sage V2

Sage V2 is a graphical aid to help visualize and quality control (QC) nitrate and pH data from BGC Argo profiling floats. It was written in MATLAB’s App Designer environment using MATLAB version 9.14.0.2337262 (R2023a) Update 5 on and Microsoft Windows 11 Pro Version 10.0 (Build 22631).

Highlights of V2 include:

Directly import Sprof netCDF files in addition to MBARI ODV style text files for QC

Automatically determine correction nodes using BIC as a guide (in sage V1.0)

Calculate, visualize and apply pH pump offset correction during QC

Eliminated use of external toolboxes for plotting, netCDF extraction (m\_map, nctoolboc, nansuite)

Updated QC List file builder. Includes meta data about QC for each parameter

I also learned that Matlab is moving its GUI framework from Java based to an HTML5-class-base framework.

**When sage\_V2 is first executed a startup function is called (StartupFcn) which does a bunch of initializations tasks. Much of this is done by the stand-alone function sageV2\_init.m which is called in StartupFcn**

* Perform non-float dependent tasks upon initial start of GUI:
* Set color maps & default colors
* Store original axis sizes to aid re-sizing (this may not be as helpful aa I first thought)
* Check to see if **ischange.m** is a recognized function in the Matlab version being used
* Run **sageV2\_init.m** to:
  + Set directory paths and add them to the matlab path
  + Set default float data file search path (Sprof or ODV)
  + Load MBARI master list & bottle look up table if appropriate (MBARI)
  + Store original axis sizes to aid re-sizing (this may not be as helpful as I first thought)
  + More initialization can be added here like installing ref functions / data if they don’t exist

**When a float is loaded with the select file button this triggers the GUI callback SelectfileButtonPushed and executes two stand-alone functions to prep for QC’ing a float (get\_sageV2\_float\_data.m , get\_sageV2\_ref\_data.m) and checks to see if there are non-bad data to QC (NO3 or PH) before continuing. If all is good four GUI functions are executed to finalize the loading and prep for QC and populating the GUI plots (init\_states, update\_anom, update\_glodap, update\_plots**

**get\_sageV2\_float\_data.m – load float data**

* Define variables to keep from Sprof or ODV txt files (MBARI)
* Check file type (Sprof or MBARI ODV style text file)
* Try and get “CONFIG\_CPAct” values for each profile
  + For Sprof files get meta file for float from the GDAC and look there
  + For MBARI ODV files look in \*.mat files for float
* Build data matrices for the GUI
* Do some data clean up to improve performance
* Only using raw P,T,S for now in GUI but this could be updated
* Build a track matrix & get min and max pressures for each profile, cycle range
* Calculate pH pump offsets and offset corrected pH if applicable and add to data matrix
* Assign function outputs to: **app.hdr**, **app.data**, **app.finfo, app.fI**, **app.pH\_PO**
* **app.hdr** column ID headers for **app.data**
* **app.data** is a matrix of float data with parameters defined by **app.info.keep\_vars** and the columns can be accessed using the indices in the structure **app.fI**
* **app.finfo** contains meta, logical & range info for the loaded float with the following fields:
  + **Log** – a log showing any issues during the float loading process
  + **keep\_vars** – the subset of Argo parameters to retain in sageV2
  + **fn**, **source** – the data file name and location
  + **ftype** – ODV or SPROF
  + **QFbad** – bad QF value ( 4 for Argo, 8 for ODV)
  + **wmo**
  + **CpActP** – Constant profiling activation pressure, n x 2 array [cycle, CpActP]
  + **ftrack\_hdr, ftrack** – [SDN, Cycle, Lon, Lat, Pmin, Pmax] & float track data matrix
  + **CycleRange**, **PminRange**, **PmaxRange** – various float ranges
  + **I** – column index structure for **app.data** (extracted float data)
  + **tf\_O2**, **tf\_NO3**, **tf\_pH**, **tf\_CpAct** – logical true or false scalars
  + **MBARI\_ID** – MBARI float name if applicable
* **App.pH\_PO** – ph pump offset data structure which uses **calc\_pH\_pump\_offset.m** to calculate the offsets:
  + **pH\_PO.hdr** = ['cycle', 'CpActP', 'spline offset', 'spline SSR', 'resid std', 'Zish', 'lin offset', 'lin diff', 'lin\_resid std', 'lin Zish'};
  + **pH\_PO.data** – cycle matrix identified by header and length of cycles
  + **pH\_PO.pH** – pump offset corrected pH using the defined method **pH\_PO.pH\_offset\_type**. Either ‘linear’ or ‘poly’ & defined in **get\_sageV2\_float\_data.m**

