Seaglider pilot training

Integrative Observational Platforms group (IOP)

Applied Physics Laboratory
University of Washington

12-16 August 2024

UNIVERSITY of WASHINGTON
Applied Physics Laboratory

Introduction

- Overview of schedule
- Logistics
- Who we are - brief history of IOP
- Introductions - experience with Seagliders (or other gliders), job / expected involvement with gliders, any specific goals/topics of interest/etc.
- To get set up on the IOP basestation, please generate a public/private key pair:

```
ssh-keygen -t ed25519 -o -a 100
```

Please ask questions at any time!

Conventions used in these slides

$PARAMETER_NAMES

Definitions

Glider/basestation filenames

Script names

`Code or commands`

GGG = glider number

dddd = dive number

ccc = call cycle

SG = older-style Seaglider (tapered nose, 52kg)

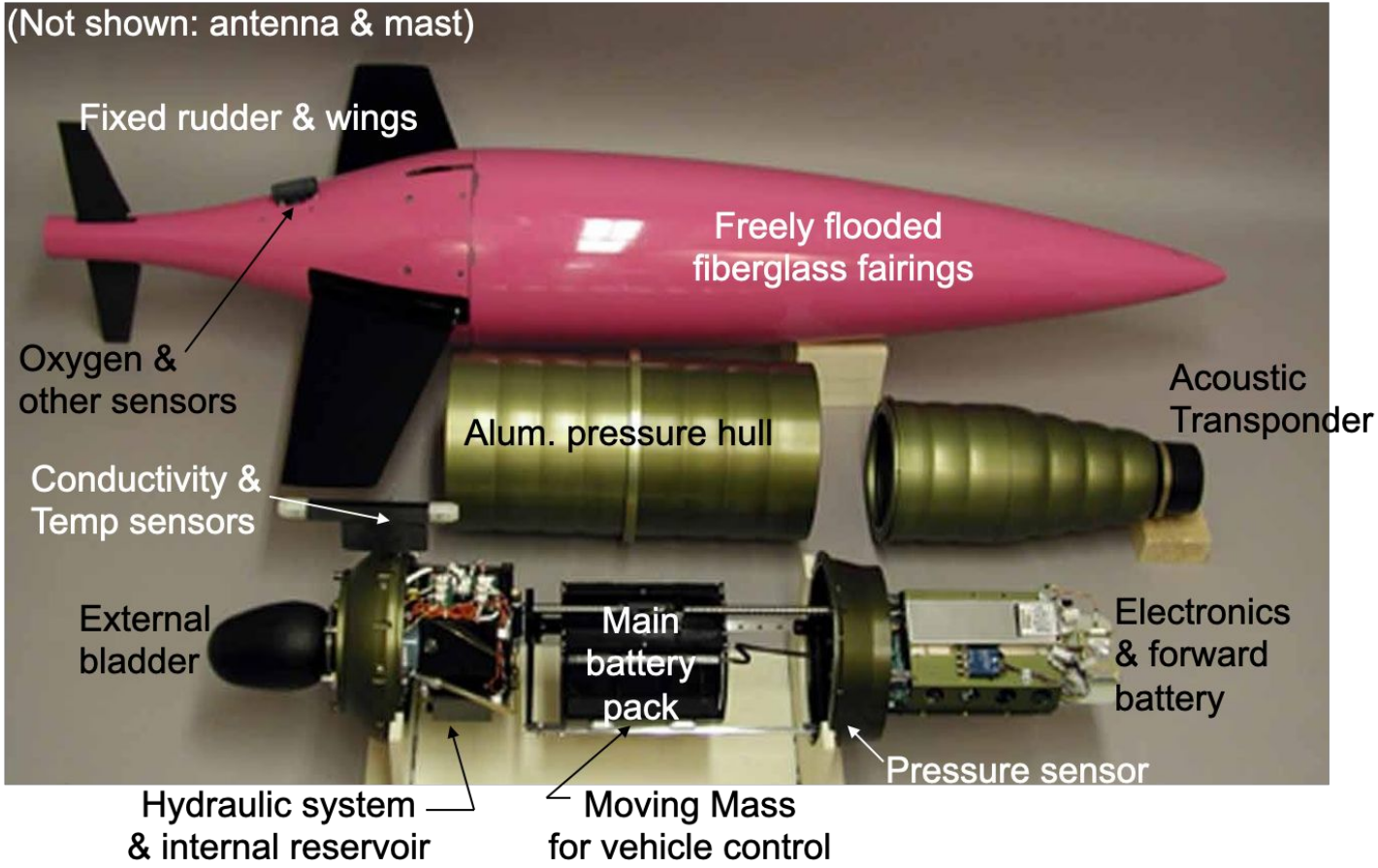
SGX = newest generation of Seaglider (bullet nose, 72kg)

Seaglider fundamentals

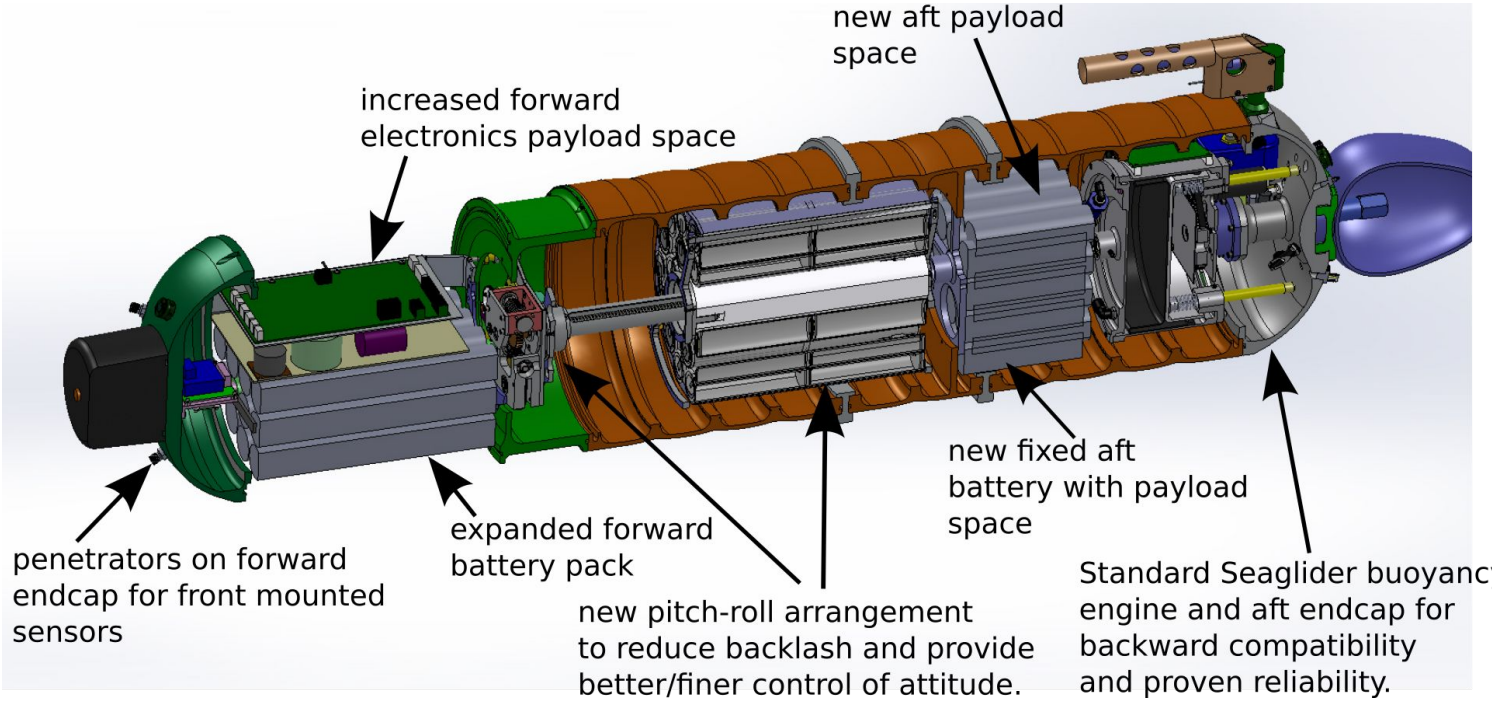
Principles of operation

- Seagliders are buoyancy-driven AUVs: they move through the water without an active thruster and are thus extremely energy efficient.
- Gliders change their buoyancy to move up and down in the water column, much like an Argo float profiles up and down. Vertical speed is ~10 cm/s.
- With a glider, some vertical motion is translated into horizontal motion by the wings
  - glide slopes from ~1:1 to ~1:5, horizontal speed ~25 cm/s.
- The glider can control its pitch and roll by shifting its large internal battery
- It uses roll to bank and turn so that it can maintain a specified heading.
- Seaglider is designed for max energy efficiency
  - Main energy draw is buoyancy pump (~50%)

Seaglider components



SGX: the newest generation of Seaglider



Buoyancy

- For the glider to move vertically, its density must change:
  - If the glider is more dense than the surrounding water, it sinks (dives)
  - If the glider is less dense than the surrounding water, it rises (climbs)
- The glider must be ballasted or configured initially so that it is very close to neutrally buoyant in the ocean water that it will operate in.
- The glider changes its density (and thereby its buoyancy) by changing its volume while keeping its mass constant.
- It is a constant mass variable volume vehicle, unlike submarines which change their mass (by taking on or ejecting water).
- We might say the vehicle is “heavy” or “light” in a particular phase of flight, but we mean denser than surrounding water (heavy) or less dense than surrounding water (light).
- The vehicle never changes its mass while operating (we hope)

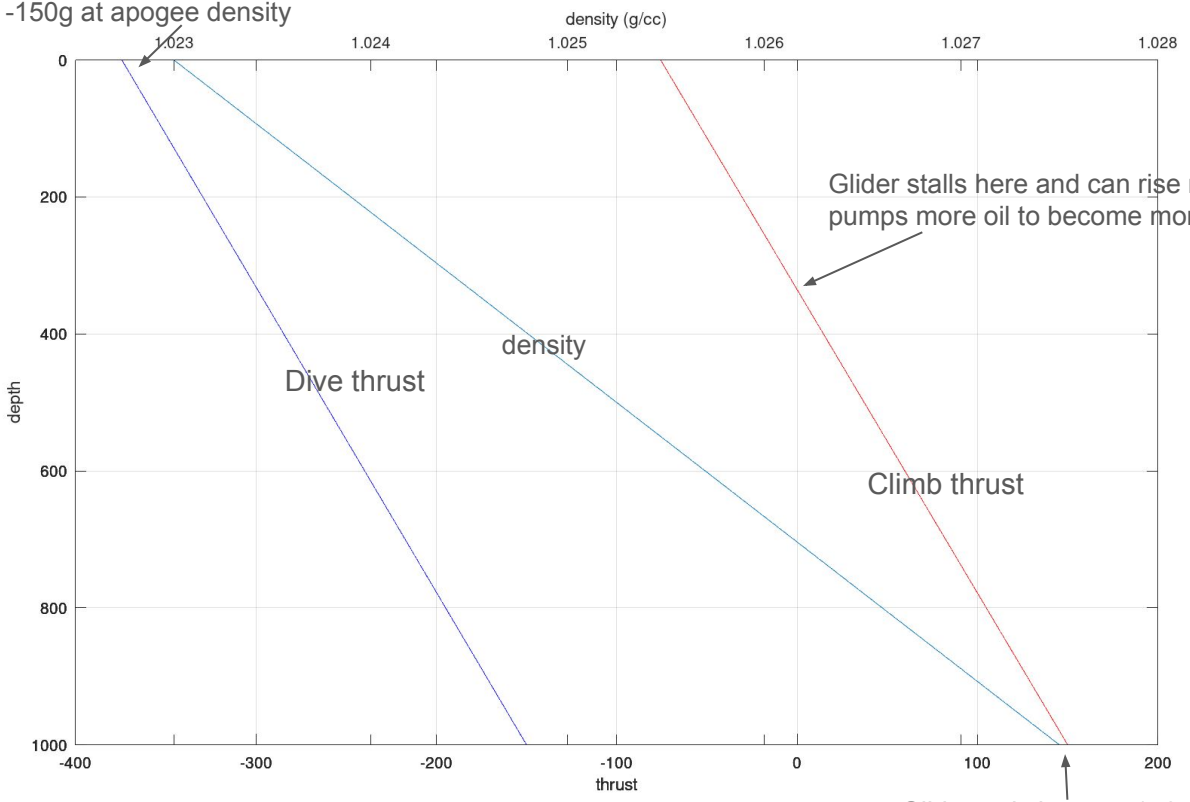
Density

- Typical 1000 m ocean density is 1027.5 kg/m^3
- Shorthand notation for density: σ = (density - 1000) kg/m^3 – so the typical 1000 m density is 27.5 in σ units (or σ_θ since density is measured based on potential temperature θ)
- We use potential density (which removes the effect of pressure on density) when talking about Seaglider because Seaglider (mostly) matches the compressibility of seawater due to its hull design.
- The most important formula in glider operations:

