Seaglider Webinar Series pt. 3

Basestation3 CLI and customization

+

Glider compass calibration

IOP Seaglider Group
APL-UW
26 June 2024

News

- We have created a new resources page to serve as a single source for documentation and common links

  https://iop.apl.washington.edu/iopsg

Basestation3 CLI primer

- Basestation3 is python based. A standard installation consists of:
  - Script files located in /usr/local/basestation3, e.g. /usr/local/basestation3/NewMission.py
  - Python3 installation located in /opt/basestation
    - We install our own version of python3 to enforce version compatibility and the availability of the required modules
- Running scripts requires ssh shell access to the basestation server. You must have login access and be able to run shell commands.
- To run a script, use the full paths to get the correct versions:
  - /opt/basestation/bin/python /usr/local/basestation3/NewMission.py ./ MyNewMission
- Most scripts require arguments and options. All scripts will report the available options and required arguments when given the --help option, e.g.:
  - /opt/basestation/bin/python /usr/local/basestation3/NewMission.py --help
- Help for installation and common tasks is available with the source on github
  - https://github.com/iop-apl-uw/basestation3

Common command-line tasks

There are numerous scripts in /usr/local/basestation3, but most are not meant to be run by users from the command-line. Typical scripts that you might use include:

- Commission.py: Set up the user account for a new glider. Run only once per glider. IOP takes care of this for seaglider.pub users. Requires sudo / root privileges.
- NewMission.py: Set up the mission directory with symbolic links for a new glider mission. Use this to run missions from a mission sub-directory while the mission is active. If NewMission.py is used, do not use MoveData.py after the mission.
- MoveData.py: Move mission data from glider home directory to an archive sub-directory after mission is complete. Use this when you did not use NewMission.py prior to the mission. Remember to update missions.yml with the new path.
- cmdedit (also sciedit, targedit): Make changes to cmdfile, science, targets, validate the changes, and log the changes. Set EDITOR environment variable to your preferred editor or defaults to vi.
- Reprocess.py: Reprocess data. This script has many options - see the help information for details.
    - /opt/basestation/bin/python /usr/local/basestation3/Reprocess.py -m ./ --force --reprocess_plots
- RegressVBD.py: Run VBD regressions (model fits for C_VBD, HD_A, HD_B) over multiple dives. Generates an html file with plots when run from command-line.
- Magcal.py: Multi-dive magnetic calibration. Generates an html file with plots when run from command-line.
- vis.py: Start a local or public instance of visualization server or test changes to missions.yml.
- BasePlot.py: Generate plots from dive and mission data. Useful sometimes for re-processing.
- SelftestHTML.py or selftest.sh: Analyze and summarize pre-launch selftest results.

Most scripts should be run from a glider's home directory or current mission directory.

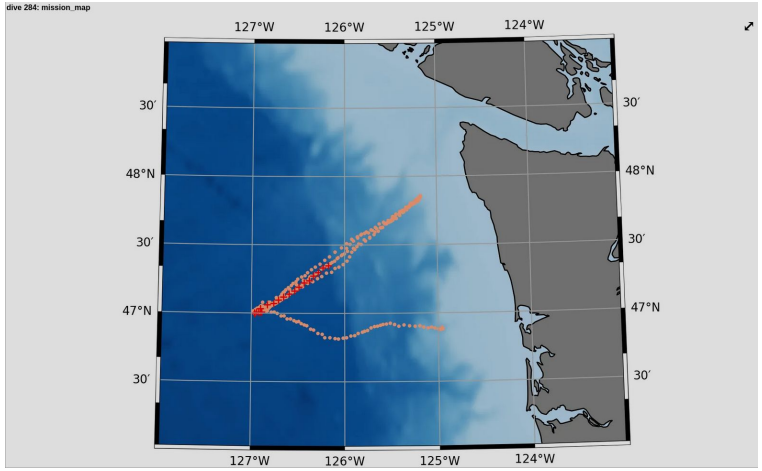
Customizing automated processing on basestation3

