OW FLOAT SALINITY CALIBRATION TOOL
Version 1.1

By Breck Owens and Annie Wong
Last updated 30 March 2009

INTRODUCTION - 2 -

DESCRIPTION OF FILE STRUCTURE - 2 -

INSTRUCTION TO SET UP THIS SYSTEM - 3 -

HOW TO RUN THE CALIBRATION - 6 -

1. update_salinity_mapping.m - 6 -

2. set_calseries.m - 7 -

3. calculate_piecewisefit.m - 10 -

4. plot_diagnostics_ow.m - 11 -

Please send any comments to Annie Wong (awong@ocean.washington.edu) and Breck Owens (bowens@whoi.edu).

INTRODUCTION

This is a package of MATLAB routines for calibrating profiling float conductivity sensor drift. A description of the algorithms can be found in "An improved calibration method for the drift of the conductivity sensor on autonomous CTD profiling floats by θ-S climatology", by W.B. Owens and A.P.S. Wong, in Deep-Sea Research Part I: Oceanographic Research Papers, 56(3), 450-457, 2009.

DESCRIPTION OF FILE STRUCTURE

The top directory contains 5 files and 2 subdirectories.

The 5 files are:
1). README.pdf (this document);
2). README_prepare_refdbase_ow.pdf (document that explains how to prepare the ref dbase);
3). ow_config.txt (for pathway configuration and mapping parameter specification);
4). ow_calibration.m (a template for running this calibration system); and
5). mapug.html (Point 10 tells you how to install m_tbase in MATLAB).

The 2 subdirectories are:
1). /matlab_codes, which contains all the necessary MATLAB routines; and
2). /data, which contains all the necessary data to run the calibration.

The subdirectory /data is organised as follows:

/data/float_source/
contains .mat files with the original data from the floats.

/data/float_mapped/
contains .mat files with the objective estimates at the float profile locations and observed θ levels.

/data/float_calib/
contains .mat files with the calibration constants and error estimates.

/data/float_plots/
contains diagnostic plots from the calibration.

/data/constants/
contains coastdat.mat, wmo_boxes.mat, and TypicalProfileAroundSAF.mat.

/data/climatology/historical_ctd, /historical_bot, /argo_profiles
are where you put your reference data. Reference data are saved as .mat files in 10×10 degree WMO boxes. Please refer to README_prepare_refdbase_ow.pdf for their data format.

INSTRUCTION TO SET UP THIS SYSTEM

1. All files are to be used in MATLAB Version 7. In addition, you will need:
a). The MATLAB Optimization Toolbox;
b). The ITS-90 version of the CSIRO SEAWATER library, which you can obtain from http://www.cmar.csiro.au/datacentre/ext_docs/seawater.htm. Make sure you obtain the version that takes ITS-90 as temperature input.
c). The M_MAP toolbox, which you can obtain from http://woodshole.er.usgs.gov/operations/sea-mat, under "Mapping Tools". Read Point (10) in **mapug.html** to install **m_tbase.m** in MATLAB. This is used to calculate water depth for potential vorticity.

2. I suggest you place the OW package in a directory called OW. Unpack it. Then set your Matlab path to that directory OW and to the subdirectory OW/matlab_codes.

3. Put your reference data in /data/climatology/historical_ctd, /historical_bot, /argo_profiles.

The file /data/constants/**wmo_boxes.mat** is a summary of the availability of reference data in the /data/climatology directory. OW uses this file to check which WMO box has what data available. The 1st column in wmo_boxes.mat is the list of all WMO box numbers. The 2nd column denotes existence of CTD data. The 3rd column denotes existence of BOTTLE data. The 4th column denotes existence of Argo data. 0 = no data, or do not use. 1 = data exist, and use them. Edit the 2nd, 3rd and 4th columns after you have added the reference data. For example, if you do not want to use BOTTLE data, simply set the 3rd column to all 0s. Or if you have added Argo data to some WMO boxes and you want to use them, then change the corresponding entries in the 4th column to 1s. The order of the WMO boxes in the 1st column is fixed, so please do not change anything in the 1st column, or delete any rows.