$$density = \frac{mass}{volume}$$

Buoyancy, density and stratification - a very contrived example

Glider starts dive by bleeding to -150g thrust, but that's -150g at apogee density



Glider stalls here and can rise no higher unless it pumps more oil to become more buoyant

Glider switches to +150g at apogee by pumping ~300cc.

Buoyancy and ballasting

The glider can be ballasted (mass changed by adding or removing lead) for

- A given range of water density
- Negative buoyancy (thrust), e.g., in strong currents
- Reserve buoyancy, e.g., in areas with low surface density like the Arctic, river plumes or Bay of Bengal (monsoon season).

Rule of thumb:

- For Seaglider SG, a ~50 cc change in glider volume produces a 1 σ_θ density change
- For SGX, a ~70 cc change in volume produces a 1 σ_θ density change

Buoyancy and ballasting

The VBD for contains around 800 cc's of moveable oil, i.e., available buoyancy (same for SG and SGX). To determine the density range (stratification) where your glider can operate:

- Start with the total buoyancy (cc's of oil) available
- Subtract the cc's of oil needed to lift the antenna out of the water
- Subtract the cc's of oil needed for negative buoyancy (thrust), usually specified at the deepest/densest water to be encountered).
- Divide the remaining cc's of oil by the rule of thumb (50 or 70 cc/σ_θ):

Total VBD available	800 *cc*
Positive buoyancy required to expose antenna	(150 *cc*)
VBD remaining	650 *cc*
Negative thrust in densest water	(250 *cc*)
VBD remaining	400 *cc*
equivalent σ_T units of stratification (SGX)	5.5
equivalent σ_T units of stratification (SG)	8

Variable buoyancy device (VBD): drives glider's vertical motion

VBD moves oil between an internal reservoir that is inside the glider's pressure housing and a bladder that is external to the pressure housing, thereby changing the total volume of the glider and hence its density

Oil pumped from the reservoir to the bladder:

- Glider's total volume increases
- Glider becomes buoyant relative to the ocean around it and rises (i.e., climbs)

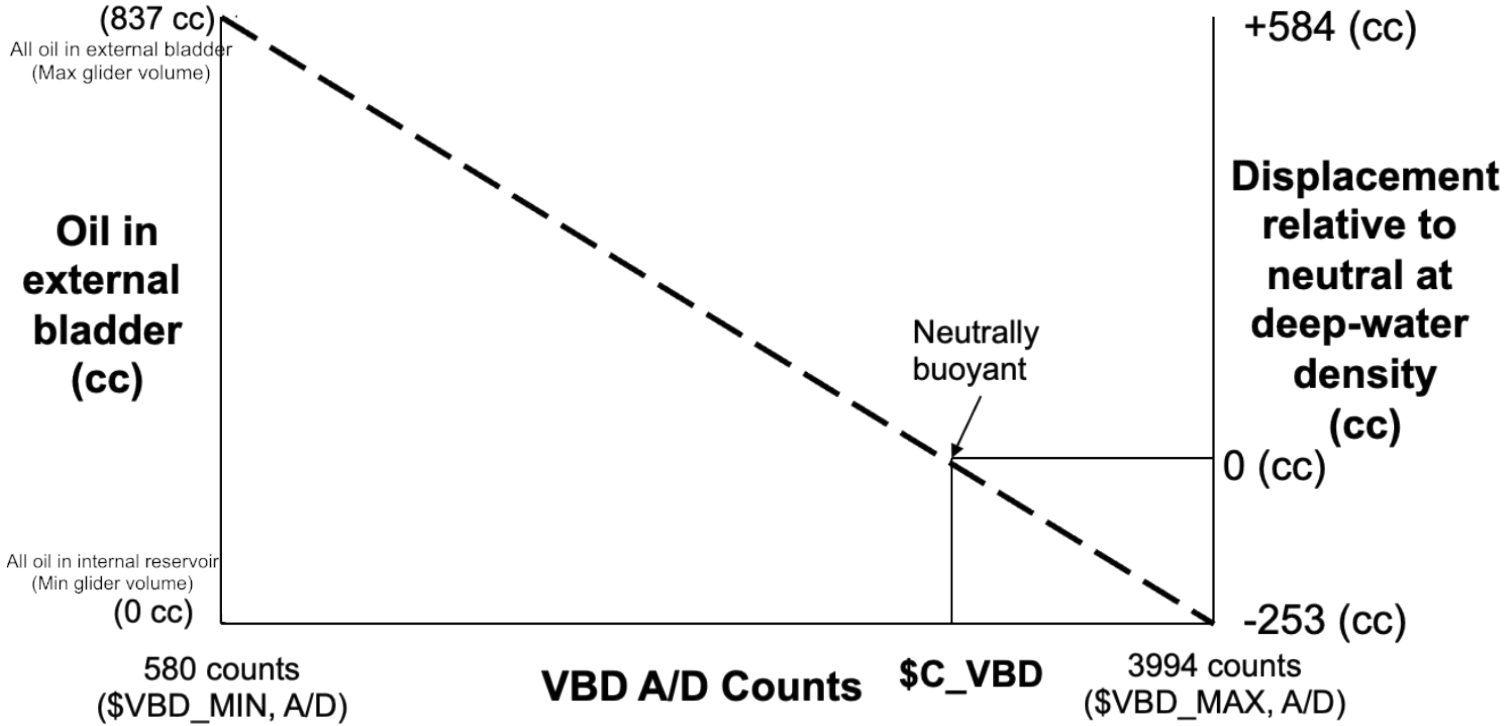
Oil bled from the bladder to the reservoir:

- Glider's total volume decreases
- Glider becomes dense relative to the ocean around it and sinks (i.e., dives)

VBD units

- For VBD there are two units that we use for talking about the control system
  (1) In *engineering units* we talk about volume in cc's, cubic centimeters
    - This could be the maximum volume of the vehicle (which we'll call volmax below)
    - Or it could the differential volume as measured by the oil in (or out of the bladder) relative to the volume at which the glider is neutrally buoyant.
  (2) At the lowest level, we use A-to-D counts. This is the system on the glider that measures the control positions. It is a 12-bit value so for all systems ranges from 0 to 4095, though in practice there are software and hardware stops that put the ranges from 100s to 3900s.
- For the VBD system, low AD counts correspond to a full bladder. Higher AD counts are an emptier bladder.
- The conversion is set by the physical travel of the linear potentiometers and geometry of the internal reservoir where the volume is measured (so large AD counts is a full internal reservoir). There are 4.0767 AD counts / cc of oil for SG and SGX.
- These same concepts of engineering units and AD counts will apply to pitch and roll as well.
- Most piloting is done in AD counts.
- Most planning / analysis is done in engineering units.
- A pilot must be familiar and comfortable translating back and forth between the two concepts.

VBD oil vs AD counts (example values)



Definition of important VBD variables

Volmax: maximum volume of the glider (i.e., volume when all moveable oil is in the external bladder), in cc's. Getting an accurate estimate of volmax is key to ballasting. We get an initial estimate of volmax in a tank, and a more refined estimate during the test deployment. Volmax is a constant for a given glider configuration (external payload and VBD settings).

Neutral volume: This is the volume of the glider including external bladder position at which the glider is neutrally buoyant at the desired apogee density. By definitions below, the bladder position at this point is 0 cc and **$C_VBD** AD counts.

Thrust: buoyancy in grams relative to the surrounding water. Negative thrust means the glider is "heavy" or denser than the surrounding water and will descend. Sometimes also described in cc of oil relative to **$C_VBD**.

Target thrust (**$MAX_BUOY**): This is the desired amount of negative thrust that we want to have at apogee so that we can maintain speed throughout the entire dive.

$C_VBD: The position of the oil reservoir in AD counts that defines where the glider has zero thrust (is neutrally buoyant) at apogee density.

Apogee density (**$RHO**): Reference density in kg/m^3. Typically set to the most dense water expected to be seen.

$VBD_MIN, **$VBD_MAX**: AD limits of the VBD system (**$VBD_MIN** means pumped to max, i.e., bladder is full of oil; **$VBD_MAX** means bled to minimum, i.e., internal reservoir full of oil)

Working with VBD values

1. Target thrust fixes **$C_VBD** because we never need to get heavier than that thrust, so we can set **$C_VBD** such that when bled to **$VBD_MAX** (minimum buoyancy) we will be at target thrust:

   C_VBD = VBD_MAX – (thrust/$RHO)/VBD_CNV

2. Vol0, glider total neutral volume, is defined by glider mass and target apogee density:

   Vol0 = mass/$RHO

3. Now we know how positive we can get: it's the difference between neutrally buoyant at **$C_VBD** (when glider has total volume vol0) and having maximum buoyancy when pumped to **$VBD_MIN**:

   volmax = vol0 + (VBD_MIN – C_VBD)*VBD_CNV

4. If planning or ballasting, we can rearrange (2) and (3) to solve for mass to get desired total mass of glider:

   mass = RHO*(volmax – (VBD_MIN – C_VBD)*VBD_CNV)

5. We can also confirm that the glider will be able to surface and call - we need to be able to lose 150cc (the volume of the antenna) and still be buoyant in the lowest density surface water that we expect to see:

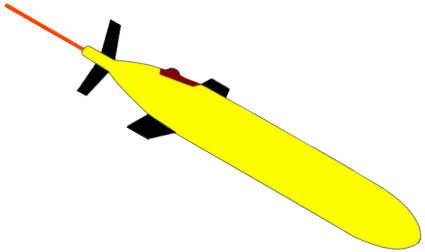
   RHO_min = mass / (Volmax - 150)

* **$VBD_CNV** = cc's of oil per AD count = -0.2453 (constant, same for SG and SGX)

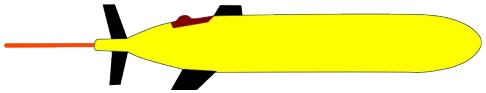
Mass shifter: controls glider's pitch and roll

The mass shifter controls the glider's pitch and roll by using electric motors to move the battery pack and a brass weight (~13 kg total mass).