- missions.yml (basestation or group wide): control the missions and map options displayed on vis
- pagers.yml (basestation wide, group wide, per glider): control notifications (email, SMS, etc.) for glider events
- sections.yml (per glider): customize the section plots that are generated for the vis plot ribbon
- .login, .logout (per glider): change default processing options (not recommended, use sgNNN.conf instead)
- sgNNN.conf (per glider): specify command-line options to override defaults during automated processing
- /usr/local/basestation/glider_login, /usr/local/basestation/glider_logout (basestation/jail wide)
    - Change default processing options
    - Note “basestation” not “basestation3”
- Data distribution/notifications (basestation wide, group wide, per glider)
    - .ftp, .sftp
    - .mailer
    - .urls
- .extensions (basestation wide, group wide, per glider): specify additional processing modules (e.g., DAC submission)
- sg_plot_constants.m (per glider): set the limits for the static map
- Hook scripts (per glider). These are scripts that can be written in any language, usually shell, and be used for additional custom processing (inserting into or modifying netcdf files for example).
    - .pre_login (must run very fast)
    - .post_dive (runs when per dive files are finished creation (eng, log, nc, etc.))
    - .post_mission (runs when whole mission products have been updated (KML, mission timeseries, etc.))

Examples of many of these files can be found in /usr/local/basestation3/sg000/

sections.yml

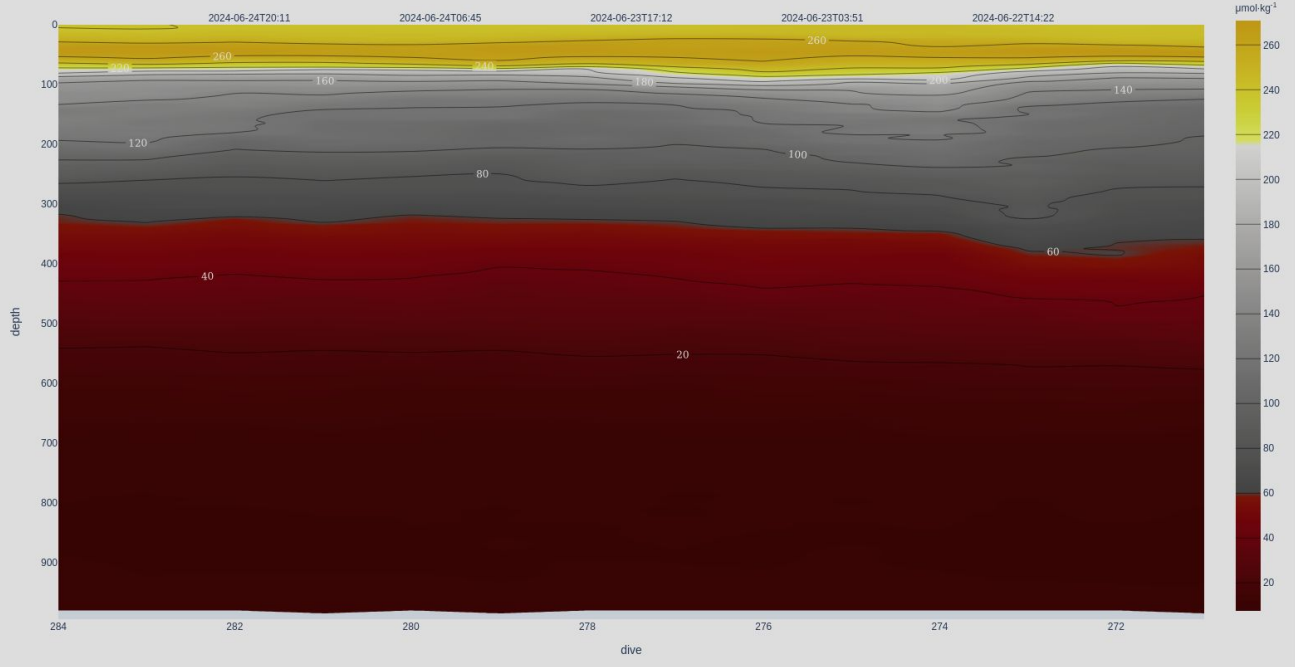
```
---
defaults:
  top: 0
  bottom: 990
  bin: 5    ← bin size (meters)
  step: 1   ← dive increment
  which: 4 ← which profiles (1=dive, 2=climb, 3=both, 4=combine, default is 4)

variables:
  temperature: { colormap: thermal, units: '&#8451;', min: 4, max: 15 }
  salinity: { colormap: haline }
  wlbb2fl_sig470nm_adjusted: { top: 0, bottom: 150, colormap: algae, units: 'm<sup>-1</sup>sr<sup>-1</sup>' }
  wlbb2fl_sig695nm_adjusted: { top: 0, bottom: 150, colormap: algae, units: '&mu;g&#183;l<sup>-1</sup>' }
  wlbb2fl_sig700nm_adjusted: { top: 0, bottom: 150, colormap: algae, units: 'm<sup>-1</sup>sr<sup>-1</sup>' }
  aanderaa4831_dissolved_oxygen: { top: 0, bottom: 1000, colormap: oxy, units: '&mu;mol&#183;kg<sup>-1</sup>' }

sections:
  "000": {start: 1, stop: -1} ← this will be the whole mission
  "001": {start: 1, stop: 56}
  "002": {start: 57, stop: 93, flip: true} ← flip means reverse the x-axis
  "003": {start: 94, stop: 168}
  "004": {start: 169, stop: 208, flip: true}
  "005": {start: 209, stop: 270}
  "006": {start: 271, stop: -1, flip: true} ← this is the latest section
```



