells = list(set(finden[:, 'well']))
231     bestden = []
232     coordden = []
233     dennames = []
234
235     boxlist = []
236     boxcoord = []
237
238     for well in denwells:
239         finm = finden[:, 'well'] == well
240         allm = den[:, 'well'] == well
241
242         # densest "chosen" value is representative of formation
243         if not (well == 'WIPP-29' or well == 'P-18'):
244             bestden.append(max(((finden[finm])['den']).max(), 1.0))
245             coordden.append((xyz[well]['x'], xyz[well]['y']))
246             boxcoord.append((xyz[well]['x']/scale, xyz[well]['y']/scale))
247             boxlist.append(den[allm]['den'])
248             dennames.append(well)
249
250     # $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
251     # create some box plots showing the range of densities seen at individual wells

```



```

252 boxdata = zip(boxlist , dennames , zip(*boxcoord)[0] , zip(*boxcoord)[1])
253 sortedboxdata = sorted(boxdata , key=lambda x: x[1]) # sort densities by well name
254
255 # make a series of x- or y-slice box plots with wells
256
257
258 # boxplot figure handle
259 figbox = plt.figure(2 , figsize=(24,18))
260 ax1 = figbox.add_subplot(311)
261
262
263 ax1.boxplot(zip(*sortedboxdata)[0])
264 ax1.set_xticks(np.arange(len(dennames))+1)
265 ax1.set_xticklabels(zip(*sortedboxdata)[1] , rotation=90)
266 ax1.set_ylim([0.95 , 1.25])
267 ax1.grid()
268 ax1.set_ylabel('Culebra specific gravity')
269
270 wx , wy = np.loadtxt(datadir+'wipp_boundary.dat' , unpack=True)/ scale
271
272 # projection of wells onto E-W profile
273 ax2 = figbox.add_subplot(312)
274 ax2.boxplot(zip(*sortedboxdata)[0] , positions=zip(*sortedboxdata)[2] , widths=0.2)
275 ax2.set_xticks(np.linspace(604.0 , 623.0 , 20))
276 ax2.grid()
277 ax2.axvspan(wx.min() , wx.max() , alpha=0.15 , color='gray')
278 ax2.set_ylim([0.95 , 1.25])
279 ax2.set_xlabel('UTM X NAD27 (km)')
280 ax2.set_ylabel('Culebra specific gravity')
281
282 # projection of wells on N-S profile
283 ax3 = figbox.add_subplot(313)
284 ax3.boxplot(zip(*sortedboxdata)[0] , positions=zip(*sortedboxdata)[3] , widths=0.25)
285 ax3.set_xticks(np.linspace(3565.0 , 3595.0 , 6))
286 ax3.set_xlim([3567 , 3595])
287 ax3.grid()
288 ax3.axvspan(wy.min() , wy.max() , alpha=0.15 , color='gray')
289 ax3.set_ylim([0.95 , 1.25])
290 ax3.set_xlabel('UTM Y NAD27 (km)')
291 ax3.set_ylabel('Culebra specific gravity')
292
293 del boxdata , sortedboxdata , boxcoord , boxlist
294
295 #####
296 # back to plotting contours of density in map view
297
298 bestden = np.array(bestden)
299 bestden.shape = (-1,) # remove trailing singleton dimension
300
301 coordden = np.array(coordden)/ scale
302
303 minx = coordden[:,0].min() - 0.5
304 miny = coordden[:,1].min() - 0.5
305 maxx = coordden[:,0].max() + 0.5
306 maxy = coordden[:,1].max() + 0.5
307
308 X , Y = np.mgrid[minx: maxx:150j , miny: maxy:100j]
309
310 # composite H2/H3 halite margin used as eastern boundary in MODFLOW model
311 h23x , h23y = np.loadtxt(datadir+'composite_23_margin.dat' , unpack=True)/ scale
312
313 # contour both data points and H2/H3 boundary (assign density=1.15 to boundary)
314 Z = mlab.griddata(np.concatenate((coordden[:,0] , h23x) , axis=0) ,
315                   np.concatenate((coordden[:,1] , h23y) , axis=0) ,

```

```

316         np.concatenate((bestden[:,], np.ones(h23x.shape)*1.15), axis=0),
317         X,Y, interp='nn')
318
319 # WIPP LWB
320 WIPPx,WIPPy = np.loadtxt(datadir+'wipp_boundary.dat', unpack=True)/ scale
321
322 # coordinates of middle of WIPP LWB
323 xmid = WIPPx.mean()
324 ymid = WIPPy.mean()
325
326 # find index that is closest to middle of WIPP LWB for mgrid output
327 xidx = np.argmin(np.abs(X[:,0] - xmid))
328 yidx = np.argmin(np.abs(Y[0,:] - ymid))
329
330 # profile from contours going through middle of WIPP site
331 ax3.plot(Y[0,:], Z[xidx,:], 'g-') # N-S profile
332 ax2.plot(X[:,0], Z[:,yidx], 'g-') # E-W profile
333 ax3.set_xlim([3567,3595])
334 plt.savefig('density_boxplot.eps')
335 plt.close(2)
336
337 # contour plot figure handle
338 figcon = plt.figure(1, figsize=(22,17))
339 axleft = figcon.add_subplot(111)
340 #axright = figcon.add_subplot(122, projection='3d')
341
342 # contours to plot
343 levels = [1.00,1.01,1.02,1.04,1.06,1.08,1.1,1.12,1.14,1.16,1.2]
344
345 CS = axleft.contour(X,Y,Z, levels)
346 axleft.clabel(CS, inline=1, fontsize=8, fmt='%.3g')
347 axleft.plot(coordden[:,0], coordden[:,1], 'k.') # well locations
348 axleft.axis('image')
349
350 axleft.plot(WIPPx,WIPPy, '--', color='black', linewidth=1.0)
351
352 # Nash Draw
353 x,y = np.loadtxt(datadir+'nash-draw.csv', delimiter=',', unpack=True)/ scale
354 axleft.plot(x,y, '--', color='black', linewidth=0.5)
355
356 # H2 halite margin
357 x,y = np.loadtxt(datadir+'h2_200711.bln', unpack=True, skiprows=1, delimiter=',')/ scale
358 axleft.plot(x,y, '--', color='purple', linewidth=0.5)
359
360 # H3 halite margin
361 x,y = np.loadtxt(datadir+'h3_200711.bln', unpack=True, skiprows=1, delimiter=',')/ scale
362 axleft.plot(x,y, '--', color='green', linewidth=0.5)
363
364 # H4 halite margin
365 x,y = np.loadtxt(datadir+'h4_200711.bln', unpack=True, skiprows=1, delimiter=',')/ scale
366 axleft.plot(x,y, '--', color='blue', linewidth=0.5)
367 del x,y
368
369 axleft.set_xlabel('UTM X NAD27 (km)')
370 axleft.set_ylabel('UTM Y NAD27 (km)')
371
372 for x,y,well in zip(coordden[:,0], coordden[:,1], dennames):
373     # add well labels (cluttered and hard to read in places)
374     plt.annotate(well, (x+0.1,y), fontproperties=AnnFontP)
375
376 axleft.set_title('"best" Culebra specific gravity used in ASER contour maps')
377 plt.savefig('density-contours.eps')
378 plt.close(1)
379 del bestden, coordden, dennames

```

```

380
381
382 # read in water level data from 2001–2010 ASER tables
383 # $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
384
385 # one header row
386 # mostly common column format
387 # A0 : well
388 # B1 : zone (CUL,MAG, etc.)
389 # C2 : date (no time)
390 # D3 : adjusted depth below top of casing (ft)
391 # E4 : adjusted depth below top of casing (m)
392 # F5 : water level elevation (ft amsl)
393 # G6 : water level elevation (m amsl)
394 # H7 : adjusted freshwater head (ft amsl)
395
396 # read in yearly files with csv reader
397
398 # make all well names uppercase, strip off anything in parenthesis
399 # strip "/" out of zone names
400
401 # convert dates to python date objects
402
403 # save (E) depth to water (meters)
404 # save (G) water level elevation (meters)
405 # save (H) freshwater head (feet) -> convert to meters
406
407 wldtype = np.dtype([( 'well' , 'S56' ), ( 'zone' , 'S4' ), ( 'dt' , 'O8' ),
408                    ( 'dtwm' , 'f8' ), ( 'wlem' , 'f8' ), ( 'cwlem' , 'f8' ), ( 'fwhm' , 'f8' )])
409 data = []
410
411 # NB: in 2000 there were no waterlevel data reported in ASER! obtained data from Dan Watterson.
412 # NB: in 1998 there was no freshwater head reported in ASER!
413
414 for year in [1998,1999,2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,2010]:
415
416     earliest = datetime.datetime(2100,12,31)
417     latest = datetime.datetime(1900,1,1)
418
419     fn = 'ASER-%i-waterlevel-data.csv' % year
420     print fn,
421     with open(fn, 'r') as fh:
422         f = fh.read()
423         lines = f.split('\r') # Mac line endings
424
425     print '# values', len(lines)
426     for line in lines[1:]:
427
428         r = line.split(',')
429         dt = datetime.datetime.strptime(r[2], '%m/%d/%y')
430
431         if dt < earliest:
432             earliest = dt
433         elif dt > latest:
434             latest = dt
435
436     # clean up and simplify well names to be consistent with HANALYST
437     well = r[0].upper().partition('(')[0].partition('/')[0].strip()
438     zone = r[1].upper().replace('/', ',').strip()
439
440     if 'SNL' in well:
441         # some SNL wells are not zero padded some years
442         # therefore they appear as different wells
443         num = well.split('-')[1]

```

```

444         if len(num) == 1:
445             well = 'SNL-0' + num
446
447         if zone == 'SRD':
448             zone = 'SRDL'
449         elif 'RUSS' in zone:
450             zone = 'RS'
451
452         if len(r) < 8 or r[7].strip() == "":
453             fwh = -999.0
454         else:
455             fwh = float(r[7])*0.3048
456
457         if year == 1998:
458             fwh = np.NaN
459
460         wlem = float(r[6])
461         if year < 2007 and well in rpcorr:
462             cwlem = wlem + rpcorr[well]
463         else:
464             cwlem = wlem
465
466         data.append((well, zone, dt, float(r[4]), wlem, cwlem, fwh))
467
468 data = np.array(data, dtype=wldtype)
469 data = np.sort(data, order=('dt', 'zone', 'well'))
470
471
472 # wells reported by WRES (doesn't include Gnome wells)
473 wells = list(set(data[:, 'well']))
474 wells.sort()
475
476 # can plot all wells reported by WRES
477 zones = list(set(data[:, 'zone']))
478 for zone in zones:
479     zmask = data['zone'] == zone
480     zwells = list(set(data[zmask]['well']))
481     zwells.sort()
482
483 # months used for creating ASER contour maps (no apparent ranges for 2003 & 2004)
484 # NB: 2003, 2004, 2005 & 2006 are my choice, not necessarily what used in ASER
485 # 2000 is just a guess so far
486 cmonths = {2000:(datetime.datetime(2000,12,1), datetime.datetime(2000,12,31)),
487            2001:(datetime.datetime(2001,12,1), datetime.datetime(2001,12,31)),
488            2002:(datetime.datetime(2002,12,1), datetime.datetime(2002,12,31)),
489            2003:(datetime.datetime(2003,9,1), datetime.datetime(2003,9,30)),
490            2004:(datetime.datetime(2004,8,1), datetime.datetime(2004,8,31)),
491            2005:(datetime.datetime(2005,6,1), datetime.datetime(2005,6,30)),
492            2005:(datetime.datetime(2005,6,1), datetime.datetime(2005,6,30)),
493            2006:(datetime.datetime(2006,11,1), datetime.datetime(2006,11,30)),
494            2007:(datetime.datetime(2007,5,1), datetime.datetime(2007,5,31)),
495            2008:(datetime.datetime(2008,9,1), datetime.datetime(2008,9,30)),
496            2009:(datetime.datetime(2009,6,1), datetime.datetime(2009,6,30)),
497            2010:(datetime.datetime(2010,2,1), datetime.datetime(2010,2,28))}
498
499 # exceptions to the above rules, based on looking closer at data
500 cexceptions = {'AEC-7':{2004:(datetime.datetime(2004,3,1),
501                             datetime.datetime(2004,3,31)),
502                  2005:(None, None), 2006:(None, None),
503                  2007:(None, None)},
504               'C-2737':{2003:(datetime.datetime(2003,3,1),
505                             datetime.datetime(2003,3,31))},
506               'CB-1':{2000:(None, None), 2001:(None, None),
507                      2002:(None, None), 2003:(None, None)},

```

508 'DOE-1':{2004:(None, None), 2005:(None, None)},
509 'DOE-2':{2000:(None, None)},
510 'ERDA-9':{2000:(datetime.datetime(2000,9,1),
511 datetime.datetime(2000,9,30))},
512 'H-01':{2000:(None, None)},
513 'H-02A':{2000:(None, None), 2001:(None, None),
514 2002:(None, None), 2003:(None, None),
515 2004:(None, None)},
516 'H-02C':{2000:(None, None), 2001:(None, None),
517 2002:(None, None), 2003:(None, None),
518 2004:(None, None)},
519 'H-03B3':{2000:(None, None), 2001:(None, None),
520 2002:(None, None), 2003:(None, None),
521 2004:(None, None)},
522 'H-05A':{2000:(None, None), 2001:(None, None),
523 2002:(None, None), 2003:(None, None),
524 2004:(None, None)},
525 'H-06A':{2000:(None, None), 2001:(None, None),
526 2002:(None, None), 2003:(None, None),
527 2004:(None, None)},
528 'H-07B1':{2004:(datetime.datetime(2004,9,1),
529 datetime.datetime(2004,9,30))},
530 'H-07B2':{2000:(None, None), 2001:(None, None),
531 2002:(None, None), 2003:(None, None),
532 2004:(None, None)},
533 'H-09A':{2000:(None, None), 2001:(None, None)},
534 'H-09B':{2000:(None, None), 2001:(None, None)},
535 'H-09C':{2000:(datetime.datetime(2000,6,1),
536 datetime.datetime(2000,6,30))},
537 'H-10C':{2006:(datetime.datetime(2006,8,1),
538 datetime.datetime(2006,8,31))},
539 'H-11B1':{2000:(None, None), 2001:(None, None),
540 2002:(None, None), 2003:(None, None),
541 2004:(None, None)},
542 'H-11B2':{2000:(None, None)},
543 'H-11B3':{2000:(None, None), 2001:(None, None)},
544 'H-19B2':{2000:(None, None), 2001:(None, None),
545 2002:(None, None), 2003:(None, None),
546 2004:(None, None), 2005:(None, None),
547 2006:(None, None), 2007:(None, None),
548 2008:(None, None), 2009:(None, None),
549 2010:(None, None), 2011:(None, None)},
550 'H-19B3':{2000:(None, None), 2001:(None, None),
551 2002:(None, None), 2003:(None, None),
552 2004:(None, None), 2005:(None, None),
553 2006:(None, None), 2007:(None, None),
554 2008:(None, None), 2009:(None, None),
555 2010:(None, None), 2011:(None, None)},
556 'H-19B4':{2000:(None, None), 2001:(None, None),
557 2002:(None, None), 2003:(None, None),
558 2004:(None, None), 2005:(None, None),
559 2006:(None, None), 2007:(None, None),
560 2008:(None, None), 2009:(None, None),
561 2010:(None, None), 2011:(None, None)},
562 'H-19B5':{2000:(None, None), 2001:(None, None),
563 2002:(None, None), 2003:(None, None),
564 2004:(None, None), 2005:(None, None),
565 2006:(None, None), 2007:(None, None),
566 2008:(None, None), 2009:(None, None),
567 2010:(None, None), 2011:(None, None)},
568 'H-19B6':{2000:(None, None), 2001:(None, None),
569 2002:(None, None), 2003:(None, None),
570 2004:(None, None), 2005:(None, None),
571 2006:(None, None), 2007:(None, None),

```

572         2008:(None, None) ,2009:(None, None) ,
573         2010:(None, None) ,2011:(None, None) } ,
574 'H-19B7':{2000:(None, None) ,2001:(None, None) ,
575         2002:(None, None) ,2003:(None, None) ,
576         2004:(None, None) ,2005:(None, None) ,
577         2006:(None, None) ,2007:(None, None) ,
578         2008:(None, None) ,2009:(None, None) ,
579         2010:(None, None) ,2011:(None, None) } ,
580 'H-14':{2000:(datetime.datetime(2000,9,1) ,
581         datetime.datetime(2000,9,30))} ,
582 'I-461':{2004:(datetime.datetime(2004,12,1) ,
583         datetime.datetime(2004,12,31))} ,
584 'P-15':{2000:(datetime.datetime(2000,1,1) ,
585         datetime.datetime(2000,1,31)) ,
586         2001:(None, None) } ,
587 'P-18':{2000:(None, None) ,2001:(None, None) } ,
588 'SNL-01':{2004:(None, None) } ,
589 'SNL-02':{2003:(None, None) ,2004:(None, None) } ,
590 'SNL-03':{2003:(None, None) ,2004:(None, None) } ,
591 'SNL-05':{2004:(None, None) } ,
592 'SNL-09':{2003:(None, None) ,2004:(None, None) } ,
593 'SNL-12':{2003:(None, None) ,
594         2004:(datetime.datetime(2004,12,1) ,
595         datetime.datetime(2004,12,31))} ,
596 'SNL-14':{2005:(None, None) ,
597         2007:(datetime.datetime(2007,11,1) ,
598         datetime.datetime(2007,11,30))} ,
599 'SNL-16':{2007:(datetime.datetime(2007,9,1) ,
600         datetime.datetime(2007,9,30))} ,
601 'WIPP-11':{2006:(datetime.datetime(2006,8,1) ,
602         datetime.datetime(2006,8,31))} ,
603 'WIPP-26':{2005:(None, None) } ,
604 'WIPP-27':{2000:(None, None) ,2001:(None, None) ,2002:(None, None) ,
605         2003:(None, None) ,2004:(None, None) ,
606         2005:(None, None) } ,
607 'WIPP-29':{2000:(None, None) ,2001:(None, None) ,2002:(None, None) ,
608         2003:(None, None) ,2004:(None, None) ,
609         2005:(None, None) } ,
610 'WIPP-30':{2000:(datetime.datetime(2000,6,1) ,
611         datetime.datetime(2000,6,30)) ,
612         2005:(datetime.datetime(2005,8,1) ,
613         datetime.datetime(2005,8,31))} ,
614 'WQSP-2':{2000:(datetime.datetime(2000,9,1) ,
615         datetime.datetime(2000,9,30)) ,
616         2001:(datetime.datetime(2001,9,1) ,
617         datetime.datetime(2001,9,30)) ,
618         2002:(datetime.datetime(2002,9,1) ,
619         datetime.datetime(2002,9,30)) ,
620         2004:(datetime.datetime(2004,7,1) ,
621         datetime.datetime(2004,7,31))} ,
622 'WQSP-3':{2000:(datetime.datetime(2000,9,1) ,
623         datetime.datetime(2000,9,30)) ,
624         2001:(datetime.datetime(2001,9,1) ,
625         datetime.datetime(2001,9,30)) ,
626         2002:(datetime.datetime(2002,9,1) ,
627         datetime.datetime(2002,9,30))} ,
628 'WQSP-5':{2000:(datetime.datetime(2000,10,1) ,
629         datetime.datetime(2000,10,31)) ,
630         2001:(datetime.datetime(2001,10,1) ,
631         datetime.datetime(2001,10,31))} ,
632 'WQSP-6':{2000:(datetime.datetime(2000,11,1) ,
633         datetime.datetime(2000,11,30)) ,
634         2001:(datetime.datetime(2001,10,1) ,
635         datetime.datetime(2001,10,31))} }

```

```

636
637 if plot_timeseries:
638     computeyears = [2000,2001,2002,2003,2004,2005,2006,2007]
639
640     newfwhfh = {}
641     for year in computeyears:
642         newfwhfh[year] = open('meas_head_%iASER.smp' % year, 'w')
643
644     # minimum density range for plot
645     denmax = 1.100
646     denmin = 1.000
647     denrange = denmax - denmin
648
649     ##fhrpchange = open('reference-point-change-2007.dat', 'w')
650
651     # %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
652     # $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
653     # cycle over all Culebra wells, plotting figures for each
654     zone = 'CUL'
655     zmask = data['zone'] == zone
656     zdat = data[zmask]
657     for well in wells:
658         wm = zdat['well'] == well
659         if wm.sum() == 0:
660             continue # no data for this well
661         else:
662             print 'processing', zone, well
663
664         fig = plt.figure(1, figsize=(11,8.5))
665         ax1 = fig.add_subplot(211)
666
667         # plot reported (adjusted) WRES depth-to-water measurements (small red circles)
668         ax1.plot_date(zdat[wm]['dt'], zdat[wm]['dtwm'], 'ro', markersize=2, markeredgecolor='red')
669
670         # invert depth-to-water axis (bigger numbers on bottom)
671         ymin, ymax = ax1.get_ylim()
672         ax1.set_ylim([ymax, ymin])
673
674         pmask = zdat[wm]['fwhm'] > 0.0
675         pdata = (zdat[wm])[pmask]
676
677         # uncomment below and run once on data to generate corrections file
678         ##pre2007mask = pdata['dt'] < datetime.datetime(2007,1,1)
679         ##post2007mask = np.logical_not(pre2007mask)
680         ### only compute correction to reference point elevation if data straddles January/1/2
681         ##if pre2007mask.sum() > 0 and post2007mask.sum() > 0:
682         ##    oldrpelev = (pdata[pre2007mask]['dtwm'] + pdata[pre2007mask]['wlem']).mean()
683         ##    newrpelev = (pdata[post2007mask]['dtwm'] + pdata[post2007mask]['wlem']).mean()
684         ##    earlyrpcorr = newrpelev - oldrpelev
685         ##else:
686         ##    earlyrpcorr = 0.0
687         ##fhrpchange.write('%s\t%.2f\n' % (well, earlyrpcorr))
688
689         # plot freshwater heads on second y-axis
690         ax2 = ax1.twinx()
691
692         # plot freshwater head data as reported in ASER (small blue stars)
693         ax2.plot_date(pdata[:, 'dt'], pdata[:, 'fwhm'], 'b*', markersize=3, markeredgecolor='blue')
694
695         # correct FWH for densities chosen as "correct" densities over a given time range
696         fdmask = finden[:, 'well'] == well
697         fdata = finden[fdmask]
698
699         for dval in fdata:

```

```

700 # mask off ASER values in the date range associated with this density value
701 datemask = np.logical_and(pdata['dt'] >= dval['dt0'],
702                          pdata['dt'] <= dval['dt1'])
703
704 newfwhdates = pdata[datemask]['dt']
705
706 # compute new fwh using consistent density
707 newfwh = (pdata[datemask]['cwlem']-midpt[well])*dval['den'] + midpt[well]
708
709 # plot consistent freshwater heads (large blue x's)
710 ax2.plot_date(newfwhdates, newfwh, 'bx', markersize=5)
711
712 # save 2000-2007 fwh to file for use by PEST
713 for yr in computeyears:
714     exception = False
715     skipwell = False
716     if well in cexceptions:
717         if yr in cexceptions[well]:
718             exception = True
719             if cexceptions[well][yr][0] == None:
720                 # skip this well this year, even though there is data
721                 skipwell = True
722             else:
723                 # use a different month than the one selected for all other wells
724                 maskyrfwh = np.logical_and(newfwhdates >= cexceptions[well][yr][0],
725                                           newfwhdates <= cexceptions[well][yr][1])
726         if not exception:
727             # use the standard month selected for all other wells
728             maskyrfwh = np.logical_and(newfwhdates >= cmonths[yr][0],
729                                       newfwhdates <= cmonths[yr][1])
730
731         if maskyrfwh.sum() > 0 and not skipwell:
732             # use the same name if this well was used before,
733             # otherwise strip hyphen and go for it
734             if well.upper() in aserfwh[2010]:
735                 wname = aserfwh[2010][well]['name']
736             elif well.upper() in aserfwh[2009]:
737                 wname = aserfwh[2009][well]['name']
738             elif well.upper() in aserfwh[2008]:
739                 wname = aserfwh[2008][well]['name']
740             else:
741                 wname = well.replace('-0', '-') # remove leading zero
742                 wname = wname.replace('-', '') # remove hyphen
743
744             # if there are more than one fwh during that month, select the first one
745             newfwhfh[yr].write('%s\t%s\t12:00:00\t%.3f\t%.4f\n' %
746                               (wname, newfwhdates[maskyrfwh][0].strftime("%m/%d/%Y"),
747                               newfwh[maskyrfwh][0], dval['den']))
748
749 ax1.xaxis.set_major_formatter(NullFormatter())
750 ax1.set_title('%s water levels and specific gravities' % well)
751 ax2.xaxis.set_major_locator(YearLocator())
752 ax2.xaxis.set_minor_locator(MonthLocator())
753 ax2.xaxis.set_major_formatter(DateFormatter('%Y'))
754 ax1.set_ylabel('ASER depth to water (m BTOC)', color='red', fontsize=9)
755
756 ax2.set_ylabel('freshwater head (ASER=*, this rept=x) (m AMSL)',
757               color='blue', fontsize=9)
758
759 # add date ranges used for contouring
760 for yr in cmonths.keys():
761     exception = False
762     if well in cexceptions:
763         if yr in cexceptions[well]:

```



```

764         exception = True
765     if cexceptions[well][yr][0] == None:
766         # don't use this well this year
767         continue
768     else:
769         # use a different month for this well this year
770         ax2.axvspan(cexceptions[well][yr][0], cexceptions[well][yr][1],
771                   alpha=0.25, color='green')
772     if not exception:
773         # use the standard month for this well and year
774         ax2.axvspan(cmonths[yr][0], cmonths[yr][1], alpha=0.25, color='blue')
775 axd = fig.add_subplot(212)
776
777 # apparent specific gravity is (fwh-el - midpt-el)/(wl-el - midpt-el)
778 # computed from wl elevation and freshwater head reported by WRERS +
779 # Culebra midpoint elevations (small green circles)
780 specgrav = (((zdat[wm])[pmask]['fwhm'] - midpt[well])/
781             ((zdat[wm])[pmask]['wlem'] - midpt[well]))
782 axd.plot_date((zdat[wm])[pmask]['dt'], specgrav, 'go',
783              markersize=2, markeredgcolor='green')
784
785 fdmask = finden[:, 'well'] == well
786 fdata = finden[fdmask]
787 for dval in fdata:
788     # plot the "final" gravity as a line across the figure (solid green line)
789     axd.plot_date([dval['dt0'], dval['dt1']], [dval['den'], dval['den']], 'g-')
790
791 # plot reported density values; different symbols
792 dm = den['well'] == well
793 ddat = den[dm]
794 settype = set(ddat[:, 'type'])
795 densitytypes = list(settype)
796
797 # types are saner than the "source" field was before, which
798 # had the year, and other information in with the type too.
799 for dtype in densitytypes:
800     # they are AVG-PDS, AVG-SGR, and AVG-SGR-PDS
801     # just reduce them all to AVG
802     if 'AVG' in dtype:
803         settype.remove(dtype)
804         settype.add('AVG')
805
806 densitytypes = list(settype)
807 for dtype in densitytypes:
808     # densities from this sources (all TROLLYYY just count as TROLL, etc.)
809     mddat = npchar.find(ddat[:, 'type'], dtype) >= 0 # find returns -1 if not found
810     sddat = ddat[mddat]
811
812     # densities since cutoffdate
813     msddat = sddat['dt'] > firstplot
814     dsddat = sddat[msddat]
815     nrecent = msddat.sum()
816
817     if dtype == 'TROLL':
818         if nrecent > 0:
819             tmpden = sddat[msddat]['den']
820             # move any troll-computed densities below 1.0 to fresh water (1.0)
821             tmpden[tmpden < 1.0] = 1.0
822             # troll densities plot as a date range
823             axd.plot_date([sddat[msddat]['dt'], sddat[msddat]['dt2']], [tmpden, tmpden],
824                          typesymb[dtype], linestyle='solid', color=typecolor[dtype],
825                          linewidth=2.0, label=dtype, markersize=9)
826
827 else:

```

```

828     if nrecent > 0:
829         tmpden = sddat[msddat]['den']
830         # move any densities below 1.0 to fresh water (1.0)
831         tmpden[tmpden < 1.0] = 1.0
832         # non-troll densities plot as a single date
833         axd.plot_date(sddat[msddat]['dt'], tmpden, typesymb[dtype],
834                     color=typecolor[dtype], label=dtype, markersize=9)
835
836     # densities before cutoff date (plot on edge with left-pointing triangle)
837     msddat = sddat['dt'] < firstplot
838     nold = msddat.sum()
839
840     if nold > 0:
841         axd.plot_date([firstplot]*nold, sddat[msddat]['den'], '<',
842                     color='yellow', label='pre-1999', markersize=9)
843
844     # make range of density plots at least a minimum range
845     ymin, ymax = axd.get_ylim()
846     if ymax < denmax:
847         ymax = denmax
848
849     if ymin > denmin:
850         ymin = denmin
851
852     axd.set_ylim((ymin, ymax))
853
854     # add pumping, drilling, and plugging events located at the current well
855     em = events[:, 'well'] == well
856     if em.sum() > 0:
857         ee = events[em]
858         ymin, ymax = axd.get_ylim()
859         yann = ymax - (ymax - ymin)/5.0
860         for ev in ee:
861             if ev['dt1'] == None:
862                 # no ending date
863                 axd.axvline(ev['dt0'], alpha=0.5, color=ev['c'])
864                 axd.annotate(ev['s'], (ev['dt0'], yann),
865                             rotation='vertical', fontproperties=AnnFontP)
866             else:
867                 axd.axvspan(ev['dt0'], ev['dt1'], alpha=0.25, color=ev['c'])
868                 axd.annotate(ev['s'], (ev['dt0']+(ev['dt1']-ev['dt0'])/2, yann),
869                             rotation='vertical', fontproperties=AnnFontP)
870
871
872     # add drilling and P&A events for wells within 500 m (i.e., same pad)
873     for other in ewells:
874         if not other == well:
875             em = events[:, 'well'] == other
876             ee = events[em]
877             om = np.logical_or(ee[:, 'c'] == 'red', ee[:, 'c'] == 'green')
878             dist = np.sqrt(((xyz[well]['x']-xyz[other]['x'])**2 +
879                           (xyz[well]['y']-xyz[other]['y'])**2))
880             # does other well have drilling or p&a activities?
881             if om.sum() > 0:
882                 if dist < 500:
883                     for ev in ee[om]:
884                         if ev['dt1'] == None:
885                             # no ending date
886                             axd.axvline(ev['dt0'], alpha=0.5,
887                                         linestyle='dashed', color=ev['c'])
888                         else:
889                             axd.axvspan(ev['dt0'], ev['dt1'], alpha=0.25,
890                                         linestyle='dashed', color=ev['c'])
891                             axd.annotate('%s (%im) %s' % (ev['well'], dist, ev['s']), (ev['dt0'], yann),

```

```

892 rotation='vertical', fontproperties=AnnFontP, color='gray')
893
894 # nearby pumping tests (not slug tests) within 5.0 km
895 # npchar.find() returns -1 for not found
896 om = np.logical_and( (ee[:, 'c'] == 'cyan',
897                    npchar.find( ee[:, 's'], 'slug') == -1)
898 if om.sum() > 0:
899     if dist < 5000:
900         for ev in ee[om]:
901             if ev['dt1'] == None:
902                 # no ending date
903                 axd.axvline( ev['dt0'], alpha=0.5,
904                            linestyle='dashed', color=ev['c'])
905             else:
906                 axd.axvspan( ev['dt0'], ev['dt1'], alpha=0.25,
907                             linestyle='dashed', color=ev['c'])
908                 axd.annotate( '%s (%.1fkm) %s' % (ev['well'], dist/1000.0, ev['s']),
909                              (ev['dt0'], yann), rotation='vertical',
910                              fontproperties=AnnFontP, color='gray')
911
912 ax2.set_xlim( left=datetime.datetime(1998,3,1)
913             right=datetime.datetime(2011,1,1) )
914
915 # force subplots to have same data range
916 axd.set_xlim( ax2.get_xlim() )
917 axd.set_ylabel( 'specific gravity' )
918
919 # legend for type of density measurement
920 # while removing duplicate entries from legend
921 handles, labels = axd.get_legend_handles_labels()
922 newhandles = []
923 newlabels = []
924 if len(handles) > 0:
925     for h,l in zip(handles, labels):
926         if not l in newlabels:
927             newhandles.append(h)
928             newlabels.append(l)
929
930 leg = axd.legend( newhandles, newlabels, loc=0, prop=LegFontP, numpoints=1, scatterpoints=1 )
931
932 axd.xaxis.set_major_locator( YearLocator() )
933 axd.xaxis.set_minor_locator( MonthLocator() )
934 axd.xaxis.set_major_formatter( DateFormatter('%Y') )
935 if well == 'AEC-7':
936     ax1.set_ylim( [195,180] )
937     ax2.set_ylim( [925,940] )
938 plt.savefig( '%s-%s-ASER-waterlevels.png' % (zone, well), dpi=150 )
939 plt.close(1)
940
941 for fh in newfwhfh:
942     newfwhfh[fh].close()
943 #@#rhrpchange.close()