**get\_sageV2\_ref\_data.m – load various reference data sets**

* Define master list of reference data sets – adjust as desired  
  ('CANYONB' 'ESPER\_NN' 'ESPER\_LIR' 'ESPER\_MIX' 'ESPER\_MIX\_NoO2' 'WOA')
* Load CANYON-B for float profiles – need corrected O2
* Load ESPERS for float profiles – need corrected O2 & sea water depth to do this (we use **sw\_dpth.m**)
* Load World Ocean Atlas 2023 for float profiles – this requires **get\_WOA23\_local.m**

**Load QC history for float including bad sensor and bad sample lines if available**

**Init\_states** – GUI function that executes only once when a float is loaded

* Enable all buttons, panels & checkboxes
* Clear axes and their dependents in all tabs – (i.e. clear plots before loading a new float in same GUI instance)
* Set initial cycle & pressure ranges
* Set default radio buttons
* Use loaded float data to fine tune pressure & cycle ranges
* Add QC list folder to dirs structure
* Get QC list data (**get\_QCA.m**) and add to app table if it exists (**update\_table**)
  + App.QCA – from QC list file, never modified
  + App.CGOD – modified by GUI operations
* Get bottle data for 1st profile if it exists (**get\_shipboard\_dataV2.m**)
* Build initial parameter state structures – **app.state.(param)**   
  These store or “remember” settings for each parameter in the GUI where param can be Nitrate, pH, Oxygen, Temperature, Salinity.  
  **app.state.(param)** has the following fields:
  + Cycles cycle range selected
  + Surface surface pressure range
  + Deep deep pressure range
  + Profile profile pressure range
  + Ref selected reference used
  + CGOD current QC adjustments applied [ Cycle Gain Offset Drift]
* Build QC\_info structure skeletons for NO3 & pH with the following fields:   
  Operator, Date, Type, Ref, PRange, NodeCount, BIC
* Check for all bad NO3, pH and CpAct & turn off buttons if need be
* Add pH pump offset data to pdata structure (plot data structure - more below)

**update\_anom** – GUI function that updates float-reference anomalies given changes in reference choice or pH flavor (pH pump offset corrected or not)

**update\_glodap** – update GLODAP data grab based on change in cross-over range. This uses **get\_GLODAPv2\_local\_jp.m** and a GLODAP mat file to get the data

**update\_plots** – update plots in tabs based on selected information

* **update\_plot\_data** is executed and the **app.pdata** plotting structure is populated with a data subset and relevant information for a given parameter
* Get plot type parameter and the parameters state structure
* Define logical arrays for sub setting (Pressure, cycle, QF)
* Do the same for pH pump offset corrected data if needed
* Populate the **pdata** subset structure:
  + STA = sample cycle number
  + SDN = sample Matlab serial date number
  + P = sample pressure
  + cgod = cycle gain offset drift structure
  + RD = Raw data for selected parameter
  + AD = Adjusted data for selected parameter
  + REF = selected reference data
  + RDA = float – raw data anomaly for selected parameter
  + ADA = float – adjusted data anomaly for selected parameter
  + mRDA, mADA = cycle means for RDA, ADA
  + PO\_pH = n x 2 matrix of raw pH & pump offset corrected raw pH
  + PO\_data = cycle pH pump offset estimate matrix
* Plot the data (deep, shallow or profile)
* If relevant plot pH pump offset data in pump offset tab (**update\_pHOffsetPlots**)

When correcting data there are three options:

1. Auto change point detection & correction
2. Manual node entry & automatic correction
3. Individual table entry & correction using the manual entries