4. After you have decided where you want to install the package on your computer, edit **ow_config.txt** at the following lines so the correct pathways are specified:

HISTORICAL_DIRECTORY =
FLOAT_SOURCE_DIRECTORY =
FLOAT_MAPPED_DIRECTORY =
FLOAT_CALIB_DIRECTORY =
FLOAT_PLOTS_DIRECTORY =
CONFIG_DIRECTORY =

5. The last section of ow_config.txt below the heading **"Objective Mapping Parameters"** is where you set the various parameters. Change these parameters to suit your region. If you have floats in more than one region, you may want to have several configuration files (e.g. one per ocean basin). Here are some tips on how to specify these parameters.

CONFIG_MAX_CASTS is the maximum number of historical casts used in objective mapping. I find that 300 casts are usually more than enough to produce a reliable objective estimate for a θ-S profile. More than 300 casts and the codes will take longer to run but you will not necessarily get more robust estimates. But play around with it.

MAP_USE_PV allows you to turn on/off the potential vorticity (PV) constraint in objective mapping. MAP_USE_PV = 0 means do not use PV constraint. MAP_USE_PV = 1 means use PV constraint. You should specify MAP_USE_PV = 1 in regions where you think the water mass distribution is constrained by bathymetry, e.g. the subpolar North Atlantic. Courtesy of L Bohme.

MAP_USE_SAF allows you to turn on/off the Subantarctic Front (SAF) separation criteria in objective mapping. MAP_USE_SAF = 0 means do not use the SAF separation criteria. MAP_USE_SAF = 1 means use the SAF separation criteria. You should specify MAP_USE_SAF = 1 when your float is near the Subantarctic Front in the Southern Ocean. That will select historical data from the same side of the SAF as your Argo profile for objective mapping. That is, it will prevent mixing historical data from north and south of the SAF. Courtesy of JB Sallee.

MAPSCALE_LONGITUDE_LARGE, MAPSCALE_LONGITUDE_SMALL, MAPSCALE_LATITUDE_LARGE, MAPSCALE_LATITUDE_SMALL are the spatial decorrelation scales (in degrees) used in the two stages of objective mapping. The large scales (×3) are also used to specify the ellipse used in selecting historical data. I suggest you start at a maximum scale of 8° and experiment on decreasing it to see what scales suit your region. You can also specify a circle instead of an ellipse. It's up to you.

MAPSCALE_PHI_LARGE and MAPSCALE_PHI_SMALL are the cross-isobath scales used in objective mapping if you use the PV constraint, i.e. if you specify MAP_USE_PV = 1. I don't know much about them, so you will have to read Lars Bohme's manuscript, BS(2005).

MAPSCALE_AGE (in years) is the temporal decorrelation scale used in the second stage of objective mapping. This is with respect to the ventilation age of the water masses. I assume most water masses in the main thermocline undergo decadal changes, so I usually specify MAPSCALE_AGE = 10. But if your region changes more frequently, then you will need to specify a shorter time scale. For example, in the subpolar North Atlantic, water masses change almost annually, so you may need to set MAPSCALE_AGE = 1 there. Exercise your scientific judgement.

MAP_P_EXCLUDE (in dbar) is used to exclude the shallow layers of the water column that are too variable for the mapping routine to produce reliable objective estimates; that is, the seasonal mixed layer! I usually set MAP_P_EXCLUDE = 200 for the subtropical gyre, then progressively go deeper towards higher latitudes. For example, in the Southern Ocean you may need to set MAP_P_EXCLUDE = 500 or deeper. Beware not to set MAP_P_EXCLUDE too deep unnecessarily. For example, in some parts of the subtropical gyre, the water column between 200-500 dbar has fairly small salinity variance on θ surfaces and is therefore good for calibration. If you set MAP_P_EXCLUDE too deep and end up excluding that part of the water column, you will be throwing away valuable calibration information. So, look at your float profiles carefully before deciding how much of the top layers you want to exclude in the mapping process. It is better to set MAP_P_EXCLUDE to a shallow depth, then refine your calibration range by using "use_theta_lt", "use_theta_gt", "use_pres_lt", "use_pres_gt" in calseries_*.mat. More on this on p.9–10.