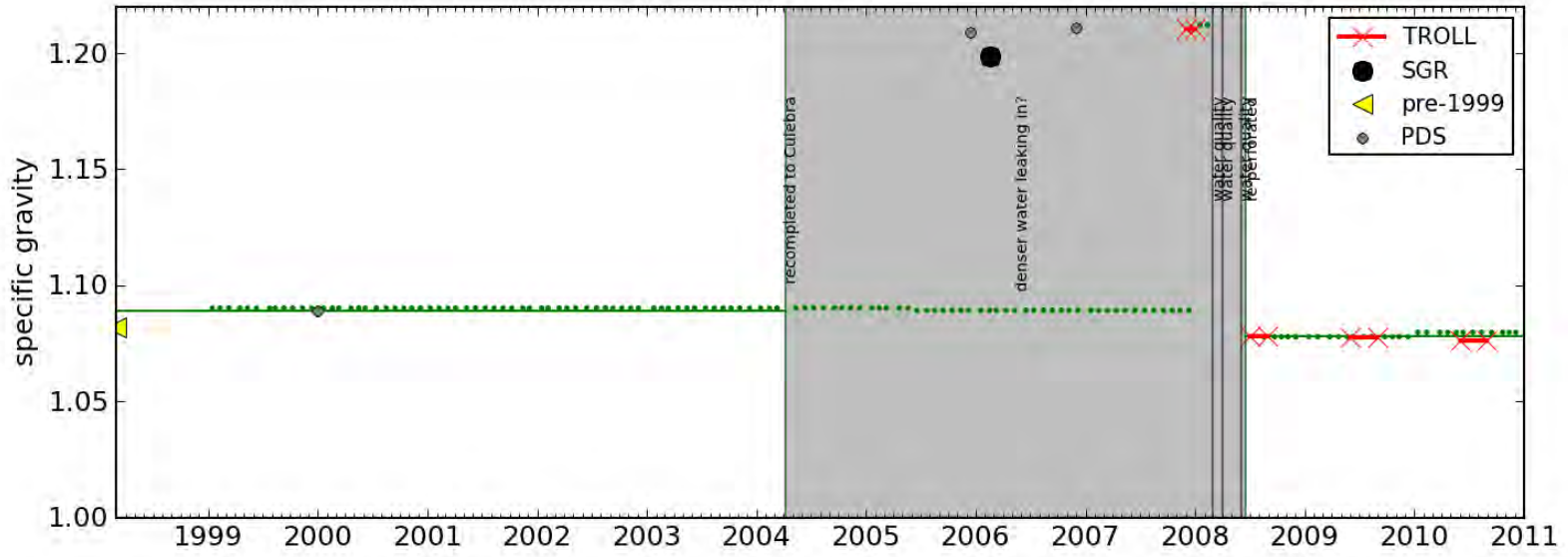
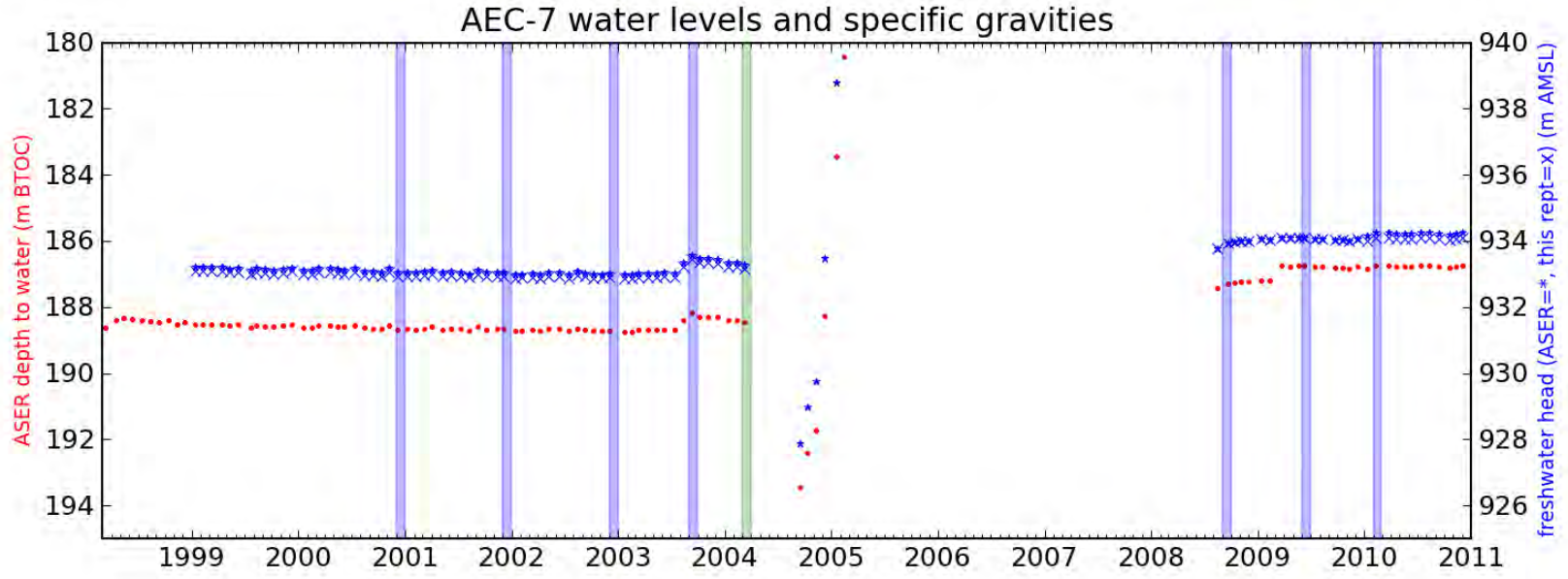
```

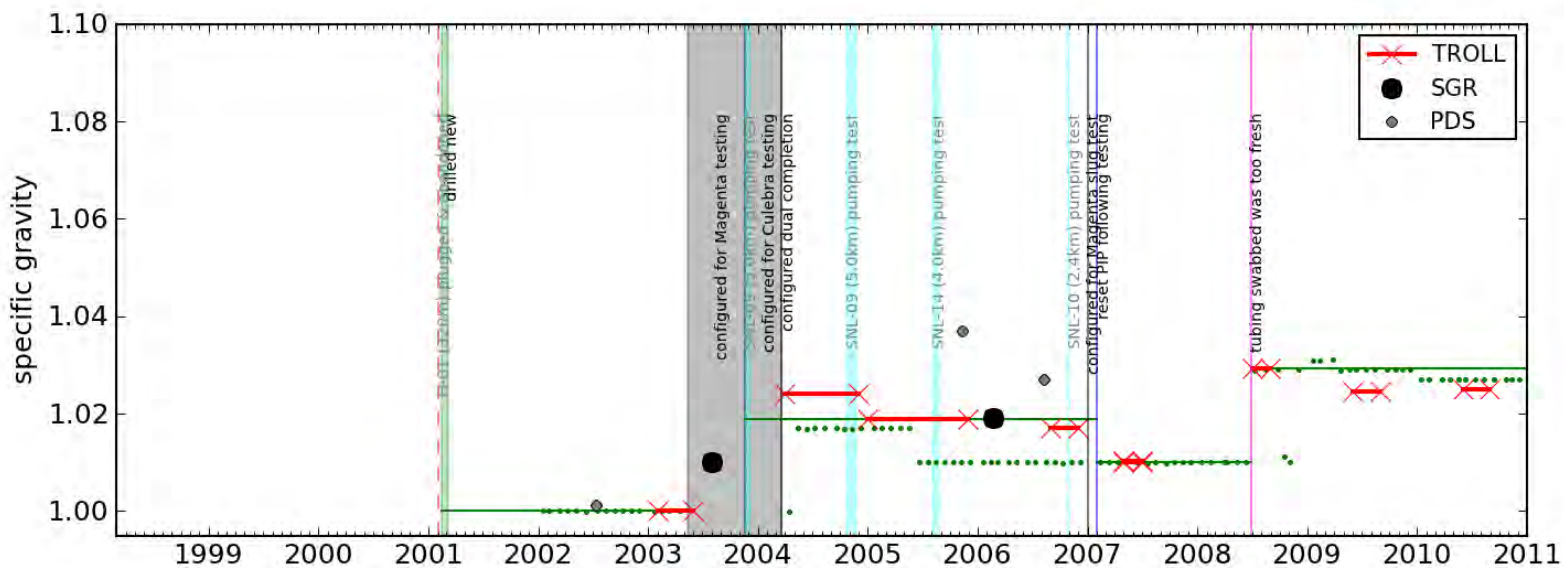
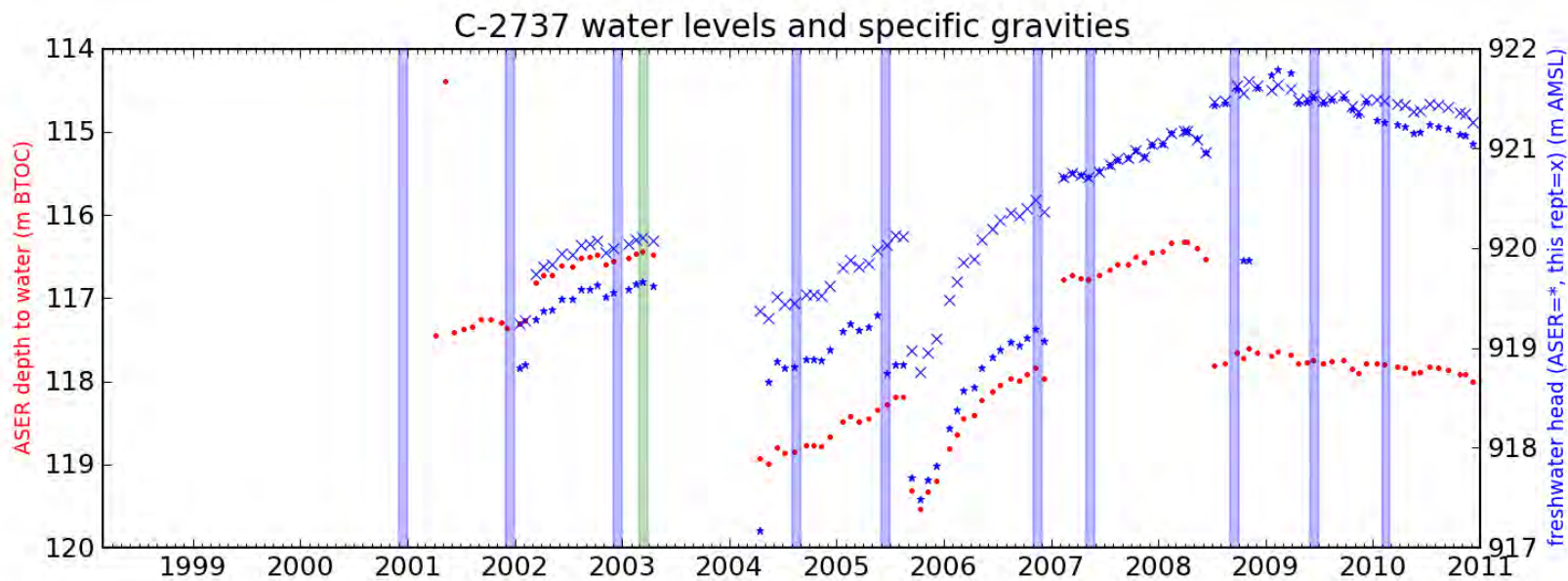
11.3 Figures Generated by Python Water Level Script

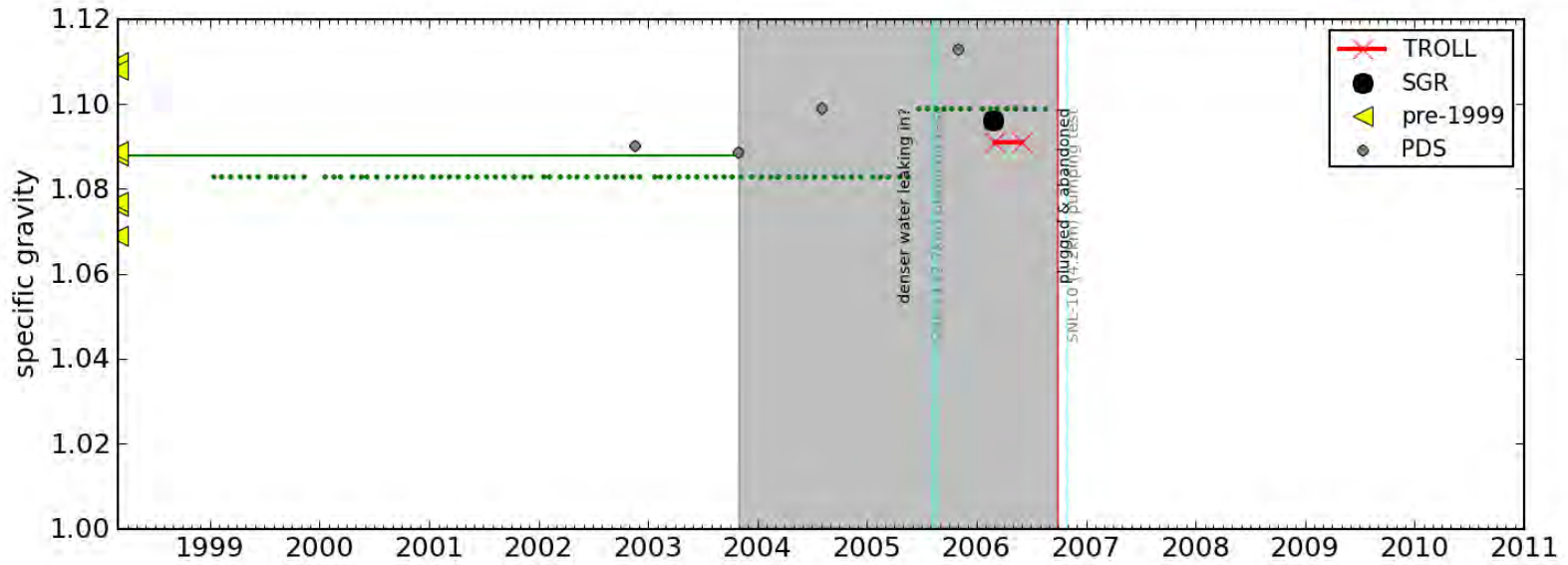
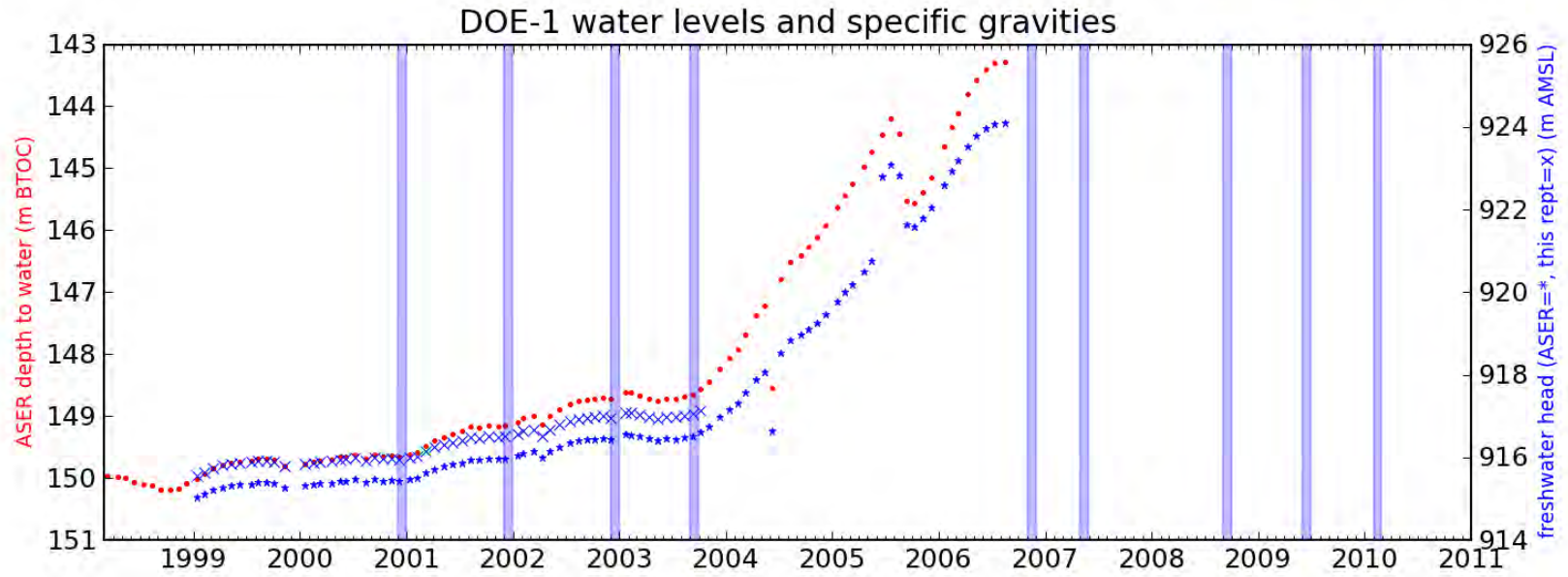
The following figures were generated by the Python script `plot-waterlevels.py` and represent the water level, density (aka specific gravity), and well-event data listed in the 2000-2010 ASERs.

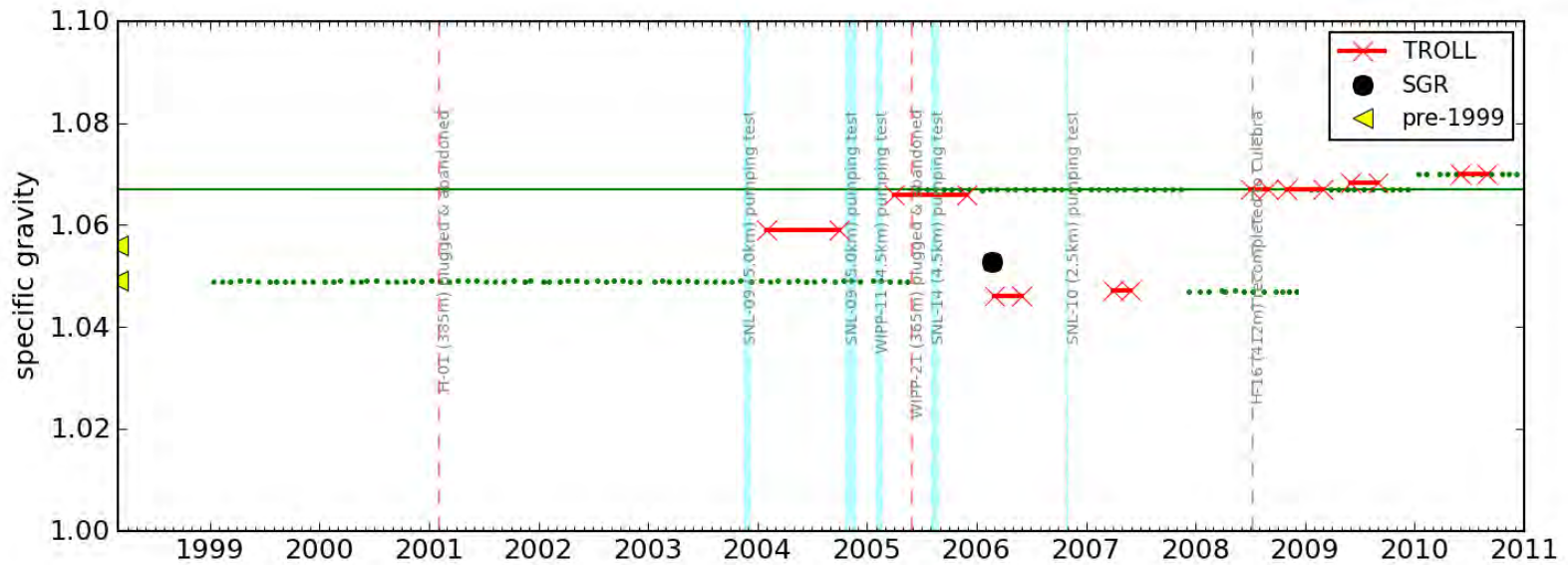
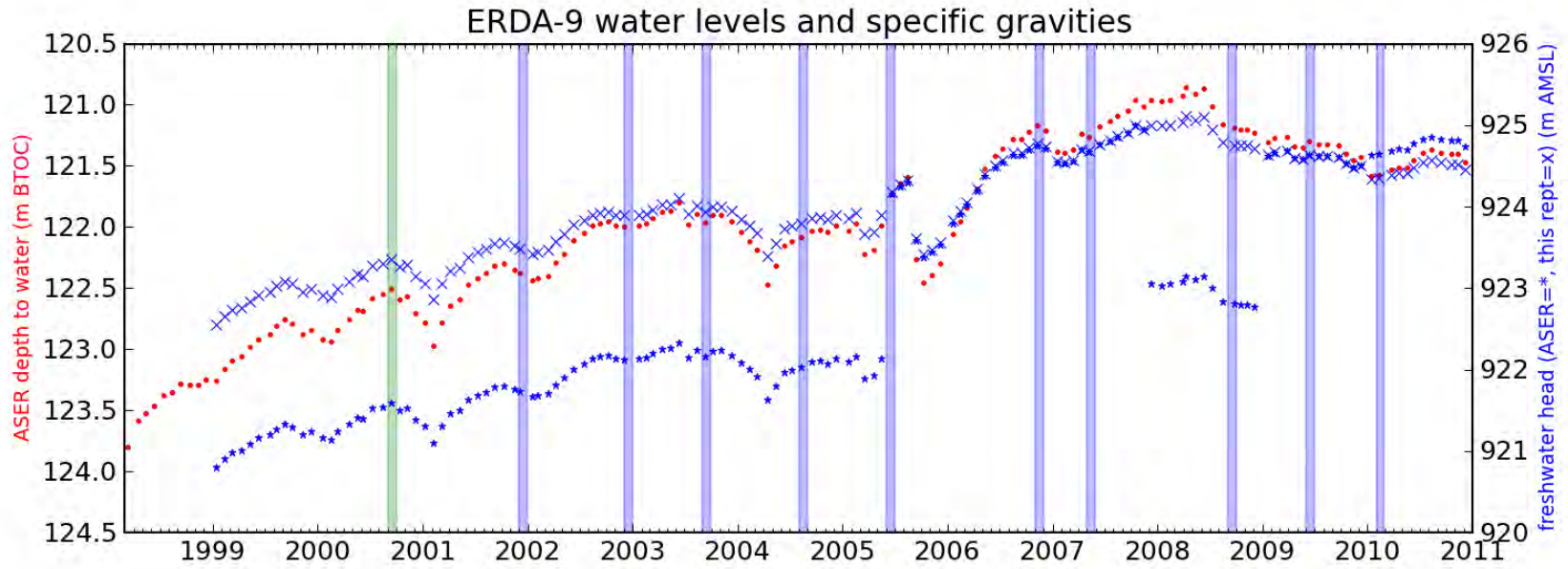
Each page represents the 2000-current ASER data for a given Culebra well. In the water level plots (top), filled red circles are reported depths to water (meters below top of casing (BTOC)), filled blue stars are ASER-reported freshwater head elevations (meters above mean sea level (AMSL)), and the blue \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.

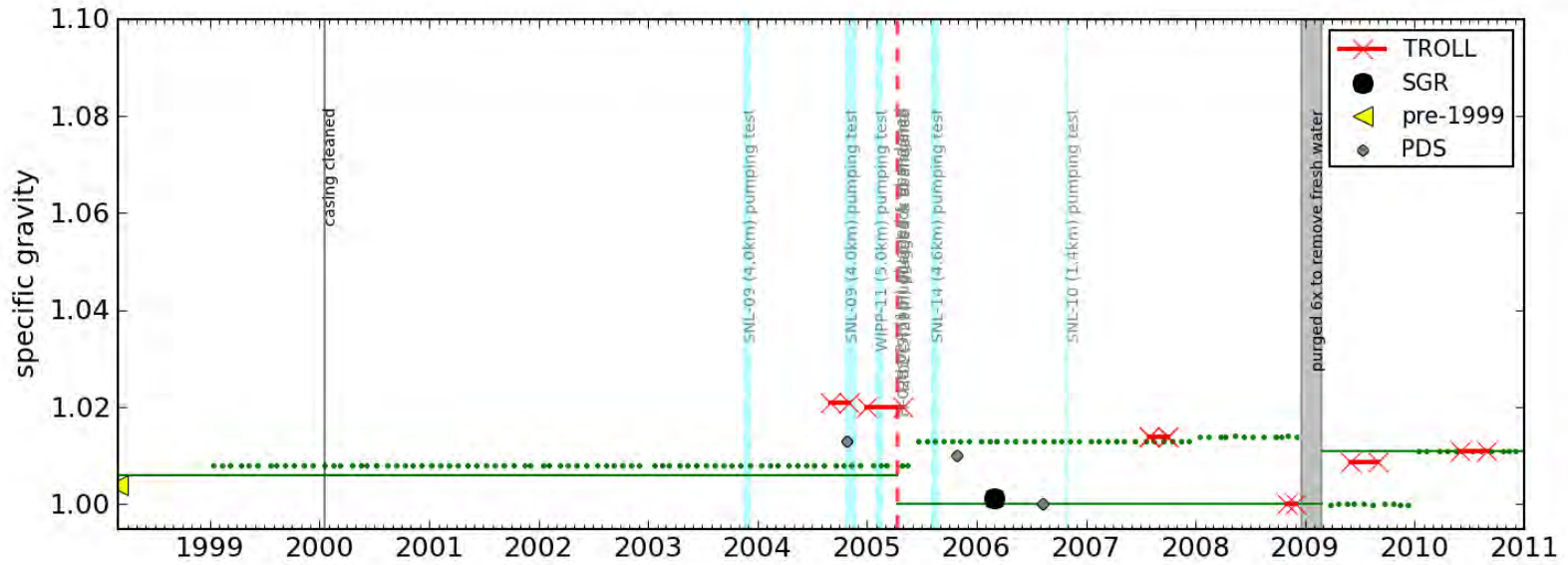
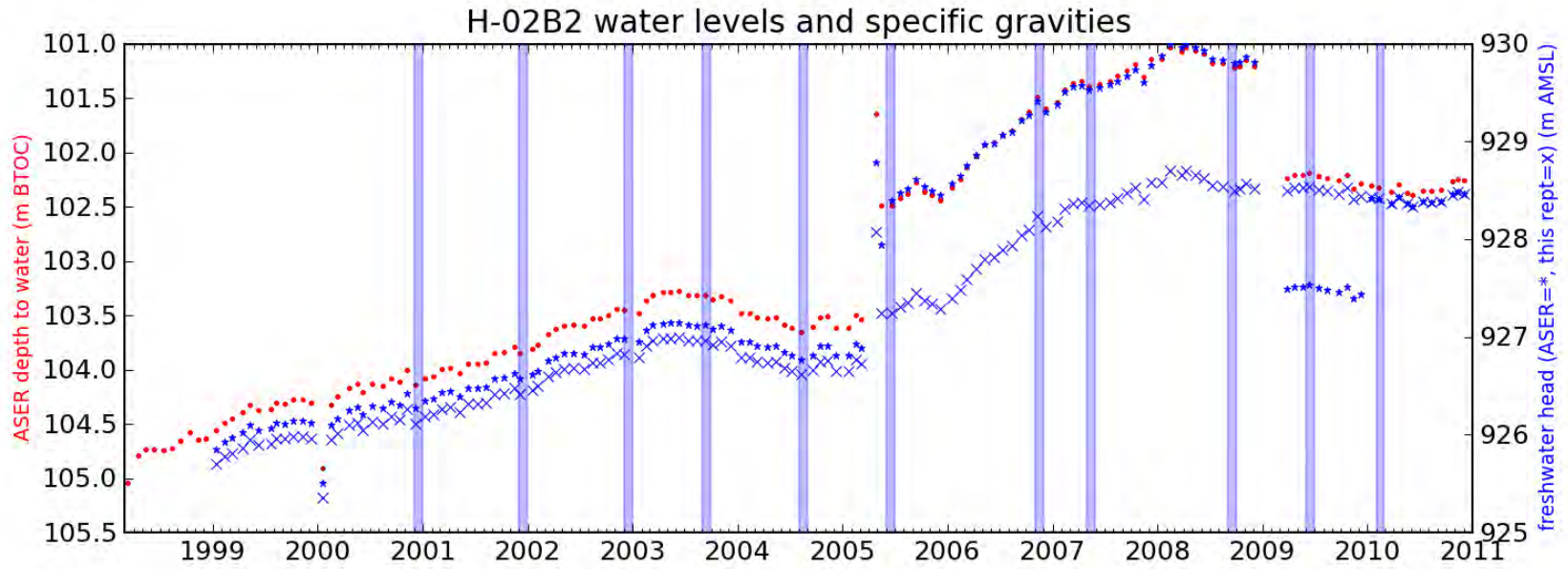
In the density plots (bottom), horizontal green lines indicate the density/specific gravity values chosen to be used at a given time (`densities-to-use.csv`), vertical lines are events in current or nearby wells (nearby wells are gray text and indicate the distance between the wells, while the current 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.

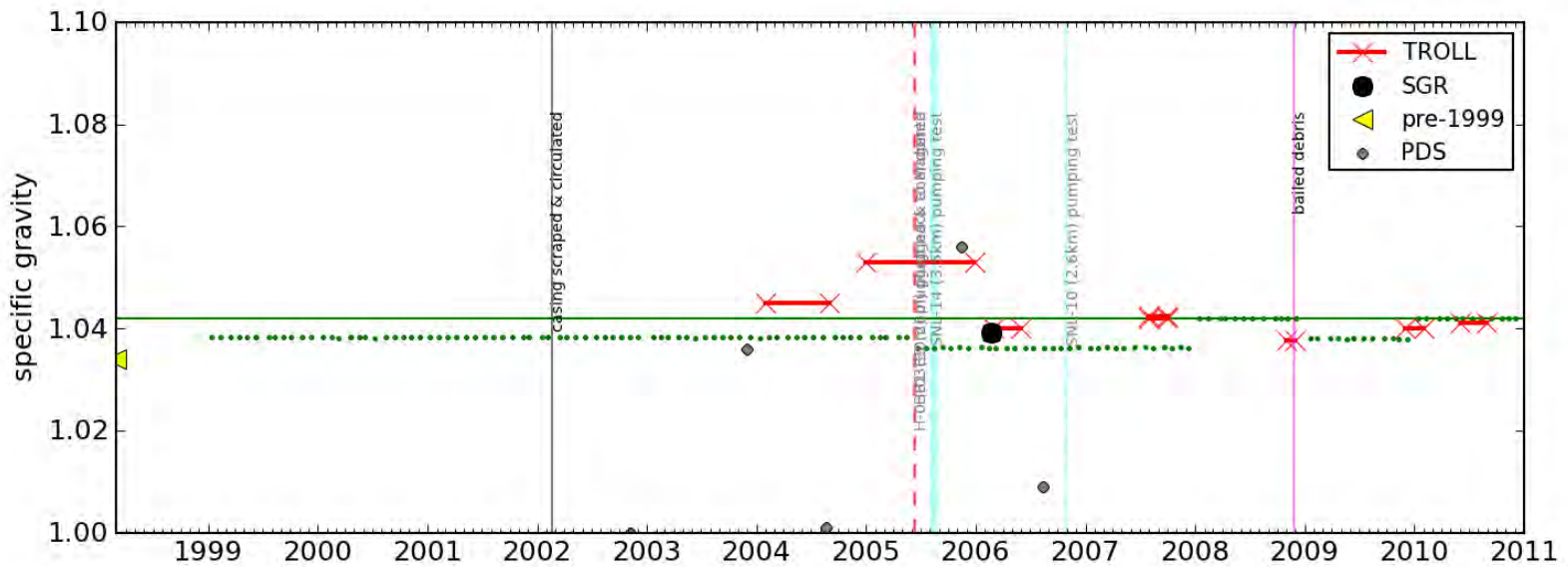
