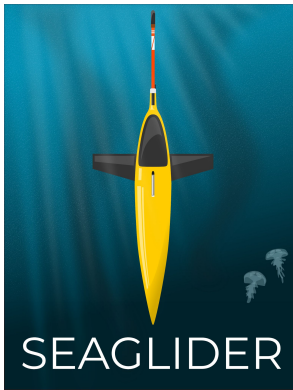
Seaglider Webinar Series pt. 2

Vis 201: Piloting with basestation3

IOP Seaglider Group
APL-UW
5 June 2024



Caveat / disclaimer

Most of what we will show in this presentation is simply our workflow for our gliders within IOP. There might be multiple valid ways to manage mission directories, archive data, handle notifications, think about roll trim, determine C_VBD, calibrate a compass, and just generally pilot gliders.

Our intent here is largely to highlight new capabilities in basestation3 that users might like to incorporate into their own workflows.

Mission management I: Directory structure

- We use a strategy of one directory per mission.
- Vis tools like the index page and the selftest review allow for easy browsing of historical test and mission data
- We encourage seaglider.pub users to leave historical missions on the server to take advantage of these tools (we do discourage archiving of other types of ancillary data (models, remote sensing)).
- Basestation3 supports the current/active mission being in a sub-directory. In this case there is a symbolic link “current” in the home directory that points to the current mission directory.
    - This approach means the mission is configured before the mission starts.
    - This is in contrast to the historical approach of using MoveData.py to archive the data off to a sub-directory or another location when the mission is complete.
- Commission.py is still used to create a new glider account, but this is only done once and is already taken care of for seaglider.pub users (because we at IOP have to create glider accounts)
- For each new mission, run NewMission.py to create the mission directory and the appropriate symbolic links. From the home directory:
    - /opt/basestation/bin/python /usr/local/basestation3/NewMission.py ./ MissionName
    - This creates “current” symlink and links to some dot files.

Mission management II: update missions.yml

missions.yml specifies the glider numbers, mission names and directories, fixed markers, and other features that appear on the map feature of vis.

- missions.yml usually resides in the directory about the glider home directories

  seaglider.pub: /home/jails/<group_name>/gliderjail/home/missions.yml

- Vis automatically detects a changed missions.yml and changes immediately take effect. Errors can be difficult to debug, so best practice is:
  - cp missions.yml to missions.new
  - Edit missions.new
  - Run vis.py in test mode to check for syntax errors:

    /opt/basestation/bin/python /usr/local/basestation3/vis.py -r ./ -f missions.new -t

  - If the changes are ok, cp missions.new to missions.yml
  - There are lots of yaml tutorials online if you want to structure the file in the way that works best for you.

```
---
organization:
 orglink: https://school.edu/labpage.html
 orgname: School
 contact: Principal Investigator
 email: contact@school.edu
 text: Seaglider data for our group

assets:
  mooring: { type: fixed, lat: 48, lon: -125, marker: square, color: "#ff0000" }
  imagery: { type: tileset, url: "https://tiles.server.net/sg/tile/{z}/{x}/{y}" }

missions:
  - { glider: 101, mission: MissionThree_Apr24, path: sg249/missionthree_Apr24 }
  - { glider: 101, mission: MissionTwo_Feb23, path: sg249/missiontwo_Feb23, status:
complete }
  -
    glider: 102
    mission: Offshore_Jan24
    path: sg102/Offshore_Jan24
    also: [ {glider: 103}, {asset: mooring}, {asset: imagery} ]
  -
    glider: 103
    mission: Offshore_Jan24
    path: sg103/Offshore_Jan24
    also: [ {glider: 102}, {asset: mooring}, {asset: imagery} ]
```

Mission management III: pagers.yml

pagers.yml - Use instead of .pagers (though .pagers is also still processed). Hierarchical, subscription based approach.

Users define endpoints that are email addresses, slack hooks, etc. that are the destination for notifications from the basesation. Users and their endpoints can be defined at multiple levels

- global (/usr/local/basestation3/etc/pagers.yml)
- group (seaglider.pub: /home/jails/<group_name>/gliderjail/etc)
- glider: pagers.yml in the glider mission directory (or symlinked from mission directory to home directory).

Subscriptions define which different types of messages are sent to each user

The motivation is to provide a single source for contact information (the global or group level file) and then in each glider, pilots subscribe as needed to turn on or turn off according to the watch schedule or duty roster. Pilots can also remove themselves from all gliders by turning off at the global or group level.

```
(/usr/local/basestation3/etc/pagers.yml)

universal:
  status: on
  mattermost: [ { filters: [gps],  hook: https://server.og/mattermost/hooks/token1 },
                { filters: [alerts], hook: https://server.org/mattermost/hooks/token2 },
                { filters: [critical], hook: https://server.org/mattermost/hooks/token3 }]

pilot:
   status: on
   ntfy: [ { filters: [critical], topic: mypilotntfytopic } ]
   email: [ {filters: [critical], address: 2065551234@telco.com, format: html} ]
            {filters: [gps, alerts], address: name@school.edu} ]

critical: universal
alerts: universal
gps: universal
```

```
(glider local pagers.yml)

pilot:
    status: off

critical: [pilot, pilot2]
alerts: [pilot]
gps: [pilot]
```

Mission management IV: sg_calib_constants.m

sg_calib_constants.m

- Have a legato?
  - sg_ct_type = 4; % Unpumped RBR legato
  - Set legato_sealevel = <value as reported in selftest>;
  - GetLegatoPressCorr.py to determine the value for legato_cond_press_correction
- Wetlabs naming - see sg000/sg_calib_constants.m
- Have an optode?
  - GetOptodeConstants.py to get the sg_calib_constants.m values for the optode from a selftest

Don’t forget sg_plot_constants.m. It is still used for drawing the static map (used on the index page and in the plot ribbon).