MAP_P_DELTA (in dbar) is used to select historical data within +/− MAP_P_DELTA from float observed levels. That is, it specifies the vertical box from which historical data are selected for

objective mapping. This is important for regions with temperature inversions and weakly stratified temperature layers. You don't want to use historical data that come from a totally different depth range than the float observed θ level that you want to map to. I usually set MAP_P_DELTA = 250 for the subtropical gyre. That gives me a vertical box of 500 dbar to play with. I then decrease it poleward as the vertical temperature structure becomes weakly stratified. For example, in the Polar Frontal Zone in the Southern Ocean, I need to set MAP_P_DELTA = 50 to get a good objective estimate. Again, just like the mapping scales, you have to play around with this parameter.

6. If this is the first time you are using this system, then the 4 directories /data/float_source, /float_mapped, /float_calib, and /float_plots should be empty. Decide how you want to organise your floats, e.g. under different project names or different investigator names. Then make identical subdirectories under each of these 4 directories. For example:

/data/float_source/project_xx
/data/float_mapped/project_xx
/data/float_calib/project_xx
/data/float_plots/project_xx

7. For each float, put the original float data in matrix form, with each column being a profile, in chronological order (i.e. column 1 contains the first profile of the float, column 2 contains the second profile, etc.), in the following variable names:

LAT	(1×n, in decimal degrees, −ve means south of the equator, e.g. 20.5S = −20.5)
LONG	(1×n, in decimal degrees, from 0 to 360, e.g. 98.5W in the eastern Pacific = 261.5E)
DATES	(1×n, in decimal year, e.g. 10 Dec 2000 = 2000.939726)
	**Note that this date format is different from that used in the reference database.*
PRES	(m×n, in dbar, monotonically increasing; i.e. 1st element of the column is the shallowest pressure, and subsequent values are unique and increasing)
SAL	(m×n, in PSS-78)
TEMP	(m×n, in-situ ITS-90)
PTMP	(m×n, potential temperature referenced to zero pressure)
PROFILE_NO	(1×n, this can go from 0 to n−1, or 1 to n)

m = maximum number of observed levels from the float
n = number of profiles in the float time series

PROFILE_NO usually is the same as CYCLE_NO in the Argo netcdf files, but PROFILE_NO has to be unique. So for floats that report two cycle 0s, I suggest you either: (a) store cycle number in a variable called CYCLE_NO = [0, 0, 1, 2, 3, 4, ...] for your own record-keeping, then store PROFILE_NO = [1, 2, 3, 4, 5, ...] correspondingly for computation in this software; or (b) remove the first cycle 0 if it does not need calibration by this software.

Note also that if there are missing cycles in your float series, you can either create extra columns with NaNs to represent the missing cycles, or you can just leave them out. For example, if a float is missing cycle 4 from a 7-cycle series, then you can just have PROFILE_NO = [1, 2, 3, 5, 6, 7] and other matrices will have the corresponding 6 entries.

Fill up the extra spaces in the columns with NaNs to make up the matrices. Bad data should also be denoted by NaNs. In particular, values in PRES have to be distinct and monotonically increasing. Save the matrices in a .mat file in MATLAB in the appropriate subdirectory in /data/float_source/. There should be one .mat file for each float. For example,

/data/float_source/project_xx/float0001.mat
/data/float_source/project_xx/float0002.mat
/data/float_source/jones/myfloat_a.mat
/data/float_source/jones/myfloat_b.mat

HOW TO RUN THE CALIBRATION

Open MATLAB in the top directory. List all the float files in a cell array "float_names", with the corresponding subdirectories in another cell array "float_dirs". For example,

float_dirs = { 'project_xx/'; 'project_xx/'; 'jones/'; 'jones/' };
float_names = { 'float0001'; 'float0002'; 'myfloat_a'; 'myfloat_b' }.