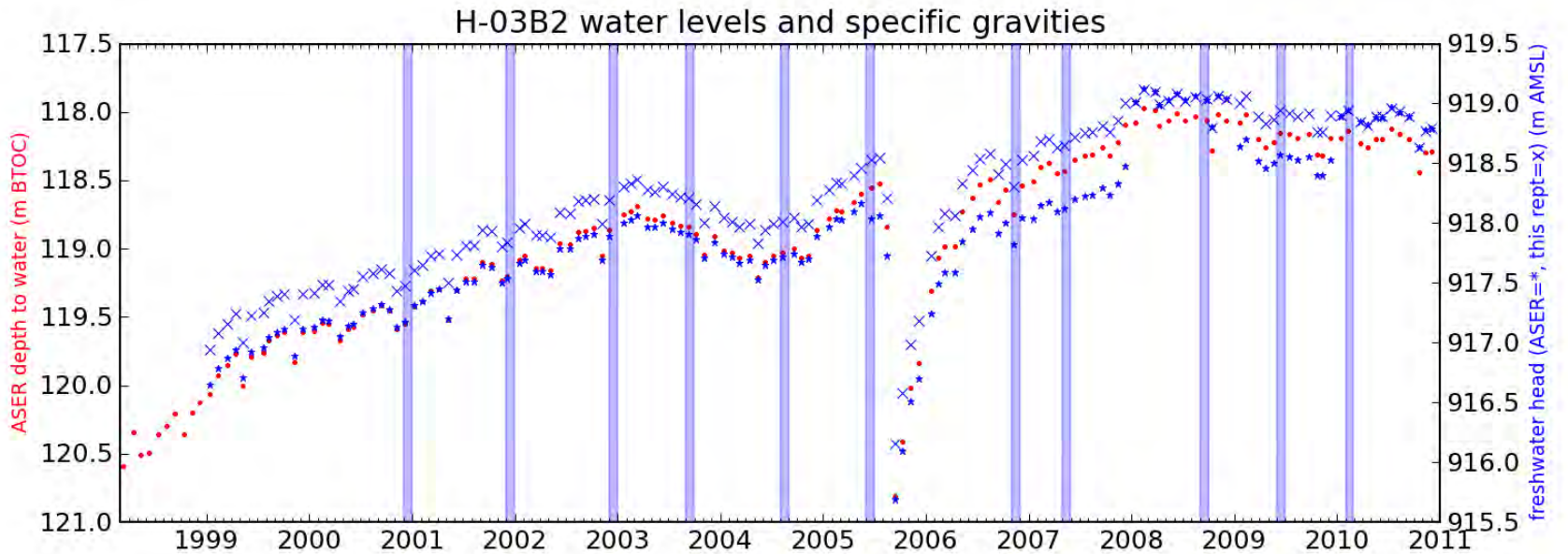


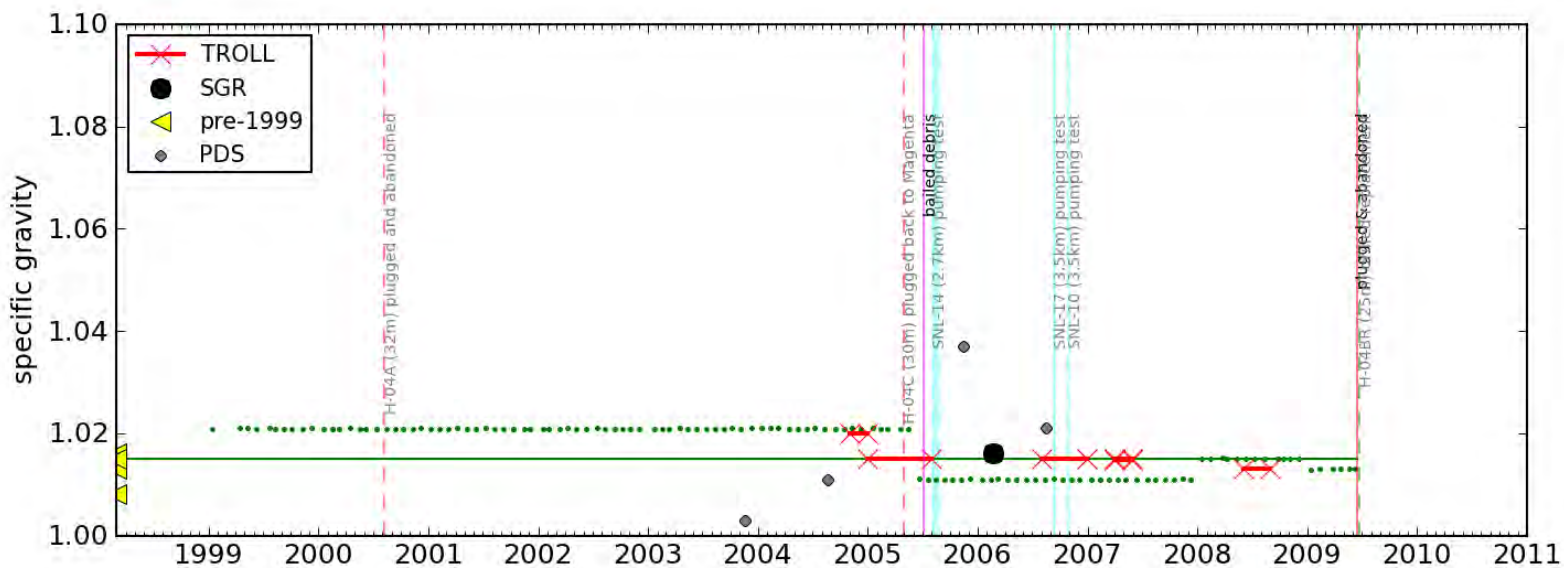
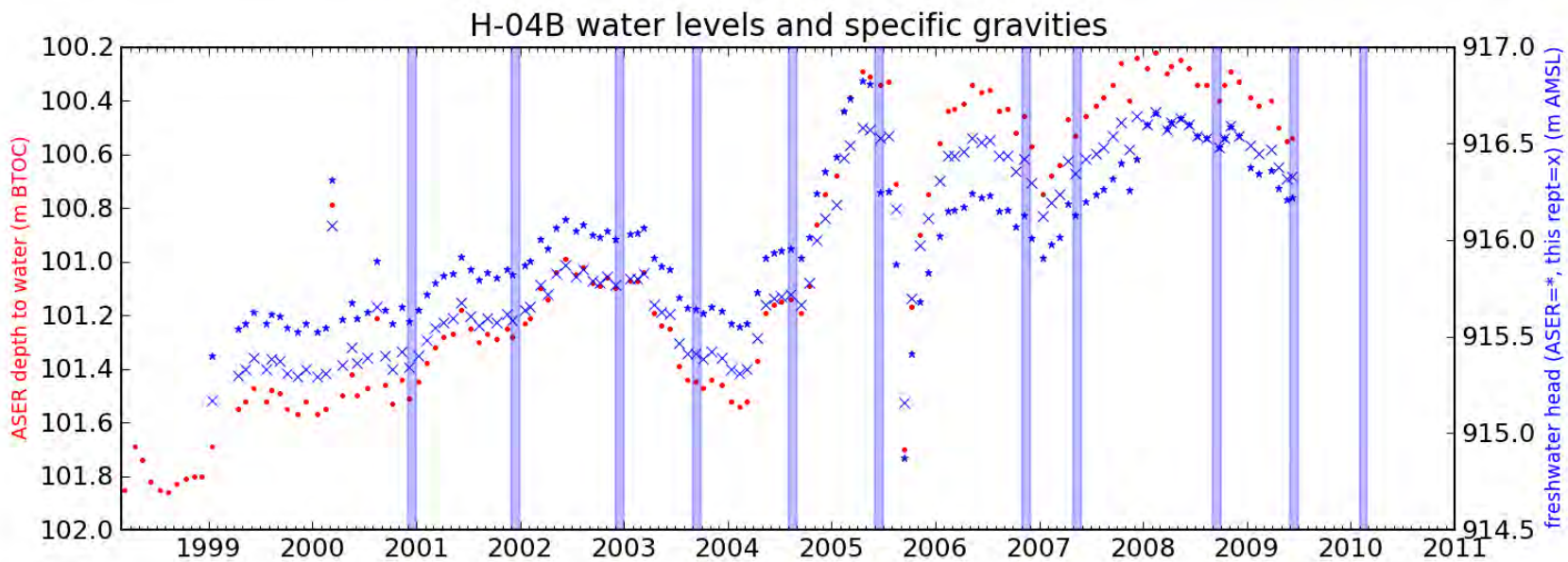


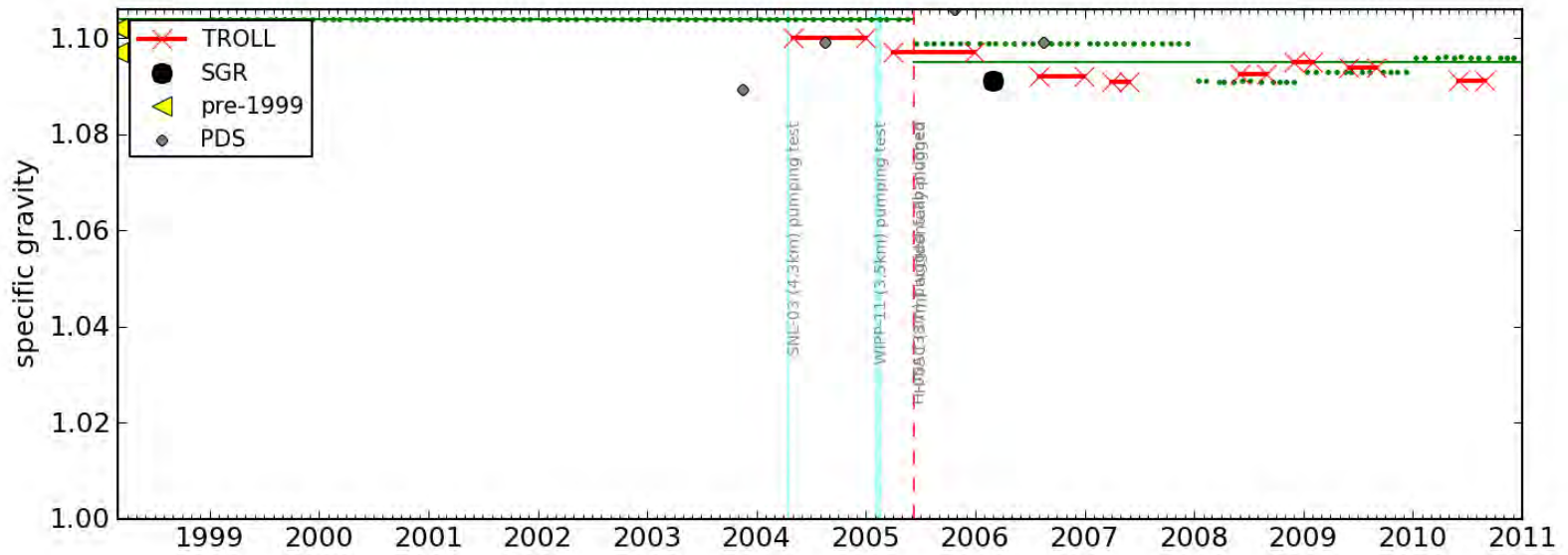
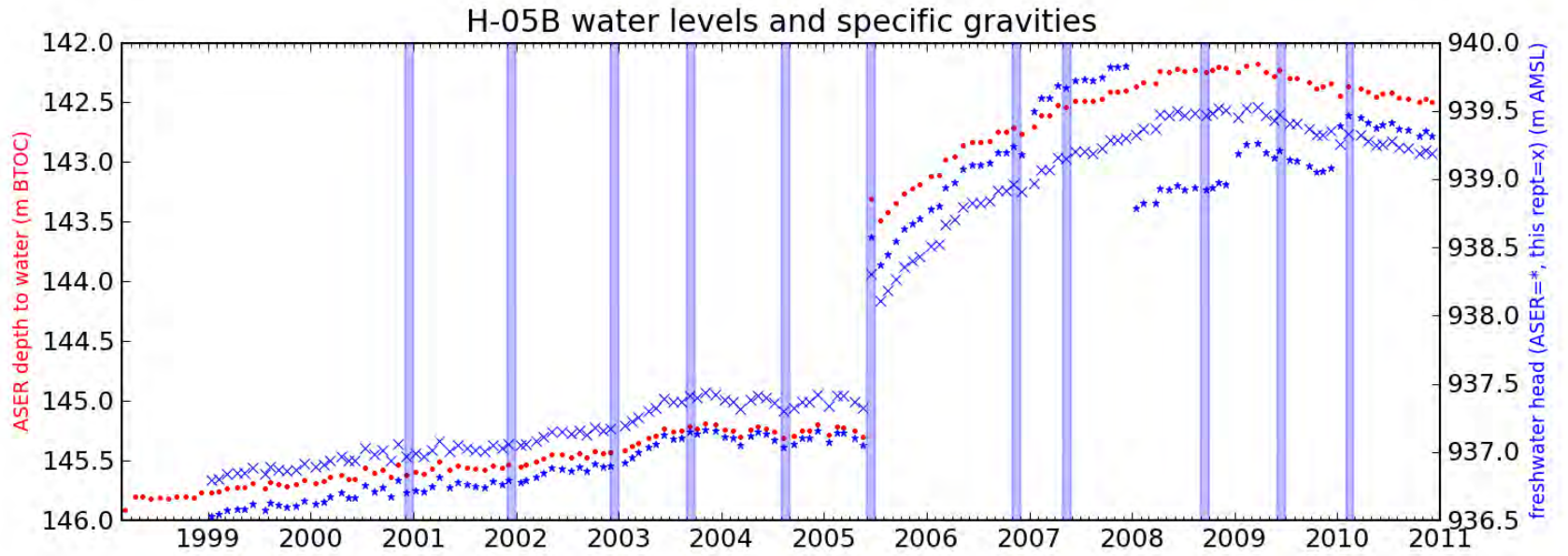


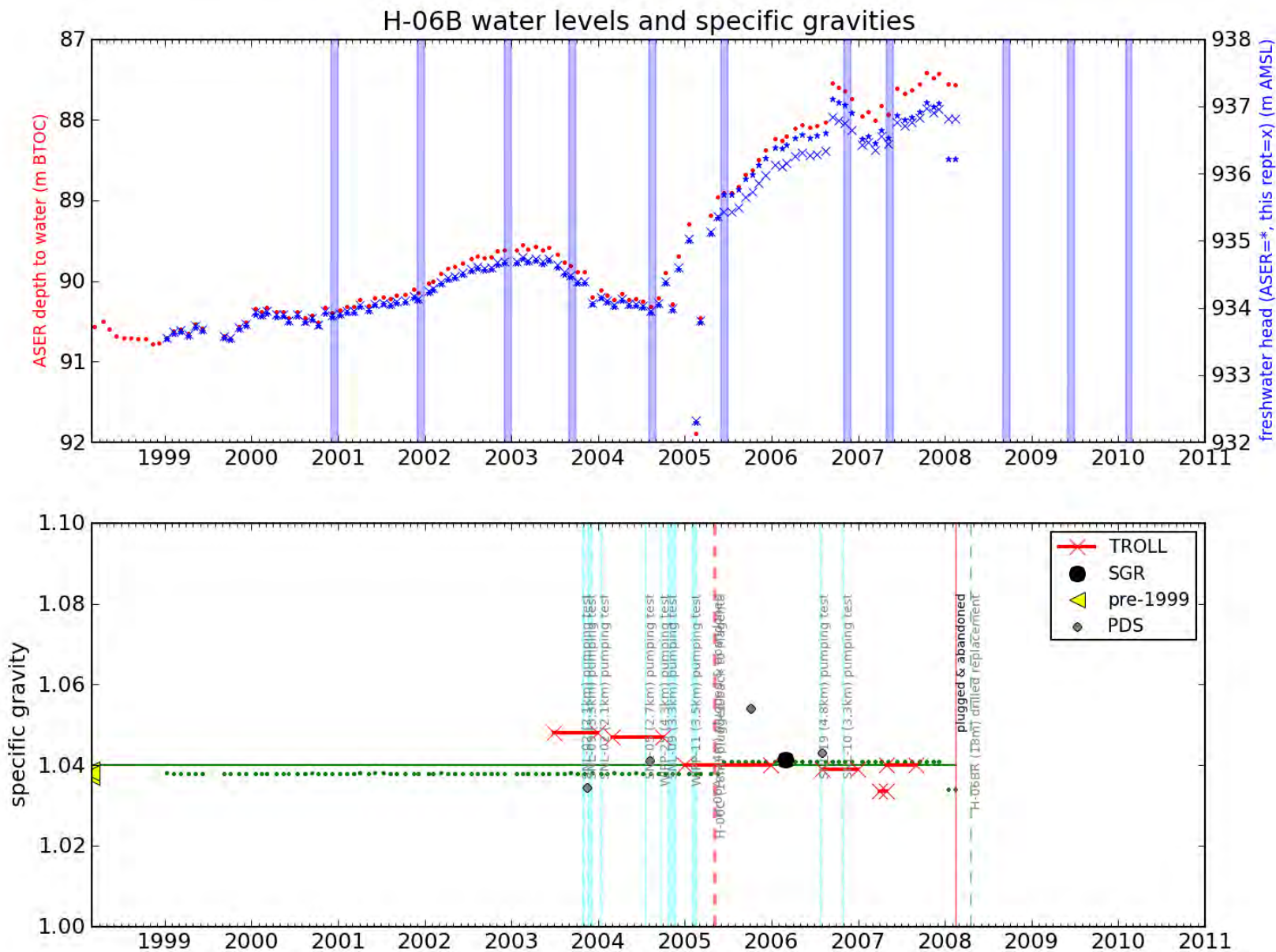


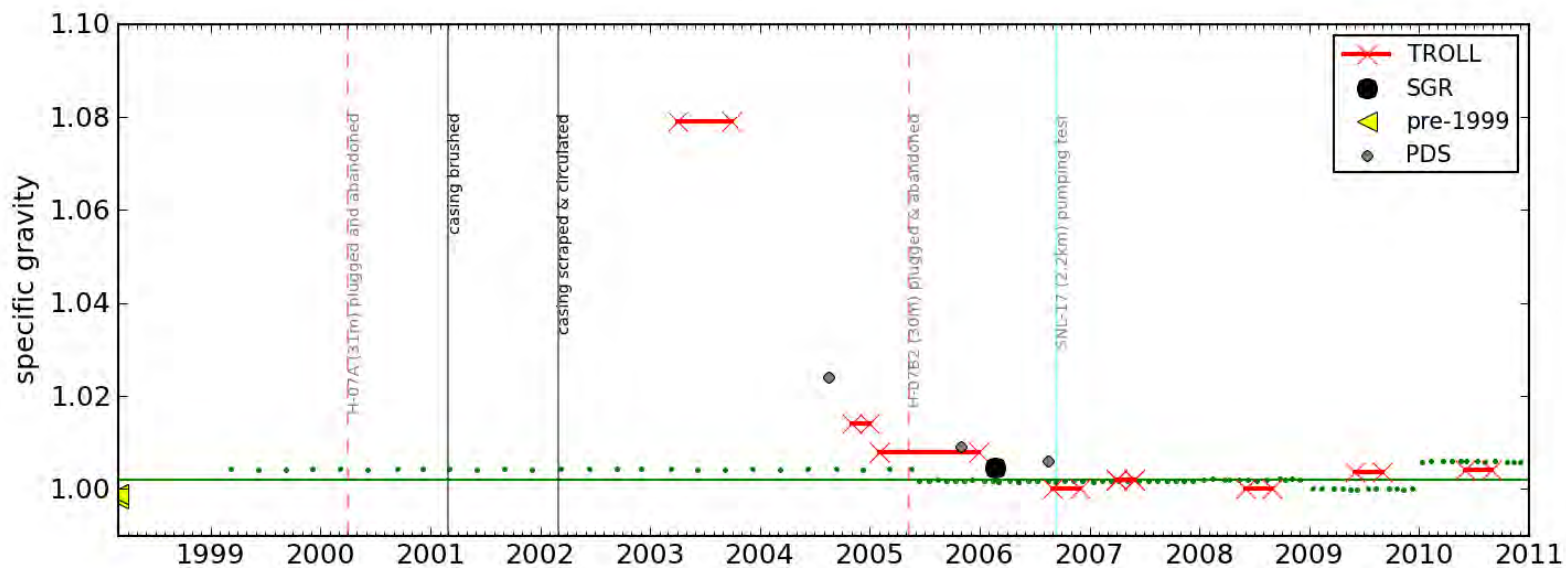
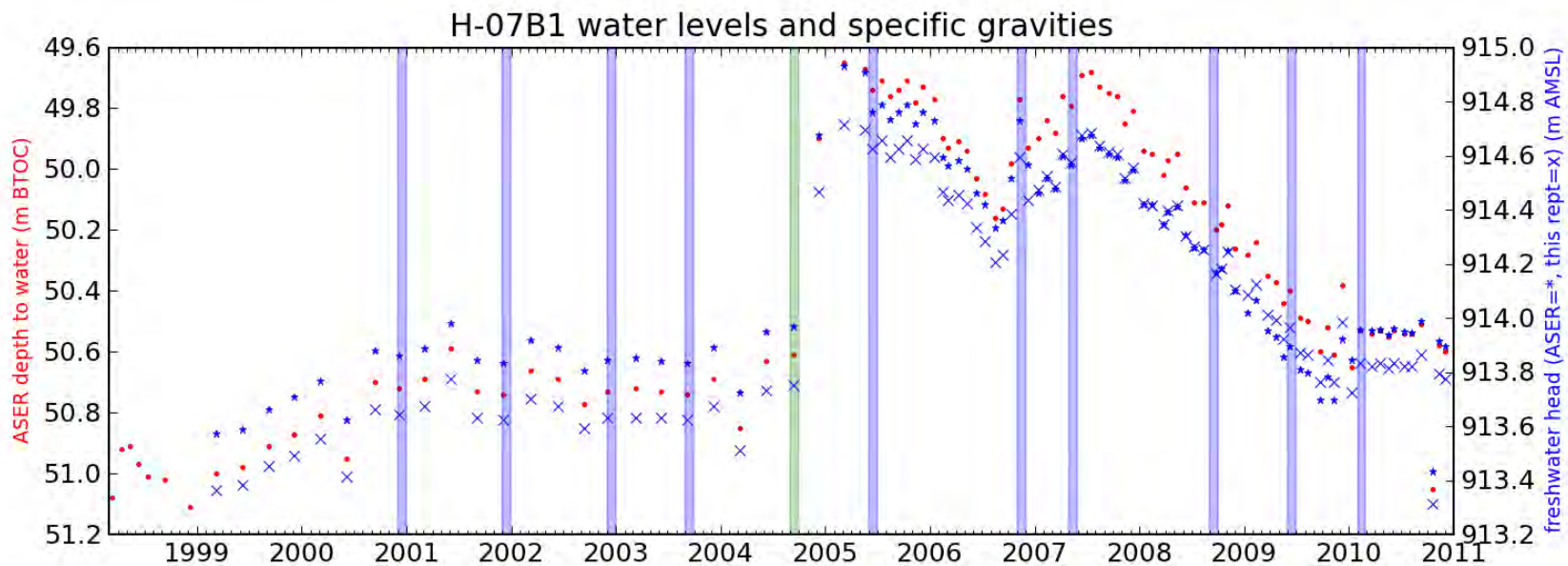


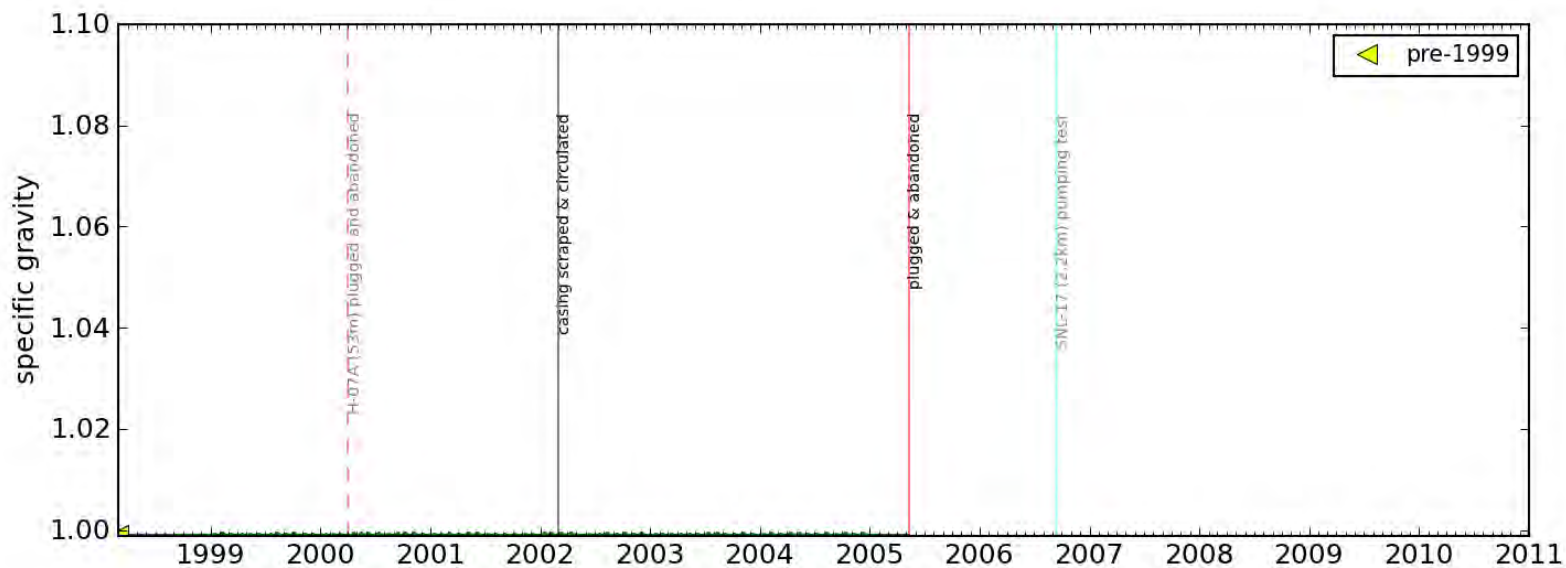
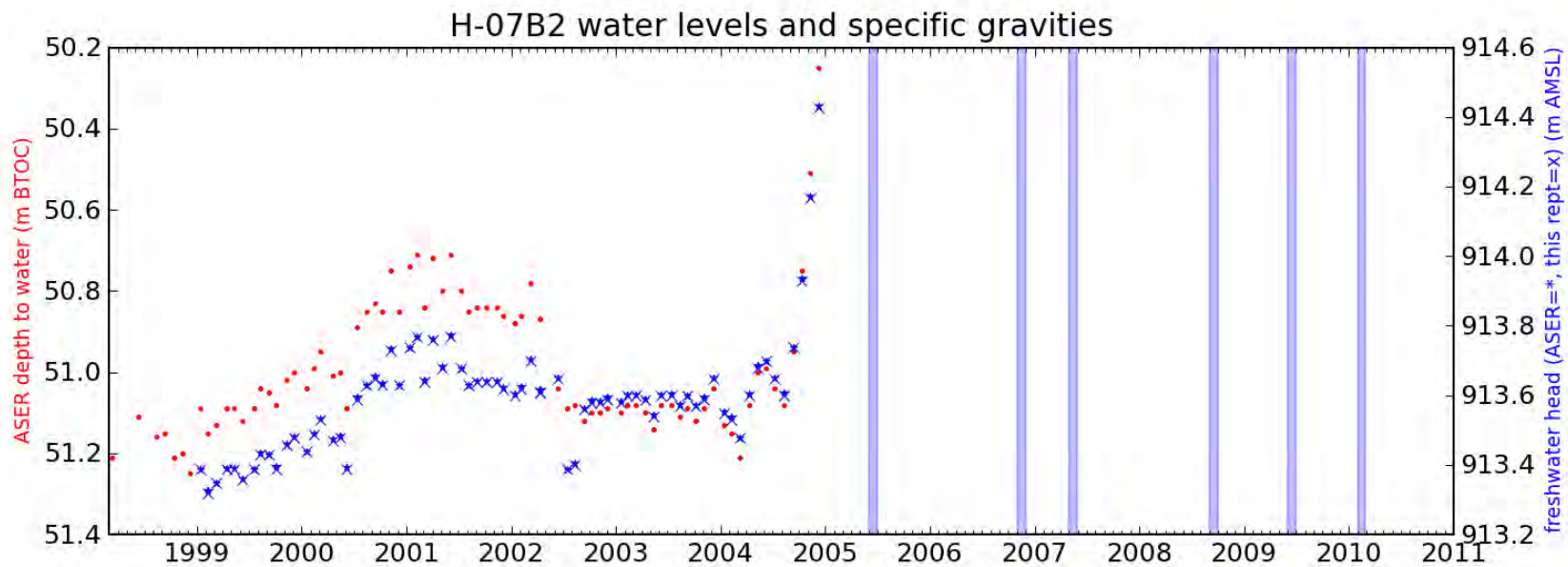


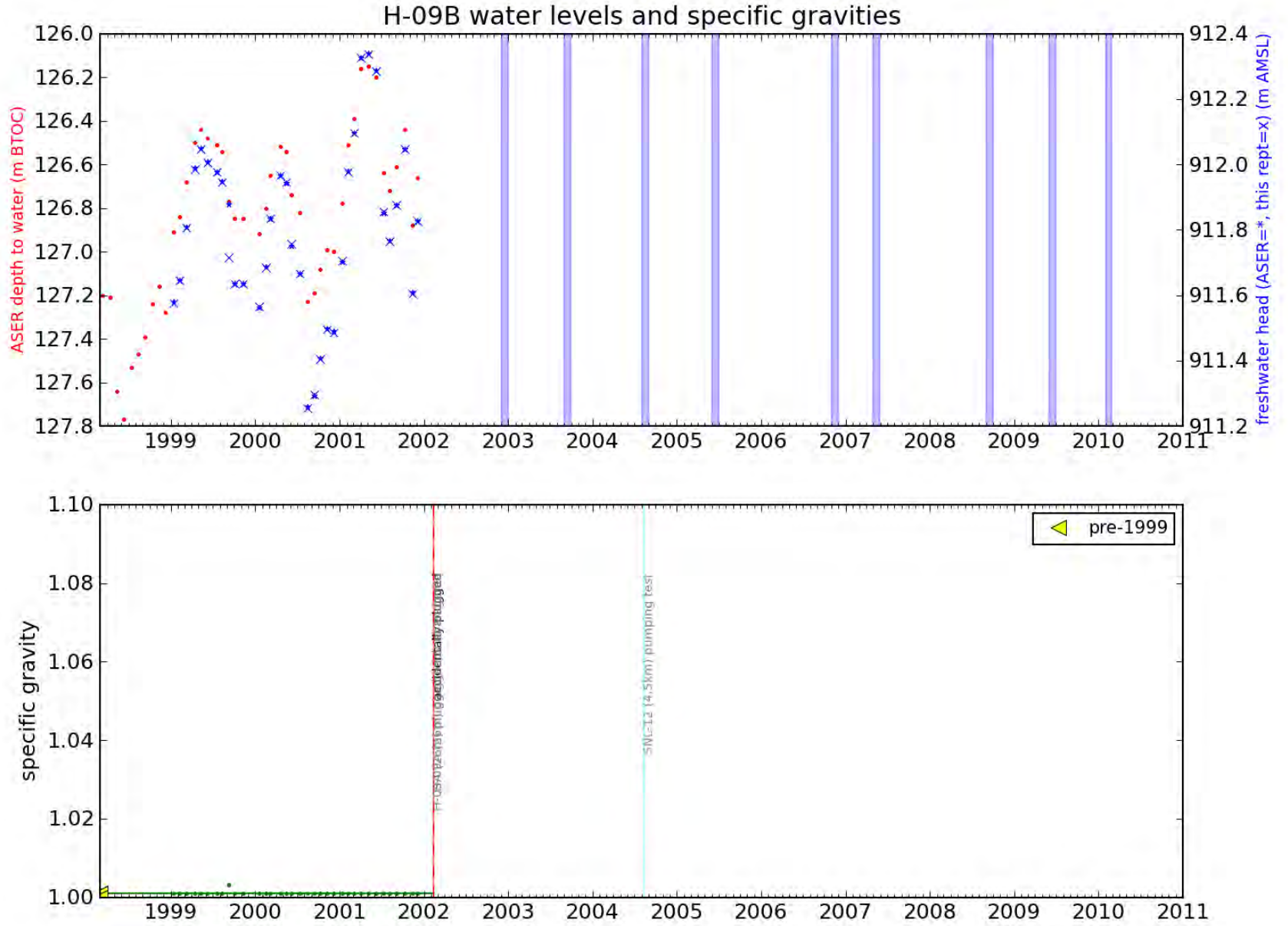


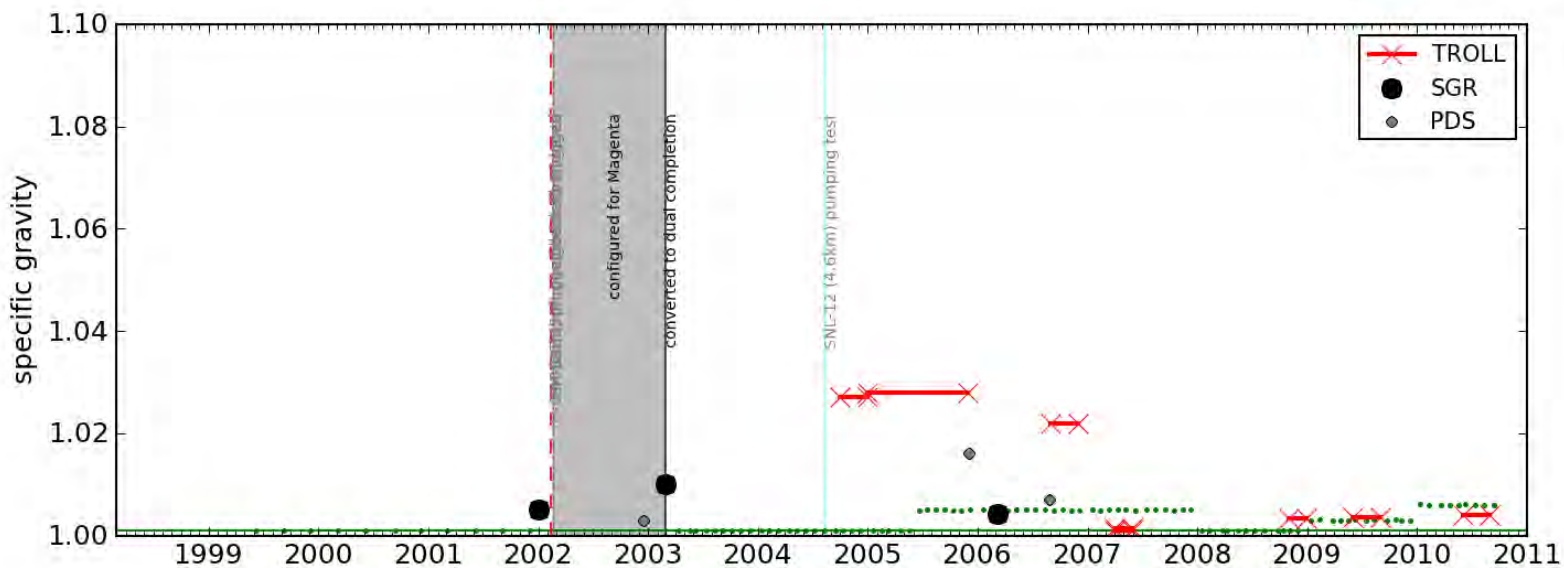
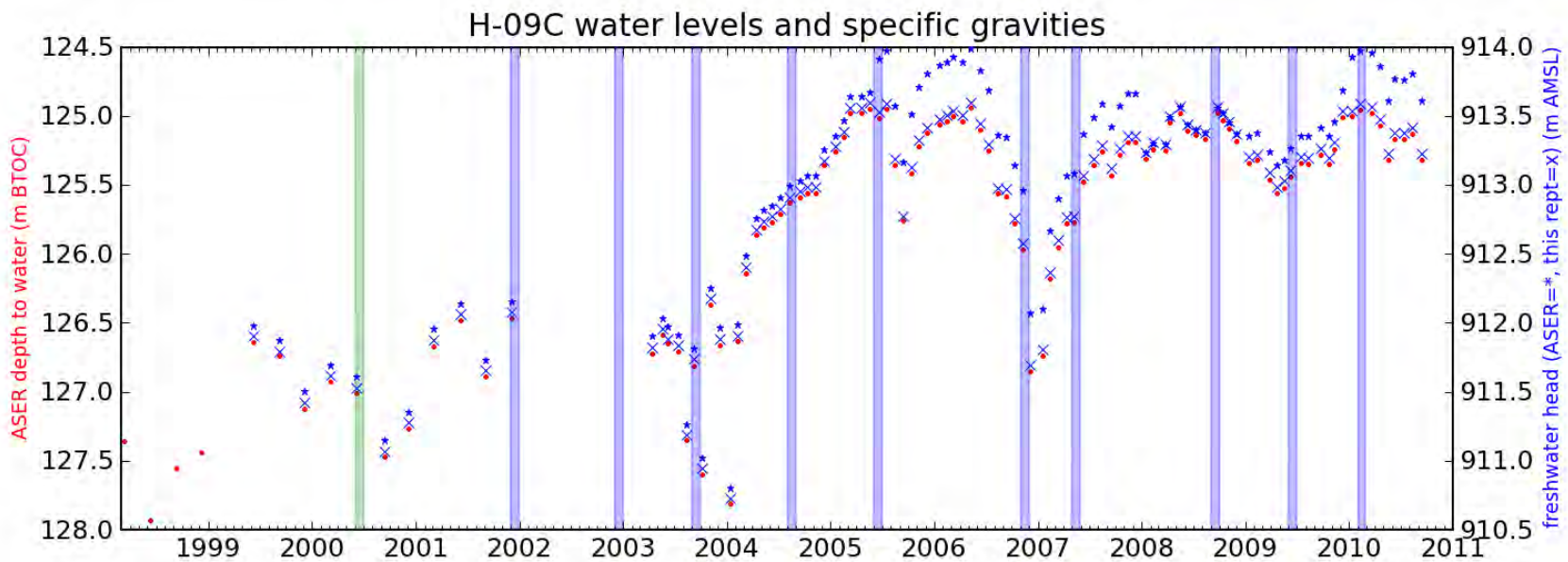


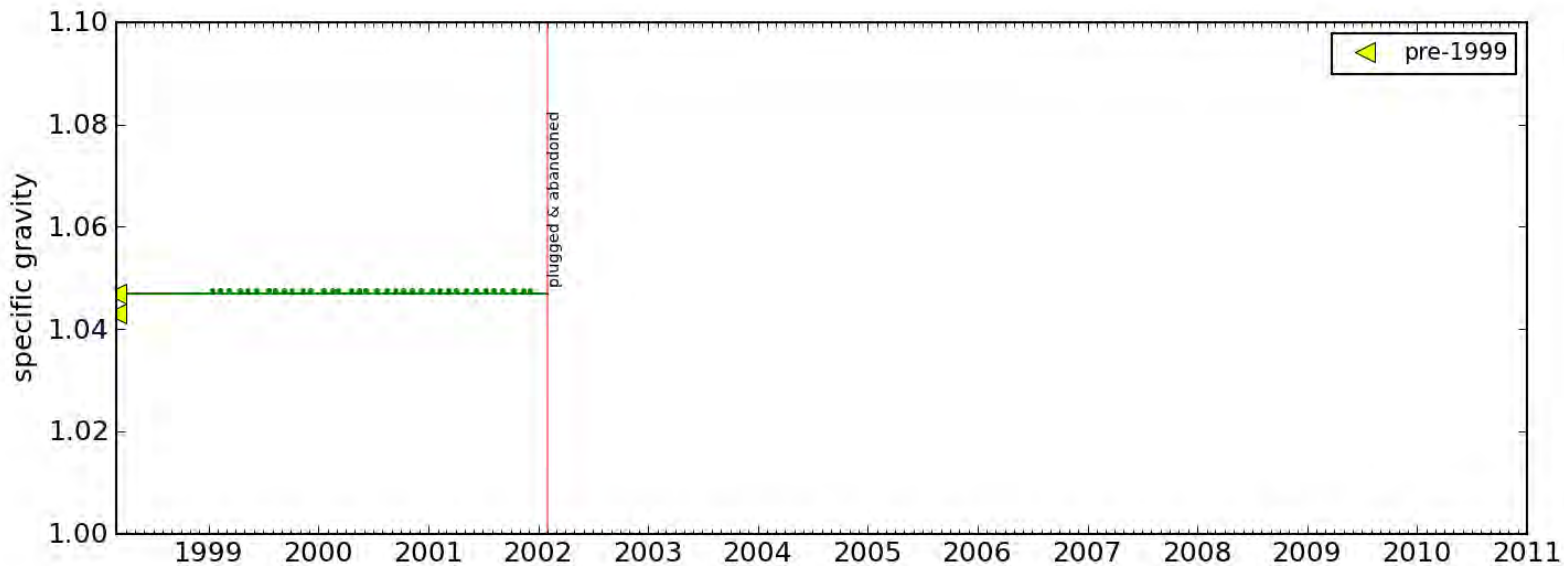
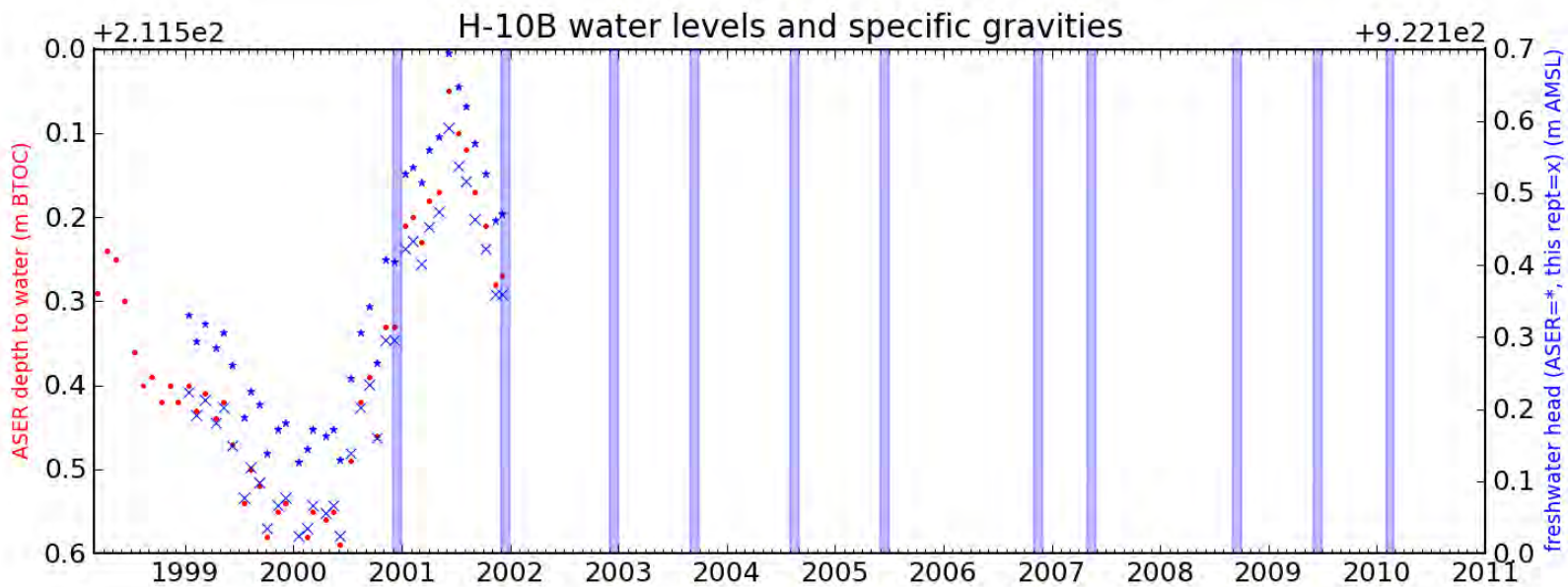


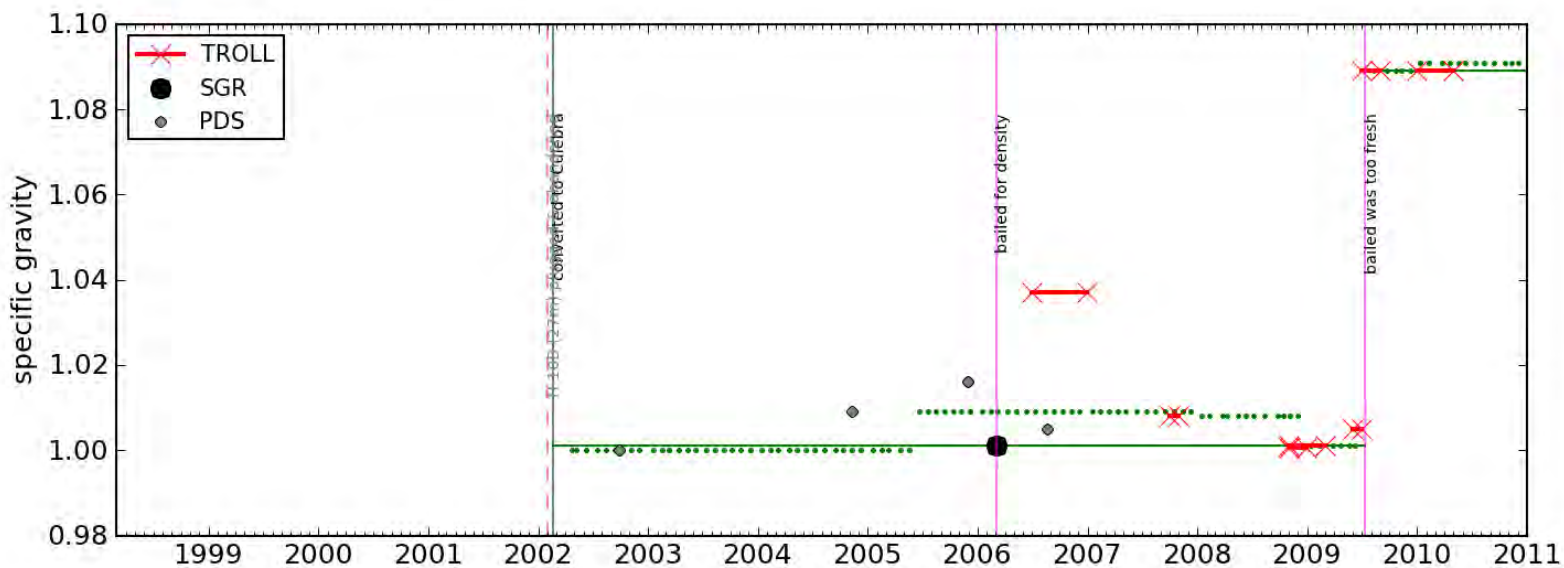
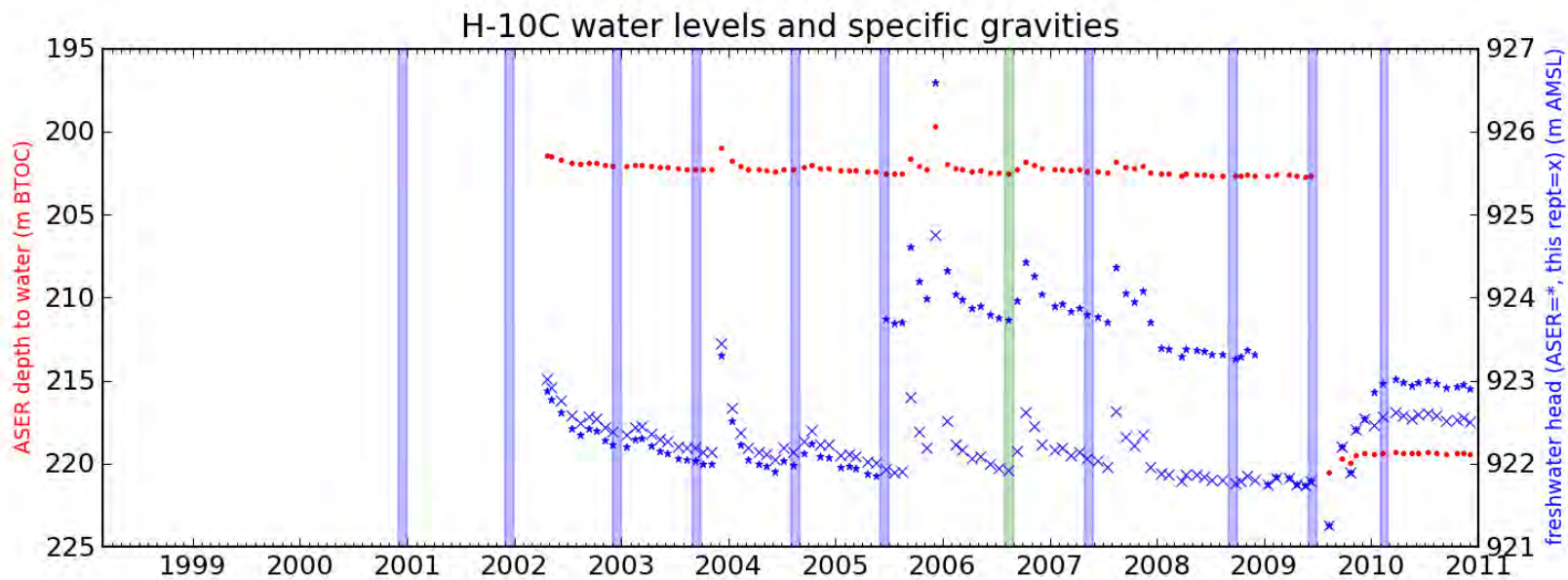


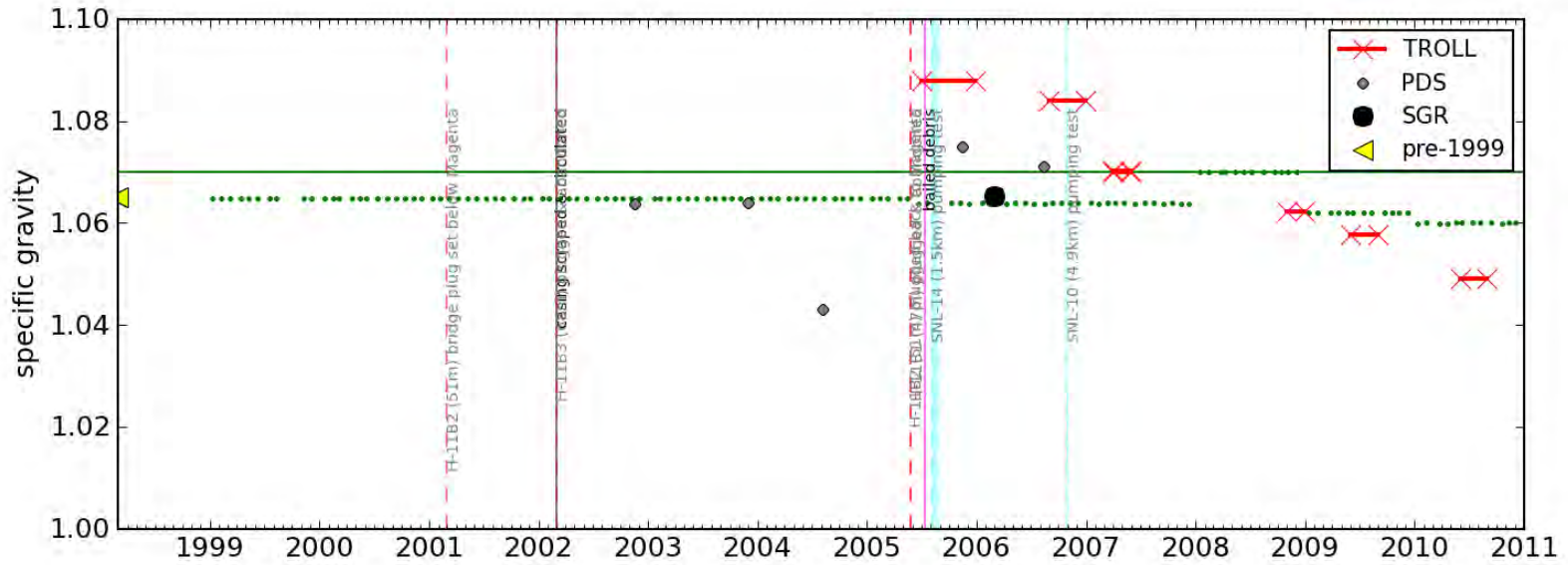
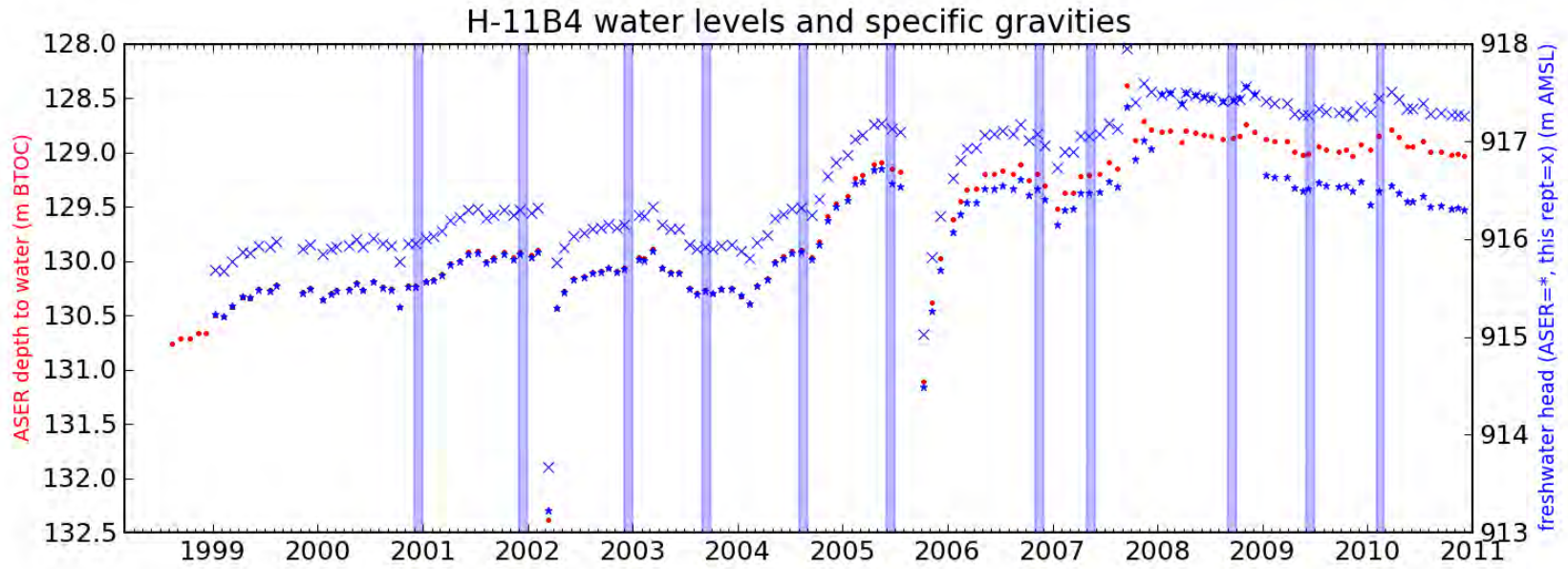


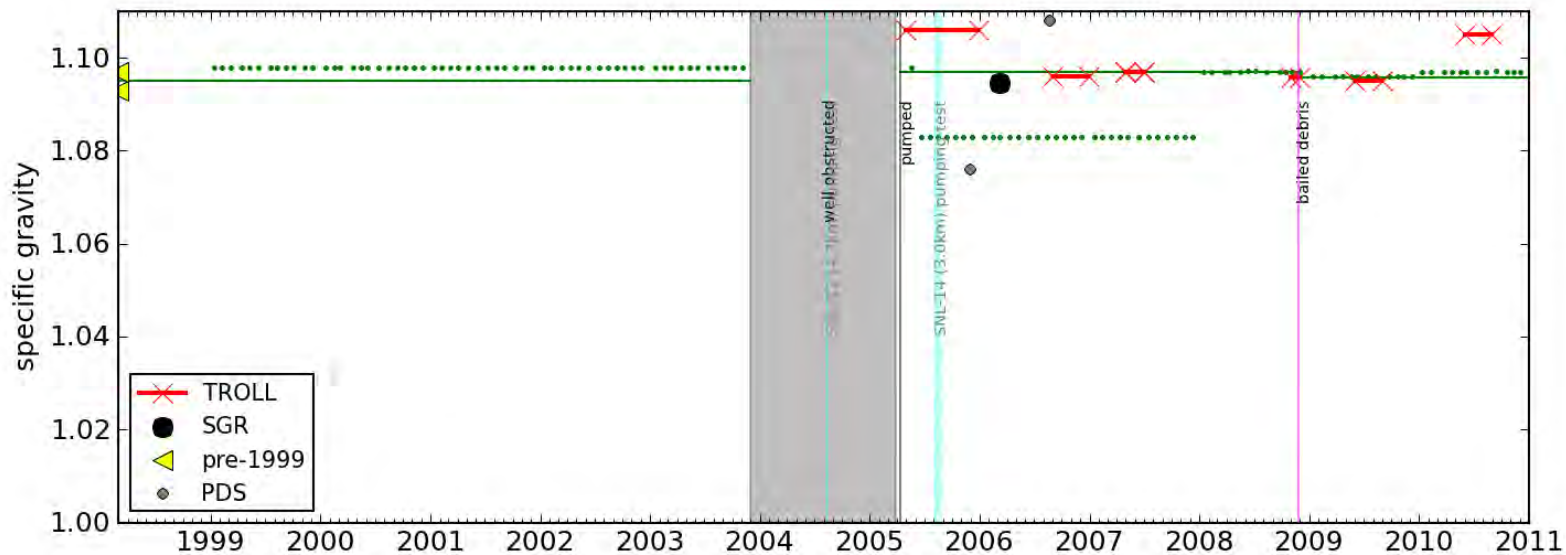
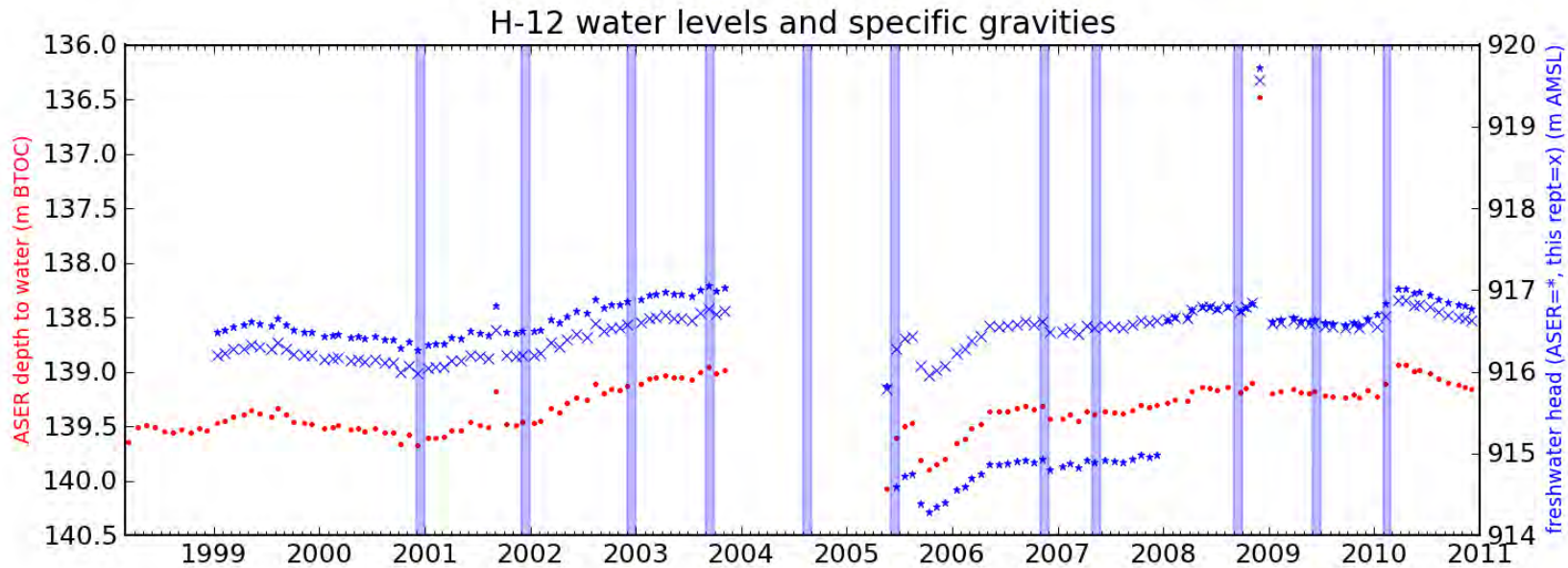


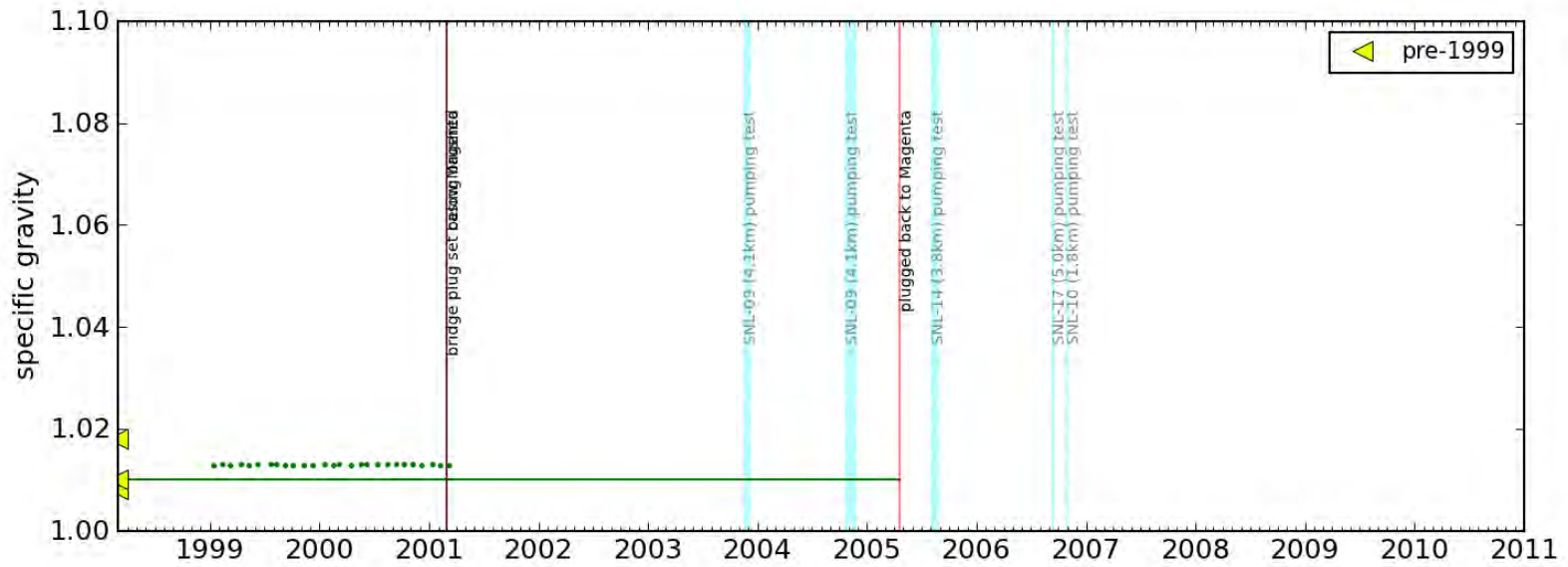
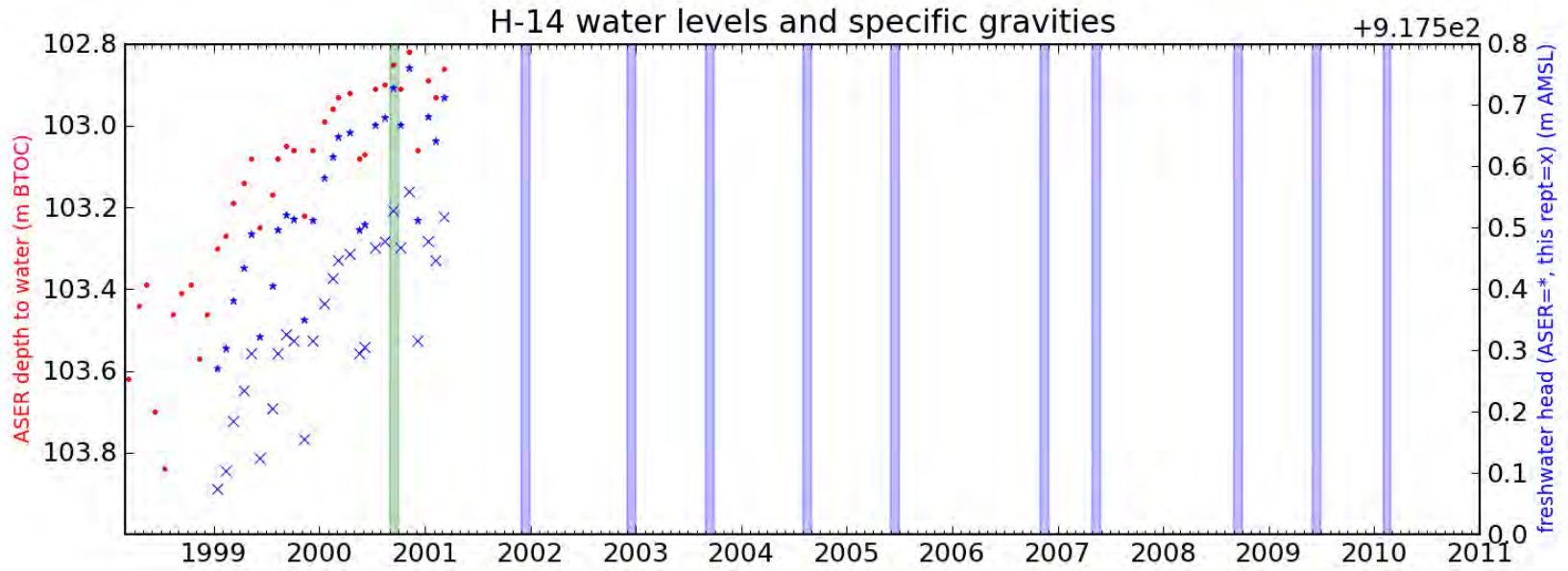


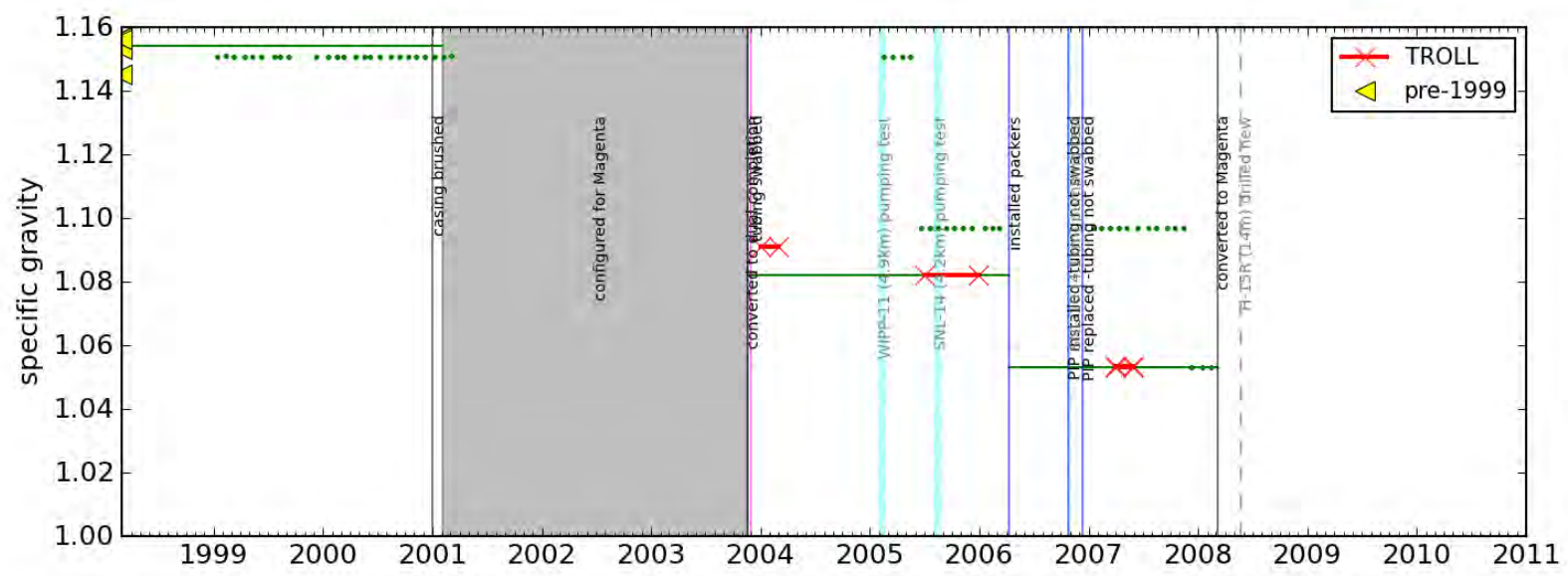
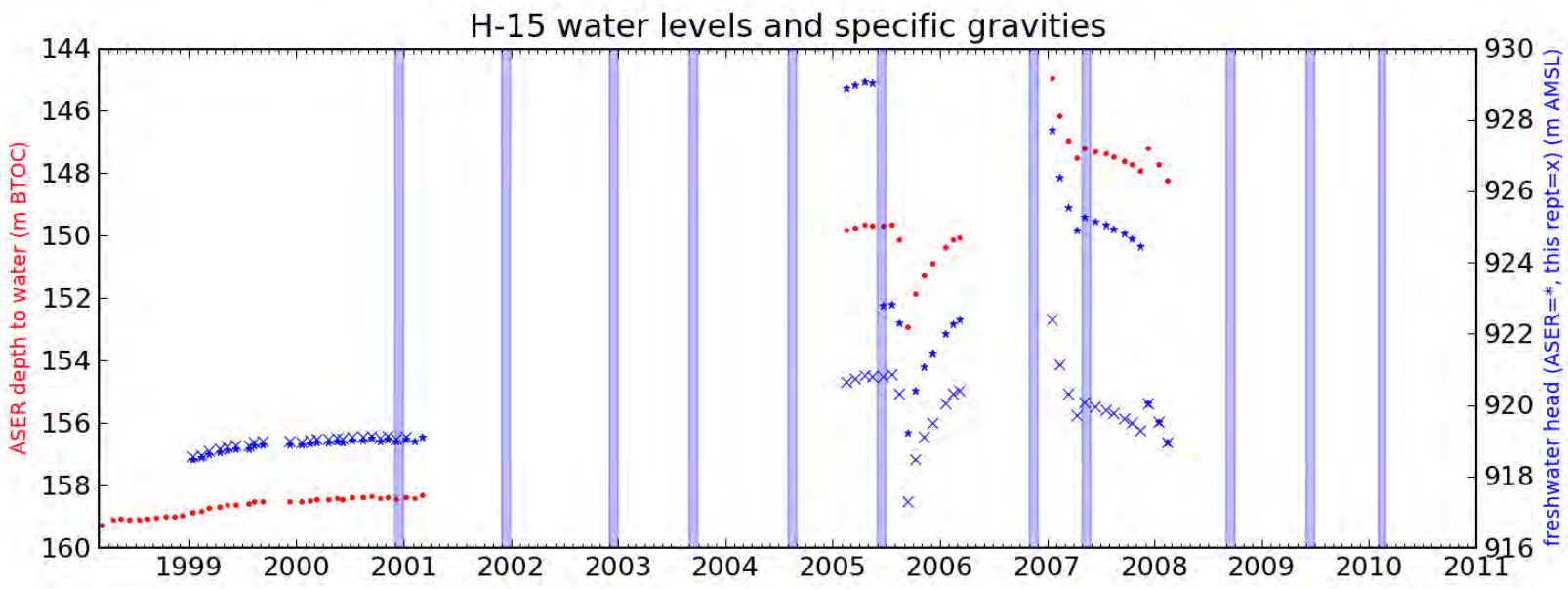


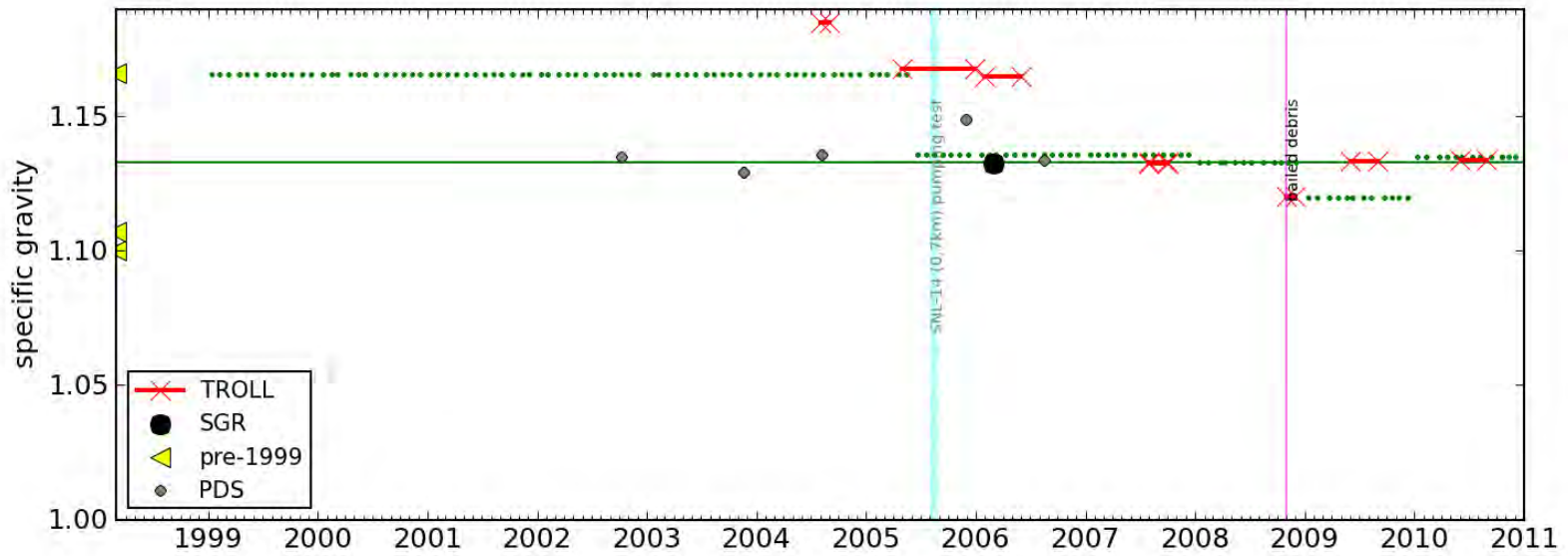
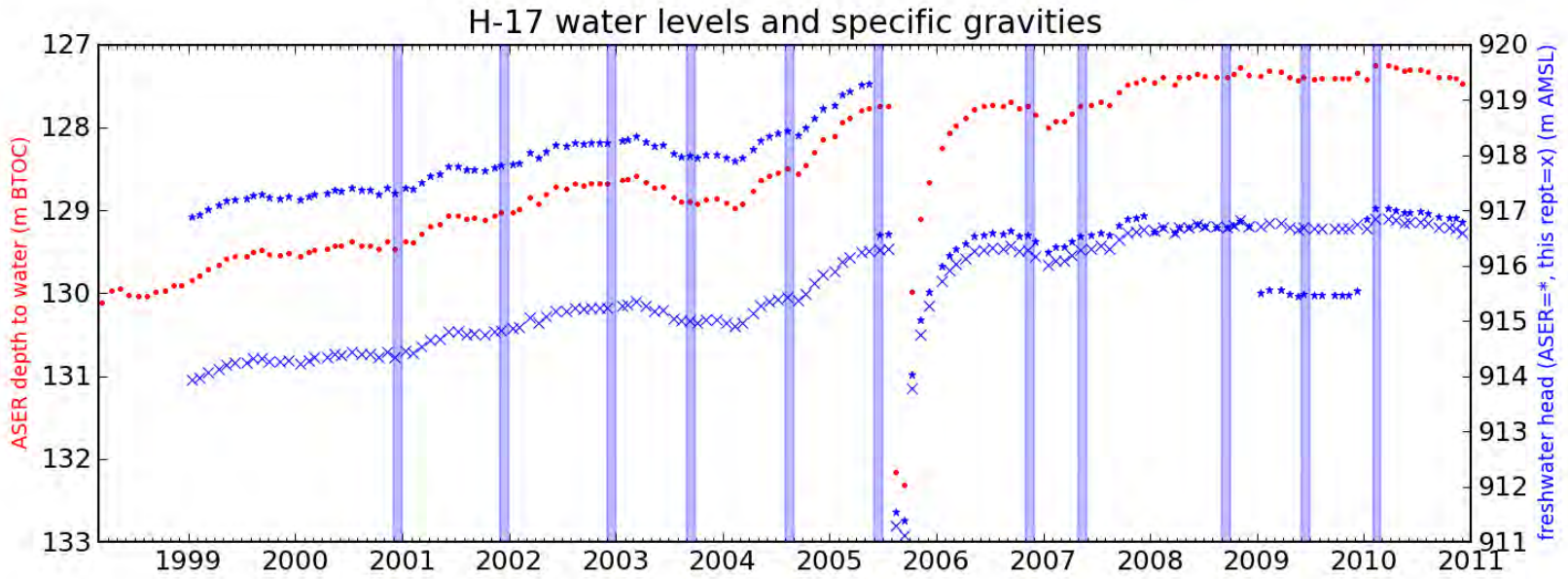


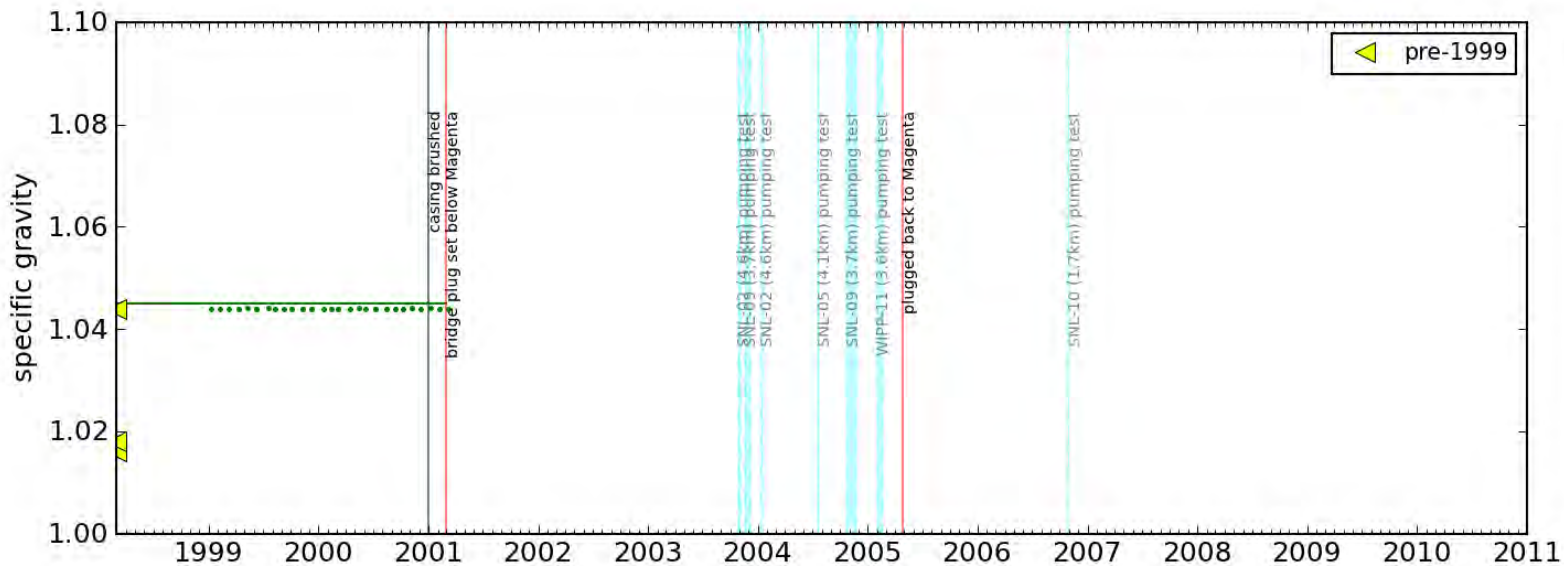
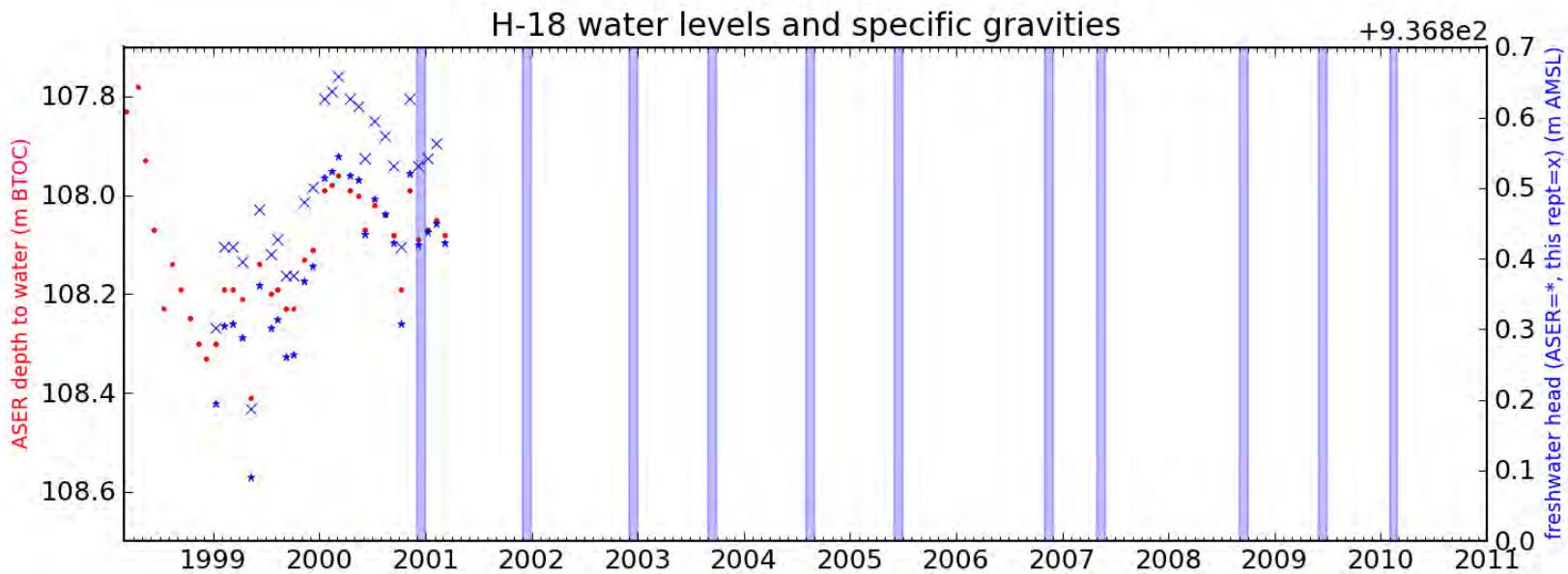


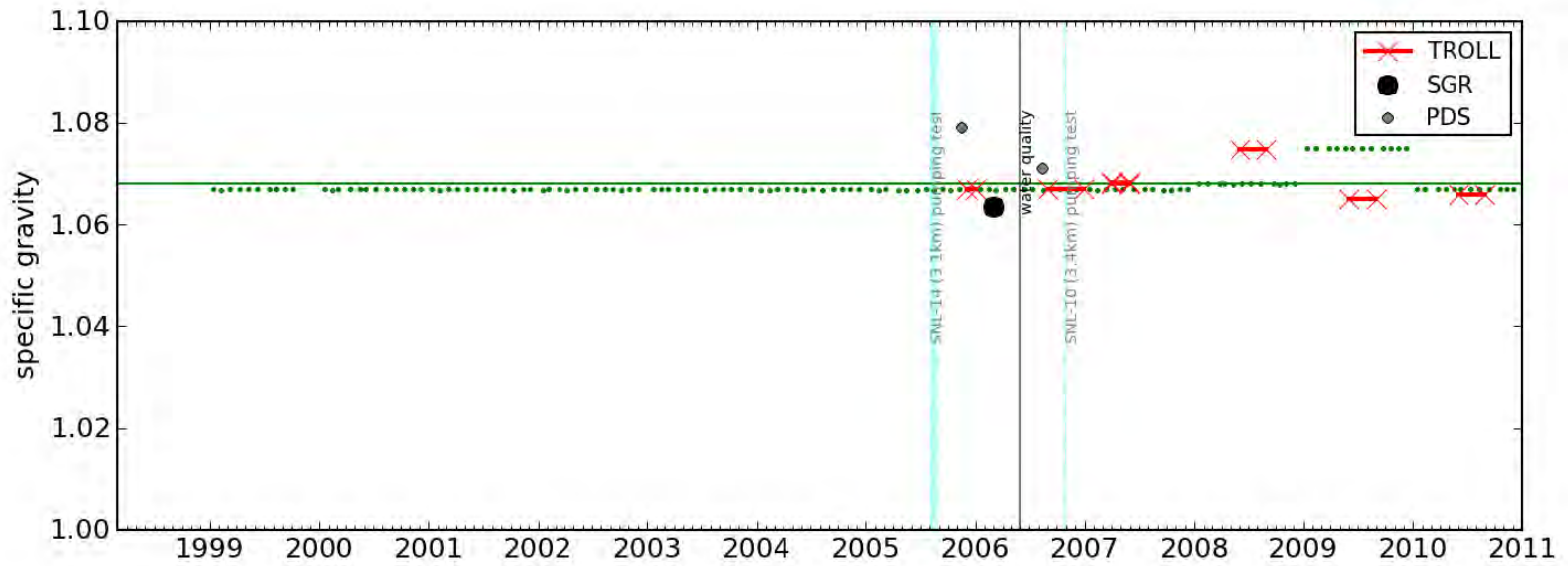
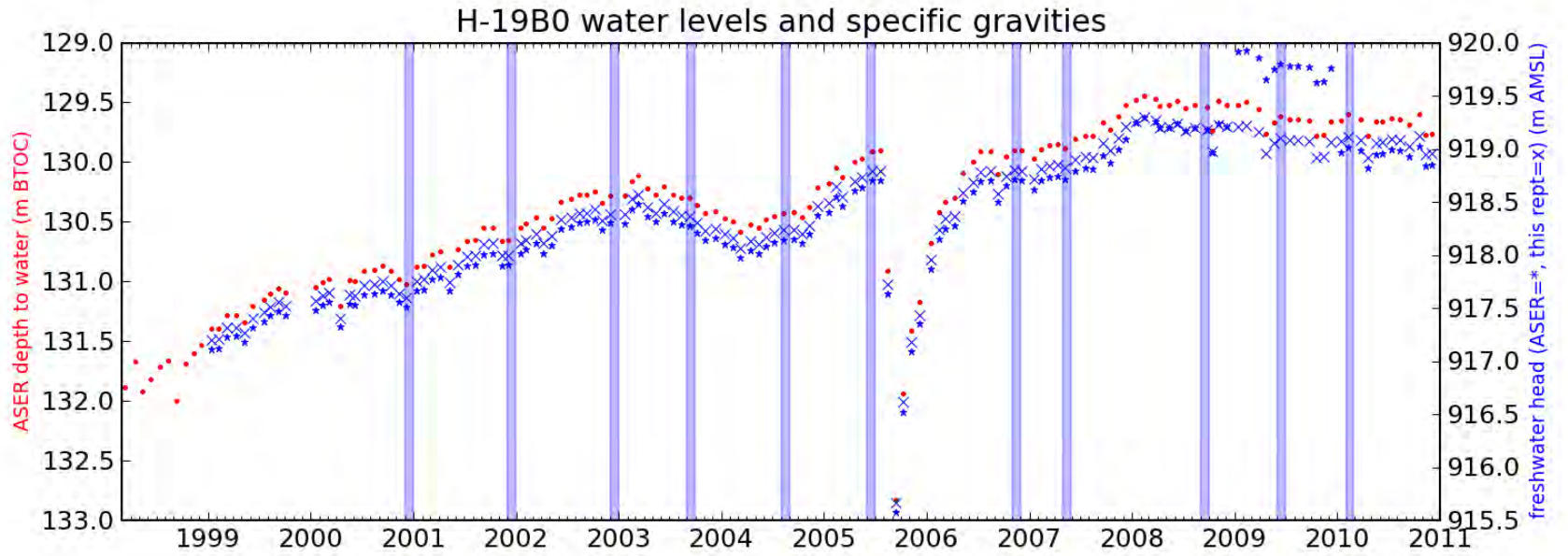


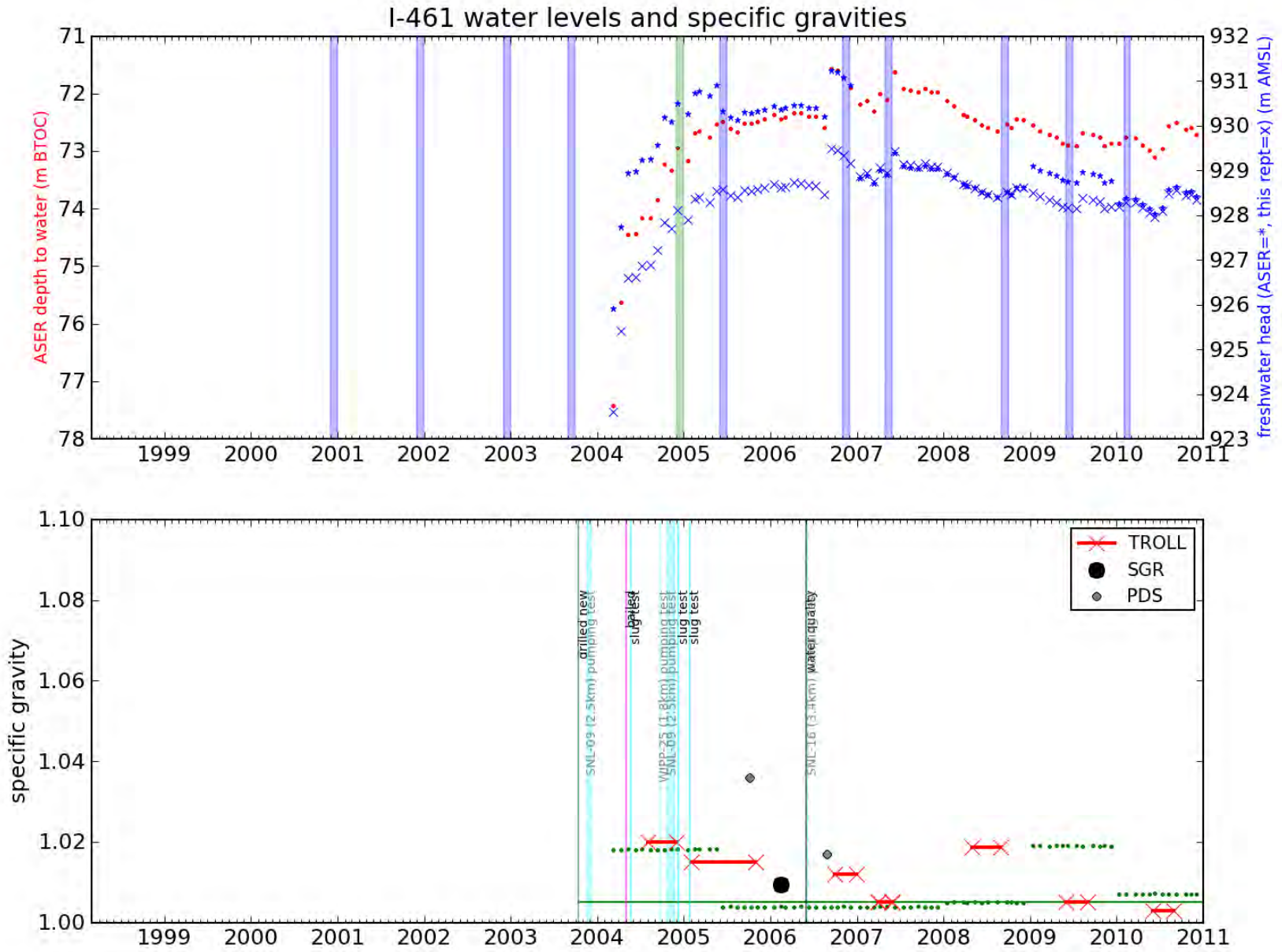


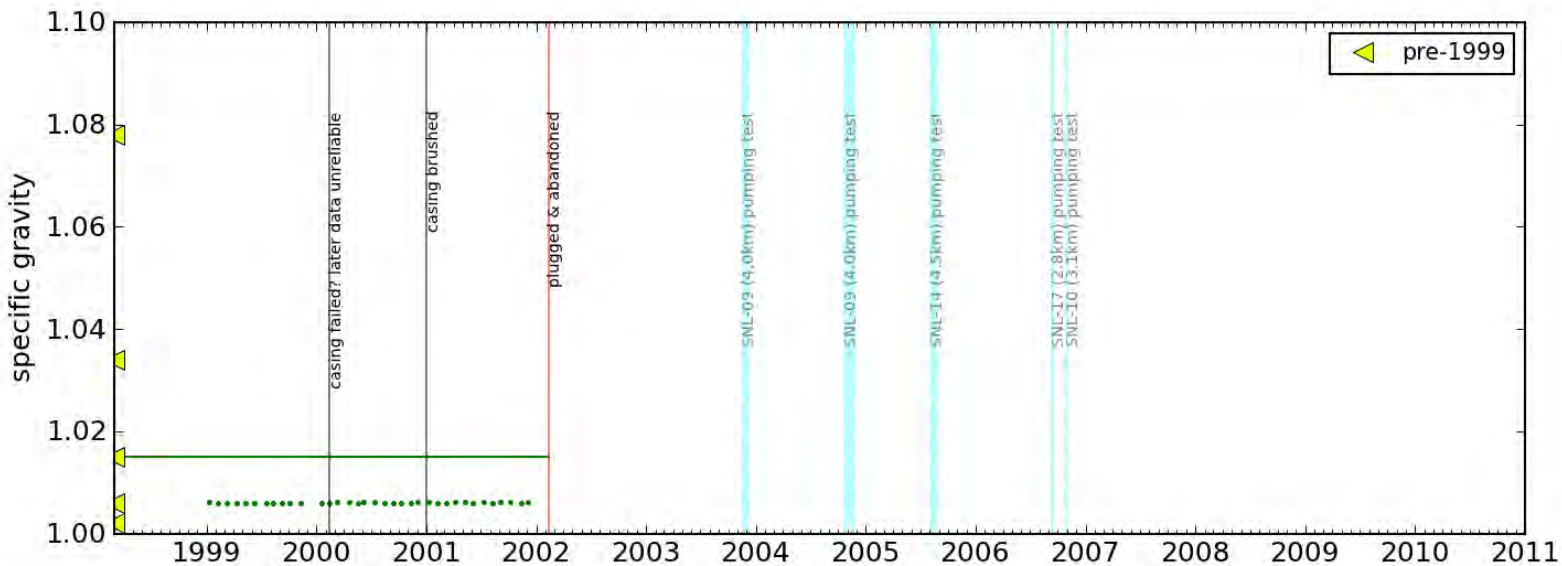
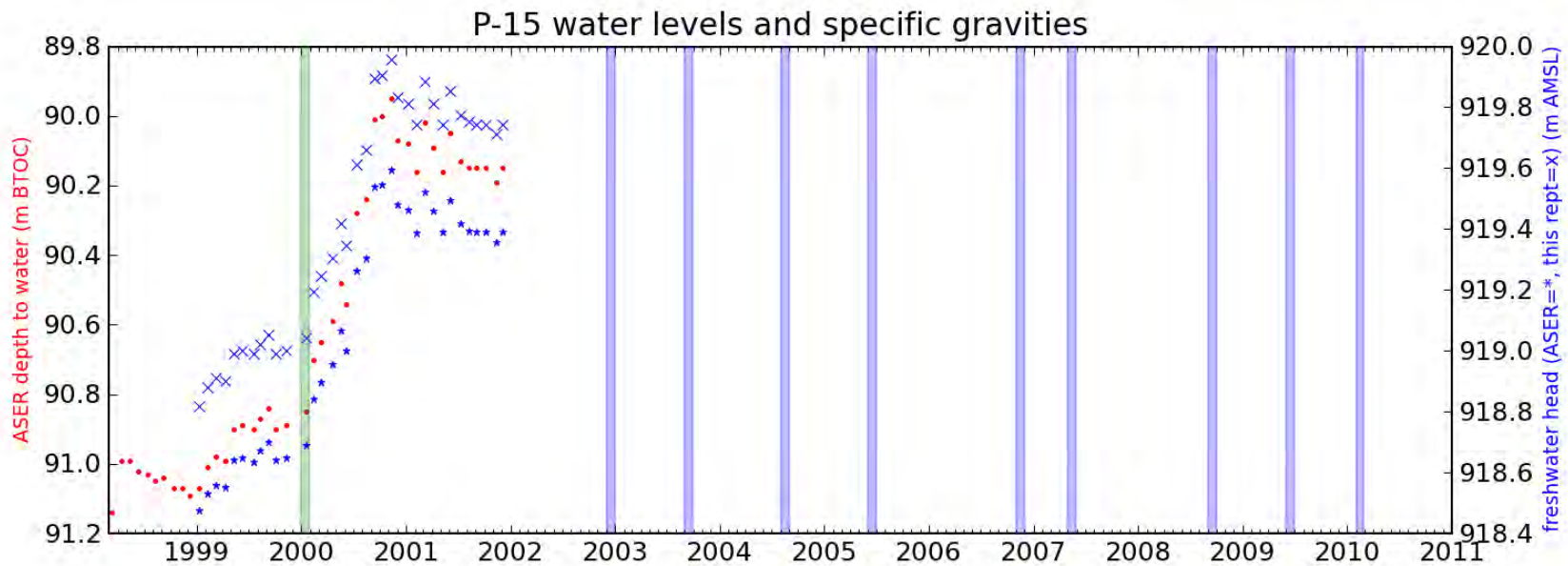