Selftest review

- Run a selftest! In the lab, on the dock, on the ship. Run them often. They are cheap and easy.
- And look at the results!
- Summaries - look for obvious errors and warnings
- Review capture for subsystem results
- If there are questions, look at historical selftests if available
    - Has this glider always done this or is this anomaly new?
- Check that needed bathymaps are loaded (loading them via Iridium is a bummer)
- Check that initial targets, science and scicon configurations are reasonable - prepare corrections as needed
- Look at the parameter comparison to confirm settings for dive 1
    - The table shows which set of canonical parameters was used for comparison
    - You can have your own if you have a specific set of initial values you like: .canonicals
    - You can also create a symbolic link from .canonicals in the glider directory to any of the various in /usr/local/basestation3/canonicals/
- Use cmdedit, targedit and sciedit to make any needed changes to parameters, targets, and science.
    - The …edit tools will validate syntax, report changes, and log which pilot made which changes
    - The tools honor the EDITOR environment variable so you can use vim, emacs, joe, pico, nano, etc.

Initial dives

- The first several plots in the vis plot ribbon are designed to help with glider trimming
- We still mostly trim in the classic order at the beginning of the mission:
    - Pitch. It's the easiest and can have the biggest impact on flight, and can be trimmed well even based on shallow dives
    - VBD. Usually easier when the glider is flying a bit deeper.
    - Roll. Trimming roll can also depend on getting a reasonable compass calibration.
- Each of those DOFs has their own plot or plots to help with trim. The first plot, the classic diveplot, is also a primary tool.
- Diveplot can be overwhelming. The version in vis tries to help by allowing you to turn traces on or off to help with information overload.
- When looking at diveplot, some simple things to consider:
    - Check for dive up/down symmetry in both time and profile shape (depth vs time)
    - Can any asymmetry be explained by big differences in pitch between dive and climb?
    - Where are the VBD pumps happening (deep/shallow), how are they affecting vertical velocity?
    - Get a sense of roll behavior (many or few? where in the dive or climb?). Is the glider heading changing a lot when the glider is not turning? Or conversely is the glider heading not changing even when the glider is rolled? Consider the turning and roll behavior that you see on diveplot when looking at the roll trim plots.
- If you are not seeing data in vis or do not see data from all sensors - check alerts and baselog
- You can often get a good first estimate of the compass calibration even from dive 1. This may not be necessary but if your heading/turning behavior looks very odd then it could be due to compass calibration and nothing you do about roll trim is going to make much sense until there is at least a first guess compass calibration onboard.
- Compass cal results are presented in the new format. You may need to translate to old tcm2mat format.

Roll heuristic - look at diveplot areas where the glider is not trying to turn

If (glider descending) then

- If (turning clockwise, i.e. to the right, heading increasing) then
  - Increase C_ROLL_DIVE (turn left)
- Elseif (turning counterclockwise, i.e. to the left, heading decreasing)
  - Decrease C_ROLL_DIVE (turn right)
- Endif

Diving

heading ↑ then C_ROLL_DIVE ↑

heading ↓ then C_ROLL_DIVE ↓

ElseIf (glider ascending) then

- If (turning clockwise, i.e. to the right, heading increasing) then
  - Decrease C_ROLL_CLIMB (turn left)
- Elseif (turning counterclockwise, i.e. to the left, heading decreasing)
  - Increase C_ROLL_CLIMB (turn right)
- Endif

Endif

Climbing

heading ↑ then C_ROLL_CLIMB ↓

heading ↓ then C_ROLL_CLIMB ↑

Later dives

- With 10+ dives or once diving to typical apogee depth, review mission energy consumption
  - Rev E gliders present both model based and fuel-gauge based results
    - Model uses hardcoded current draws or values from CURRENTS file along with measured on time (phone and motors are exceptions, those currents are measured in real-time, and scicon reports its own average current from its onboard fuel gauge)
    - Fuel gauge uses continuous real-time current measuring (typically 2 Hz) to track current draw from both battery packs.
  - Understand where energy is going: sensors, VBD, Iridium (large sensor payloads?, capture files?), etc.
  - Consider whether the projected endurance meets mission requirements
  - Mission days and the trend in mission days can help you understand how the changes you are making to sampling, thrust or dive speed are impacting mission endurance.
- Consider running multi-dive regressions to understand VBD and hydrodynamic parameters