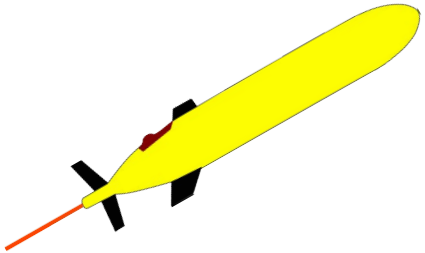
Shifting the mass forward/aft pitches the glider's nose down/up:



Mass is shifted forward
Glider's nose points down
AD counts low



Mass is between fore/aft
Glider flat
AD counts = **$C_PITCH**

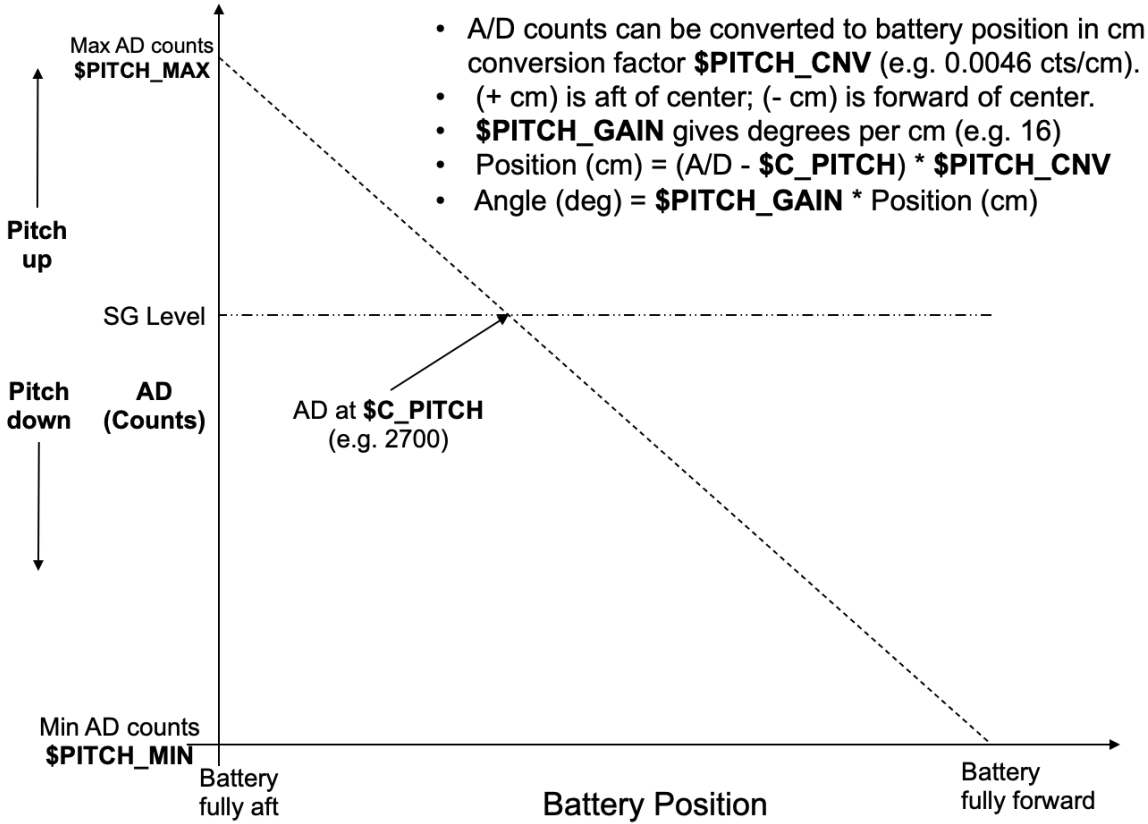


Mass is shifted aft
Glider's nose points up
AD counts high

Pitch units

- In *engineering units* we talk about the battery position along the lead screw in centimeters (cm)
- For the pitch system low AD counts correspond to the mass fully forward, glider pitched down. Higher AD counts are pitch mass aft, glider pitched up.
- The conversion (**$PITCH_CNV**) is set by the physical travel of the lead screw and the multi-turn rotary potentiometer that measures the position of the battery along the lead screw. **$PITCH_CNV** is different for SG and SGX but otherwise does not vary from glider to glider - it is constant for a given glider type.
- Pilots set the center (**$C_PITCH**) in AD counts, but pitch also has a **$PITCH_GAIN** parameter that is based on engineering units (degrees pitch / centimeter of mass movement) to describe how much the glider's pitch changes when the battery position changes.
- SGX tends to have lower gains - small battery moves produce small pitch changes. SGs are more "tippy": small battery moves produce larger pitch changes. .

Pitch relationship to AD counts

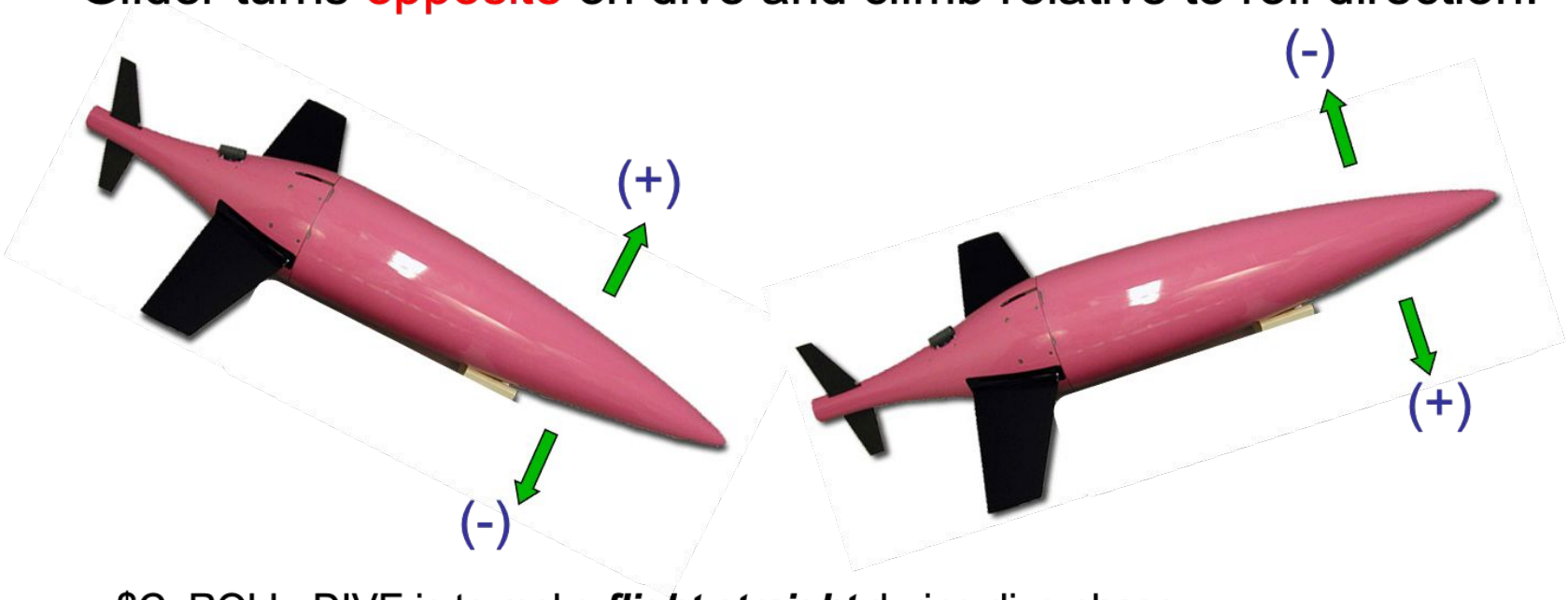


- A/D counts can be converted to battery position in cm using conversion factor **$PITCH_CNV** (e.g. 0.0046 cts/cm).
- (+ cm) is aft of center; (- cm) is forward of center.
- **$PITCH_GAIN** gives degrees per cm (e.g. 16)
- Position (cm) = (A/D - **$C_PITCH**) * **$PITCH_CNV**
- Angle (deg) = **$PITCH_GAIN** * Position (cm)

Mass shifter: roll

- The mass shifter rotates the battery pack about the center axis of the glider
- This changes its center of gravity, making the glider roll and thereby turn
- The glider always rolls in the opposite direction as the mass shifter
- However, the direction of the turn depends on whether the glider is climbing or diving

Glider turns opposite on dive and climb relative to roll direction.

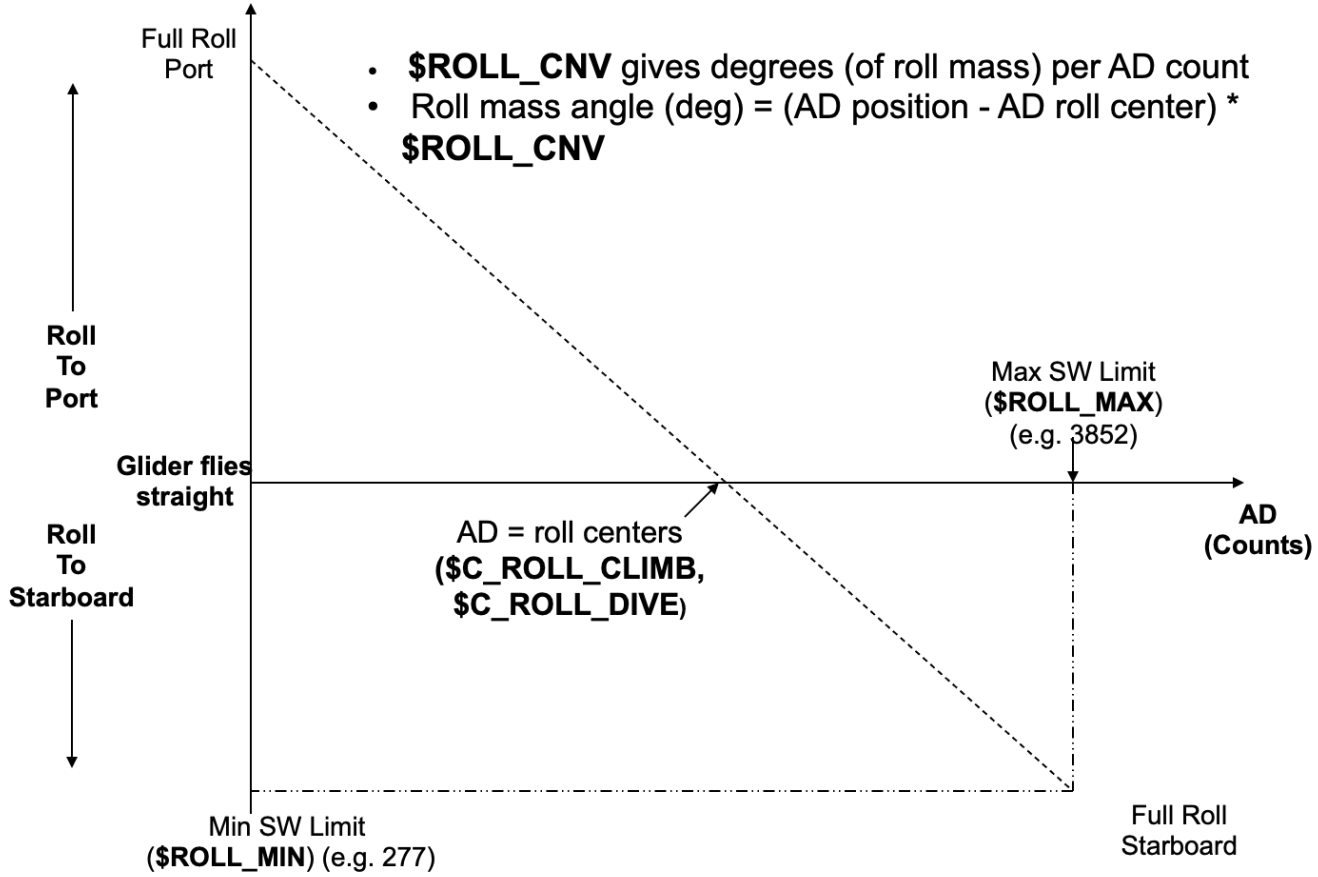


- $C_ROLL_DIVE is to make ***flight straight*** during dive phase.
- $C_ROLL_CLIMB is to make ***flight straight*** during climb phase.
- Smaller A/D shifts mass to port, glider rolls port
- Port roll on dive *turns right* (starboard), port roll on climb *turns left* (port)
- Larger A/D shifts mass to starboard, glider rolls starboard
- Starboard roll on dive turns left (port), starboard roll on climb turns right (starboard)

Roll units

- In *engineering units* we talk about the battery rotation around a central axis in degrees.
- Roll is a bang-bang controller - we typically apply full roll (moving the battery as far as it will go to one side or the other) - so we do not have a gain setting.
- For roll, we tune the centers in AD counts (**$C_ROLL_DIVE**, **$C_ROLL_CLIMB**) to specify the rotational position that best achieves straight flight when we're not trying to turn.