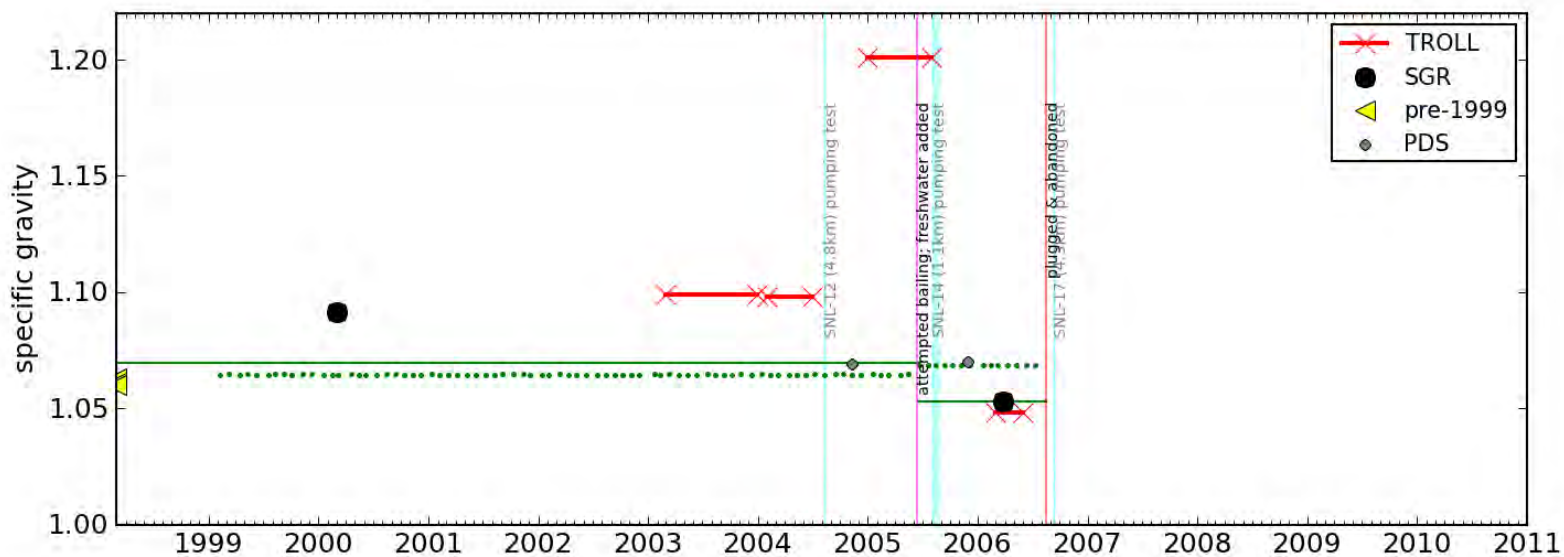
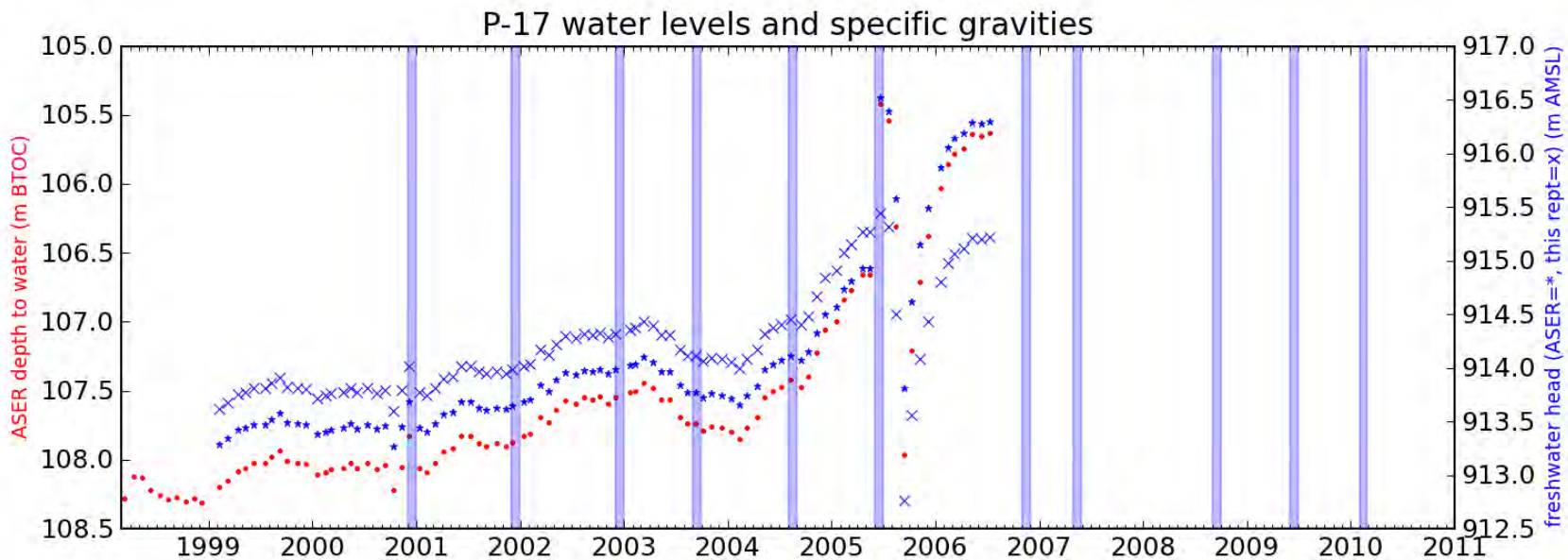


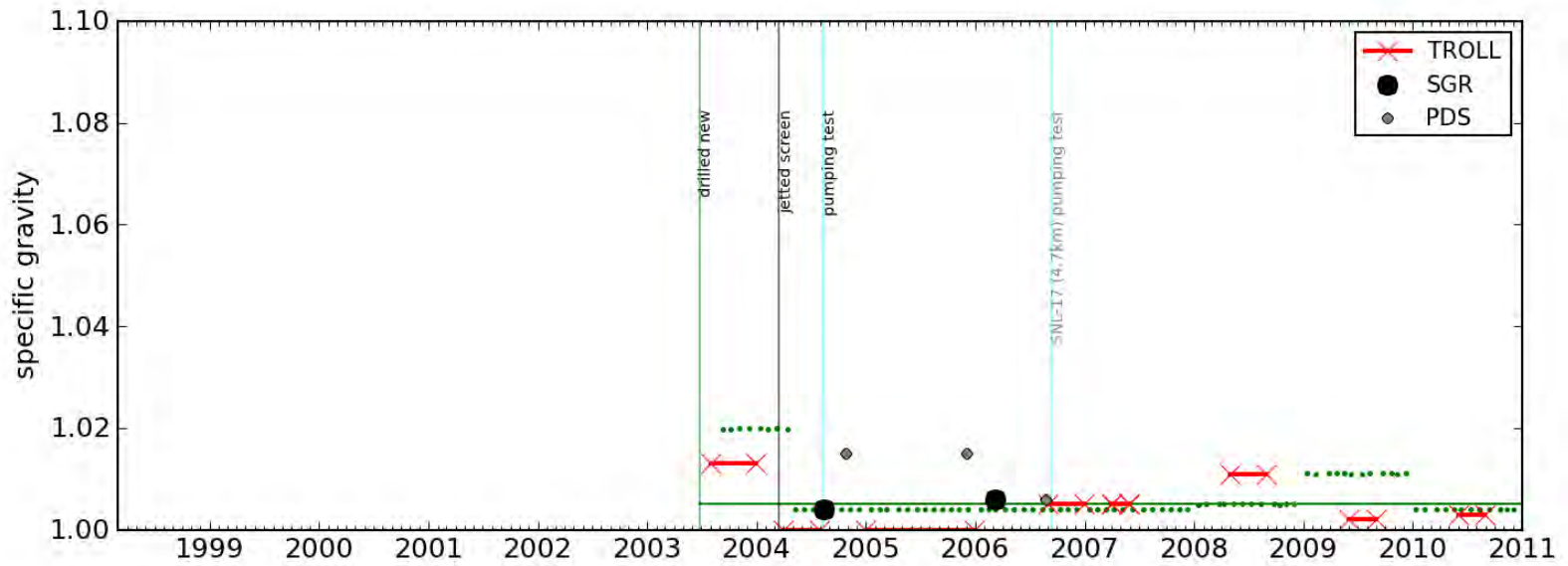
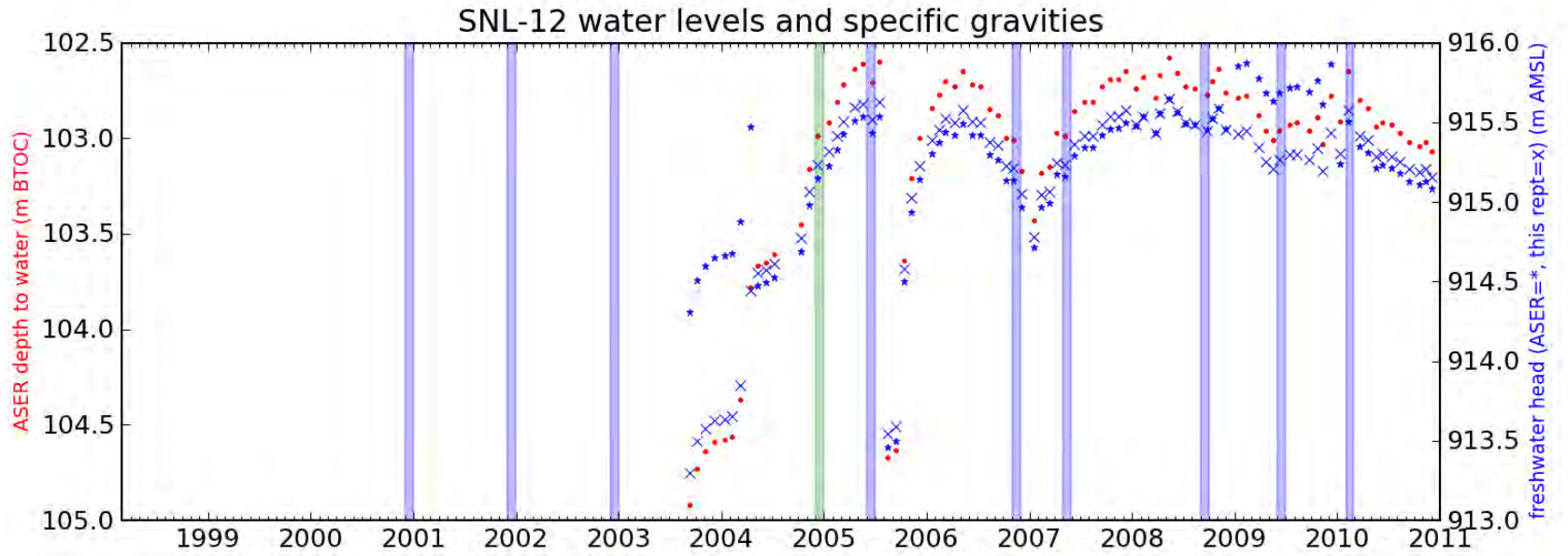


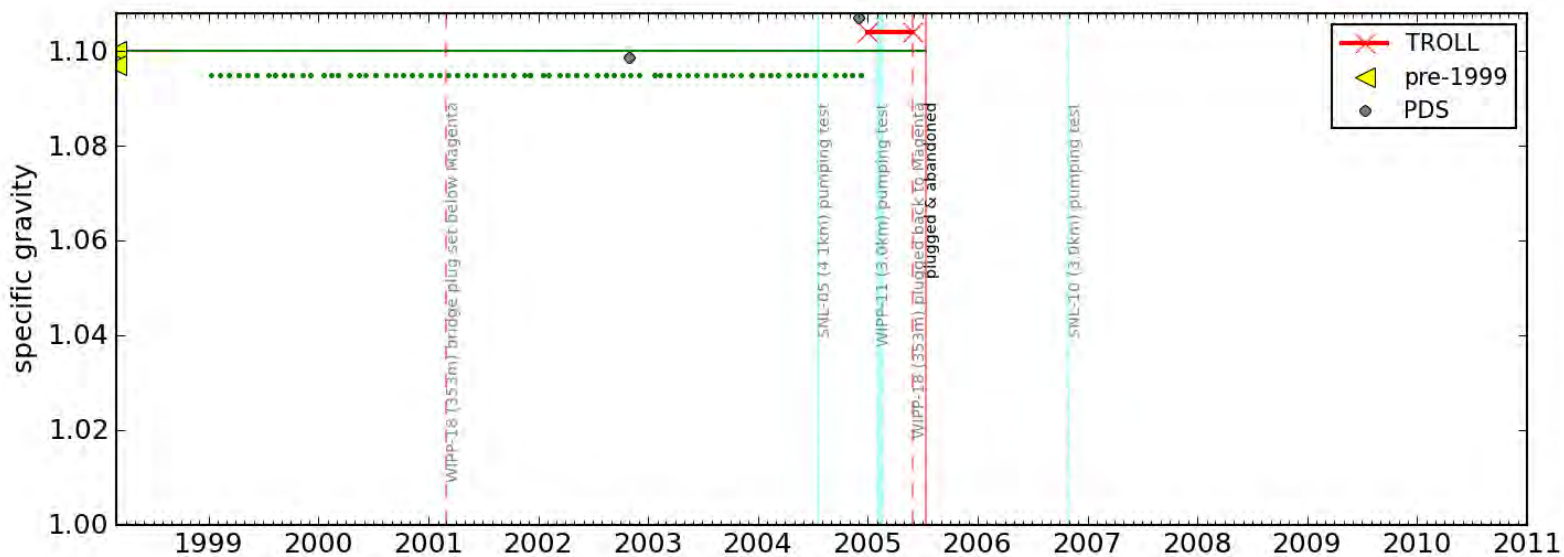
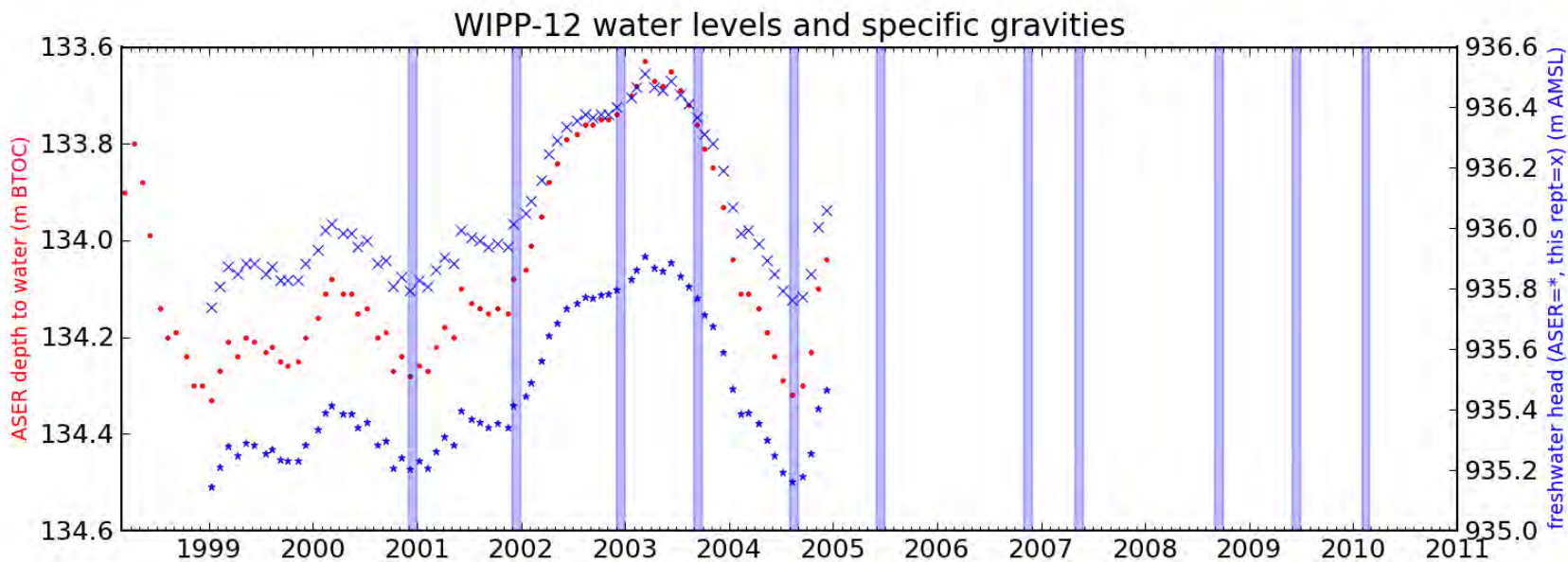


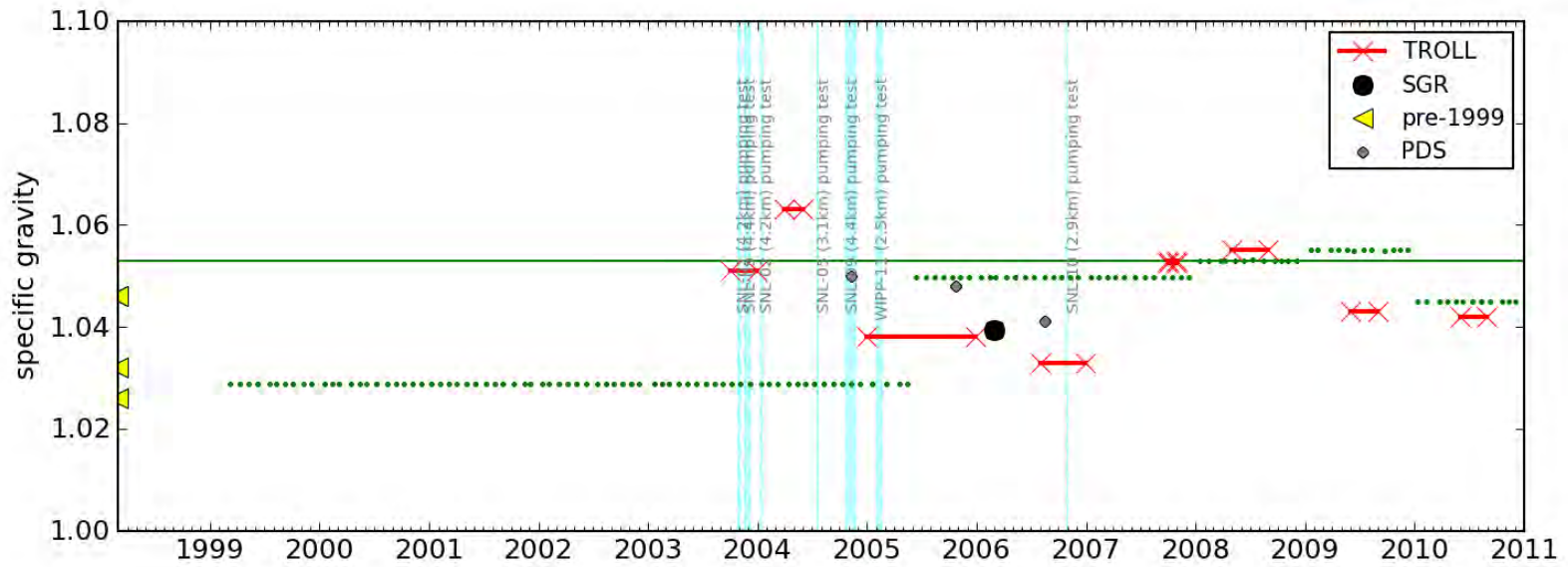
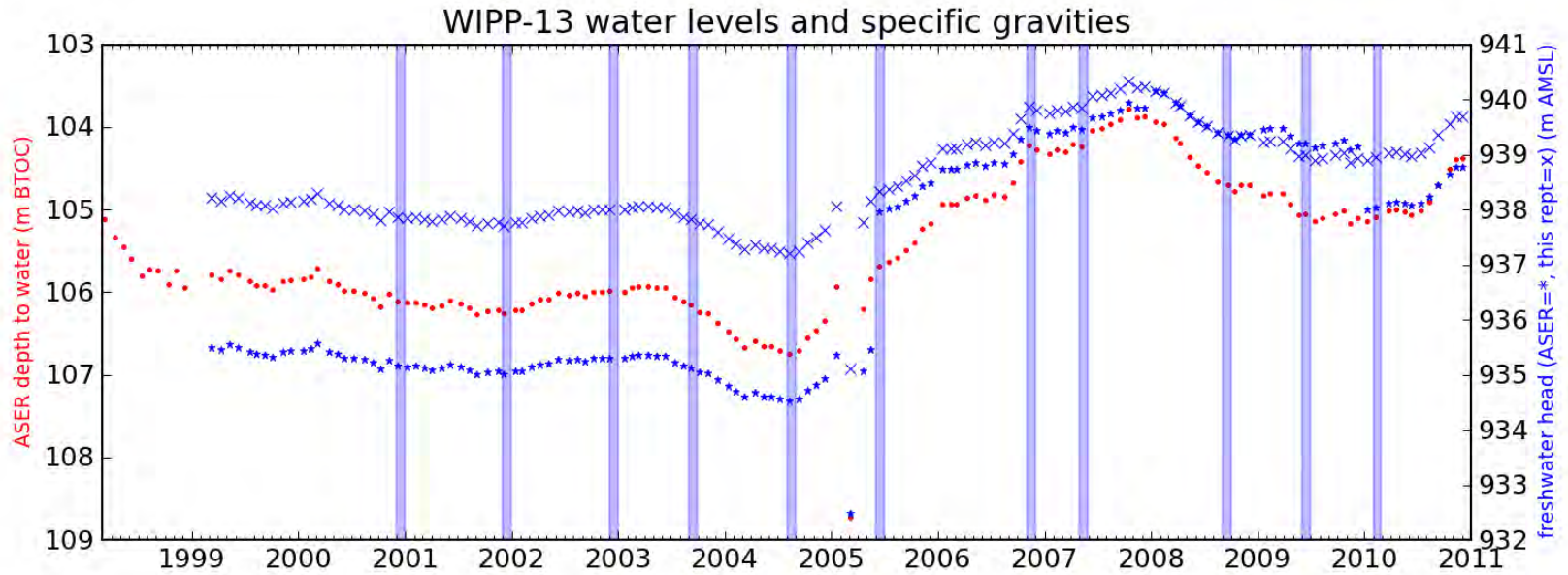


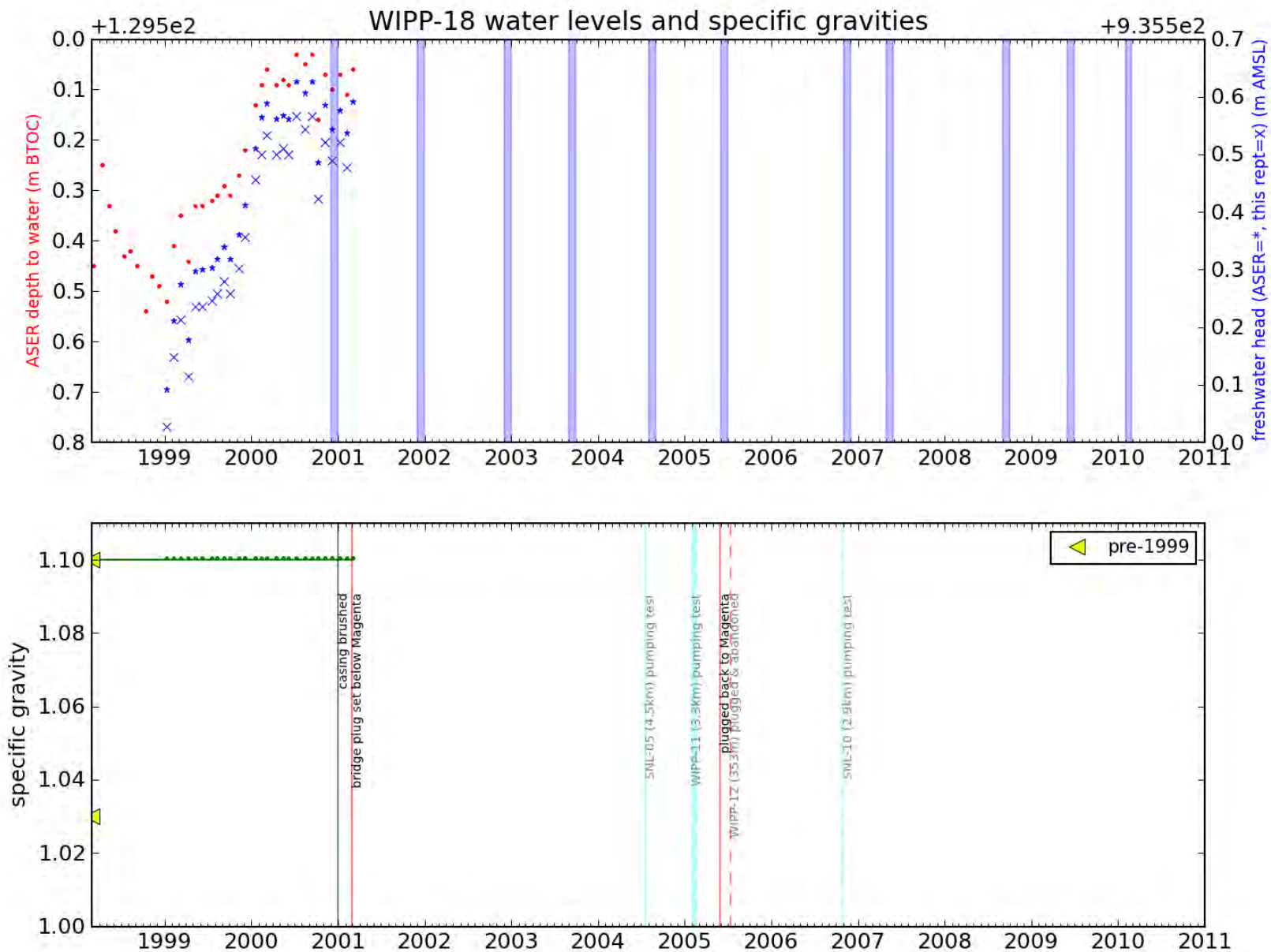


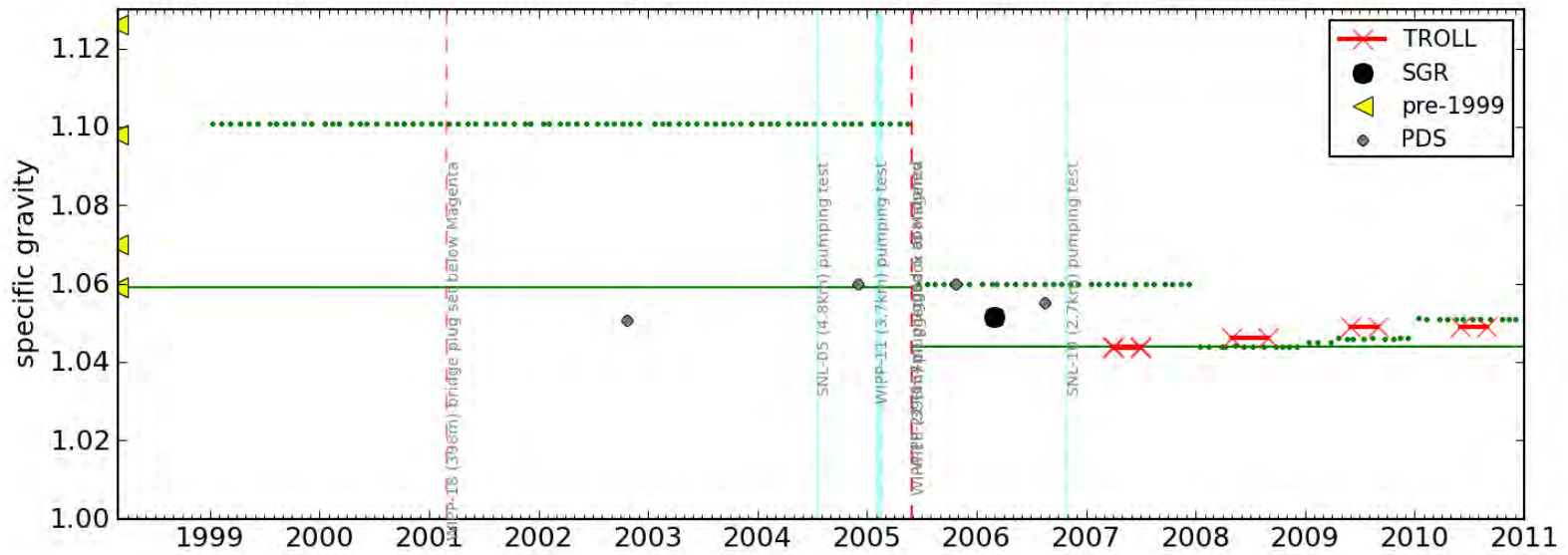
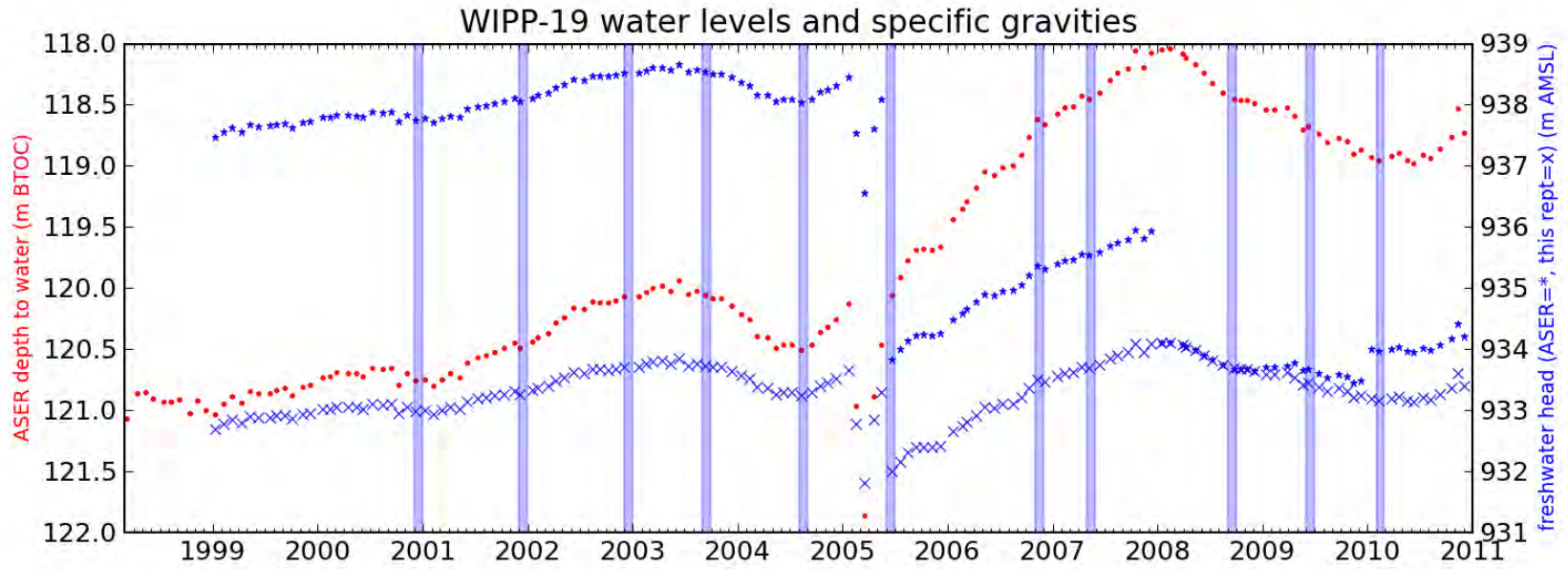


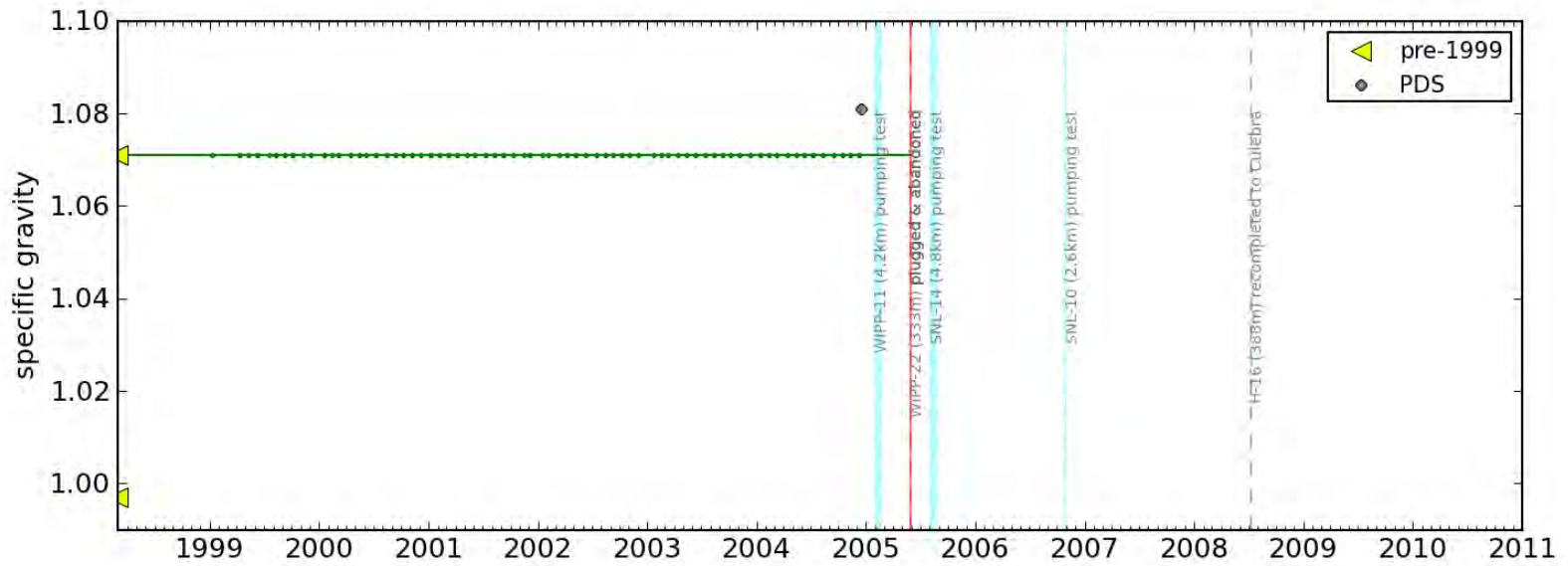
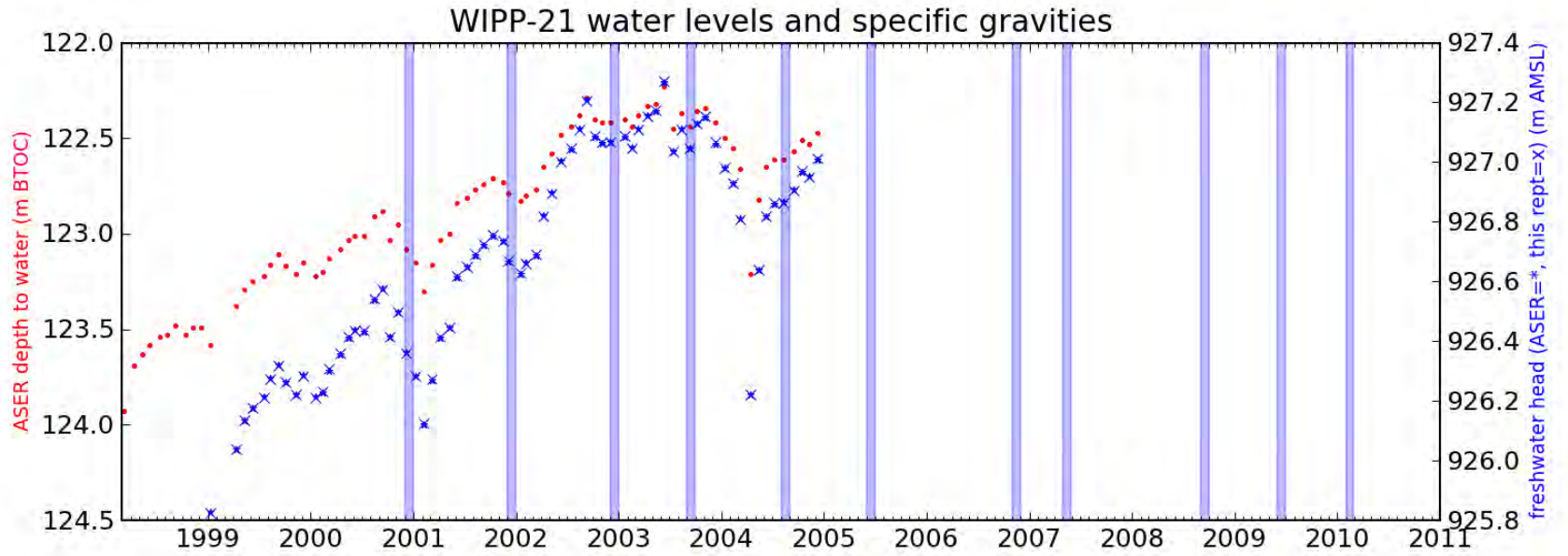


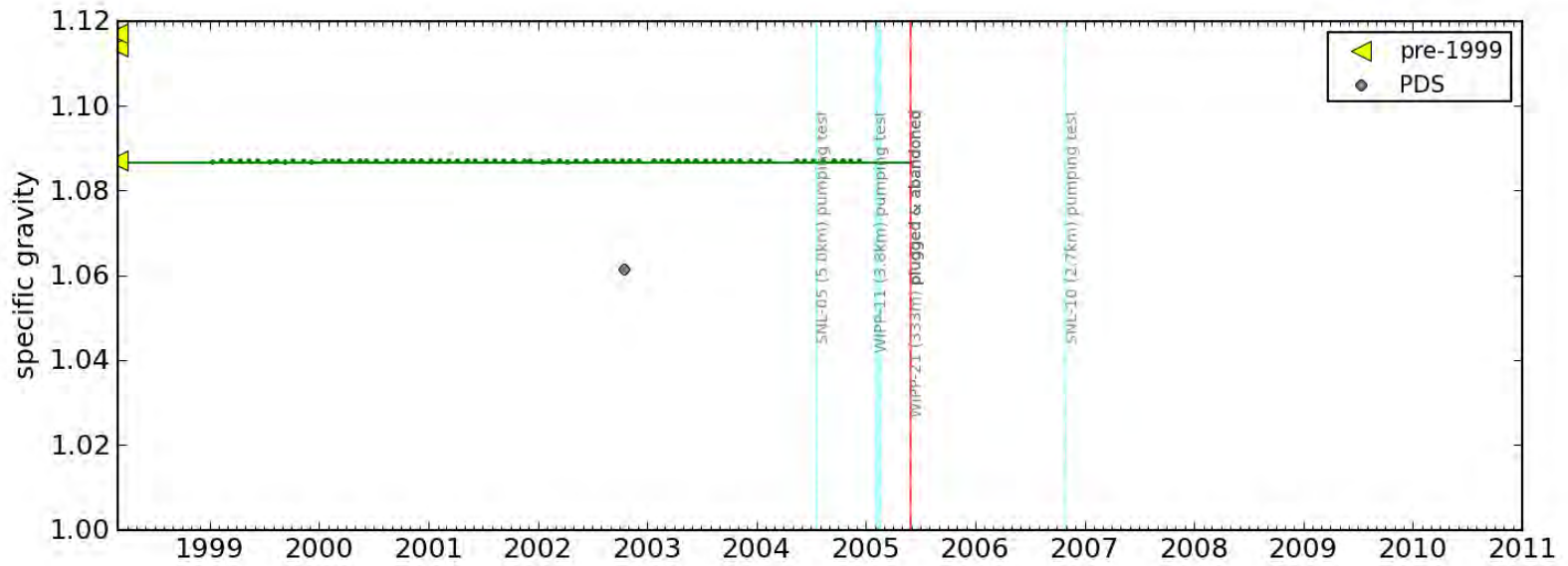
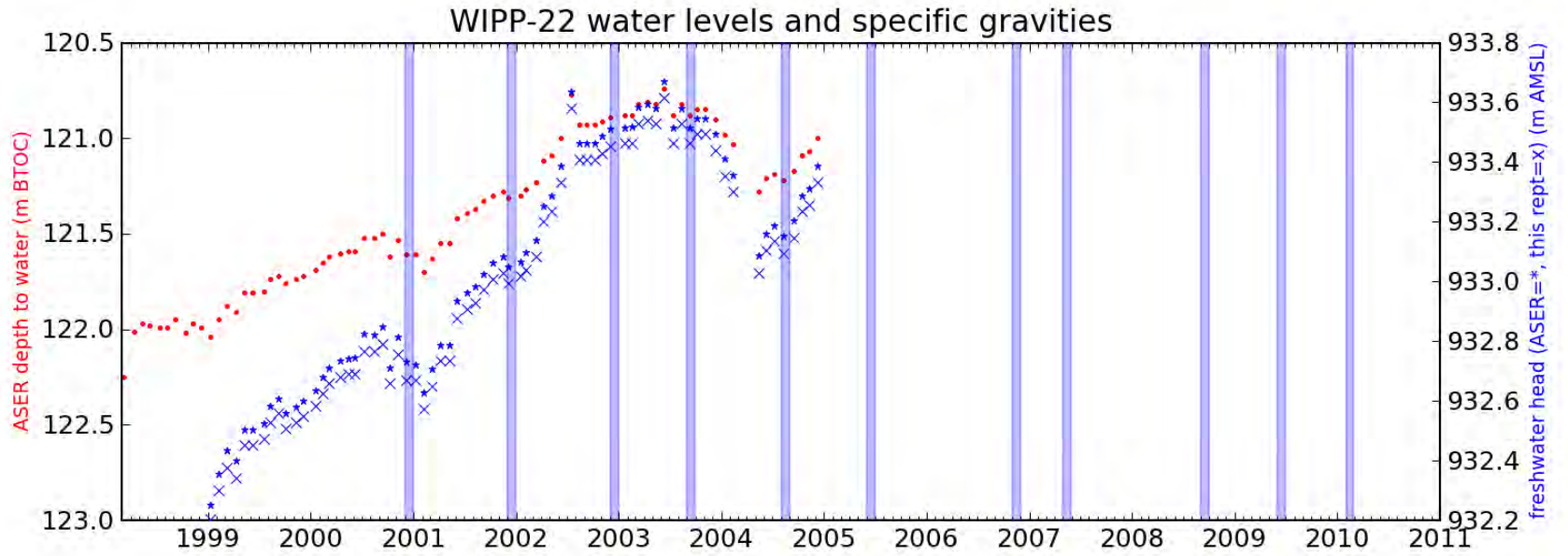


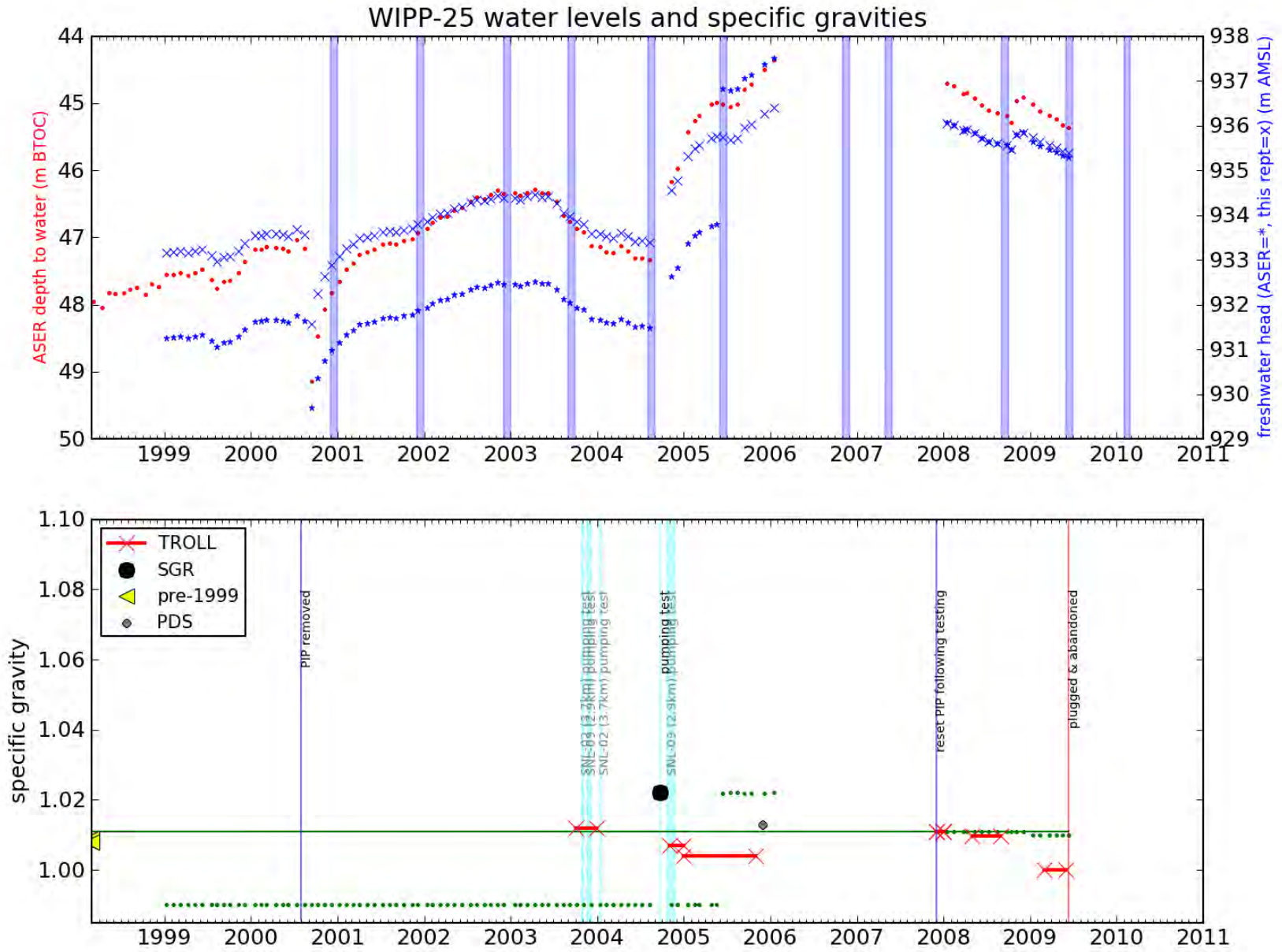


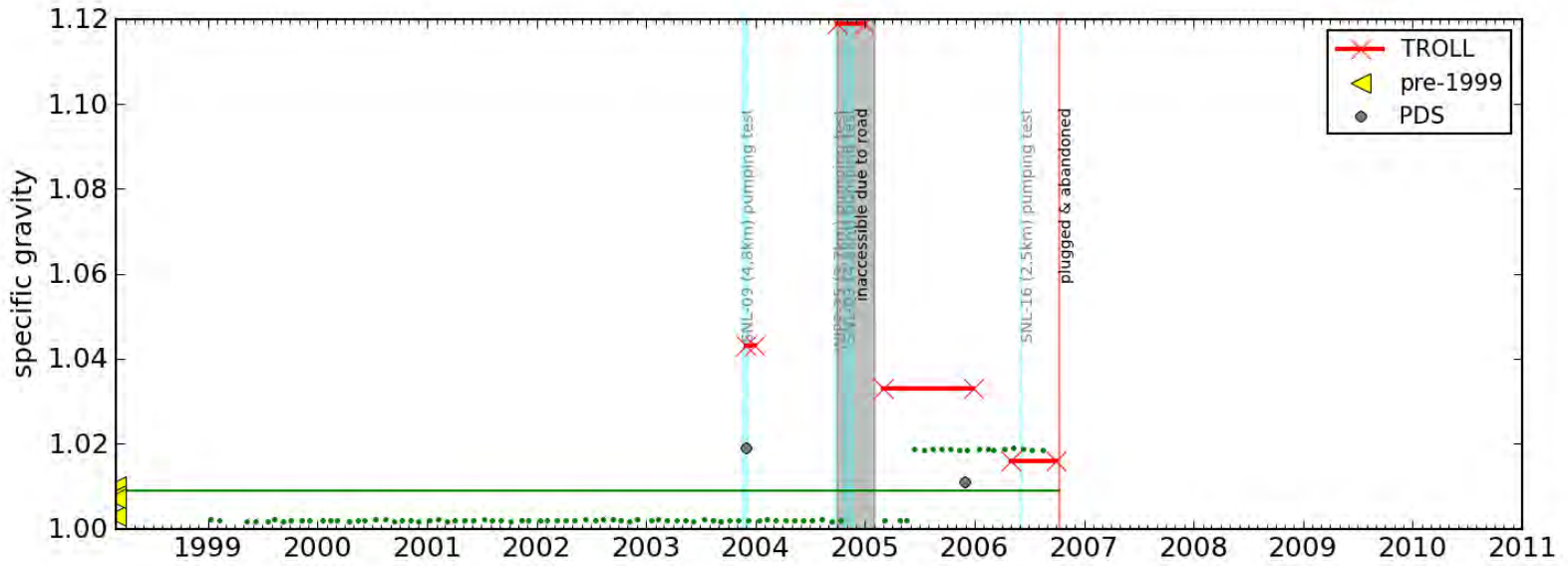
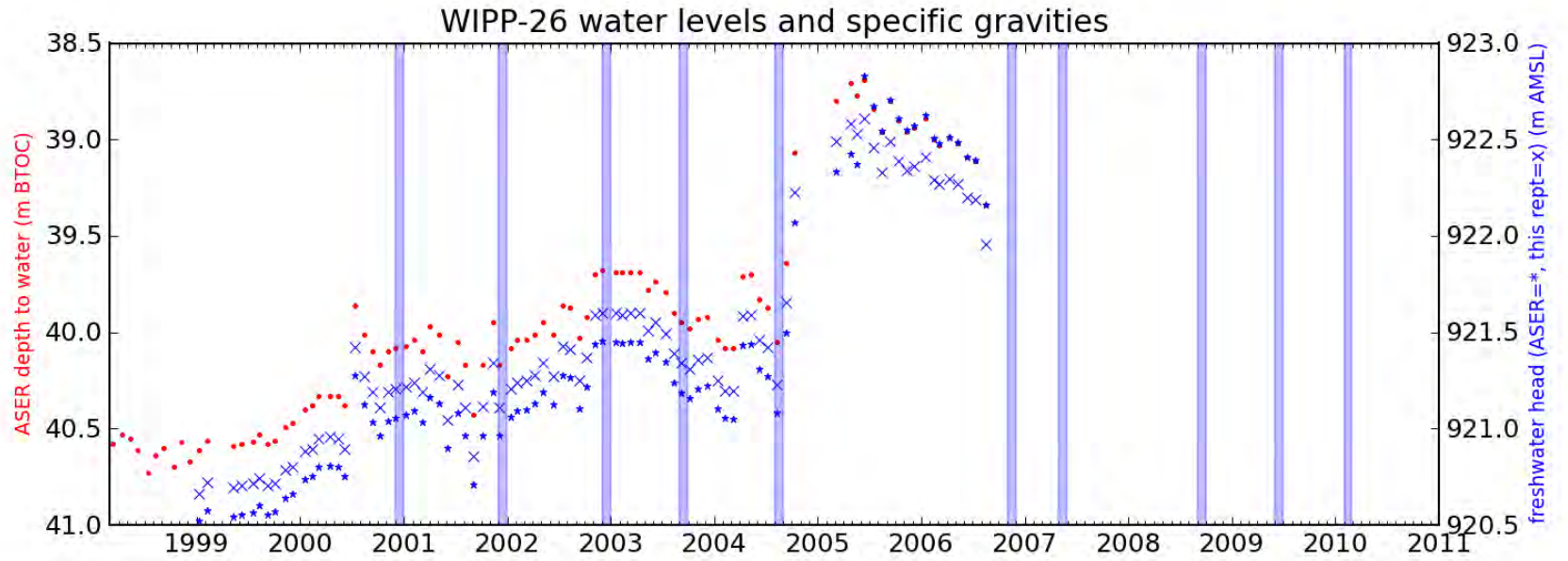


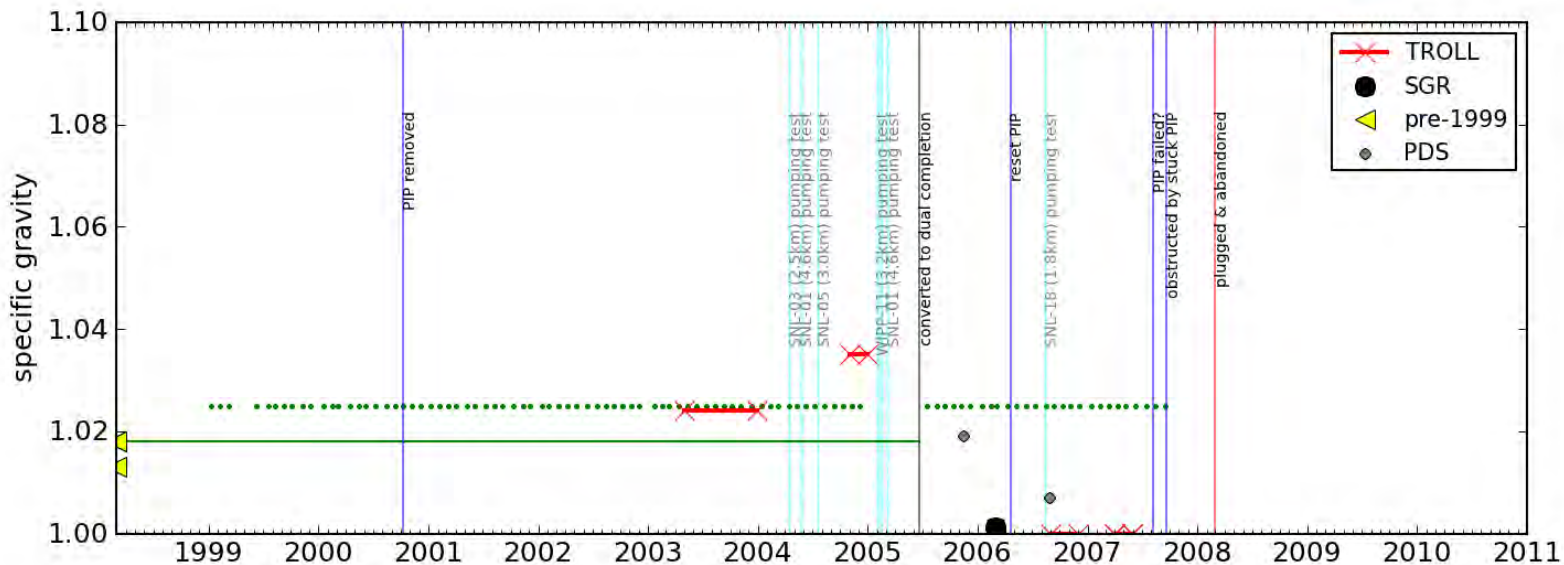
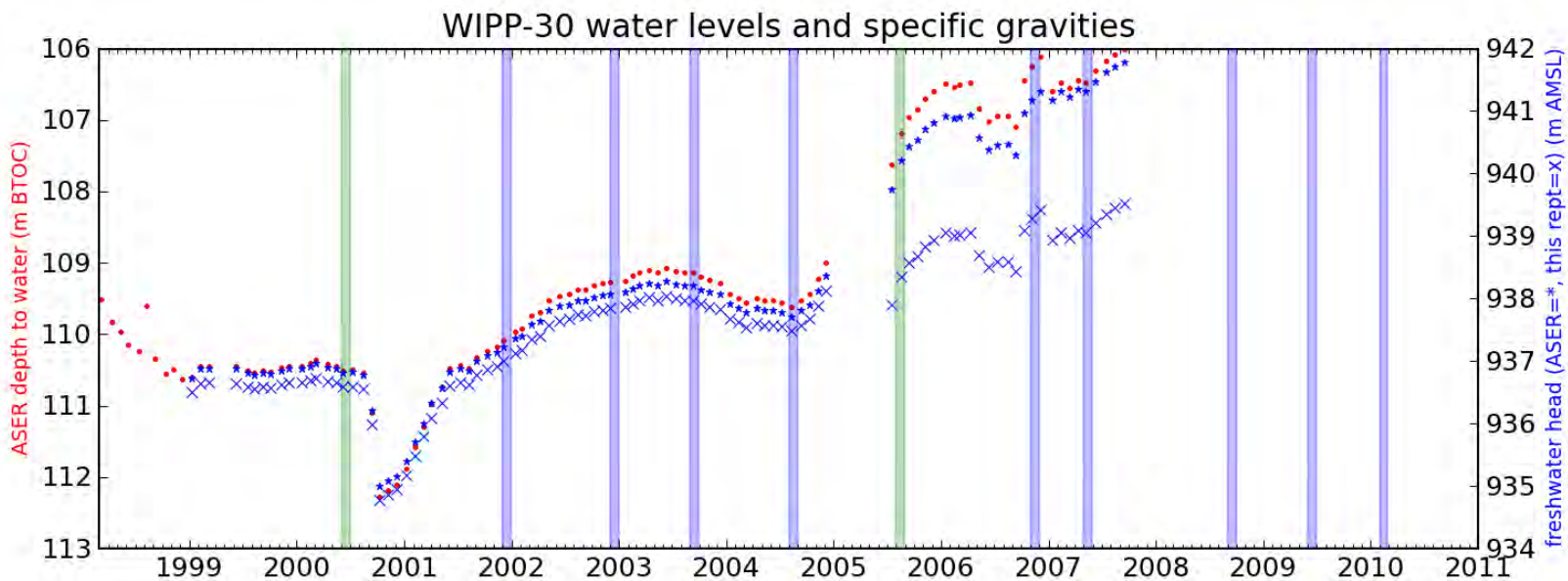


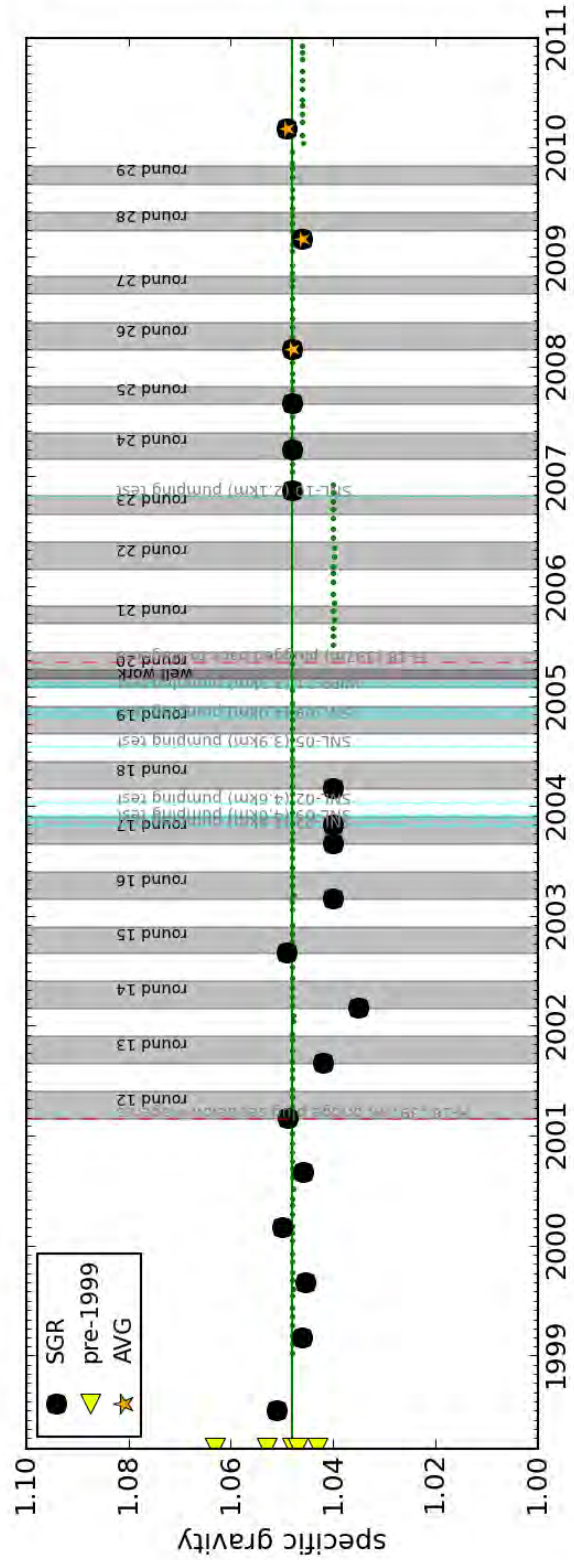
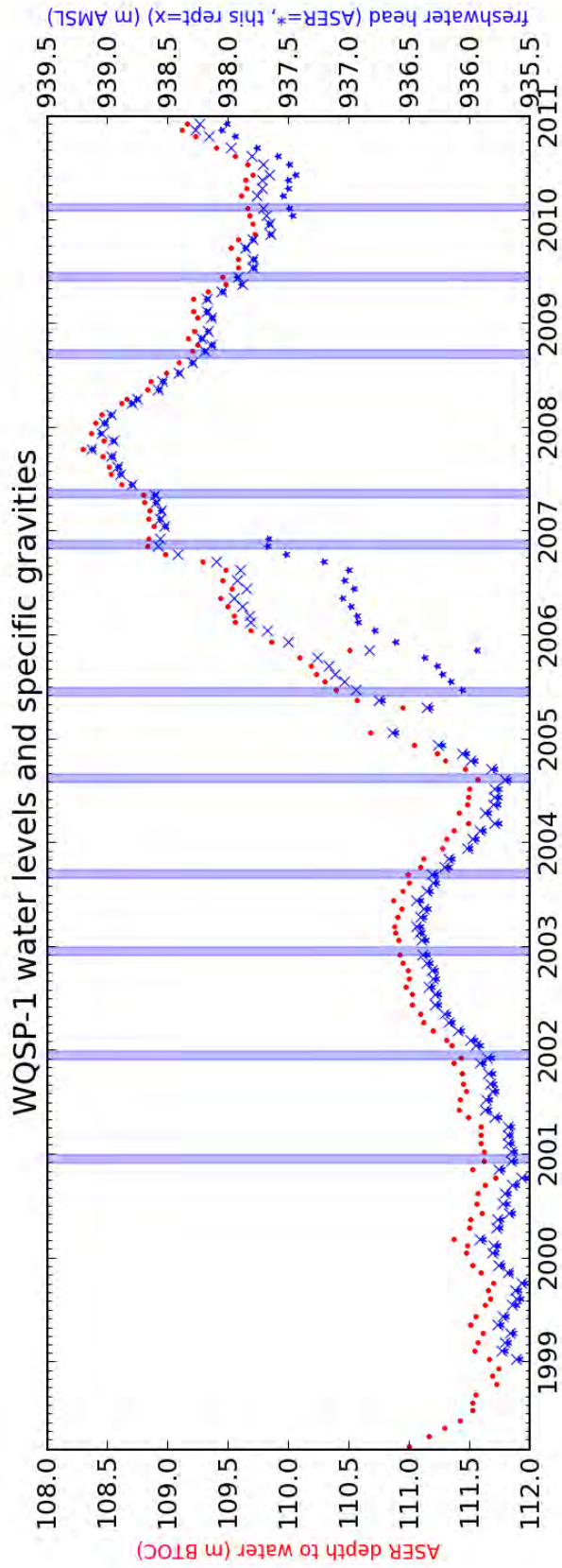


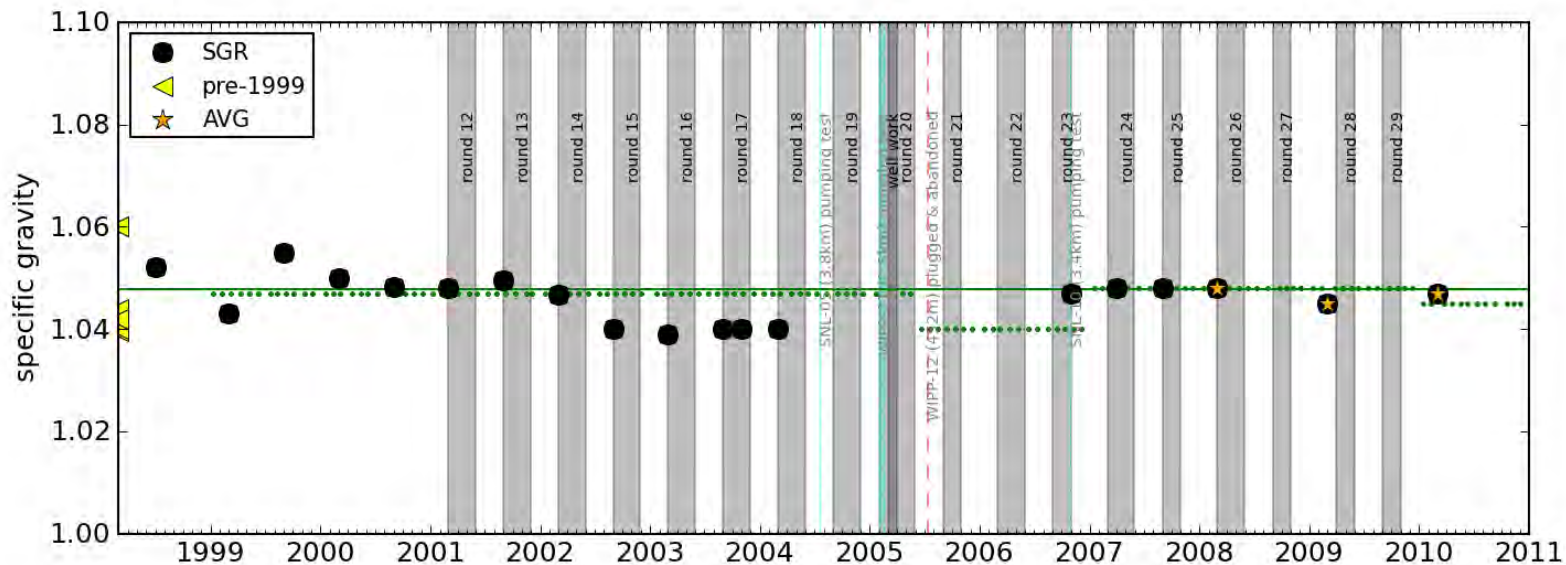
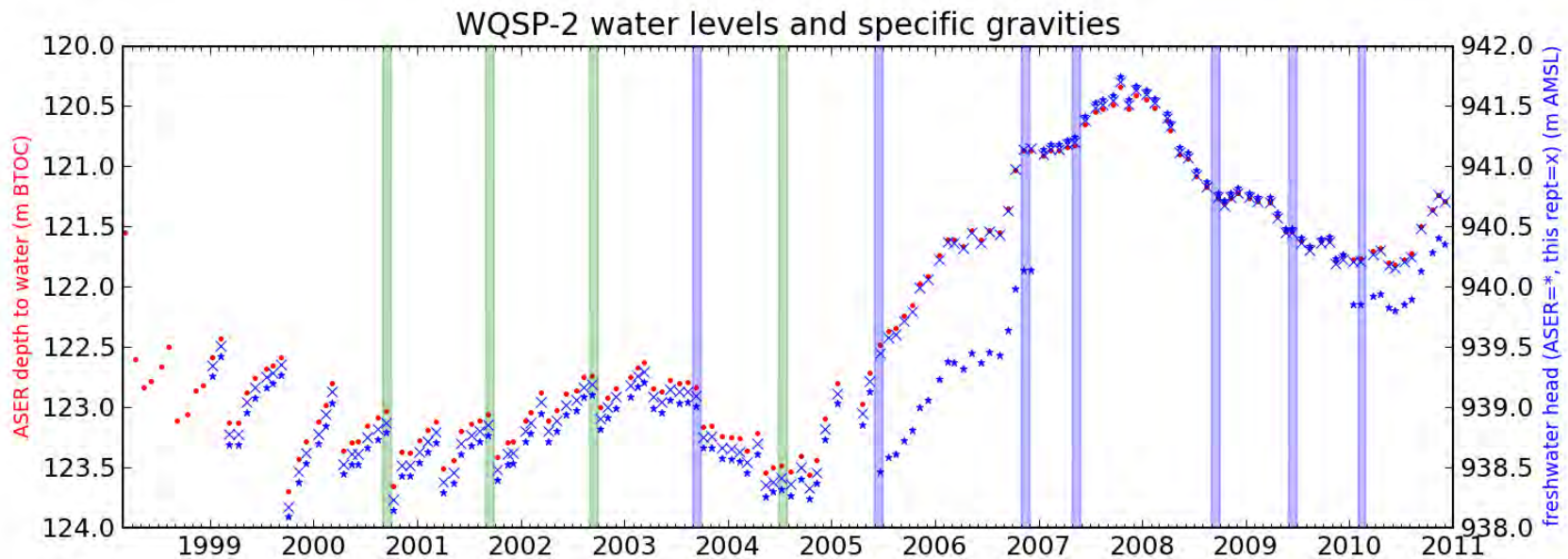


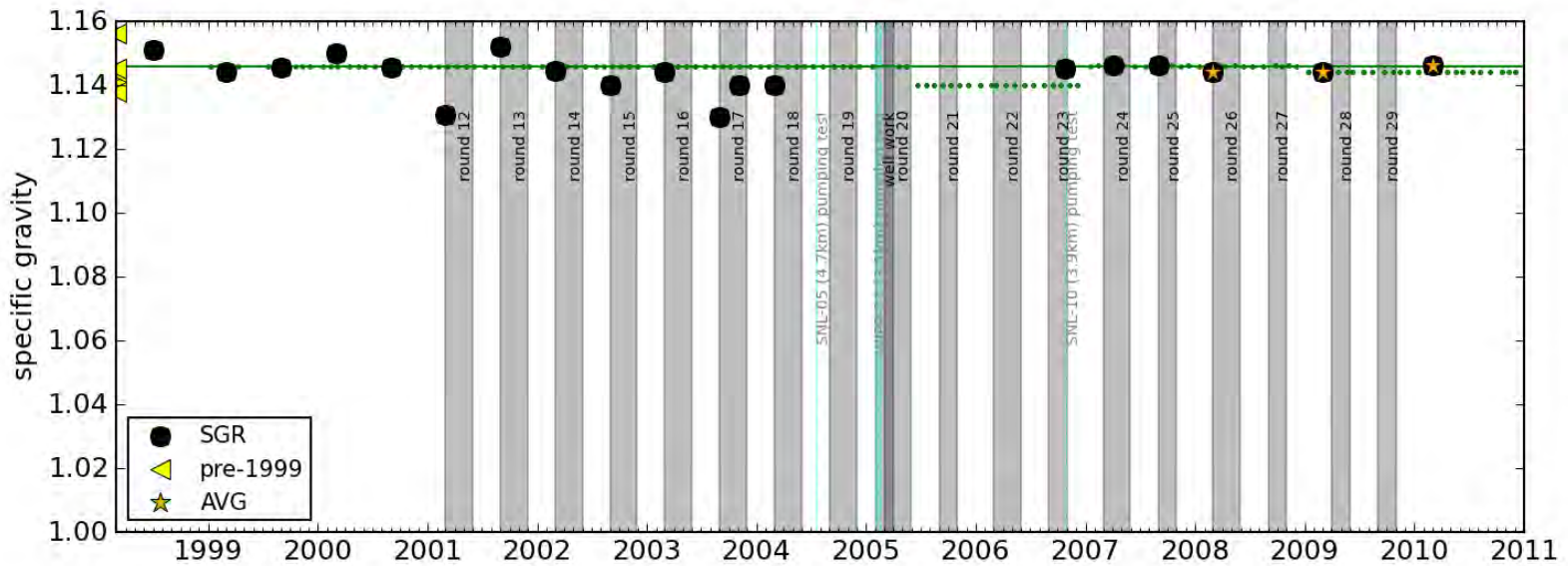
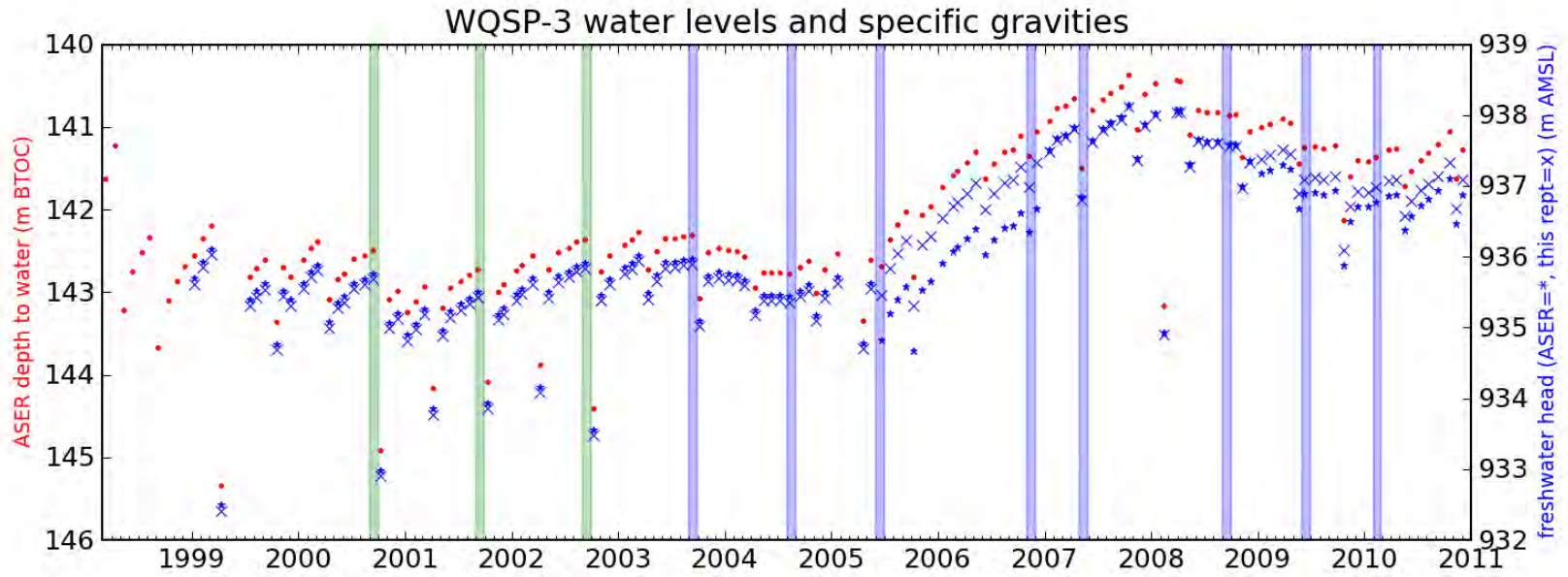


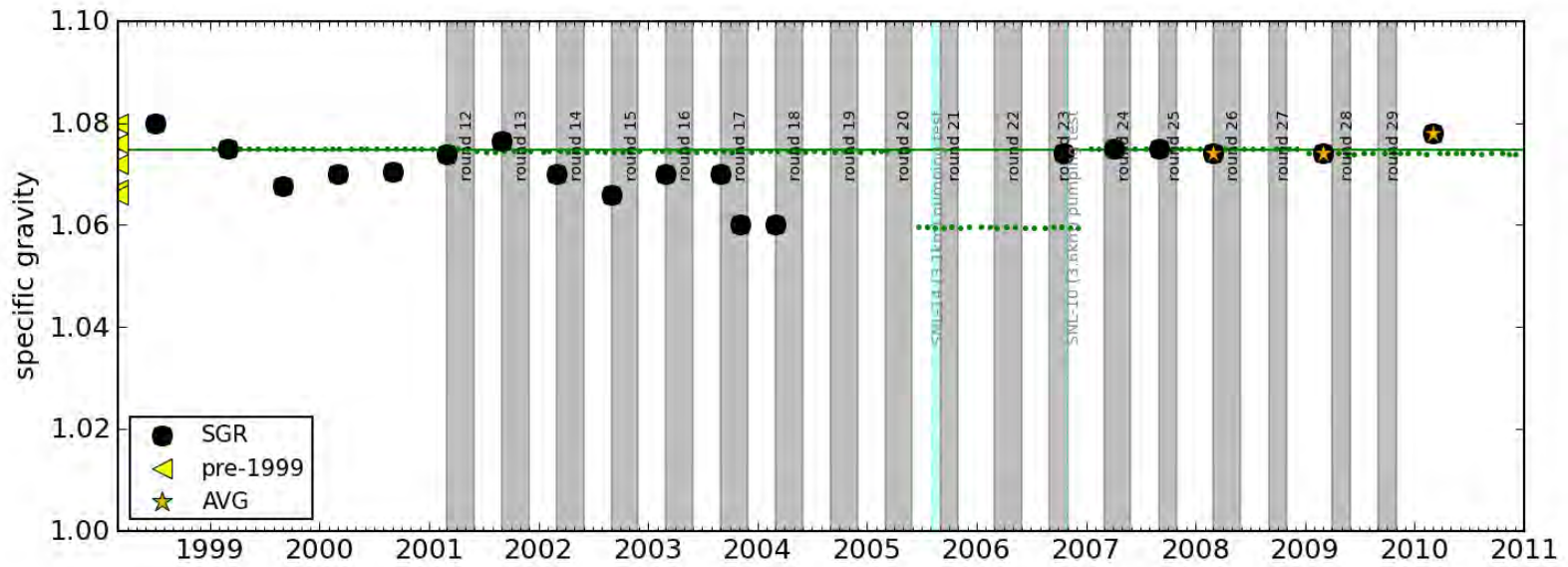
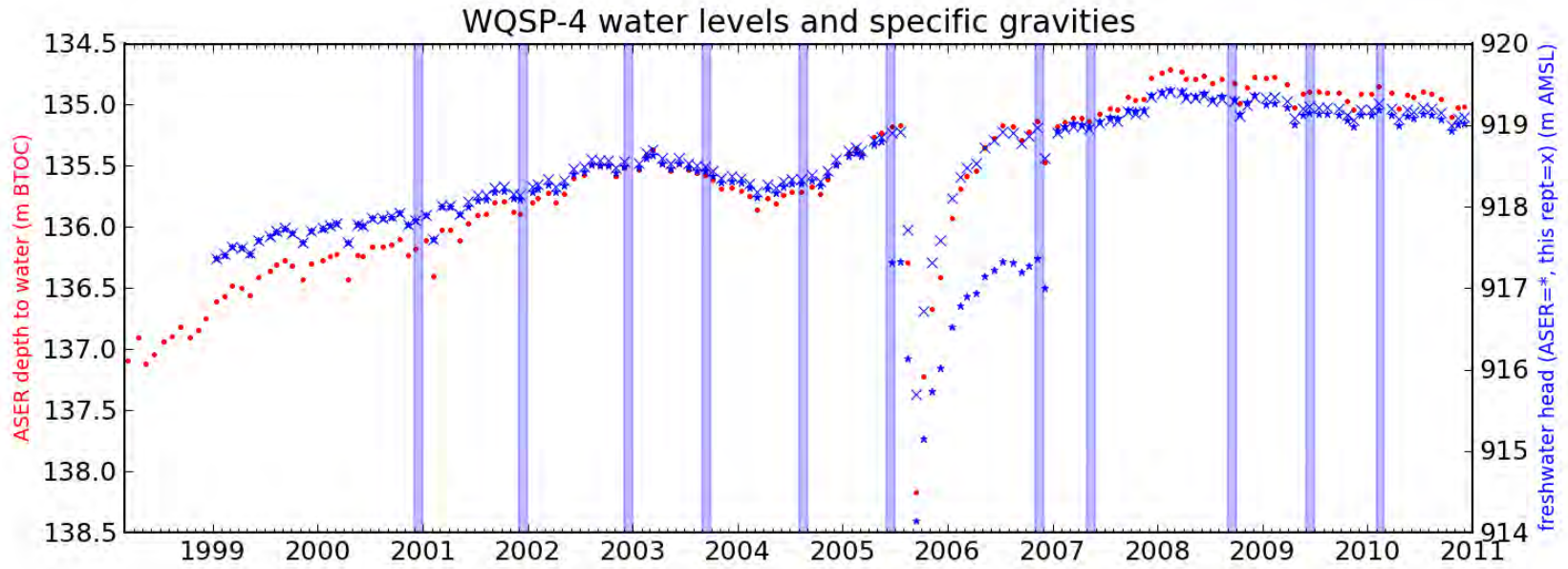


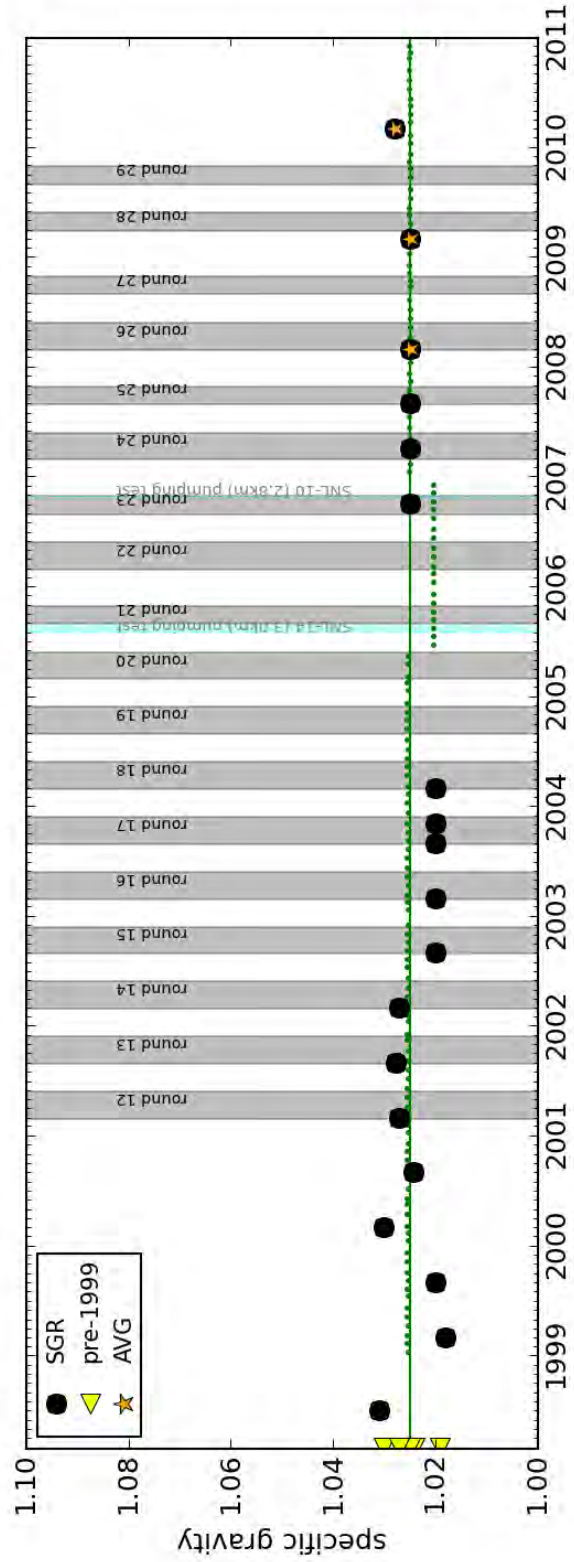
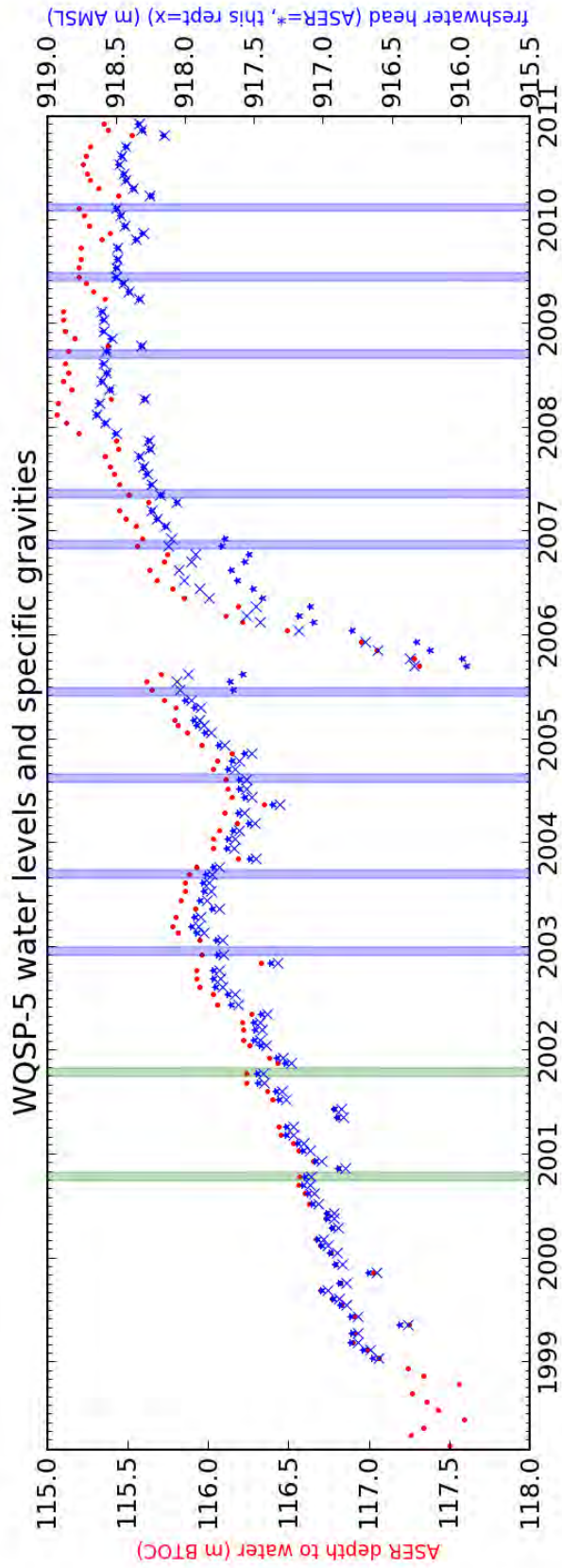


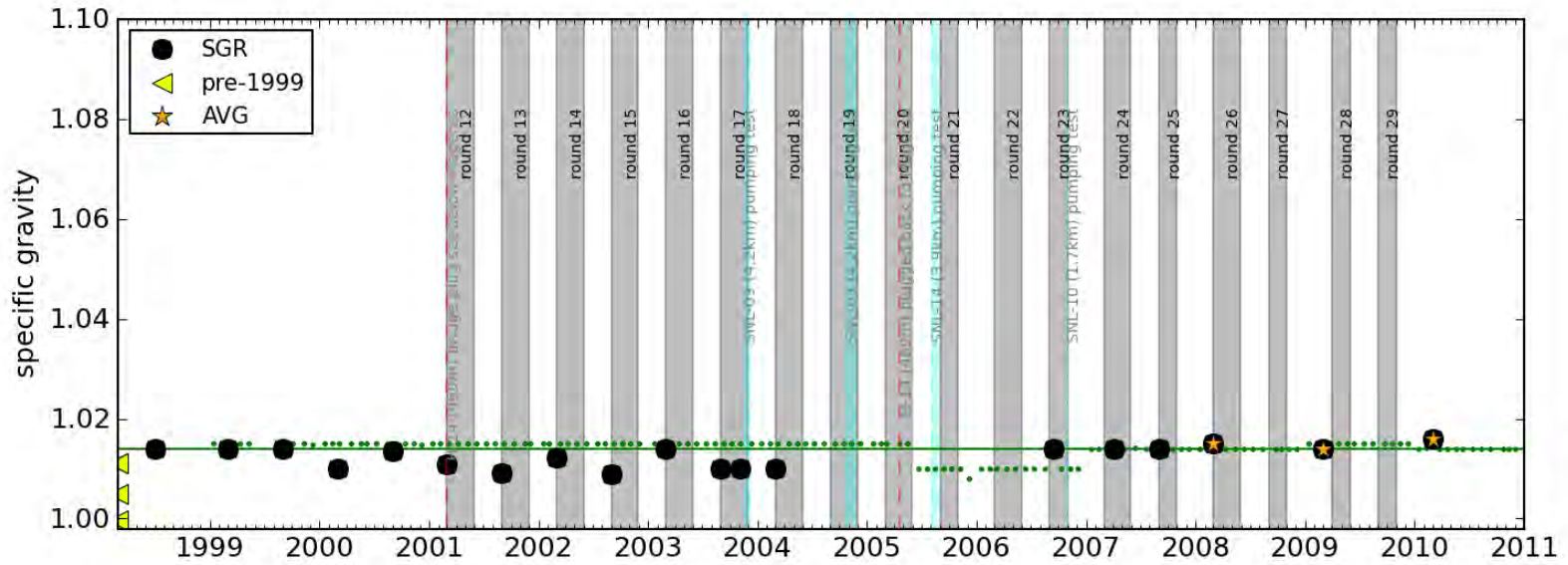
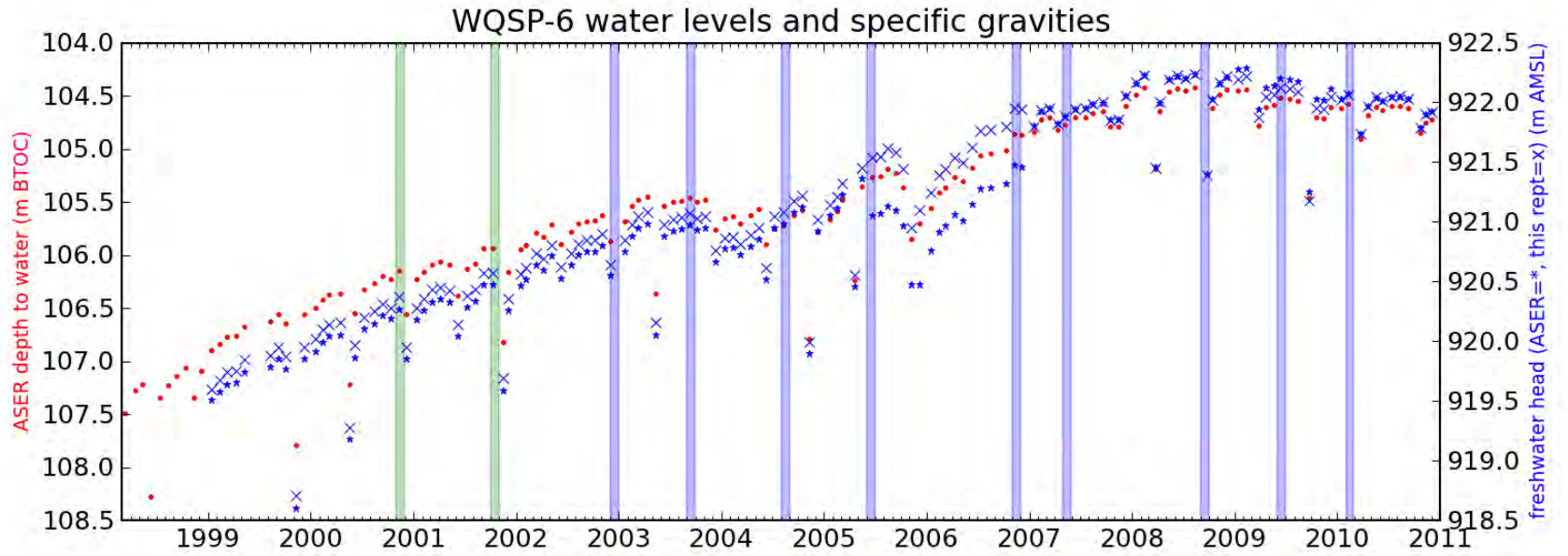












12 Appendix: MODFLOW and Pest Files and Script Source Listings

12.1 Input File Listing