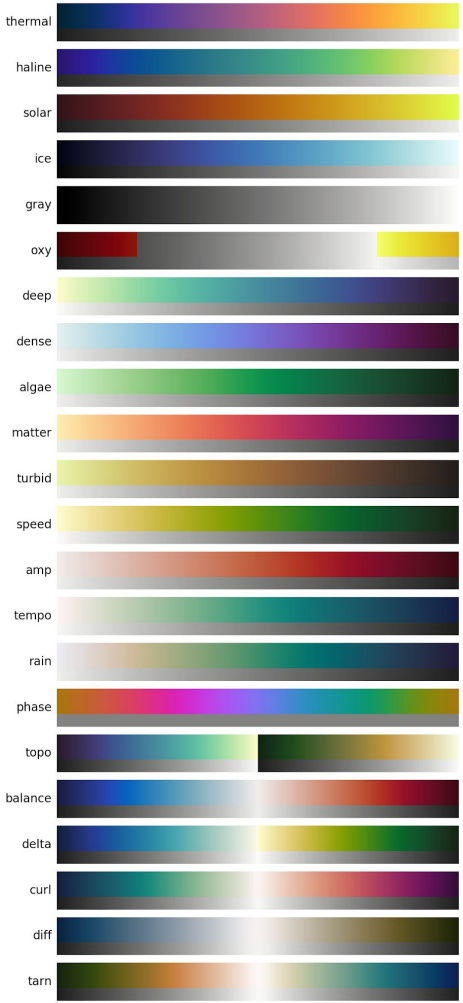
cmocean colormaps: https://matplotlib.org/cmocean/

Use sectionNumber=(normal or reverse) and sectionSort=(number or name) in the vis URL to change the sort order on the plot ribbon.

dive 284: aanderaa4831_dissolved_oxygen_section_006



diveplot…vert_vel_regression | pitch…ctw | mission_map…aa4831 | legato…ts_raw | ab_dives…commlog | depthangle…pitch_roll_motors | vbd_effic…vbd_motors | sections


thermal
haline
solar
ice
gray
oxy
deep
dense
algae
matter
turbid
speed
amp
tempo
rain
phase
topo
balance
delta
curl
diff
tarn

Useful vis API endpoints (URL queries)

Use with curl, wget, browser, or from another web app or script

- /data/nc/GGG/DDD
  - Downloads netcdf file for glider GGG dive number DDD
- /kml/GGG
  - Download latest glider KML
- /kml/GGG?kmz
  - Download latest glider KML in compressed KMZ format
- /kml/GGG?network
  - Use this as the URL in google earth when adding a network location (update interval 2 minutes)
- /pos/poll/GGG
  - Latest position in JSON format
- /pos/poll/GGG?format=csv
  - Latest position in CSV format
- /query/GGG/var1,var2,var3…
  - query for per dive variables, results in JSON format; see the plot tool for variable names
- /query/GGG/var1,var2,var3…?format=csv
  - e.g., /query/249/dive,log_gps_time,log_gps_lat,log_gps_lon?format=csv
- See docs/vis.txt in the basestation sources for the complete list of endpoints and available parameters