Roll relationship to AD counts



Recap of control concepts for buoyancy, pitch, and roll

- Electric motors drive the VBD and mass shifter systems
- Potentiometers (“pots”) are used to measure the position of each of these systems at any given time
- Potentiometers have a range of 0 to 4095 analog-to-digital (AD) counts
- AD counts are converted to physical displacements – cc’s of oil (for the VBD), cm of travel (pitch), degrees of roll (of the battery) (roll)
- Conversions are based on mechanical design of the system and do not change over time (i.e., the pilot shouldn’t change these)

Sub-system	**Conversion factor**	**Parameter name**
VBD	-0.2453 cc/AD (same for SG and SGX)	$VBD_CNV
Pitch	0.00313 cm/AD (for SG) 0.00413 cm/AD (for SGX)	$PITCH_CNV
Roll	0.028270 degrees/AD (for SG) 0.054945 degrees/AD (for SGX)	$ROLL_CNV

Recap A/D counts for control of buoyancy, pitch, and roll

Sub-system	**A/D counts**	
	Low	**High**
VBD	Oil in bladder More buoyant Glider climbs	Oil in reservoir Less buoyant Glider dives
Pitch	Mass shifter forward Glider pitches down	Mass shifter aft Glider pitches up
Roll	Center of mass shifts to port (negative degrees) Glider rolls to port	Center of mass shifts to starboard (positive degrees) Glider rolls to starboard

Recap centers of buoyancy, pitch and roll

- The mass shifter and VBD have center points that are the neutral position for that sub-system (expressed in A/D counts)
- The pilot adjusts the centers to control the glider.
- Seagliders are asymmetrical, so the roll centers are generally different when the glider is diving versus climbing

Sub-system	**Parameter describing the center point**	**Glider position at center point**
VBD	**$C_VBD**	Neutrally buoyant at apogee
Pitch	**$C_PITCH**	Flat
Roll	**$C_ROLL_DIVE** **$C_ROLL_CLIMB**	Flying straight (not turning) during dive or climb

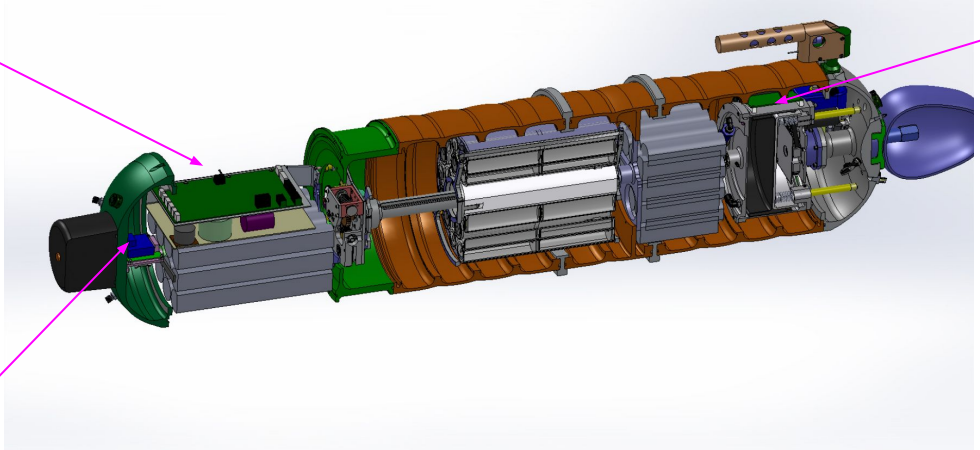
Seaglider payload

Seaglider electronics payload

- Core systems:
  - Mainboard - main processor and IO hub
  - Iridium modem
  - GPS
  - Acoustic transponder/altimeter
  - Compass
- Required sensors:
  - Conductivity-temperature sensor - science and engineering
    - Historically, Sea-Bird “CT sail” or “glider payload CTD”
    - More recently, RBRLegato CTD
  - Pressure sensor:
    - Typically, Kistler pressure sensor
    - Can also rely on RBRLegato’s pressure sensor if available
- Internal sensors (engineering):
  - Internal pressure - monitors internal vacuum
  - Relative humidity - leak monitoring
  - Temperature
- Typical sensors include:
  - Oxygen (e.g., Aanderaa optode, RBR optode, Sea-Bird DO)
  - Backscatter/fluorescence sensors (e.g., WETLabs puck, RBR Tridente)
- Other: Seaglider supports a wide range of sensors
  - AD2CP, passive acoustics, microstructure, irradiance, …
- More sensors = more power consumption and often more drag on the vehicle

Seaglider electronics

Motherboard sits above the forward battery pack



Aft end of vehicle, attached to VBD cylinder:

- Tailboard
- Scicon board (if present)
- Seabird CT sail and oxygen sensor boards (if present)

Compass in forward endcap

Ribbon cables running inside the length of the hull attach the forward and aft electronics

Most sensors are wired to the tailboard or scicon

Payload configuration

- Three options for sensor payload configuration:
  - Attach to mainboard as “serdev” - a serdev is a simple serial device that outputs one line of data per sample and typically has no onboard storage of its own.
  - Attach to mainboard as “logdev” - a logdev is a device capable of independently logging data. Typically the glider provides power and event notification (start of dive, end of dive, etc.) and the logdev provides an entire data file for telemetry at the end of the dive cycle.
  - Attach to scicon - scicon is a special logdev that provides more features and flexibility than the serdev and logdev interfaces. It also provides asynchronous sampling capability so can reduce power consumption when configured properly along with power saving measures on the core platform.
- All three options provide end-user capabilities for adding new sensor types.
- serdev and logdev sensors are added via `param/config` in the menu system and with *.cnf* files on the glider memory card.
- Scicon sensors are configured in *scicon.ins* and *scicon.att* files on the scicon memory card.

serdev

- Configured with a .cnf file on the glider memory card
- *.cnf* file syntax: **https://iop.apl.washington.edu/iopsg/serdev.txt**
- .cnf file must be registered with the glider so that it becomes available as an option
  - Copy .cnf file to memory card, either directly or via ymodem
  - Register the .cnf file in the sensor library
    `param/config/seradd`
- Attach the sensor to an available serial port (USART2, 4, 5A,B,C,D, 6). If voltage is jumperable or software selectable (USART4) be sure to select the proper voltage or wire the appropriate pin.
- Configure the sensor in the glider software
  - `param/config/sensor`
  - The sensor should be available in the list of sensors
  - Choose the correct serial port
- Test the sensor
  - `hw/sensors/xx` (where xx is the prefix as defined in the .cnf file)
  - Use the edit menu option to test changes to .cnf values
    `edit timeout=300`
    This is a temporary change. Make the change permanent by editing the .cnf file.
  - You can edit the .cnf file directly on the glider:
    `!vi sensor.cnf`
  - Or make changes off the glider and re-upload the modified file to the memory card
    `!yr` (followed by ymodem send from your terminal program)
  - Use capture level debug to see additional sensor interactions
    `!capvec HNAME DEBUG BOTH`
- Sensor sampling is controlled in the *science* file

```
name=legato
prefix=rbr
timeout=1000
baud=19200
warmup=8000
voltage=10
headerlines=0
current=0.02
format="%d-%d-%d %d:%d:%f, %00, %01, %02, %03"
query="%F%n%1%F%n%[Ready: ]fetch sleepafter=true%r%n"
column=conduc(10000,0)
column=temp(10000,0)
column=pressure(1000,0)
column=conducTemp(10000,0)
power-policy=1
cycles=0
```

logdev

- Configured with a .cnf file on the glider memory card
- .cnf file syntax: https://iop.apl.washington.edu/iopsg/logdev.txt
- .cnf file must be registered with the glider so that it becomes available as an option
  - Copy .cnf file to memory card, either directly or via ymodem
  - Register the .cnf file in the sensor library
    `param/config/logadd`
- Attach the device to an available serial port (USART2, 4, 5A,B,C,D, 6). If voltage is jumperable or software selectable (USART4) be sure to select the proper voltage or wire the appropriate pin.
- Configure the device in the glider software
  - `param/config/logger`
  - The device should be available in the list of loggers
  - Choose the correct serial port
- Test the device
  - `hw/logger/xx` (where xx is the prefix as defined in the .cnf file)
  - Use the edit menu option to test changes to .cnf values
    `edit timeout=300`
    This is a temporary change. Make the change permanent by editing the .cnf file.
  - You can edit the .cnf file directly on the glider:
    `!vi sensor.cnf`
  - Or make changes off the glider and re-upload the modified file to the memory card
    `!yr` (followed by ymodem send from your terminal program)
  - Use capture level debug to see additional sensor interactions
    `!capvec HNAME DEBUG BOTH`
- Logger sampling is controlled by parameters and any script files defined in .cnf
  - $LOGGERS - bit-field to activate/deactivate each logger slot
  - $XX_RECORDABOVE - depth above which the logger is activated (use 2000 to always run)
  - $XX_XMITPROFILE - which profile's data to send via Iridium (1=dive, 2=climb, 3=both)
  - $XX_PROFILE - which profile to run on (1=dive, 2=climb, 3=both)
  - $XX_NDIVE - which dives to run on (modulo, 1= every dive)
  - $XX_PARAM… - user parameters defined in .cnf file

```
name=NCP
prefix=cp
timeout=15000
baud=38400
warmup=400
voltage=15
current=0.025
wakeup=%F@@@@@%(200)@@@@@%(200)K1W%%!Q%(400)K1W%%!Q%(1000)%r%nMC%r%n%[OK]%F%(1000)%F
cmdprefix=$CP_
powerup-timeout=0
prompt=OK
datatype=u
start=%X
stop=%FMC%r%n%[OK]
profiles-download=merged
dive-start=%FMC%r%n%[OK]%(500)ERASETM,CODE=9999%r%n%[OK]%(500)
cleanup=%FMC%r%[OK]%(500)ERASETM,CODE=9999%r%n%[OK]%(500)
download="telemetryfile.bin"
downloader=ad2cp
metadata=%FMC%r%n%[OK]GETALL%r%n%[OK]
selftest=%FMC%r%n%[OK]GETALL%r%n%[OK]
clock-set=%FMC%r%n%[OK]SETCLOCK,YEAR=20%{%y,MONTH=%m,DAY=%d,HOUR=%H,MINUTE=%M,SECOND=%S}%r%n%[OK]
clock-read=%FMC%r%n%[OK]GETCLOCK%r%n%[OK]
clock-sync=gps2
post-clock=off
post-transfer=off
post-stop=off
script-x=NCP_GO
log-cmd-0=%FMC%r%n%[OK]RECSTAT,FC%r%n%[OK]
log-resp-0=%00
log-param-0=FREE
log-cmd-1=%FMC%r%n%[OK]GETPWR%r%n%[OK]
log-resp-1=%01,%f,%f,%02,%f,%f,%f
log-param-1=POWER
log-param-2=POWER1
```