The following table lists the input files for the 2000 contour map. The 2001-2004 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_2000ASER.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_2000ASER.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

12.2 Output File Listing

The following table lists the input files for the 2000 contour map. The 2001-2004 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.blm
16K	PEST output	matrix condition numbers	bc_adjust_2000ASER.cnd
2.7K	PEST output	binary intermediate file	bc_adjust_2000ASER.drf
7.4K	PEST output	binary intermediate file	bc_adjust_2000ASER.jac
7.5K	PEST output	binary intermediate file	bc_adjust_2000ASER.jco
9.9K	PEST output	binary intermediate file	bc_adjust_2000ASER.jst
3.8K	PEST output	parameter statistical matrices	bc_adjust_2000ASER.mtt
477	PEST output	parameter file	bc_adjust_2000ASER.par
62K	PEST output	optimization record	bc_adjust_2000ASER.rec
4.6K	PEST output	model outputs for last iteration	bc_adjust_2000ASER.rei
8.4K	PEST output	summary of residuals	bc_adjust_2000ASER.res
28	PEST output	binary restart file	bc_adjust_2000ASER.rst
24K	PEST output	relative parameter sensitivities	bc_adjust_2000ASER.sen
4.0K	PEST output	absolute parameter sensitivities	bc_adjust_2000ASER.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

12.3 Individual MODFLOW and Pest Script Listings

12.3.1 Bash shell script checkout_average_run_modflow.sh