Tips: If the files are not saved under a subdirectory and are only saved under ./float_source/, specify float_dirs = { ''; ''; ''; '' }, etc.

Run **ow_calibration.m**. The specifications in ow_config.txt will be loaded. Then it will run four main functions in the following order:

1. update_salinity_mapping.m

update_salinity_mapping.m uses wmo_boxes.mat and the reference database in /data/climatology/, together with the objective mapping parameters specified in ow_config.txt, to map reference salinity to float profile locations and observed θ levels. The output is saved in /data/float_mapped/"float_dirs"/map_"float_names".mat. The output variables are:

la_mapped_sal	(objectively mapped reference salinity)
la_mapsalerrors	(objective mapping errors in salinity)
la_ptmp	(float PTMP at levels where objective mapping is done)
la_noise_sal	(noise variance of salinity in the reference data)
la_signal_sal	(signal variance of salinity in the reference data)
scale_lat_large	(large latitudinal scale used in objective mapping)
scale_long_large	(large longitudinal scale used in objective mapping)
scale_lat_small	(small latitudinal scale used in objective mapping)
scale_long_small	(small longitudinal scale used in objective mapping)
scale_phi_large	(large cross-isobath scale used in objective mapping with PV)
scale_phi_small	(small cross-isobath scale used in objective mapping with PV)
scale_age	(temporal scale used in objective mapping)

use_pv (1 means PV is used, 0 means PV is not used, in objective mapping)
use_saf (1 means SAF separation is used, 0 means SAF separation is not used)
p_delta (reference data from within +/- p_delta of float obs are used in mapping)
p_exclude (the top p_exclude dbar of the water column is excluded in mapping)
la_profile_no (same as PROFILE_NO)
selected_hist (positions of reference data used in mapping, and corresponding profile no)

Note (a): Objective mapping is done on every float observed θ level. For Iridium floats, users should subsample the vertical profiles to reduce the number of vertical levels, so that the objective mapping routine doesn't spend time mapping 500+ levels!

Note (b): You should always check "la_mapped_sal" against "SAL" to make sure the objective salinity estimates are realistic. This is where you can experiment with the various mapping parameters.

Note (c): update_salinity_mapping.m does not re-calculate existing profiles. It updates the matrices with new profiles by comparing /float_source/filename and /float_mapped/map_filename. Because of this, if you are experimenting with different maps, you have to manually remove the relevant profiles from /float_mapped/map_filename before news maps will be calculated.

Note (d): The codes are currently written so that if you have included Argo data in the reference database, they will exclude the Argo float that is analysed from being selected as historical data for objective mapping. This is done by comparing "float_names" with "source" in the ref dbase. For example, if you enter float_names = { 'annie49999' } in ow_calibration.m, it will exclude all Argo profiles with the string '49999' in the "source" variable. Please refer to README_prepare_refdbase_ow.pdf on how to specify "source" in the ref dbase.

2. set_calseries.m

set_calseries.m sets the default values for 8 variables that can be changed manually for calibration. The output is saved in /data/float_calib/"float_dirs"/calseries_"float_names".mat. The output variables are:

calseries (tags used to split series and isolate bad points - same as "cal_series_flags" in WJO)
breaks (index in the series where you think your breakpoints should be) **Note this is the index, which is different from the profile number if there are missing cycles.*
max_breaks (maximum number of breakpoints allowed in the piecewise linear fit)
use_theta_lt (the upper bound of θ used in the calibration)
use_theta_gt (the lower bound of θ used in the calibration)
use_pres_lt (the upper bound of pressure used in the calibration)
use_pres_gt (the lower bound of pressure used in the calibration)
calib_profile_no (same as PROFILE_NO)
use_percent_gt (percentage of valid measurements required on a level for it to be included in the calibration)

The tags in "calseries" are used for grouping various profiles into separate unique cal series for calibration, and to exclude unsuitable profiles. The codes will form separate unique cal series by picking up profiles with the same tags in "calseries". The default tag is 1. If you feel that certain profiles should form another series, or an unsuitable profile should be excluded, or there has been a calibration jump (i.e. discontinuity), then manually change the tag to another number other than 1. Similarly the series should be split when there have been so many missing cycles that you believe the sensor trend is no longer continuous. The piece-wise linear fit will fit each unique cal series with continuous segments (i.e. where the end-points are joined). Lastly, set_calseries.m will automatically tag any profile with PRES = NaN or TEMP = NaN or SAL = NaN with calseries = 0. These bad profiles are skipped and are not included in any unique cal series.