Piloting 101

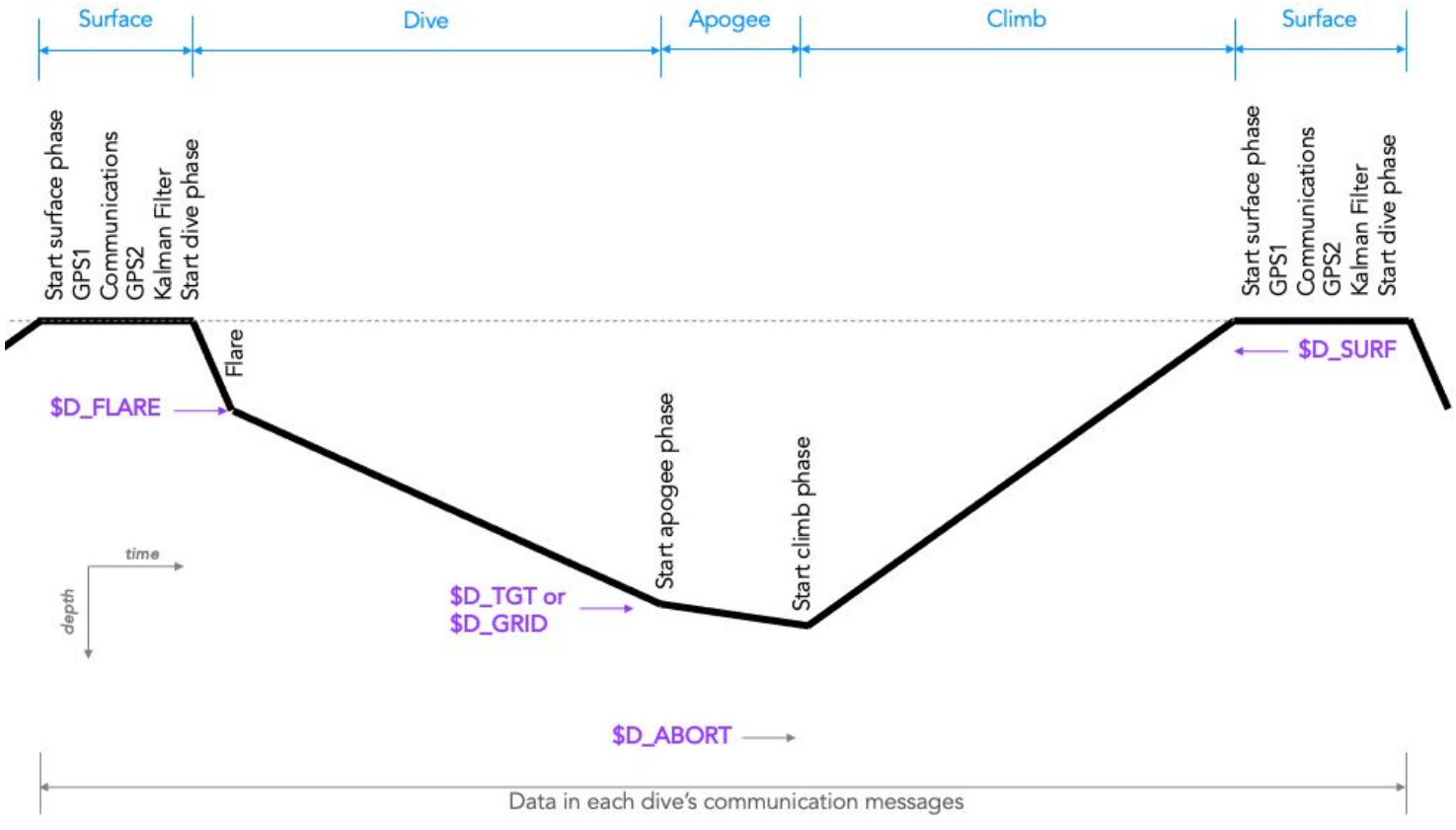
Pilot’s responsibilities: I

- Command glider to gather desired data from target area at proper space & time intervals
- Keep glider flying efficiently until end of mission (batteries depleted, reached pickup location, completed n-th dive, etc.)
- Modify glider’s commands/files/behavior to match environmental changes (e.g., currents, stratification, bottom depth, etc.)
- Continuously monitor flight and engineering data to determine cause of errors & problems and decide whether to continue mission
- Continuously monitor science data to ensure that sensors are functioning properly and mission science goals are being met

Pilot’s responsibilities: II

- Set up the mission directory on the basestation
- Adjust files controlling:
  - parameters for phases of dive, centers of systems, control modes (*cmdfile* file)
  - waypoints and associated details (*targets* file)
  - sampling scheme for each sensor (*science* file and/or *scicon.sch*)
- Use special files for interactive debugging or resending dive data (*pdoscmds.bat* file)

Phases of a typical Seaglider dive cycle



Controlling dive time and depth - fundamental **$PARAMETERS**

- Specify the desired dive time in minutes and dive depth in meters:
  - **$T_DIVE** (minutes)
  - **$D_TGT** (meters)
- These two parameters set the desired vertical velocity. A convenient bit of math is that **$T_DIVE** = **$D_TGT**/3 sets a vertical velocity of 10 cm/s.
- The actual target depth will be a function of **$D_TGT** and bathymetry (if loaded).
- The first two failsafes on these values are:
  - **$T_MISSION** (also minutes)
  - **$D_ABORT** (also meters)
- If time into the dive reaches **$T_MISSION**/2 before it reaches apogee (target depth) it will move to climb phase. If it reaches **$T_MISSION** before it reaches the surface it will move to surface phase.
- We typically set **$T_MISSION** to be 15-60 minutes longer than $T_DIVE
- If the glider reaches **$D_ABORT** at any time, it will immediately pump to maximum buoyancy and move to surface phase (go into recovery)

1. Launch (beginning of mission)

- Launch phase begins after field team has performed a selftest and initiated Sea Launch (more details later)
- Seaglider is in its surface position:
  - Rolled to neutral
  - Pitched fully forward (typically -30 to -60° pitch angle)
  - VBD pumped to max volume, $SM_CC is automatically reset to max cc's at launch. VBD AD counts will equal $VBD_MIN.
- Seaglider acquires the first GPS fix, "GPS1", and initiates a communications session via Iridium

2. Surface

Surface phase begins at the end of the climb-phase data acquisition and has the following steps:

- Surface maneuver (pitch fully forward, rolled to **$C_ROLL_CLIMB**, VBD pumped to **$SM_CC**)
- GPS1 acquired (unless **$T_GPS** minutes elapse without a GPS fix)
- Glider initiates Iridium call to basestation, files transferred between glider & basestation (more on this tomorrow)
- Surface depth and angle measured (average over 10 readings) and written to log file
- Second GPS fix (GPS2) acquired - this is the most recent position of the glider before diving
- Navigation & flight calculations: buoyancy, pitch angle, and heading computed for the next dive

After these steps, surface phase is completed and a new dive phase (new profile) starts

Path planning and heading determination

Once the glider has GPS2, it calculates the heading it will follow for the next dive.

- If using targets, desired goal heading is the bearing to current target
- If **$HEADING** >= 0 then targets are ignored and desired goal heading = **$HEADING**

Once the glider has a true heading to follow

- If **$NAV_MODE** = 0, glider uses the goal heading as the heading to steer in flight
- If **$NAV_MODE** = 2, glider applies a set correction, calculating a ferry angle using the presumed speed through water and the depth-averaged current from the previous dive to optimize travel toward the goal. **$FERRY_MAX** provides an upper bound on the ferry angle.
- **$NAV_MODE**,1 (Kalman) and **$NAV_MODE**,3 (current relative) are not commonly used but are also available.

Glider applies local magnetic declination from the IGRF model to convert to magnetic heading as observed with the compass to true heading for steering purposes.

Buoyancy and pitch calculations

Once the glider has GPS2, it knows the distance from its current position to its goal (either current target or a synthetic target 20 km distant if flying by **$HEADING**). It also knows its SPEED_LIMITS (as reported in the log file), the slowest and fastest possible speeds it can achieve without stalling and within the limits of $**GLIDE_SLOPE** and $**MAX_BUOY**. SPEED_LIMITS ➗ **$T_DIVE** provide bounds on the shortest and furthest distances the glider can travel in one dive.

- If distance to goal is greater than the maximum distance the glider can travel, then it will choose to fly at **$MAX_BUOY** and the shallowest glide angle it can achieve to maximize horizontal distance.
- If distance to goal is shorter than the maximum distance possible (generally the case when the target is close but not yet achieved), the glider will choose glide slope and buoyancy to try to surface as close as possible to the target while still diving to **$D_TGT** (or **$D_GRID** if shallower). This can lead to very steep, fast dives when the glider is close to the target.
- Practically, if **$HEADING** >= 0 the glider will always fly to maximize horizontal distance because the synthesized target is always 20 km away and the glider can never fly 20 km in a single dive.

3. Dive

- VBD bleeds (oil moves from bladder into reservoir) until depth **$D_FLARE** (typically ~3 m). Pitch is still full forward.
- At **$D_FLARE**, glider starts a regular G&C operation (pitch, VBD, roll - see next slide) to come to correct pitch for the desired glide slope and continues to bleed as necessary to come to desired negative thrust for the dive.
- If glider speed is too fast on the dive (i.e., glider is too dense), VBD pumping is not allowed (this is an energy consideration)
- If glider speed is too slow on the dive (i.e., glider is too buoyant), VBD bleeding is allowed only above **$D_NO_BLEED**. This is unusual because of stratification. Glider thrust is calculated for apogee density, so in the light water at the surface the glider is usually heavy and will be moving fast relative speeds as it moves deeper into denser water.
- Science sampling at interval specified in the *science* file (or *scicon.sch* file for gliders with scicon)
- Glider performs guidance and control (G&C) operations at the interval given in the *science* file

Seaglider guidance and control (G&C)

The glider performs G&C operations while it is profiling (dive, apogee,and climb phases)

- G&C operations occur at intervals defined in the science file
- Three operations during G&C, if necessary, in this order:
  - pitch adjustment
  - VBD adjustment
  - roll adjustment
- When G&C operations occur, the Seaglider is said to be in *active* G&C mode. In an active G&C phase, the glider moves each actuator sequentially as needed.
- When G&C corrections are not being made, the Seaglider is said to be in *passive* G&C mode
- G&C operations do not affect science sampling
- A summary of each G&C move is sent back in log file for each dive in the $GC lines
- The capture file contains detailed information about all G&C moves during the dive, including detailed records of motor movement AD counts, motor currents, etc. - this can be a useful diagnostic tool
- When a heading correction is needed, glider uses active mode to roll then switches to passive mode while turning. When desired heading is achieved, uses active mode to roll back to center.