GGG=glider number, DDD=dive number

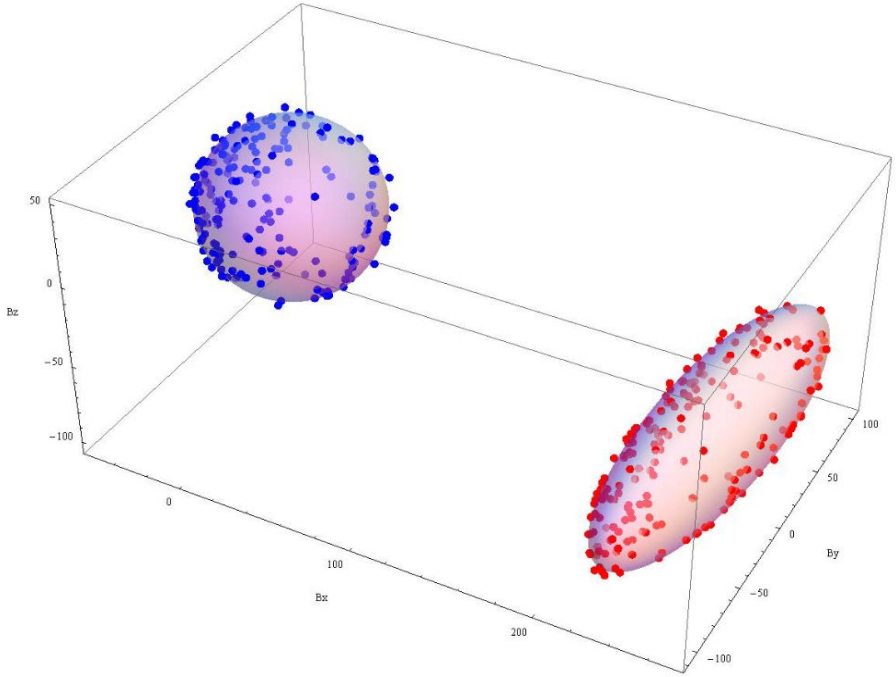
Reminder that any of these can be user/group/password protected on a per site or per mission basis.

Compass calibration 1

- The glider uses a 3D accelerometer and 3D magnetometer to compute heading.
  - We use the accelerometer to calculate pitch and roll
  - The magnetometer measures 3 axes of magnetic field in a local coordinate system (x,y,z). We use the compass pitch and roll to rotate that local field into earth coordinates (X,Y,Z).
  - The field we measure is a the sum of earth's field (which is what we want to measure to calculate heading) and hard and soft iron effects.
- Compass calibration is the process of removing hard and soft iron effects from the compass magnetometer data so that we can isolate just the earth field (X_e, Y_e) and calculate heading
  - Heading = $\tan^{-1}(Y_e/X_e)$
- Hard iron effects come from materials with their own local magnetic fields (like steel in the battery cell casings).
  - They are fixed in the local coordinate system of the compass, i.e., they move and rotate with the glider in the same way that compass does.
  - Hard iron manifests as an offset in the measured magnetic field.
- Soft iron comes from materials which produce a magnetic field in the presence of another magnetic field.
  - Soft iron effects are not fixed in the local coordinate system.
  - They distort the measured field through stretching and rotation.
- Hard iron is usually (but definitely not always) a larger source of error on the glider.
- Some form of compass calibration should be applied for every mission.
  - Compass calibration can be critical for roll trimming
  - Good heading data is required to calculate depth-averaged current (and the nav modes that rely on it)
  - Compass calibration can change even with no changes on the glider (i.e., even without changing batteries).

Compass calibration 2

- In an environment with no hard or soft iron and a glider that is spinning in circles, pitching up and down, and rolling side to side, the glider magnetic field data would map out a sphere as the (x,y,z) field values at each measurement point would represent a vector equal in magnitude to the the earth's field intensity at that location.
- When rotated to earth coordinates (X,Y,Z), the X,Y components would form a circle centered at (0,0)
- Hard iron effects offset that sphere and circle away from the origin.
- Soft iron effects turn that sphere into an ellipsoid.



Compass calibration 3

- In the calibration we represent the hard iron as an offset vector (p,q,r) and the soft iron as a 3x3 transformation matrix (a,b,c,d,e,f,g,h,i)

$$\begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x-p \\ y-q \\ z-r \end{bmatrix}$$

  - In tcm2mat.cal
    - hard0="p q r"
    - soft0="a b c d e f g h i"
  - Hard iron is easier to calculate numerically. Sometimes the solution for soft iron will not converge and we only get a hard iron result.
  - In our perfect world example, hard iron = [0,0,0] and soft iron is the identity matrix.