Example 1. If the 5th profile needs to be excluded in a 10-profile series, set "calseries" to 1 1 1 1 2 1 1 1 1 1.

Example 2. If after the 6th profile, there is a sudden jump and you want to split the series into two separate cal series for calibration, then set "calseries" to 1 1 1 1 2 3 3 3 3 3.

Example 3. If the 10th profile has all bad SAL, then set_calseries.m will automatically tag the 10th profile with 0. i.e. "calseries" = 1 1 1 1 2 3 3 3 3 0. The 10th profile will be automatically excluded from all unique cal series, ie. 0 is not counted as a unique series.

The two variables "breaks" and "max_breaks" are used to specify breakpoints in the piece-wise continuous linear fit. The default is to look for a maximum of 4 breakpoints. If you are not happy with the optimal breaks, you can specify your own.

Example 4. breaks = [ ]; max_breaks = 4;
These are the defaults. This means let the optimization program find whatever breakpoints it thinks the series should have, with a maximum of 4 breakpoints. This means the solution can have 0, 1, 2, 3, or 4 breakpoints.

Example 5. breaks = [ ]; max_breaks = 0;
This means fit the series with one linear line with no breakpoints.

Example 6. breaks = [ ]; max_breaks = X;
This means let the optimization program find whatever breakpoints it thinks the series should have, with a maximum of X breakpoints. The solution can have 0, 1, 2, ... or X breakpoints.

For m>1 unique cal series, you can specify "max_breaks" as a 1×m or m×1 vector, with each element applying to each corresponding unique cal series, sorted in ascending order of the tags. If you only specify one value, that will be applied to every unique cal series.

Example 7. breaks = [ ]; max_breaks = [3, 4] or [3; 4]; a 10-profile series has "calseries" set to 8 8 8 8 5 0 5 5 5 5. This means up to 3 breakpoints (ie. the 1st element in "max_breaks") will be fitted to

the unique cal series with tag = 5 (ie. the 1st unique series after sorting); up to 4 breakpoints (ie. the 2nd element in "max_breaks") will be fitted to the unique cal series with tag = 8 (ie. the 2nd unique series after sorting). Note that profiles with tag = 0 are ignored. This example is to illustrate that the tags do not have to follow any numerical order. They are just tags. But if you use "max_breaks" in conjunction with multiple series, you will have to be aware of their order.

To specify breakpoints, specify "breaks" as a m×n matrix, where m = number of unique cal series, and n = number of specified breakpoints for each unique series, with extra spaces filled by NaNs. The rows "m" need to correspond to the order of the unique cal series sorted in ascending order. Within each unique calseries, the value of "max_breaks" must then be greater than or equal to the number of specified "breaks". When the value of "max_break" is equal to the number of specified "breaks", a straight line will be fitted between each specified "breaks". When the value of "max_break" is greater than the number of specified "breaks", a straight line will be fitted between each specified "breaks", and the codes will look for additional breakpoints beyond the last specified "breaks".

Example 8. breaks = [4, 16, 42]; max_breaks = 6; there is only 1 unique cal series. The codes will fit a straight line between profiles 1 and 4, 4 and 16, 16 and 42, then look for more breakpoints beyond profile 42. Note here "breaks" is a 1×3 row vector, meaning 1 unique cal series, 3 specified breakpoints. The value of "max_breaks" is 6, which is greater than the number of specified "breaks" (= 3).