```
1  #!/bin/bash
2
3  set -o nounset # explode if using an un-initialized variable
4  set -o errexit # exit on non-zero error status of sub-command
5
6  prefix=2000ASER
7
8  # this script makes the following directory substructure
9  #
10 #  current_dir \----- Outputs (calibrated parameter fields - INPUTS)
11 #               \----- Inputs (other modflow files - INPUTS)
12 #               \----- original_average (foward sim using average fields)
13 #               \----- bin (MODFLOW and DTRKMF binaries)
14 #               \----- pest_0? (pest-adjusted results)
15
16 ##set -o xtrace # loads of verbose debugging info
17
18 echo " ~~~~~~ "
19 echo " checking out T fields"
20 echo " ~~~~~~ "
21
22 # these will checkout the calibrated parameter-field data into subdirectories
23 # checkout things that are different for each of the 100 realizations
24 for d in `cat keepers`
25 do
26     cvs -d /nfs/data/CVSLIB/Tfields checkout Outputs/${d}/modeled_{K,A,R,S}_field.mod
27 done
28
29 ## checkout MODFLOW input files that are constant for across all realizations
30 cvs -d /nfs/data/CVSLIB/Tfields checkout Inputs/data/elev_{top,bot}.mod
31 cvs -d /nfs/data/CVSLIB/Tfields checkout Inputs/data/init_{bnds.inf,head.mod}
32 cvs -d /nfs/data/CVSLIB/Tfields checkout Inputs/modflow/mf2k_culebra.{lmg,lpf}
33 cvs -d /nfs/data/CVSLIB/Tfields checkout Inputs/modflow/mf2k_head.{ba6,nam,oc,dis,rch}
34
35 ## modify the path of "updated" T-fields, so they are all at the
36 ## same level in the directory structure (simplifying scripts elsewhere)
37
38 if [ -a keepers_short ]
39 then
40     rm keepers_short
41 fi
42 touch keepers_short
43
44 for d in `cat keepers`
45 do
46     bn=`basename ${d}`
47     # test whether it is a compound path
48     if [ ${d} != ${bn} ]
49     then
50         dn=`dirname ${d}`
51         mv ./Outputs/${d} ./Outputs/
52
53         # put an empty file in the directory to indicate
54         # what the directory was previously named
55         touch ./Outputs/${bn}/${dn}
56     fi
57
58     # create a keepers list without directories
59     echo ${bn} >> keepers_short
60 done
61
```