  - Within IOP we prefer an in situ calibration:
    - Accounts for hard iron picked up during transit or storage
    - Saves time and cost associated with stand or fixture based "whirly" calibration procedures performed on land
    - No absolute heading reference required

In situ compass calibration techniques on Seaglider 1

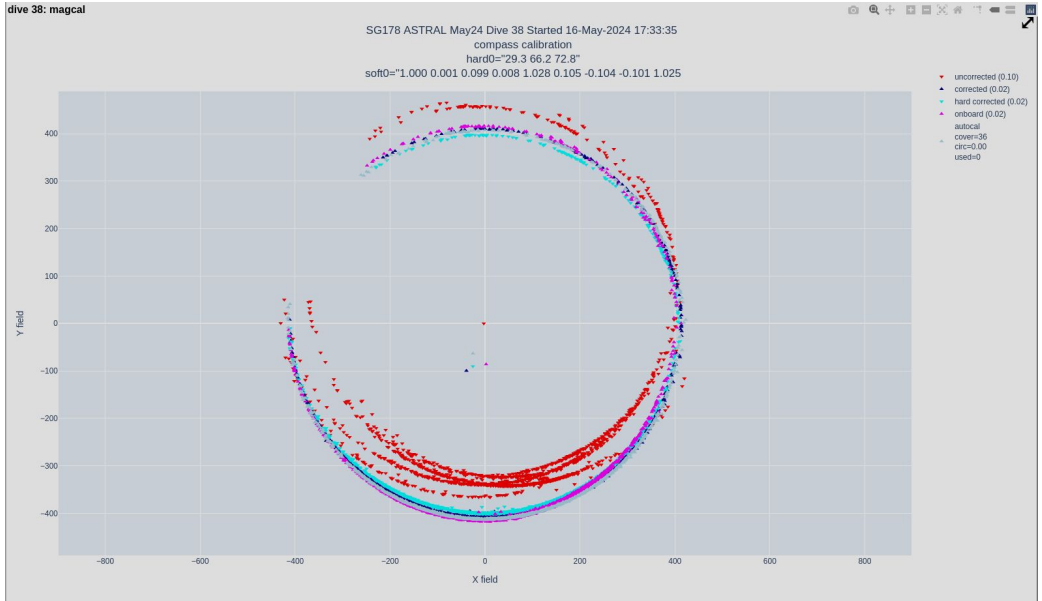
- $COMPASS_USE,4 (at least four) critical to send back magnetometer data
- Multiple ways for a pilot to compute a compass calibration:
  - vis computes a per-dive calibration result and plot in the plot ribbon
  - Magcal.py can be run from the command line
  - Multi-dive calibration (same as Magcal.py) is available from the tools menu on vis
- Transfer the result from any of these into a file named tcm2mat.cal and upload to glider by putting the file into the glider mission or home directory. The updated file will be transferred automatically during the next glider call.

- The glider can run calibration onboard according to $COMPASS_USE bits:
  - Bit 2 (4): return magnetic field data (always set at least this)
  - Bit 14 (16384): run calibration onboard after every dive
  - Bit 15 (32768): prefer tcm2mat.aut (automatic cal result) if available when loading coefficients
  - Bit 16 (65536): accept a good automatic calibration as the calibration to use and save to tcm2mat.aut
- Additional variables in tcm2mat.cal can be set to control the calibration (rare)
  - min-quality: RMS deviation from perfect circle (default 0.2)
  - min-cover: minimum coverage around the compass rose in 10s of degrees (default 18)
  - tail: number of dives (default 3)
  - maxpts: number of data points to use (default 2000)

- The calibration used is reported every dive in $IRON in log file
- Automatic calibration results (applied or not) are reported in $MAGCAL

In situ compass calibration techniques on Seaglider 2

- The goal when selecting dives for calibration is to choose dives that provide good coverage around the complete circle
  - Consider selecting dives on either side of a target change (dives on reciprocal or orthogonal headings).
  - Results are often better earlier in a mission when glider might be spinning more.
  - You can guide the glider through deliberate calibration dives to achieve a range of headings and attitudes.
- When evaluating a calibration result:
  - look for hard iron results that are consistent from dive to dive (or group of dives to group of dives)
  - soft iron matrix where the diagonal terms are reasonably close to 1 and the off-diagonal terms are relatively small.
  - Is the plotted field is nicely circular and well centered on (0,0)?