Control adjustments during G&C mode

During a G&C interval, the glider might adjust

- Roll
  - to initiate a turn (if current heading is outside **$HEAD_ERRBAND**)
  - to roll back when a turn is complete
- VBD
  - If descending too slowly during dive and still above **$D_NO_BLEED**
  - If ascending too slowly climb
  - To adjust depth while loitering
  - To switch phase (dive to climb, dive to loiter, loiter to climb, etc.)
- Pitch
  - At **$D_FLARE**
  - To switch phase (dive to climb, etc.)
  - According to **$PITCH_ADJ_GAIN** and **$PITCH_ADJ_DBAND** to maintain desired pitch angle
  - According to **$PITCH_W_GAIN** and **$PITCH_W_DBAND** to maintain desired vertical speed (only on climb unless **$OPTIONS** bit is set)

Or the glider might do nothing if the glider is flying nominally.

4. Apogee

- Begins when **$D_TGT** (target depth) is reached or the glider reaches **$D_GRID** (see slides on bathymaps)
- Two G&C cycles to transition from dive to climb phase without stalling:
  (1) Glider pitched to a small intermediate angle (**$APOGEE_PITCH**, typically -5°), rolled to neutral, and VBD pumped to 0 cc. (No course adjustment)
  (2) Glider pitched and VBD pumped to inverse positions of the dive (pitch = -pitch, VBD = -VBD).

Data sampling is continued during apogee

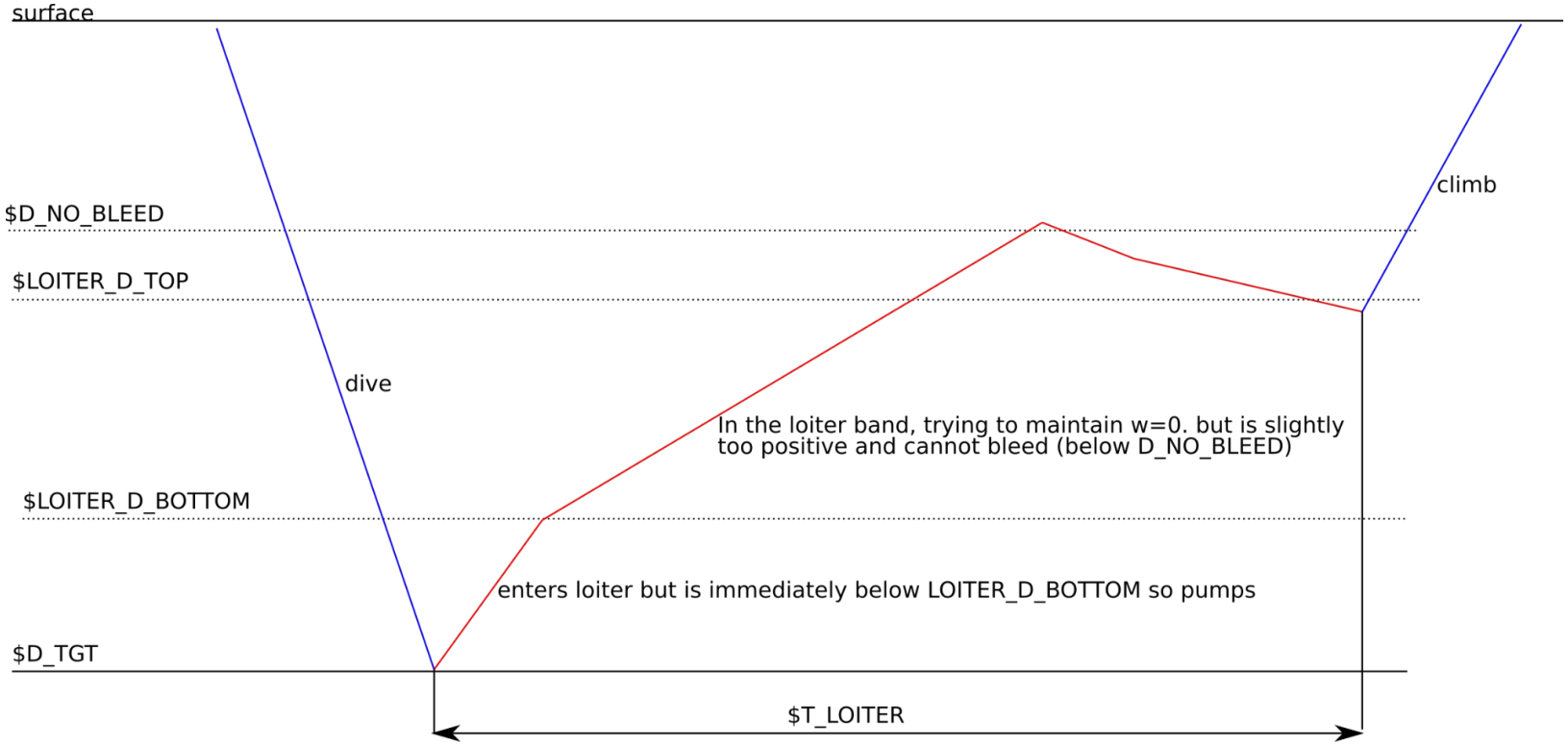
4.5. Loiter

If **$T_LOITER** and **$LOITER_N_DIVES** are set, then the glider will enter the loiter phase between the two apogee G&C phases (i.e., after the glider has gone to neutral buoyancy. While loitering the glider will attempt to maintain zero vertical velocity, pumping and bleeding as necessary to when velocity exceeds **$LOITER_W_DBAND** (or **$W_DBAND** if the loiter specific value is not set). **$D_NO_BLEED** is honored.

Additionally, the glider will pump and bleed regardles of vertical velocity if **$LOITER_D_TOP**, and or **$LOITER_D_BOTTOM** are set and the depth is above or below those bounds.

Use the special depth bin “loiter” in the science file to specify sampling and GC intervals while loitering.

Loiter behavior



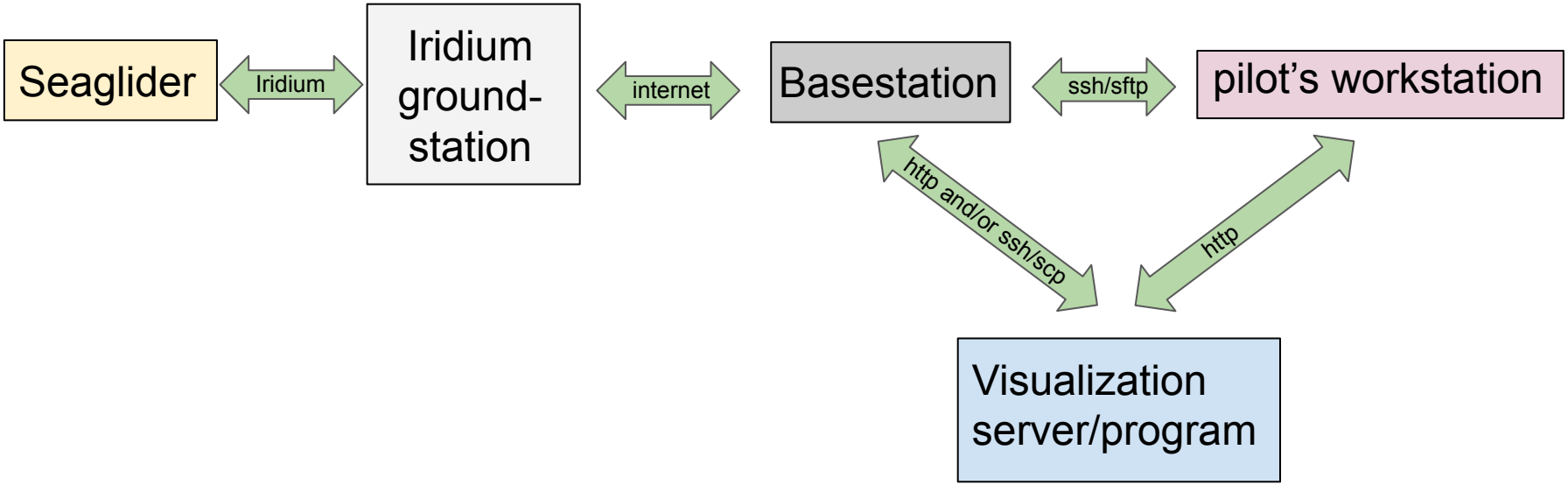
5. Climb

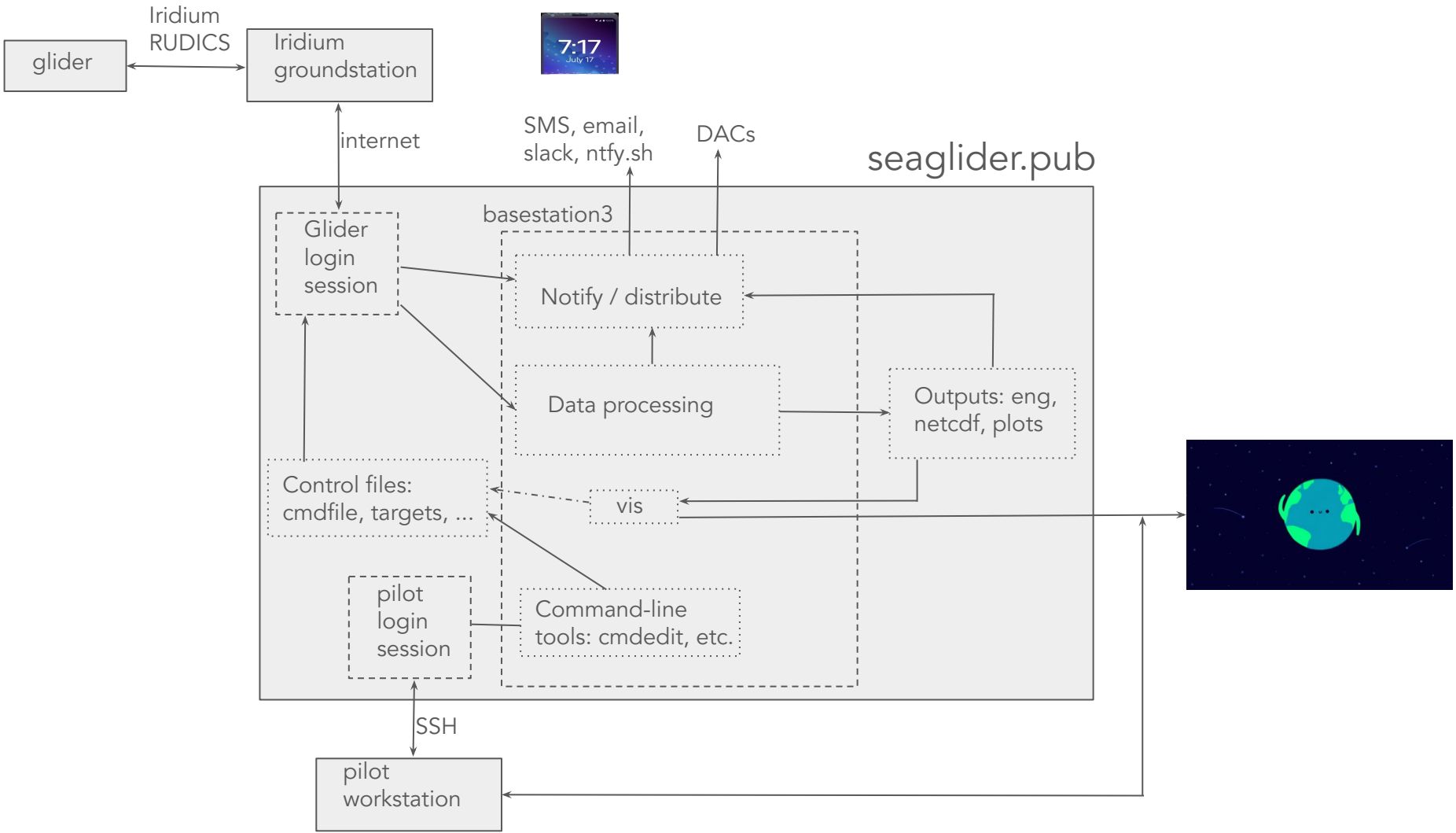
- Glider is positively buoyant and pitched up
- G&C at interval specified in the *science* file
- Science sampling at interval specified in the *science* or *scicon.sch* file
- If glider speed is too fast (i.e., glider too buoyant), VBD bleeding not allowed
- If glider speed is too slow (glider too dense), VBD will pump
- At depth **$D_SURF** glider enters passive G&C mode for time equal to **$D_SURF**/(observed vertical speed) or a maximum of 50 data points (whichever is shorter) then enters the surface phase