```

62 # -----
63 # the averaging was a slow step, when done in python
64
65 echo " ~~~~~ "
66 echo " perform averaging across all realizations "
67 echo " ~~~~~ "
68
69 python average_realizations.py
70
71 # checkout MODFLOW / DTRKMF executables
72 cvs -d /nfs/data/CVSLIB/MODFLOW2K checkout bin/mf2k/mf2k_1.6.release
73 cvs -d /nfs/data/CVSLIB/MODFLOW2K checkout bin/dtrkmf/dtrkmf_v0100
74
75 # check out pest and obs2mod binaries
76 cd bin
77 cvs -d /nfs/data/CVSLIB/PEST checkout Builds/Linux/pest.exe
78 cvs -d /nfs/data/CVSLIB/PEST checkout Builds/Linux/mod2obs.exe
79 cd ..
80
81 # -----
82
83 echo " ~~~~~ "
84 echo " setup copies of files constant between all realizations "
85 echo " ~~~~~ "
86
87 # directory for putting original base-case results in
88 od=original_average
89
90 if [ -d ${od} ]
91 then
92     echo ${od}" directory exists: removing and re-creating"
93     rm -rf ${od}
94 fi
95
96 mkdir ${od}
97 cd ${od}
98 echo `pwd`
99
100 # link to unchanged input files
101 for file in `cat ../modflow_files.dat`
102 do
103     ln -sf ${file} .
104 done
105
106 # link to averaged files computed in previous step
107 for f in {A,R,K,S}
108 do
109     ln -sf ../modeled_${f}_field.avg ./modeled_${f}_field.mod
110 done
111
112 ln -sf elev_top.mod fort.33
113 ln -sf elev_bot.mod fort.34
114
115 echo " ~~~~~ "
116 echo " run original MODFLOW and DTRKMF and export results for plotting "
117 echo " ~~~~~ "
118
119 # run MODFLOW, producing average head and CCF
120 ../bin/mf2k/mf2k_1.6.release mf2k_head.nam
121
122 # run DTRKMF, producing particle track (from ccf)
123 ../bin/dtrkmf/dtrkmf_v0100 <dtrkmf.in
124
125 # convert binary MODFLOW head output to Surfer ascii grid file format

```