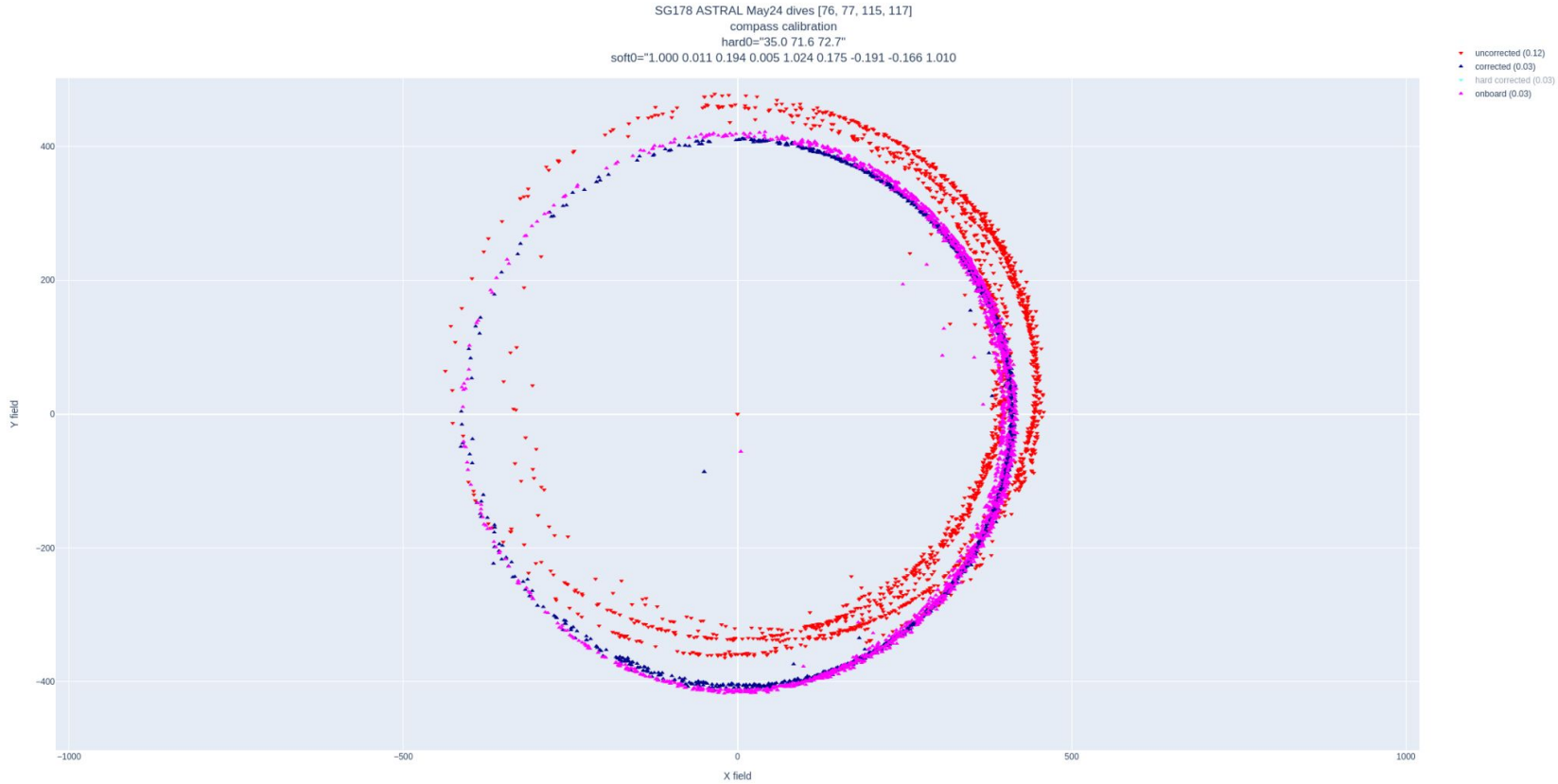


uncorrected = glider's measured magnetometer data
corrected = field corrected for hard and soft iron
hard corrected = field corrected by hard iron only
onboard = corrected field calculated for this dive (using tcm2mat.cal/tcm2mat.aut)
autocal = autocalibration calculated onboard glider (used=0 means not applied)