Example 9. breaks = [3, 5; 10, NaN]; max_breaks = [2, 4] or [2; 4]; a 14-profile series has "calseries" set to 1 1 1 1 1 1 1 1 2 2 2 2 2 2. This means the codes will fit a straight line between profiles 1 and 3, 3 and 5, 5 and 8 (8 is the last element of the unique series with tags = 1). The codes will fit a straight line between profiles 9 and 10, then look for additional breakpoints between profiles 10 and 14 (9 and 14 are the first and last elements respectively of the unique series with tags = 2). Note that the value of the 1st element of "max_breaks" is 2, which is equal to the number of specified "breaks" for the unique series with tags = 1. The value of the 2nd element of "max_breaks" is 4, which is greater than the number of specified "breaks" for the unique series with tags = 2.

The four variables "use_theta_lt", "use_theta_gt", "use_pres_gt", "use_pres_lt" are used to refine the theta range and/or the pressure range used in the calibration. These four variables are applied to each unique cal series. Their default values are [ ], which means use the entire water column below "p_exclude" to choose 10 float levels for the piece-wise linear fit. The 10 float levels are θ levels with the minimum salinity variance.

Example 10. When only "use_theta_gt" is specified (e.g. use_theta_gt = 5; use_theta_lt = [ ]), then it is used as a one-sided limit; that is, only measurements with $\theta > 5°C$ are used.

Example 11. When only "use_theta_lt" is specified (e.g. use_theta_gt = [ ]; use_theta_lt = 16), then it is used as a one-sided limit; that is, only measurements with $\theta < 16°C$ are used.

Example 12. When both "use_theta_gt" and “use_theta_lt” are specified and use_theta_gt < use_theta_lt, (e.g. use_theta_gt = 5; use_theta_lt = 16), then they are used as a two-sided limit to specify that measurements in the band 5°C < θ < 16°C are used.

Example 13. When both "use_theta_gt" and “use_theta_lt” are specified and use_theta_gt > use_theta_lt, (e.g. use_theta_gt = 16; use_theta_lt = 5), then measurements in the band 5°C < θ < 16°C are excluded.

Example 14. “use_pres_gt” and “use_pres_lt” are used in similar manners, except the values specified are in dbar and not in °C. All four variables “use_theta_gt”, “use_theta_lt”, “use_pres_gt”, “use_pres_lt” can be used together.

The variable “use_percent_gt” is used to specify the percentage of valid measurements required on a level for it to be included in the 10-level selection. The default value is 0.5 (= 50%); that is, only levels with percentage of good data greater than 50% are included. This default is set at Line 35 in set_calseries.m.

Example 15. You can increase this value to favour levels that have a greater number of good measurements. On the other hand, there are also instances where you may want to decrease this value. For example, the old U Washington floats sample deep every 4th profile. These deep levels often tap into homogeneous water masses, and so are exactly the ones I want to use in the calibration. So for those old UW floats, I would set use_percent_gt = 0.25 (=25%).

3. calculate_piecewisefit.m

calculate_piecewisefit.m uses the variables in the float_calib/calseries_.mat files to choose 10 float θ levels below “p_exclude”, then computes the piece-wise linear fit between the float values in the float_source files and the objectively estimated values in the float_mapped files. The output is saved in data/float_calib/"float_dirs"/cal_"float_names".mat. The output variables are:

PROFILE_NO	(same as before)
cal_COND	(calibrated float potential conductivity)
cal_COND_err	(estimated calibration error for float potential conductivity)
cal_SAL	(calibrated float salinity)
cal_SAL_err	(estimated calibration error for float salinity)
pcond_factor	(multiplicative calibration factor in potential conductivity - same as "condslope" in WJO)
pcond_factor_err	(estimated error for pcond_factor - same as "condslope_err" in WJO)
time_deriv	(time derivative of the potential conductivity correction)
time_deriv_err	(estimated error associated with time_deriv)
sta_mean	(1-1 profile by profile fit of pcond_factor = weighted sum of observed conductivity ratios, where the weights are the inverse of the estimated error variance)
sta_rms	(rms differences for each station of the observed conductivity ratios)

sta_SAL	(1-1 profile by profile estimate of the adjusted salinity)
sta_SAL_err	(error based on sta_rms for sta_SAL)
fcoef	(coefficients for the fit)
fbreaks	(breaks for the fit. This allows the user to exactly specify the breaks at a future analysis rather than trying to visually pick it off a plot.)