```

126 ln -sf ../head_bin2ascii.py .
127 python head_bin2ascii.py
128 mv modeled_head_asciimed.grd modeled_head- $\{od\}$ .grd
129
130 # convert DTRKMF output from cells to X,Y and
131 # save in Surfer blanking file format
132 ln -sf ../convert_dtrkmf_output_for_surfer.py .
133 python convert_dtrkmf_output_for_surfer.py
134 mv dtrk_output.blm dtrk_output- $\{od\}$ .blm
135
136 # extract head results at well locations and merge with observed
137 # head file for easy scatter plotting in Excel (tab delimited)
138 for file in `cat ../mod2obs_files.dat`
139 do
140     ln -sf  $\{file\}$  .
141 done
142
143 ln -sf ../meas_head- $\{prefix\}$ .smp .
144 ln -sf ../obs_loc- $\{prefix\}$ .dat .
145 ../bin/Builds/Linux/mod2obs.exe <mod2obs_head.in
146 ln -sf ../merge_observed_modeled_heads.py
147 python merge_observed_modeled_heads.py
148 mv both_heads.smp modeled_vs_observed_head- $\{od\}$ .txt
149
150
151 # go back down into root directory
152 cd ..
153 echo `pwd`
154
155 echo " ~~~~~~"
156 echo "  setup and run PEST to optimize parametric surface to set BC "
157 echo " ~~~~~~"
158
159 for p in pest_02
160 do
161
162     if [ -d  $\{p\}$  ]
163     then
164         echo  $\{p\}$ " directory exists: removing and re-creating"
165         rm -rf  $\{p\}$ 
166     fi
167
168     mkdir  $\{p\}$ 
169     cd  $\{p\}$ 
170     echo `pwd`
171
172     # link to unchanged input files
173     for file in `cat ../modflow_files.dat`
174     do
175         ln -sf  $\{file\}$  .
176     done
177
178     # link to averaged files computed in previous step
179     for f in {A,R,K,S}
180     do
181         ln -sf ../modeled- $\{f\}$ _field.avg ./modeled- $\{f\}$ _field.mod
182     done
183
184     # link to mod2obs files (needed for pest)
185     for file in `cat ../mod2obs_files.dat`
186     do
187         ln -sf  $\{file\}$  .
188     done
189

```

```

190 # link to pest files
191 for file in `cat ../${p}_files.dat`
192 do
193     ln -s ${file} .
194 done
195
196 # rename 'original' versions of files to be modified by pest
197 rm init_head.mod
198 ln -sf ../Inputs/data/init_head.mod ./init_head_orig.mod
199 rm init_bnds.inf
200 ln -sf ../Inputs/data/init_bnds.inf ./init_bnds_orig.inf
201
202 # create new ibound array for easier modification during PEST
203 # optimization iterations
204 python boundary_types.py
205
206 # create the necessary input files from observations
207 python create_${p}_input.py
208
209 # run pest
210 ../bin/Builds/Linux/pest.exe bc_adjust_${prefix}
211
212 # last output files should be best run
213 # extract all the stuff from that output
214 #####
215
216 ln -sf elev_top.mod fort.33
217 ln -sf elev_bot.mod fort.34
218
219 ../bin/dtrkmf/dtrkmf_v0100 <dtrkmf.in
220
221 ln -sf ../head_bin2ascii.py .
222 python head_bin2ascii.py
223 mv modeled_head_asiihed.grd modeled_head_${p}.grd
224
225 ln -sf ../convert_dtrkmf_output_for_surfer.py .
226 python convert_dtrkmf_output_for_surfer.py
227 mv dtrk_output.blm dtrk_output_${p}.blm
228
229 for file in `cat ../mod2obs_files.dat`
230 do
231     ln -sf ${file} .
232 done
233
234 ../bin/Builds/Linux/mod2obs.exe <mod2obs_head.in
235 ln -sf ../merge_observed_modeled_heads.py
236 python merge_observed_modeled_heads.py
237 mv both_heads.smp modeled_vs_observed_head_${p}.txt
238
239 cd ..
240 done

```

12.3.2 Python script average_realizations.py

```
1 from math import log10 ,pow
2
3 nrow = 307
4 ncol = 284
5 nel = nrow*ncol
6 nfr = 100 # number of fields (realizations)
7 nft = 4 # number of field types
8
9 def floatload(filename):
10     """Reads file (a list of strings, one per row) into a list of strings."""
11     f = open(filename, 'r')
12     m = [float(line.rstrip()) for line in f]
13     f.close()
14     return m
15
16 types = ['K', 'A', 'R', 'S']
17
18 # get list of 100 best calibrated fields
19 flist = open('keepers_short', 'r')
20 runs = flist.read().strip().split('\n')
21 flist.close()
22
23 # initialize to help speed lists up a bit
24 # nfr (100) realizations of each
25 fields = []
26 for i in xrange(nft):
27     fields.append([None]*nfr)
28     for i in xrange(nfr):
29         # each realization being nel (87188) elements
30         fields[-1][i] = [None]*nel
31
32 # read in all realizations
33 print 'reading ...'
34 for i,run in enumerate(runs):
35     print i,run
36     for j,t in enumerate(types):
37         fields[j][i][0:nel] = floatload('Outputs/'+ run +'/modeled_'+ t +'_field.mod')
38
39 # open up files for writing
40 fh = []
41 for t in types:
42     fh.append(open('modeled_'+ t +'_field.avg', 'w'))
43
44 # transpose fields to allow slicing across realizations, rather than across cells
45 for j in range(len(types)):
46     fields[j] = zip(*(fields[j]))
47
48 print 'writing ...'
49 # do averaging across 100 realizations
50 for i in xrange(nel):
51     if i%10000 == 0:
52         print i
53     for h,d in zip(fh, fields):
54         h.write('%18.11e\n' % pow(10.0, sum(map(log10, d[i]))/ nfr) )
55
56 for h in fh:
57     h.close()
```

12.3.3 Python script boundary_types.py

```
1 nx = 284          # number columns in model grid
2 ny = 307          # number rows
3 nel = nx*ny
4
5 def intload(filename):
6     """Reads file (a 2D integer array) as a list of lists.
7     Outer list is rows, inner lists are columns."""
8     f = open(filename, 'r')
9     m = [[int(v) for v in line.rstrip().split()] for line in f]
10    f.close()
11    return m
12
13 def intsave(filename, m):
14    """Writes file as a list of lists as a 2D integer array, format '%3i'.
15    Outer list is rows, inner lists are columns."""
16    f = open(filename, 'w')
17    for row in m:
18        f.write(' '.join(['%2i' % col for col in row]) + '\n')
19    f.close()
20
21 def floatload(filename):
22    """Reads file (a list of real numbers, one number each row) into a list of floats."""
23    f = open(filename, 'r')
24    m = [float(line.rstrip()) for line in f]
25    f.close()
26    return m
27
28 def reshapev2m(v):
29    """Reshape a vector that was previously reshaped in C-major order from a matrix,
30    back into a matrix (here a list of lists)."""
31    m = [None]*ny
32    for i, (lo, hi) in enumerate(zip(xrange(0, nel-nx+1, nx), xrange(nx, nel+1, nx))):
33        m[i] = v[lo:hi]
34    return m
35
36 #####
37
38 # read in original MODFLOW IBOUND array (only 0,1, and -1)
39 ibound = intload('init_bnds_orig.inf')
40
41 # read in initial heads
42 h = reshapev2m(floatload('init_head_orig.mod'))
43
44 # discriminate between two types of constant head boundaries
45 # -1) CH, where value > 1000 (area east of halite margin)
46 # -2) CH, where value < 1000 (single row/column of cells along edge of domain)
47
48 for i, row in enumerate(ibound):
49     for j, val in enumerate(row):
50         # is this constant head and is starting head less than 1000m ?
51         if ibound[i][j] == -1 and h[i][j] < 1000.0:
52             ibound[i][j] = -2
53
54 # save new IBOUND array that allows easy discrimination between types in python script during
55 # PEST optimization runs, and is still handled the same by MODFLOW
56 # since all ibound values < 0 are treated as constant head.
57 intsave('init_bnds.inf', ibound)
```

12.3.4 Python script create_pest_02_input.py

```
1 prefix = '2000ASER'
2
3 #####
4 ## pest instruction file reads output from mod2obs
5 fin = open('meas_head_%s.smp' % prefix, 'r')
6
7 # each well is a [name,head] pair
8 wells = [[line.split()[0],line.split()[3]] for line in fin]
9 fin.close()
10
11 fout = open('modeled_head.ins','w')
12 fout.write('pif @\n')
13 for i,well in enumerate(wells):
14     fout.write("l1 [%s]39:46\n" % well[0])
15 fout.close()
16
17 # exponential surface used to set initial head everywhere
18 # except east of the halite margins, where the land surface is used.
19 # initial guesses come from AP-114 Task report
20 params = [928.0, 8.0, 1.2, 1.0, 1.0, -1.0, 0.5]
21 pnames = ['a', 'b', 'c', 'd', 'e', 'f', 'exp']
22
23 fout = open('avg_NS_res.ins','w')
24 fout.write("""pif @
25 l1 [medianN]1:16
26 l1 [medianS]1:16
27 l1 [meanN]1:16
28 l1 [meanS]1:16
29 """)
30 fout.close()
31
32
33 #####
34 ## pest template file
35 ftmp = open('surface_par_params.ptf','w')
36 ftmp.write('ptf @\n')
37 for n in pnames:
38     ftmp.write('@      %s      @\n' % n)
39 ftmp.close()
40
41
42 #####
43 ## pest parameter file
44
45 fpar = open('surface_par_params.par','w')
46 fpar.write('double point\n')
47 for n,p in zip(pnames,params):
48     fpar.write('%s %.2f 1.0 0.0\n' % (n,p))
49 fpar.close()
50
51
52 #####
53 ## pest control file
54
55 f = open('bc_adjust_%s.pst' % prefix, 'w')
56
57 f.write("""pcf
58 * control data
59 restart estimation
60 %i %i 1 0 2
61 1 2 double point 1 0 0
62 5.0 2.0 0.4 0.001 10
63 3.0 3.0 1.0E-3
```

```

64 0.1
65 30 0.001 4 4 0.0001 4
66 1 1 1
67 * parameter groups
68 bc relative 0.005 0.0001 switch 2.0 parabolic
69 """ % (len(params), len(wells)+4))
70
71 f.write('* parameter data\n')
72 for n,p in zip(pnames, params):
73     if p > 0:
74         f.write('%s none relative %.3f %.3f %.3f bc 1.0 0.0 1\n' %
75                 (n, p, -2.0*p, 3.0*p))
76     else:
77         f.write('%s none relative %.3f %.3f %.3f bc 1.0 0.0 1\n' %
78                 (n, p, 3.0*p, -2.0*p))
79
80 f.write("""* observation groups
81 ss_head
82 avg_head
83 * observation data
84 """)
85
86 ## read in observation weighting group definitions
87 fin = open('obs_loc_%s.dat' % prefix, 'r')
88 lines = fin.readlines()
89 fin.close()
90
91 weightidx = []
92 location = []
93 for line in lines:
94     w,l = line.strip().split()[1:3]
95     weightidx.append(w)
96     location.append(l)
97
98 weights = []
99
100 for l in weightidx:
101     # inside LWB (+H-4, very near the LWB)
102     if l == '0':
103         weights.append(2.5)
104     # near (<= 3km from) LWB
105     if l == '1':
106         weights.append(1.0)
107     # distant to LWB
108     if l == '2':
109         weights.append(0.4)
110     if l == '99':
111         weights.append(0.01) # AEC-7
112
113
114 for name, head, w in zip(zip(*wells)[0], zip(*wells)[1], weights):
115     f.write('%s %s %.3f ss_head\n' % (name, head, w))
116
117 numnorth = len([x for x in location if x=='N'])
118 numsouth = len([x for x in location if x=='S'])
119
120 f.write("""medianN 0.0 %.1f avg_head
121 medianS 0.0 %.1f avg_head
122 meanN 0.0 %.1f avg_head
123 meanS 0.0 %.1f avg_head
124 """) % (numnorth, numsouth, numnorth, numsouth))
125
126 f.write("""* model command line
127 ./run_02_model

```

```
128 * model input/output
129 surface_par_params.ptf surface_par_params.in
130 modeled_head.ins modeled_head.smp
131 avg_NS_res.ins avg_NS_res.smp
132 """)
133 f.close()
```


12.3.5 Python script surface_02_extrapolate.py

```
1 from itertools import chain
2 from math import sqrt
3
4 def matload(filename):
5     """Reads file (a 2D string array) as a list of lists.
6     Outer list is rows, inner lists are columns."""
7     f = open(filename, 'r')
8     m = [line.rstrip().split() for line in f]
9     f.close()
10    return m
11
12 def floatload(filename):
13     """Reads file (a list of real numbers, one number each row) into a list of floats."""
14     f = open(filename, 'r')
15     m = [float(line.rstrip()) for line in f]
16     f.close()
17    return m
18
19 def reshapem2v(m):
20     """Reshapes a rectangular matrix into a vector in same fashion as numpy.reshape().
21     which is C-major order"""
22    return list(chain(*m))
23
24 def sign(x):
25     """ sign function """
26     if x<0:
27         return -1
28     elif x>0:
29         return +1
30     else:
31         return 0
32
33 #####
34
35 # read in modified IBOUND array, with the cells to modify set to -2
36 ibound = reshapem2v(matload('init_bnds.inf'))
37
38 h = floatload('init_head_orig.mod')
39
40 # these are relative coordinates, -1 <= x,y < +1
41 x = floatload('rel_x_coord.dat')
42 y = floatload('rel_y_coord.dat')
43
44 # unpack surface parameters (one per line)
45 # z = A + B*(y + D*sign(y)*sqrt(abs(y)))+C*(E*x**3 - F*x**2 - x)
46
47 finput = open('surface_par_params.in', 'r')
48 try:
49     a,b,c,d,e,f,exp = [float(line.rstrip()) for line in finput]
50 except ValueError:
51     # python doesn't like 'D' in 1.2D-4 notation used by PEST sometimes.
52     finput.seek(0)
53     lines = [line.rstrip() for line in finput]
54     for i in range(len(lines)):
55         lines[i] = lines[i].replace('D','E')
56     a,b,c,d,e,f,exp = [float(line) for line in lines]
57
58 finput.close()
59
60 # file to output initial/boundary head for MODFLOW model
61 fout = open('init_head.mod', 'w')
62 for i in xrange(len(ibound)):
63     if ibound[i] == '-2' or ibound[i] == '1':
```

```

64     # apply exponential surface to active cells (ibound=1) -> starting guess
65     # and non-geologic boundary conditions (ibound=-2) -> constant head value
66     if y[i] == 0:
67         fout.write('%.7e \n' % (a + c*(e*x[i]**3 + f*x[i]**2 - x[i])))
68     else:
69         fout.write('%.7e \n' % (a + b*(y[i] + d*sign(y[i])*abs(y[i])**exp) +
70                                 c*(e*x[i]**3 + f*x[i]**2 - x[i])))
71 else:
72     # use land surface at constant head east of halite boundary
73     # ibound=0 doesn't matter (inactive)
74     fout.write('%.7e\n' % h[i])
75
76 fout.close()

```

12.3.6 Bash shell script run_02_model

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

12.3.7 Python script head_bin2ascii.py

```
1 import struct
2 from sys import argv, exit
3
4 class FortranFile(file):
5     """ modified from May 2007 Enthought-dev mailing list post by Neil Martinsen-Burrell """
6
7     def __init__(self, fname, mode='r', buf=0):
8         file.__init__(self, fname, mode, buf)
9         self.ENDIAN = '<' # little endian
10        self.di = 4 # default integer (could be 8 on 64-bit platforms)
11
12    def readReals(self, prec='f'):
13        """Read in an array of reals (default single precision) with error checking"""
14        # read header (length of record)
15        l = struct.unpack(self.ENDIAN+'i', self.read(self.di))[0]
16        data_str = self.read(l)
17        len_real = struct.calcsize(prec)
18        if l % len_real != 0:
19            raise IOError('Error reading array of reals from data file')
20        num = l/len_real
21        reals = struct.unpack(self.ENDIAN+str(num)+prec, data_str)
22        # check footer
23        if struct.unpack(self.ENDIAN+'i', self.read(self.di))[0] != 1:
24            raise IOError('Error reading array of reals from data file')
25        return list(reals)
26
27    def readInts(self):
28        """Read in an array of integers with error checking"""
29        l = struct.unpack('i', self.read(self.di))[0]
30        data_str = self.read(l)
31        len_int = struct.calcsize('i')
32        if l % len_int != 0:
33            raise IOError('Error reading array of integers from data file')
34        num = l/len_int
35        ints = struct.unpack(str(num)+'i', data_str)
36        if struct.unpack(self.ENDIAN+'i', self.read(self.di))[0] != 1:
37            raise IOError('Error reading array of integers from data file')
38        return list(ints)
39
40    def readRecord(self):
41        """Read a single fortran record (potentially mixed reals and ints)"""
42        dat = self.read(self.di)
43        if len(dat) == 0:
44            raise IOError('Empty record header')
45        l = struct.unpack(self.ENDIAN+'i', dat)[0]
46        data_str = self.read(l)
47        if len(data_str) != l:
48            raise IOError('Didn't read enough data')
49        check = self.read(self.di)
50        if len(check) != 4:
51            raise IOError('Didn't read enough data')
52        if struct.unpack(self.ENDIAN+'i', check)[0] != 1:
53            raise IOError('Error reading record from data file')
54        return data_str
55
56    def reshapev2m(v, nx, ny):
57        """Reshape a vector that was previously reshaped in C-major order from a matrix,
58        back into a C-major order matrix (here a list of lists)."""
59        m = [None]*ny
60        n = nx*ny
61        for i, (lo, hi) in enumerate(zip(xrange(0, n-nx+1, nx), xrange(nx, n+1, nx))):
62            m[i] = v[lo:hi]
63        return m
```

```

64
65 def floatmatsave(filehandle,m):
66     """Writes array to open filehandle , format '568%e12.5'.
67     Outer list is rows, inner lists are columns."""
68
69     for row in m:
70         f.write(''.join([' %12.5e' % col for col in row]) + '\n')
71
72 # open file and set endian-ness
73 try:
74     infn ,outfn = argv[1:3]
75 except:
76     print '2 command-line arguments not given, using default in/out filenames'
77     infn = 'modeled_head.bin'
78     outfn = 'modeled_head_asciihed.grd'
79
80 ff = FortranFile(infn)
81
82 # currently this assumes a single-layer MODFLOW model (or at least only one layer of output)
83
84 # format of MODFLOW header in binary layer array
85 fmt = '<2i2f16s3i'
86 # little endian, 2 integers, 2 floats,
87 # 16-character string (4 element array of 4-byte strings), 3 integers
88
89 while True:
90     try:
91         # read in header
92         h = ff.readRecord()
93
94     except IOError:
95         # exit while loop
96         break
97
98     else:
99         # unpack header
100         kstp ,kper ,pertim ,totim ,text ,ncol ,nrow ,ilay = struct.unpack(fmt,h)
101
102         # print status/confirmation to terminal
103         print kstp ,kper ,pertim ,totim ,text ,ncol ,nrow ,ilay
104
105         h = ff.readReals()
106
107     ff.close()
108
109     xmin , xmax = (601700.0,630000.0)
110     ymin , ymax = (3566500.0,3597100.0)
111     hmin = min(h)
112     hmax = max(h)
113
114     # write output in Surfer ASCII grid format
115     f = open(outfn , 'w')
116     f.write("""DSAA
117 %i %i
118 %.1f %.1f
119 %.1f %.1f
120 %.8e %.8e
121 """ %(ncol , nrow , xmin , xmax , ymin , ymax , hmin , hmax) )
122     hmat = reshapev2m(h, ncol , nrow)
123
124     # MODFLOW starts data in upper-left corner
125     # Surfer expects data starting in lower-left corner
126     # flip array in row direction
127

```

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

12.3.8 Python script merge_observed_modeled_heads.py

```
1 fobs = open('meas_head_2000ASER.smp','r') # measured head
2 fmod = open('modeled_head.smp','r') # modeled head
3 fwgt = open('obs_loc_2000ASER.dat','r') # weights
4 fdb = open('spec_wells.crd','r') # x/y coordinates
5
6 fout = open('both_heads.smp','w') # resulting file
7
8 # read in list of x/y coordinates, key by well name
9 wells = {}
10 for line in fdb:
11     well,x,y = line.split()[0:3] # ignore last column
12     wells[well.upper()] = [x,y]
13 fdb.close()
14
15 fout.write('\t'.join(['#NAME','UTM-NAD27-X','UTM-NAD27-Y',
16                     'OBSERVED','MODELED','OBS-MOD','WEIGHT'])+'\n')
17
18 for sobs,smod,w in zip(fobs,fmod,fwgt):
19     obs = float(sobs.split()[3])
20     mod = float(smod.split()[3])
21     name = sobs.split()[0].upper()
22     fout.write('\t'.join([name,wells[name][0],wells[name][1],
23                         str(obs),str(mod),str(obs-mod),
24                         w.rstrip().split()[1]])+'\n')
25
26 fobs.close()
27 fmod.close()
28 fwgt.close()
29 fout.close()
```

12.3.9 Python script `convert_dtrkmf_output_for_surfer.py`

```
1
2 # grid origin for dtrkmf cell -> x,y conversion
3 x0 = 601700.0
4 y0 = 3597100.0
5
6 dx = 100.0
7 dy = 100.0
8
9 fout = open('dtrk_output.blm', 'w')
10
11 # read in all results for saving particle tracks
12 fin = open('dtrk.out', 'r')
13 results = [l.split() for l in fin.readlines()[1:]]
14 fin.close()
15
16 npts = len(results)
17
18 # write Surfer blanking file header
19 fout.write('%i,1\n' % npts)
20
21 # write x,y location and time
22 for pt in results:
23     x = float(pt[1])*dx + x0
24     y = y0 - float(pt[2])*dy
25     t = float(pt[0])/7.75*4.0 # convert to 4m Cuelbra thickness
26     fout.write('%0.1f,%0.1f,%0.8e\n' % (x,y,t))
27
28 fout.close()
```


12.3.10 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.

```
1 import numpy as np
2 #import matplotlib
3 #matplotlib.use('Agg')
4 import matplotlib.pyplot as plt
5
6 fprefix = 'pest_02/'
7 mprefix = '../wipp-polyline-data/'
8 fname = fprefix + 'modeled_vs_observed_head_pest_02.txt'
9
10 ofname = 'original_average/modeled_vs_observed_head_original_average.txt'
11
12 M2FT = 0.3048
13 year = '2000'
14
15 # load in observed, modeled, obs-mod, (all in meters)
16 res = np.loadtxt(fname, skiprows=1, usecols=(3,4,5))
17 ores = np.loadtxt(ofname, skiprows=1, usecols=(3,4,5))
18
19 # load in weights
20 weights = np.loadtxt(fname, skiprows=1, usecols=(6,), dtype='int')
21 # load in names
22 names = np.loadtxt(fname, skiprows=1, usecols=(0,), dtype='|S6')
23
24 # load in N/S/C/X zones
25 zones = np.loadtxt('obs_loc_%sASER.dat' % year, usecols=(2,), dtype='|S1')
26
27 ## checking locations / zones
28 # *****
29 wipp = np.loadtxt(mprefix+'wipp_boundary.dat')
30 x,y = np.loadtxt(fname, skiprows=1, usecols=(1,2), unpack=True)
31
32 fig = plt.figure(2, figsize=(18,12))
33 ax1 = fig.add_subplot(121)
34 ax1.plot(x,y, 'k*') # wells
35 ax1.plot(wipp[:,0], wipp[:,1], 'r-') # WIPP LWB
36 buff = np.loadtxt(mprefix+'wipp_boundary.dat')
37 buff[1:3,0] -= 3000.0
38 buff[0,0] += 3000.0
39 buff[3:,0] += 3000.0
40 buff[2:4,1] -= 3000.0
41 buff[0:2,1] += 3000.0
42 buff[-1,1] += 3000.0
43 colors = {'N': 'red', 'S': 'blue', 'C': 'green', 'X': 'gray'}
44 ax1.plot(buff[:,0], buff[:,1], 'g--') # WIPP LWB+3km
45 for xv,yv,n,w,z in zip(x,y,names,weights,zones):
46     print xv,yv,n,w,z
47     plt.annotate('%s %i'%(n,w),xy=(xv,yv), fontsize=8,color=colors[z])
48 plt.axis('image')
49 ax1.set_xlim([x.min()-1000,x.max()+1000])
50 ax1.set_ylim([y.min()-1000,y.max()+1000])
51 ax2 = fig.add_subplot(122)
52 ax2.plot(x,y, 'k*') # wells
53 ax2.plot(wipp[:,0], wipp[:,1], 'r-') # WIPP LWB
54 ax2.plot(buff[:,0], buff[:,1], 'g--') # WIPP LWB+3km
55 for xv,yv,n,w,z in zip(x,y,names,weights,zones):
56     plt.annotate('%s %i'%(n,w),xy=(xv,yv), fontsize=8,color=colors[z])
57 plt.axis('image')
58 ax2.set_xlim([wipp[:,0].min()-100,wipp[:,0].max()+100])
59 ax2.set_ylim([wipp[:,1].min()-100,wipp[:,1].max()+100])
60 plt.suptitle('well weights check '+year)
61 plt.savefig('check-well-weights-'+year+'.png')
```

```

62
63 # convert lengths to feet
64 res /= M2FT
65 ores /= M2FT
66
67 # create the histogram of residuals for ASER
68 # *****
69
70 # -10,-9,...8,9,10
71 bins = np.arange(-10,11)
72 rectfig = (15,7)
73 squarefig = (8.5,8.5)
74
75 fig = plt.figure(1,figsize=rectfig)
76 ax = fig.add_subplot(111)
77 # all the data, all but distant wells
78 ax.hist([res[weights < 2,2], res[:,2]], bins=bins, range=(-10.0,10.0),
79         rwidth=0.75, align='mid',
80         color=['red','blue'],
81         label=['Inside LWB & <3km from WIPP LWB','All wells'])
82 ax.set_xlabel('Measured-Modeled (ft)')
83 ax.set_ylabel('Frequency')
84 ax.set_xticks(bins)
85 ax.set_ylim([0,10])
86 ax.set_yticks(np.arange(0,10,2))
87 plt.grid()
88 ax.yaxis.grid(True, which='major')
89 ax.xaxis.grid(False)
90 plt.legend(loc='upper left')
91 plt.title('Histogram of Model Residuals '+year)
92 plt.annotate('AEC-7 @ %.1f'%res[0,2], xy=(-9.75,5.0), xytext=(-8.5,5.0),
93            arrowprops={'arrowstyle':'->'}, fontsize=16)
94 plt.savefig('model-error-histogram-'+year+'.png')
95 plt.close(1)
96
97 # create bar chart plot of individual residual for ASER
98 # *****
99
100 m0 = weights==0
101 m1 = weights==1
102 m2 = np.logical_or(weights==2,weights==99)
103
104 # separate wells into groups
105 resin = res[m0,2]
106 resnear = res[m1,2]
107 resfar = res[m2,2]
108
109 nin = resin.size
110 nnear = resnear.size
111 nfar = resfar.size
112
113 # separate names into groups
114 namin = names[m0]
115 namnear = names[m1]
116 namfar = names[m2]
117
118 print 'in',namin
119 print 'near',namnear
120 print 'far',namfar
121
122 print 'resfar',resfar
123
124 # get indices that sort vectors
125 ordin = np.argsort(namin)

```

```

126 ordnear = np.argsort(namnear)
127 ordfar = np.argsort(namfar)
128
129 # put vectors back together (groups adjacent and sorted inside each group)
130 resagg = np.concatenate((resin[ordin], resnear[ordnear], resfar[ordfar]), axis=0)
131 namagg = np.concatenate((namin[ordin], namnear[ordnear], namfar[ordfar]), axis=0)
132
133 fig = plt.figure(1, figsize=rectfig)
134 ax = fig.add_subplot(111)
135
136 wid = 0.6
137 shift = 0.5 - wid/2.0
138 ab = np.arange(res.shape[0])
139
140 print ab.shape
141 print ab
142
143 ax.bar(left=ab+shift, height=resagg, width=0.6, bottom=0.0, color='gray')
144 ax.set_ylim([-15.0, 15.0])
145 ax.spines['bottom'].set_position('zero')
146 ax.spines['top'].set_color('none')
147 ax.xaxis.set_ticks_position('bottom')
148 plt.xticks(ab+wid, namagg, rotation=90)
149 # vertical lines dividing groups
150 ax.axvline(x=nin, color='black', linestyle='dashed')
151 ax.axvline(x=nin+nnear, color='black', linestyle='dashed')
152 ax.axhline(y=0, color='black', linestyle='solid')
153 ax.axhline(y=-15, color='black', linestyle='dotted')
154 plt.grid()
155 ax.yaxis.grid(True, which='major')
156 ax.xaxis.grid(False)
157 ax.set_xlim([0, res.shape[0]])
158
159 plt.annotate('', xy=(0.0, 12.0), xycoords='data',
160             xytext=(nin, 12.0), textcoords='data',
161             arrowprops={'arrowstyle': '<->'})
162 plt.annotate('inside WIPP LWB', xy=(nin/3.0, 12.5), xycoords='data')
163
164 plt.annotate('', xy=(nin, 12.0), xycoords='data',
165             xytext=(nin+nnear, 12.0), textcoords='data',
166             arrowprops={'arrowstyle': '<->'})
167 plt.annotate('<3km WIPP LWB', xy=(nin+nnear/3.0, 12.5), xycoords='data')
168
169 plt.annotate('', xy=(nin+nnear, 12.0), xycoords='data',
170             xytext=(nin+nnear+nfar, 12.0), textcoords='data',
171             arrowprops={'arrowstyle': '<->'})
172 plt.annotate('>3km WIPP LWB', xy=(nin+nnear+nfar/3.0, 12.5), xycoords='data')
173
174 ax.set_ylabel('Measured-Modeled (ft)')
175 ax.set_title('individual residuals '+year)
176 plt.annotate('AEC-7 @ %.1f'%res[0, 2], xy=(nin+nnear+1.0, -14.5), xycoords='data')
177
178 plt.savefig('model-error-residuals-'+year+'.png')
179 plt.close(1)
180
181
182 # create scatter plot of measured vs. modeled
183 # *****
184 m = 1.0/M2FT
185 sr = [2980, 3120]
186
187
188 print 'modeled-vs-measured correlation coefficients'
189 print 'all data: %.4f' % np.corrcoef(res[:, 0], res[:, 1])[1, 0]**2

```

```

190 print 'inside WIPP: %.4f' % np.corrcoef(res[m0,0], res[m0,1])[1,0]**2
191 print 'inside 3km: %.4f' % np.corrcoef(res[weights < 2,0], res[weights < 2,1])[1,0]**2
192
193 print 'uncalibrated model'
194 print 'all data: %.4f' % np.corrcoef(ores[:,0], ores[:,1])[1,0]**2
195 print 'inside WIPP: %.4f' % np.corrcoef(ores[m0,0], ores[m0,1])[1,0]**2
196 print 'inside 3km: %.4f' % np.corrcoef(ores[weights < 2,0], ores[weights < 2,1])[1,0]**2
197
198 fig = plt.figure(1, figsize=squarefig)
199 ax = fig.add_subplot(111)
200 ax.plot(res[m0,0], res[m0,1], color='red', markersize=10,
201         marker='+', linestyle='none', label='Inside LWB')
202 ax.plot(res[m1,0], res[m1,1], color='green', markersize=10,
203         marker='x', linestyle='none', label='< 3km From LWB')
204 ax.plot(res[m2,0], res[m2,1], color='blue', markersize=10,
205         marker='*', linestyle='none', label='distant')
206 ax.plot(sr, sr, 'k-', label='$45^\{\degree\}$ Perfect Fit')
207 ax.plot([sr[0], sr[1]], [sr[0]+m, sr[1]+m], 'g-', linewidth=0.5, label='$\pm$ 1m Misfit')
208 ax.plot([sr[0], sr[1]], [sr[0]-m, sr[1]-m], 'g-', linewidth=0.5, label='__nolegend__')
209 ax.set_xticks(np.linspace(sr[0], sr[1], 8))
210 ax.set_yticks(np.linspace(sr[0], sr[1], 8))
211 ax.set_xlim(sr)
212 ax.set_ylim(sr)
213 plt.minorticks_on()
214 plt.legend(loc='lower right', scatterpoints=1, numpoints=1)
215 plt.grid()
216 a = []
217 for j, lab in enumerate(names):
218     if res[j,2] < -1.5*m:
219         # plot labels to left of value far above 45-degree line
220         a.append(plt.annotate(lab, xy=(res[j,0], res[j,1]),
221                               xytext=(res[j,0]-(2.9*len(lab)), res[j,1]-2.0), fontsize=14))
222     elif res[j,2] > 1.5*m:
223         # plot labels to right of value far below 45-degree line
224         plt.annotate(lab, xy=(res[j,0], res[j,1]),
225                      xytext=(res[j,0]+2.0, res[j,1]-2.0), fontsize=14)
226
227 ax.set_xlabel('Observed Freshwater Head (ft AMSL)')
228 ax.set_ylabel('Modeled Freshwater Head (ft AMSL)')
229 ax.set_title('modeled vs. measured '+year)
230
231 # manually adjust overlapping labels
232 for lab in a:
233     lab.draggable()
234 plt.show()
235 # #####
236
237 plt.savefig('scatter_pest_02_'+year+'.png')

```

12.3.11 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.

```
1 import numpy as np
2 #import matplotlib
3 #matplotlib.use('Agg')
4 import matplotlib.pyplot as plt
5 from mpl_toolkits.basemap import pyproj
6
7 # http://spatialreference.org/ref/epsg/26713/
8 # http://spatialreference.org/ref/epsg/31013/
9 putm = pyproj.Proj(init='epsg:26713') # UTM Zone 13N NAD27 (meters)
10 pstp = pyproj.Proj(init='epsg:32012') # NM state plane east NAD27 (meters)
11
12 def transform(xin, yin):
13     """does the default conversion from utm -> state plane
14     then also convert to feet from meters"""
15     xout, yout = pyproj.transform(putm, pstp, xin, yin)
16     xout /= M2FT
17     yout /= M2FT
18     return xout, yout
19
20 year = '2000'
21 fprefix = 'pest_02/'
22 mprefix = '../wipp-polyline-data/'
23 cfname = fprefix + 'modeled_head_pest_02.grd'
24 pfname = fprefix + 'dtrk_output_pest_02.blm'
25 wfname = fprefix + 'modeled_vs_observed_head_pest_02.txt'
26
27 M2FT = 0.3048
28
29 # read in well-related things
30 # %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
31 # load in observed, modeled, obs-mod, (all in meters)
32 res = np.loadtxt(wfname, skiprows=1, usecols=(3, 4, 5))
33 res /= M2FT # convert heads to feet
34 wellx, welly = transform(*np.loadtxt(wfname, skiprows=1, usecols=(1, 2), unpack=True))
35 names = np.loadtxt(wfname, skiprows=1, usecols=(0, ), dtype='|S6')
36
37 # read in head-related things
38 # %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
39 h = np.loadtxt(cfname, skiprows=5) # ASCII matrix of modeled head in meters AMSL
40 h[h<0.0] = np.NaN # no-flow zone in northeast
41 h[h>1000.0] = np.NaN # constant-head zone in east
42 h /= M2FT # convert elevations to feet
43
44 # surfer grid is implicit in header
45 # create grid from min/max UTM NAD27 coordinates (meters)
46 utmy, utmx = np.mgrid[3566500.0:3597100.0:307j, 601700.0:630000.0:284j]
47
48 # head contour coords
49 hx, hy = transform(utmx, utmy)
50 del utmx, utmy
51
52 # read in particle-related things
53 # %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
54 px, py = transform(*np.loadtxt(pfname, skiprows=1, delimiter=',', usecols=(0, 1), unpack=True))
55 part = np.loadtxt(pfname, skiprows=1, delimiter=',', usecols=(2, ))
56
57 # read in MODFLOW model, WIPP LWB & ASER contour domain (UTM X & Y)
58 # %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
59 modx, mody = transform(*np.loadtxt(mprefix+'total_boundary.dat', unpack=True))
60 wippx, wippy = transform(*np.loadtxt(mprefix+'wipp_boundary.dat',
61                                 usecols=(0, 1), unpack=True))
```



```

126 ax.plot(wippx, wippy, 'k-')
127 ax.plot(modx, mody, '-', color='purple', linewidth=2)
128 ax.plot(wellx, welly, linestyle='none', marker='o',
129         markeredgecolor='green', markerfacecolor='none')
130 ax.plot(px, py, linestyle='solid', color='blue', linewidth=4)
131 plt.arrow(x=px[-3], y=py[-3], dx=-10, dy=-50,
132          linewidth=4, color='blue', head_length=500, head_width=500)
133 plt.axis('image')
134 ax.set_xlim([ aserx.min(), aserx.max() ])
135 ax.set_ylim([ asery.min(), asery.max() ])
136 ax.clabel(CS, lev[:,2], fmt='%i', inline_spacing=2)
137 ax.set_xticks(660000 + np.arange(5.0)*5000)
138 ax.set_yticks(485000 + np.arange(5.0)*5000)
139 labels = ax.get_yticklabels()
140 for label in labels:
141     label.set_rotation(90)
142 for j, (x, y, n) in enumerate(zip(wellx, welly, names)):
143     # only plot labels of wells inside the figure area
144     if aserx.min() < x < aserx.max() and asery.min() < y < asery.max():
145         # name above
146         a.append(plt.annotate(n, xy=(x, y), xytext=(0, 5),
147                             textcoords='offset points',
148                             horizontalalignment='center',
149                             fontsize=10))
150         # observed FW head below
151         a.append(plt.annotate('%.1f'%res[j, 0], xy=(x, y), xytext=(0, -15),
152                             textcoords='offset points',
153                             horizontalalignment='center',
154                             fontsize=6))
155 ax.set_title('Freshwater Heads WIPP Area '+ year)
156 ax.set_xlabel('NAD27 NM East State Plane Easting (ft)')
157 ax.set_ylabel('NAD27 NM East State Plane Northing (ft)')
158
159 ### >>>>>>>manually fix labels>>>>>
160 for lab in a:
161     lab.draggable()
162 plt.show()
163 ### <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
164
165 plt.savefig('aser-area-contour-map'+year+'.png')
166 plt.close(1)

```