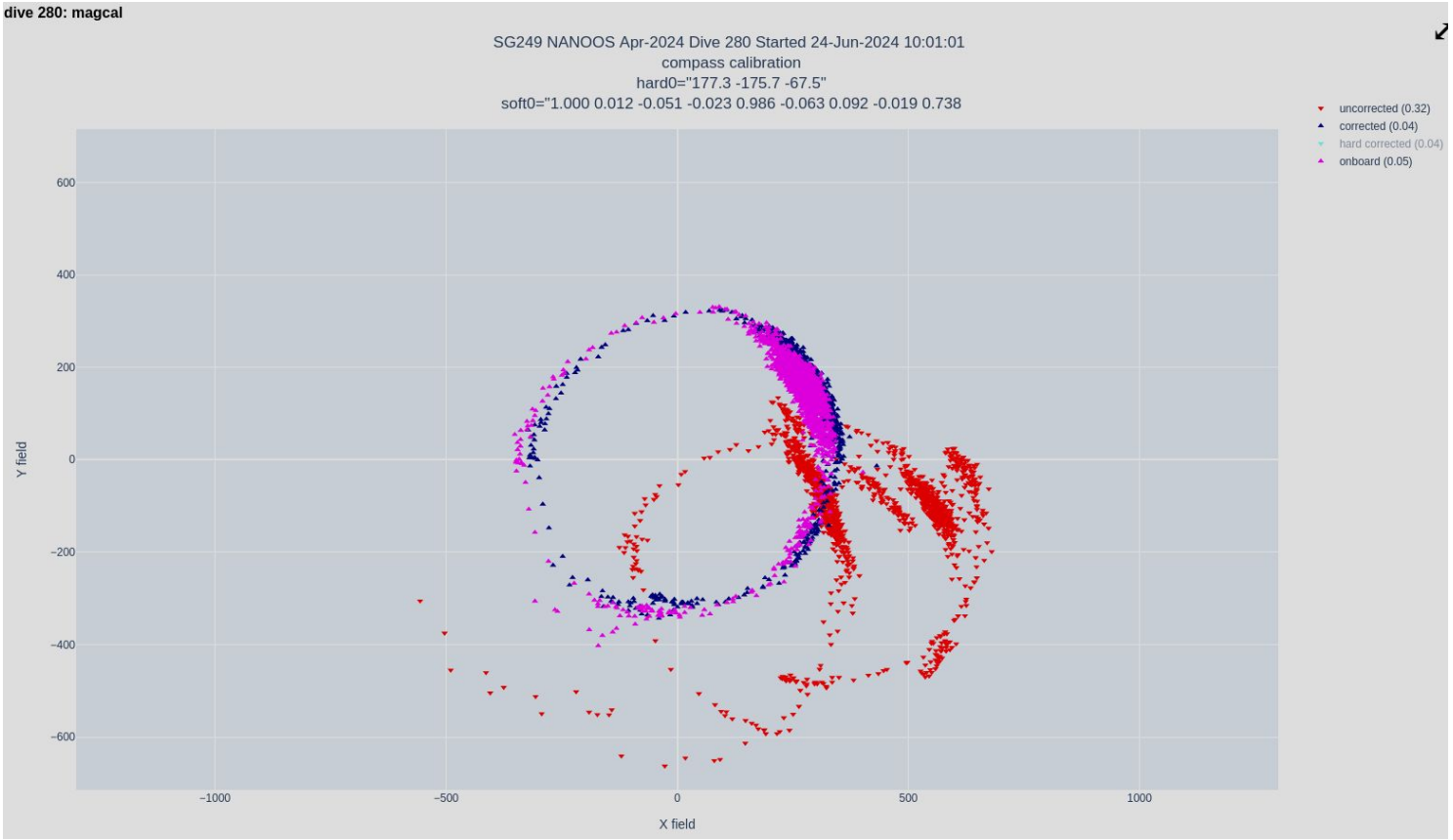
(numbers in the legend are RMS deviation from perfect circle - smaller is better)

Example: Run using the compass calibration tool from vis. Dives selected on either side of multiple target turns.