The variable "cal_SAL" is the calibrated salinity you want. "cal_SAL_err" is the associated errors. To compare the original float salinity with the calibrated float salinity at profile i, simply compare cal_SAL(:, i) with SAL(:, i).

Note that within each unique cal series, when there are fewer than 5 profiles, a mean offset will be fitted. That means when there is only 1 profile, a fit will still be produced.

Tips: Line 173 in calculate_piecewisefit.m that calls the routine build_ptmp_cov.m is a time consuming step. When experimenting with different break points, it will speed up the process if you use Line 178 to save the covariance matrix during the initial run. Then when you re-run the fitting, comment out Line 173 & Line 178, and use Line 179 to load the covariance matrix. There should be one covariance matrix per unique cal series.

4. plot_diagnostics_ow.m

plot_diagnostics_ow.m produces 8 diagnostic plots. These are saved in /data/float_plots/"float_dirs"/"float_names"_1-8.eps. No plots will be produced when there is no data in the float_source matrices, or when there is no output from update_salinity_mapping.m. Plots 3 to 8 will not be produced when there is no output from calculate_piecewisefit.m.

Note that these diagnostic plots are only meant to help you evaluate the suggested salinity drift calibration. They are not sufficient to check point-wise measurements, or other instrument errors such as temperature drift or pressure drift.

_1.eps plots the location of the float profiles and the reference data selected for mapping. Use this figure to check that no inappropriate historical data are used.

_2.eps plots the original float salinity and the objectively estimated reference salinity at the 10 float θ levels that are used in calibration, assuming there is no split series.

_3.eps plots the evolution of the suggested adjustment with time. The top panel plots the potential conductivity multiplicative adjustment. The bottom panel plots the equivalent salinity additive adjustment. The red line denotes one-to-one profile fit that uses the vertically weighted mean of each profile. The red line can be used to check for anomalous profiles relative to the optimal fit.

_4.eps plots the calibrated float salinity and the objectively estimated reference salinity at the 10 float θ levels that are used in calibration, assuming there is no split series.

_5.eps is Brian King's salinity anomaly plot, based on original float salinity. Generally the plot will be noisy in the seasonally-varying upper ocean. In the main thermocline, the signal will be noisy or stable depending on the homogeneity of that water mass at scales sampled by the float. Salinity on θ levels will vary either (a) because of genuine changes of water mass properties observed as the float migrates or as the ocean changes with time, or (b) because of sensor drift. This plot helps to distinguish between (a) or (b). In (a), genuine changes of water mass properties will be seen as a shift in salinity anomaly at some levels only. In (b), sensor drift will be seen as a change in salinity anomaly at all levels, i.e. an apparent shift by the same amount (or a systematic bias) in several different water masses. Courtesy of B King. DO NOT USE THIS PLOT WHEN THE PROFILE HAS TEMPERATURE INVERSIONS!

_6.eps plots the evolution of salinity with time along two selected θ levels with minimum salinity variance, assuming there is no split series. If you want to look at other selected θ levels, go to Line 383 in plot_diagnostics_ow.m, then specify "tplot" = any numbers between 1 and 10. e.g. tplot = [1:10] will plot time series on all 10 selected θ levels.

_7.eps is Brian King's salinity anomaly plot, based on calibrated float salinity. DO NOT USE THIS PLOT WHEN THE PROFILE HAS TEMPERATURE INVERSIONS!

_8.eps gives an idea of salinity variances on θ levels, assuming there is no split series. It also shows the 10 float θ levels that are chosen for calibration. You can use this plot to refine your calibration range by changing the variables "use_theta_gt", "use_theta_lt", "use_pres_gt", "use_pres_lt", "use_percent_gt" in /float_calib/calseries_*.mat. Courtesy of Paul Robbins.

--- Last updated March 2009, Annie Wong ---