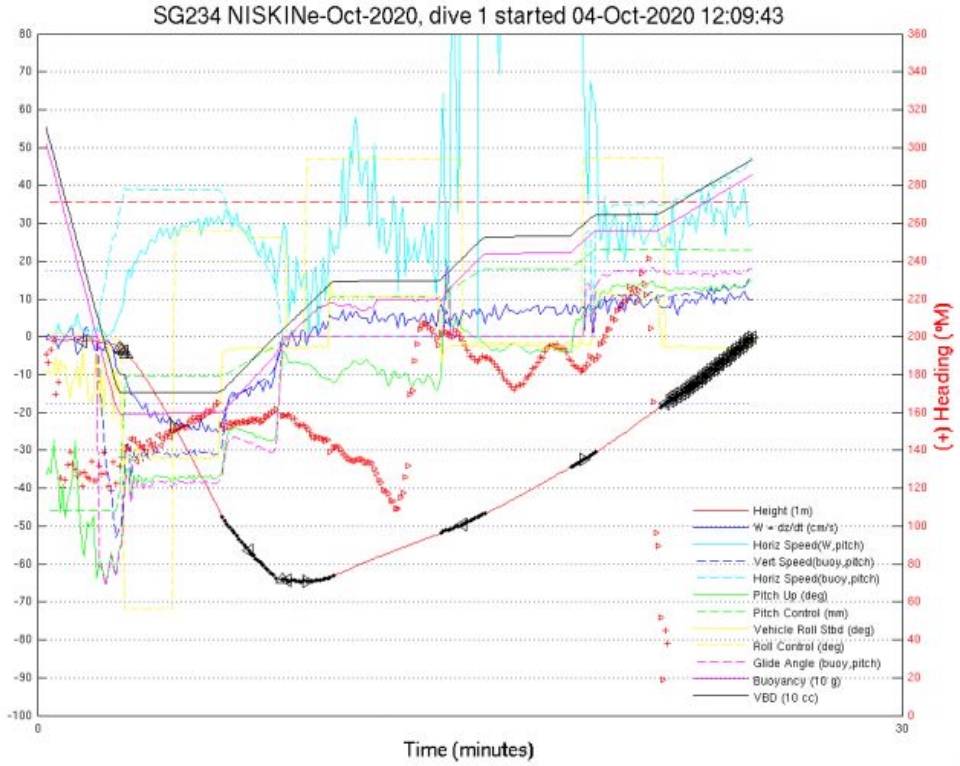
hard0="35.0 71.6 72.7"
soft0="1.000 0.011 0.194 0.005 1.024 0.175-0.191 -0.166 1.010"

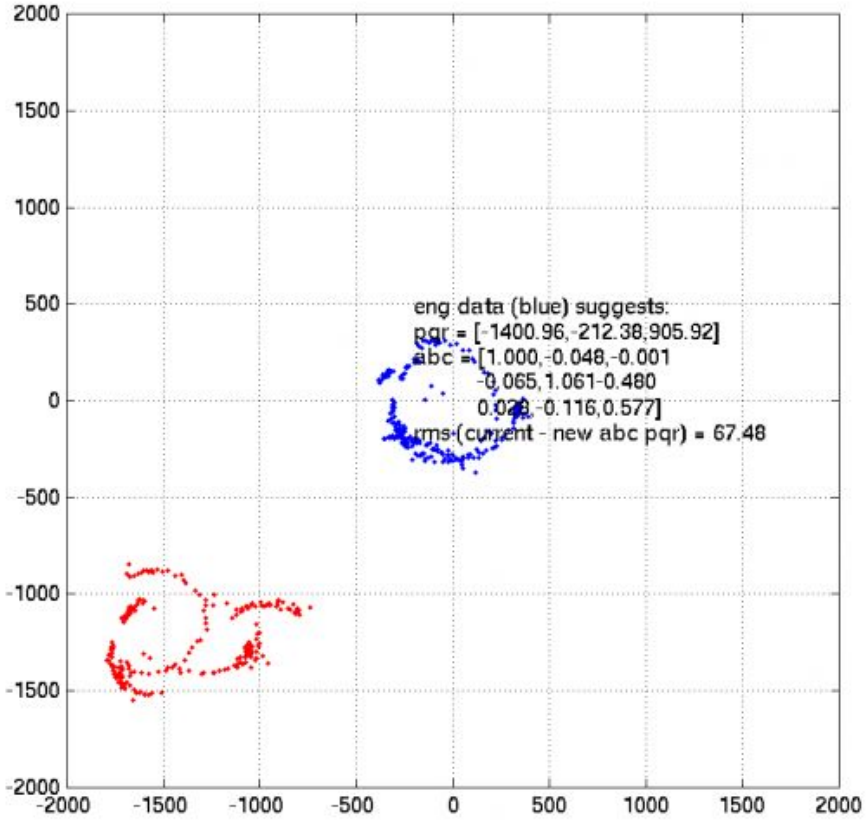


Example: Good coverage, but spare. Coverage strongly biased in one direction. The “circle” is not very circular and the 3rd diagonal soft iron term is suspiciously low.



Example: Significant hard iron, glider heading appears relatively constant when glider is actually spinning in circles. Hard iron offset same order of magnitude as (actually greater than) observed field values.





IOP webinar series going forward

Please fill out this short survey to help guide future webinar content

https://forms.gle/gQ75vH6vk5GeLbkg6