6. Recovery (end of mission)

- Recovery is entered either
  (1) When the pilot gives the **$QUIT** directive (when it is necessary/desirable to keep the glider at the surface), or
  (2) When the glider detects certain error condition, e.g.,:
    (a) Motor timeouts
    (b) **$D_ABORT** exceeded
    (c) Batteries exhausted
- In recovery, the glider obtains a GPS fix and makes an Iridium call every **$T_RSLEEP** minutes. (This process takes about 2 minutes, so in practice the time between phone calls is **$T_RSLEEP** + 2 minutes)
- The recovery loop is exited by replacing the **$QUIT** directive with the **$RESUME** directive in *cmdfile*; the glider will resume diving after its next call

Introduction to the Seaglider basestation


Seaglider
Iridium
Iridium ground-station
internet
Basestation
ssh/sftp
pilot's workstation
http and/or ssh/scp
http
Visualization server/program


glider
Iridium RUDICS
Iridium groundstation
7:17
July 17
internet
SMS, email, slack, ntfy.sh
DACs
seaglider.pub
basestation3
Glider login session
Notify / distribute
Data processing
Outputs: eng, netcdf, plots
Control files: cmdfile, targets, ...
vis
pilot login session
Command-line tools: cmdedit, etc.
SSH
pilot workstation

Seaglider basestation: Definitions

- Seaglider basestation is the shoreside infrastructure to support Seaglider communications, piloting and data processing. After each dive, Seaglider initiates calls to the basestation to pick up new commands for the next dive and download data from the previous dive. The basestation generates data files and plots, and can be used for reprocessing Seaglider data.
- Basestation server is the computer hardware serving as the network node on which basestation activities occur. Can be a cloud based resource or a computer in your lab. Every Seaglider has a user account on the server, as does every pilot. Seagliders connect via Iridium RUDICS or dial-up modem. Pilots usually connect via ssh.
- Basestation3 is the latest version of Seaglider basestation software. Open source, available at https://github.com/iop-apl-uw/basestation3. The design intent of Basestation3 is to be self-contained, providing all components necessary for glider operations, piloting and visualization. No matlab required.
- Vis is the visualization application that comes bundled with basestation3 to provide a web based interface for piloting and real-time and archival mission presentation. It runs on the basestation server, provides a separate public-facing web server (behind a proxy or directly), or over an ssh tunnel for single pilot use. No separate web or database server infrastructure required.
- seaglider.pub is the cloud-based community basestation server run by IOP. which all Seaglider users are welcome to use at no charge. It runs basestation3 software and provides a per-group instance of the vis application.

ssh, scp, sftp, WinSCP

- To connect to the basestation, use ssh (mac or linux terminal) or a Windows-based ssh program (https://learn.microsoft.com/en-us/windows/terminal/tutorials/ssh, or https://www.putty.org/, Teraterm, etc.)
- Options to transfer files from the basestation to your local machine:
  - connect to the basestation using sftp (mac/linux) or WinSCP (windows), navigate to the Seaglider's mission directory on the basestation, and use `mget` to copy files to your local machine
  - use scp (mac/linux) or putty scp/WinSCP (Windows) to copy files from the basestation to your local machine, e.g. to transfer all .nc files from the mission directory on Seaglider.pub to your local machine:

```
scp username@seaglider.pub:/home/jails/group_name/gliderjail/home/sgGGG/MyMissionName/\*nc /path/on/local/machine
```

Linux basics

- The basestation is a Linux server. Piloting requires some basic knowledge of Linux command-line operations.
- Pilots use ssh to access a command-line on the basestation server.
- Pilots should be familiar with
  - An editor, usually vi, but can be any of vi, emacs, joe, pico, nano, etc.
  - Directory (folder) commands)
    - `cd` (change directory)
    - `gcd` (special on seaglider.pub)
    - `pwd` (show working directory)
  - File commands
    - `ls` (list directory contents, like Windows “dir” command)
    - `mv` (move or rename a file, like Windows “ren” command)
    - `rm` (remove/delete a file, like Windows “del” command)
    - `cat` and `less` (to display file contents, like Windows “type” and “more” commands)
  - `grep` (to search through files), e.g.:
    - `grep C_VBD p2*.log`
  - `tar` and `gzip` to pack files for sharing, e.g.:
    - `tar cvzf engfiles.tgz p2*.eng`
    - `gzip -dc sg0001kz.x`
  - `scp` to transfer files to another machine
- Help on file commands can be found by typing `man command_name` e.g., `man rm`
- There are also lots of great tutorials online

vi basics

- Basestation programs to edit *cmdfile*, *targets*, and *science* files (`cmdedit`, `targedit`, and `sciedit`) by default use vi
- To open a file in vi, type `vi filename` (or just `cmdedit, sciedit` or `targedit` if using those tools)
- vi has 3 modes:
  - **vi mode** – the mode vi starts in. Regular keyboard arrows work in this mode. You can quickly navigate through the file and delete/copy/paste. Basic commands:
    - `gg` go to start of file
    - `G` go to end of file
    - `dd` delete current line
    - `5dd` delete 5 lines
    - `yy` copy (“yank”) current line
    - `p` paste copied or deleted line (below current line)
    - `u` undo
    - `<CTRL>+r` redo
  - **command mode** – get to command mode by typing the colon key (“`:`”). Basic commands:
    - `:w` write (save) file
    - `:q` quit
    - `:q!` quit without saving
    - `:wq` write (save) file and quit
  - **input mode** – edit text. Leave input mode and enter vi mode by typing the `<esc>` key. Some ways to get to input mode:
    - `i` input text at cursor
    - `o` insert line below cursor
    - `O` insert line above cursor

nano basics

- Basestation programs to edit *cmdfile*, *targets*, and *science* files (`cmdedit`, `targedit`, and `sciedit`) by default use vi
- To open a file in nano, type `nano filename`(or just `cmdedit, sciedit` or `targedit` if using those tools and default EDITOR is nano)
- User cursor (arrow keys to move around)
- Typing inserts at cursor location
- Use menu shortcuts shown on bottom of screen: `^` means ctrl key, `M` means esc (escape) key.

Basestation structure

- Every glider has its own home directory: `sgGGG`

  On seaglider.pub the home directory is:

  ```
  /home/jails/group_name/gliderjail/home/sgGGG
  ```

  On the IOP basestation used in this training the home directory is:

  ```
  /home/trainjail/gliderjail/home/sgGGG
  ```

  On the typical basestations the home directory is:

  ```
  /home/sgGGG
  ```

- Glider sends files to the current mission directory, which is usually a sub-directory inside the home directory. Set this up using **NewMission.py** script (otherwise, default mission directory is the glider's home directory)
- On seaglider.pub use the "`gcd sgGGG`" command shortcut to change to the glider's current mission directory.
- On a basestation without jails, use "`cd ~GGG`" to change to glider's home directory.

Basestation CLI basics

- 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`
- seaglider.pub has a shortcut installed that handles the paths for you: **`basepy:`**
  - `basepy 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.:
  - `basepy 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.

```
basepy 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.
- **compare.py**: Compare current parameter values to canonical values

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

Basestation files

File types

There are several categories of files on the basestation:

(1) Control files: the pilot can make edits/changes to the glider's operational parameters using the control files, which are located on the basestation. The glider downloads the control files each time that it calls the basestation.
(2) Log and Data files: The glider uploads log and data files it generated on the previous dive(s) to the basestation.
(3) Basestation control files: Files located on the basestation that control the generation of messages and emails containing glider health and location information, instructions on file types to be produced by basestation software, or IP addresses where data should be sent for archive or further processing.
(4) Basestation output files: Processed science and engineering data files, intermediate products and plots generated on the basestation from the glider data files.
(5) Basestation log files: Files generated on the basestation that record all interactions between the glider and the basestation, the interaction between the basestation and the RUDICS connection, and the status of glider data file processing on the basestation.

Control files

Command file	*cmdfile*
Science file	*science*
Targets file	*targets*
Scicon files (if using the science controller)	*scicon.sch* *scicon.ins* *scicon.att*
PicoDOS command file	*pdoscmds.bat*
Compass calibration file	*tcm2mat.cal*

Every time the glider calls in, the basestation uploads any control files onto the glider and renames the file on the basestation <name>.ddd.nnn, where ddd is the dive number and nnn is the call cycle. This allows the pilot to see a complete history of the control files that have been sent to the glider.

Control files: Command file (*cmdfile*)

- The command file (*cmdfile*) is used to modify parameter values during a mission to achieve the desired glider behavior.
- Parameters are specified as a list of **$NAME,value** pairs (see Parameter and Command File Directive Reference Manual: https://iop-apl-uw.github.io/basestation3/html/Parameter_Reference_Manual.html)
- *cmdfile* should always have a valid state directive (**$GO**, **$RESUME** or **$QUIT**) as the last line, but need not have anything else
- Edit on basestation with `cmdedit`, which provides some bounds checking and verification of changes
- Typical example early in mission:

- Can be dangerous as pilot can override safeguards, e.g.:

```
$D_TGT,4500
$D_ABORT,6000
$T_MISSION,6000
```

… Will this glider ever resurface?

Control files: Command file directives

$GO	causes the glider to continue in its current mode of operation.
$RESUME	causes the glider to resume diving from within recovery phase, using its current set of parameters. It acts like the **$GO** directive if the glider is diving. Takes one optional comma separated argument that is inserted into the log file as a comment.
$QUIT	causes the glider to go immediately to recovery mode. The glider will hold at the surface, sleeping **$T_RSLEEP** minutes between the end of one communications session and the start of the next.

cmdfile parameters commonly edited

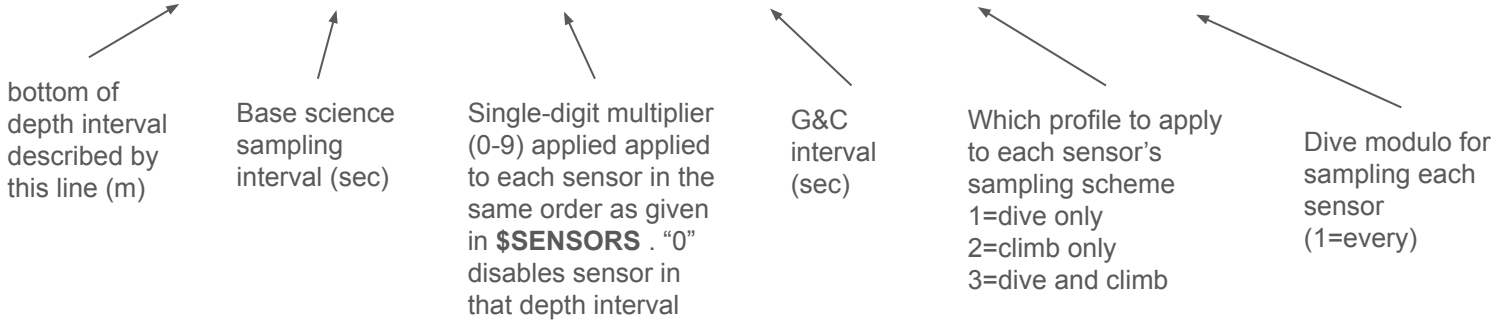
Depth/time of dive	Flight and trim	Buoyancy boundaries	Navigation	Comms behavior
$D_TGT $T_DIVE $T_MISSION $D_ABORT $N_DIVES	$C_PITCH $PITCH_GAIN $C_VBD $C_ROLL_CLIMB $C_ROLL_DIVE	$MAX_BUOY $SM_CC	$HEADING $NAV_MODE	$T_RSLEEP $N_NOCOMM $NOCOMM_ACTION

Control files: Science file (*science*)

The science file specifies intervals for guidance and control (G&C) and for science sampling (if glider doesn't have scicon; if scicon is installed, the glider will use the G&C interval supplied in *science* but will ignore the science sampling)

Each line of the science file describes one depth interval. Each term is separated by whitespace. Example:

```
200   seconds=5    sensors=12   gc=120   profile=33   dives=11
500   seconds=10   sensors=12   gc=200   profile=33   dives=11
1000  seconds=30   sensors=11   gc=300   profile=31   dives=11 compass=2 pressure=1
```



Edit *science* on the basestation with **sciedit**, which provides some bounds checking and verification of changes

Control files: Science file (*science*)

- Practical limits on sampling:
  - No shorter than 2 seconds for CT
  - Other sensors may take longer due to sensor warmup time or timeout (e.g., WETlabs)
  - Typical “fast” sampling interval is 5 seconds
  - Upper bound is dependent on pressure sensing, with respect to depth triggers on behavior

Scicon scheme file (*scicon.sch*)

- For gliders with a science controller (scicon) installed, *scicon.sch* provides the science sampling intervals. science is still used to control truck sampling and GC interval.
- Scicon enables greater flexibility and specificity:
  - Sampling intervals can be controlled as a function of depth bins, profile (dive, climb, loiter), and dive number.
  - Per-scheme configuration information can also be specified (for example to change ADCP parameters on dive and climb).
  - An attached instrument can have multiple schemes defined. The active scheme will be chosen in order of decreasing specificity of profile and dive definitions.
- Each sensor has its own sampling protocol. Example (see scicon manual for details)

```
ct = {
   50, 4.0
   200, 7.5
   1000, 14.0
}
ad2cp = {
   profile = b
   conf = SETAVG,CH=234%r%n%[OK]
   conf = SAVE,ALL%r%n[OK]
   1000,15.000000
}
```

Sensor type as defined in *scicon.ins*

bottom of depth bin, sampling interval (sec)

sensor type as defined in *scicon.ins*

which profile this scheme applies to (a=dive, b=climb, c=loiter)

configuration strings to send before startup

bottom of depth bin, sampling interval (sec)

Control files: Targets file (*targets*)

Targets file provides waypoints for the glider's flight path **$HEADING** < 0 (i.e., when navigating by waypoints). It includes (one line per waypoint:

- Waypoint name
- Waypoint latitude in degrees and decimal minutes (no spaces)
- Waypoint longitude in degrees and decimal minutes (no spaces)
- Watch circle (m) surrounding the waypoint that the glider must be in to be considered reaching the waypoint
- Name of next waypoint

Example:

```
/Shilshole targets (small rectangle)
SE  lat=4743.0 lon=-12224.0 radius=100 goto=NE
NE  lat=4743.5 lon=-12224.0 radius=100 goto=NW
NW  lat=4743.5 lon=-12225.0 radius=100 goto=SW
SW  lat=4743.0 lon=-12225.0 radius=100 goto=SE
```

Edit *targets* on the basestation with **targedit**, which provides some bounds checking and verification of changes

Control files: *pdoscmds.bat*

pdoscmds.bat file is created by the pilot only when needed. It contains one or more lines, each of which is a command that the glider runs in order.

If a pdoscmds.bat file is on the basestation, the glider picks it up near the beginning of its next call, renames the file on the basestation *pdoscmds.bat.nnn.ccc* (where nnn is the dive number and ccc is the call cycle), and immediately runs the specified commands. Any output from the action will be printed to a file called *pGGGdddd.ccc.pdos*

Some common uses of the pdoscmds.bat file:

- Resend select files from the glider (e.g., dive files or capture file)
  - `resend_dive /c 233`
- Turn on/off debug mode of glider subsystems in capture file
  - `capvec HCOMPASS DEBUG BOTH`
- Change the target the glider is heading to in the loaded targets file
  - `target NW`
- Write the specified value of a specified parameter to NVRAM
  - writenv SMSemail user@school.edu
- Search `capture files for critical errors`
  - `xargs dv001? sg*kz.a grep /s ,C,`
- Query a single parameter
  - `$T_DIVE`

A complete list of commands that are available through epdos is included in
https://iop-apl-uw.github.io/basestation3/html/epdos_Reference_Manual.html

Seaglider data file naming conventions

In general, all files generated on the seaglider (and a few generated on the basestation) follow a naming convention:

```
ooxxxxtf.s
```

Where:

- `oo` - the instrument or system that created the file
  - sg - seaglider, st - seaglider selftest, sc - scicon, pm - PMAR are examples
- `xxxx` - the dive number
- `t` - the type of file
  - `l` - logfile, `d` - data file, `k` - capture file, `a|b|c|d` - profile files from a logger
- `f` - the file packing or compression
  - `u` - uncompressed, `z` - gzip, `t` - tarfile, `g` - gzipped tarfile
- `s` - the transmission status of the file
  - `a` - archive (leave on glider), `x` - transmit to basestation, `xii` - fragment (part of a file) to transmit, `r` - received on basestation

Basestation file naming conventions

In general, files on that are created on the basestation, all follow a naming convention:

```
p|pt[oo]xxxdddd[a|b|c|d][_class_instance].ext
```

Where:

- `p|pt` - indicates “processed” or “processed selftest”
- `oo` is the the instrument or system that created the file (omitted for `sg`)
- `xxx` - seaglider 3 digit serial number
- `dddd` - dive number
- `[a|b|c|d]` profile identifier for loggers
- `_class_instance` identifies the sensor type (class) and instance name for scion
- `.ext` - file type

File flow to the basestation

- Files that are sent to the basestation are split into smaller files (fragments), named with the `.xii` extension (where *ii* is a hexadecimal number starting with `.x00`)
- The basestation reassembles the fragments into the full original file and checks for integrity resulting in the original file (`.r`)
- Files are decompressed (and extracted if a tarfile)
- Logfiles (`l`) and capture files (`k`) are copied to the basestation naming (`.log` and `.cap`)
- Datafiles (`d`) are converted to engineering files (`.eng`) by summing columns and reversing any scaling and offset applied on the Seaglider (or logger) and converted to basestation naming
- Logfiles and engineering files from each dive are processed (quality control, calibrations applied, corrections applied) into a per-dive netcdf file (`.nc`)

Data files

Capture file	*.cap*
Log file	*.log*
Glider data file	*.dat*
ASCII file	*.asc*
Engineering file	*.eng*
Private file	*.pvt*
NetCDF file	*.nc*
Scicon data directories*	*scxxxxa/* *scxxxxb/*
Scicon data files*	*.dat* *.eng*

Files generated by Seaglider

Files generated by the basestation

Directories/files generated by the basestation using scicon data

* only if glider has a science controller (scicon)

Basestion control files

Calibration file	*sg_calib_constants.m*
Map plot config file	*sg_plot_constants.m*
Sections file	*sections.yml*
Pagers file	*.pagers* *pagers.yml* (basestatation3)
Basescript script config file	*sgGGG.cnf*
Email file*	*.mailer*
URL file*	*.urls*
Extensions files*	*.extensions, .pre_extensions*

* not commonly used - all functions are available in pagers file

Basestation control files: *sg_calib_constants.m*

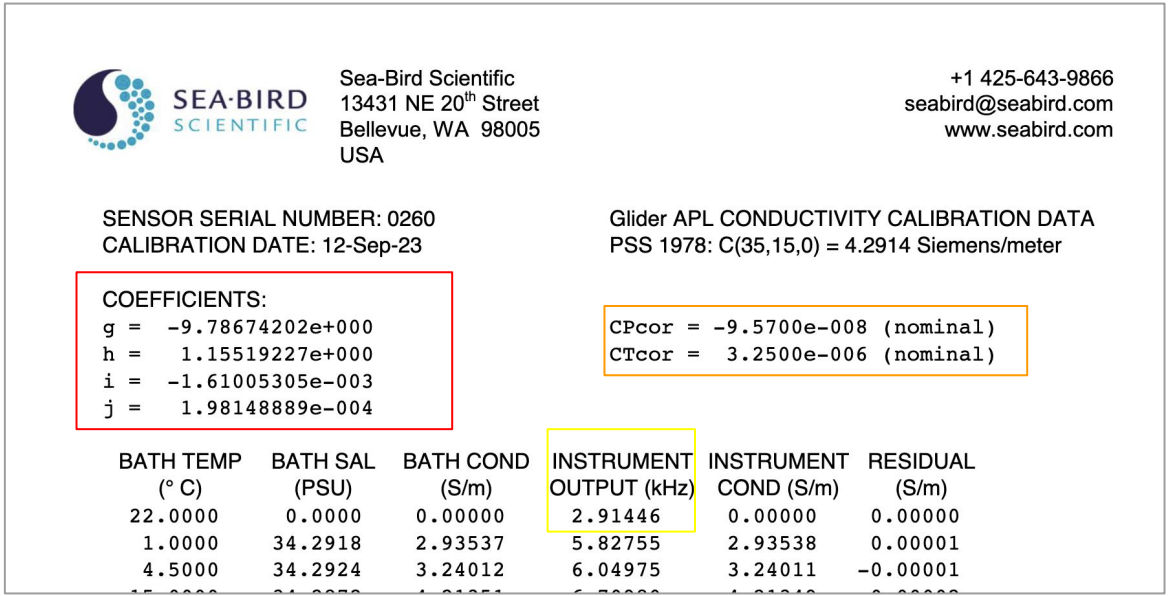
sg_calib_constants.m is used to specify sensor calibration coefficients and basic vehicle configuration that is not captured by glider parameters. It uses matlab-style .m syntax for historical reasons.

- Always include the following:
  - `mass = 72654;` % scale mass in grams as flying
  - `mission_title = 'string';` % usually 'ProjectName Month-Year', used in plot titles, netcdf file names
  - `id_str = '249';`
- Have a RBRlegato CTD?
  - `sg_ct_type = 4;` % Unpumped RBR legato
  - `legato_sealevel = <value as reported in selftest>;`
  - Run **GetLegatoPressCorr.py** to determine the sg_calib_constants.m value for `legato_cond_press_correction`
- Wetlabs calibration coefficient naming is messy - see **sg000/sg_calib_constants.m**
- Have an Aanderaa optode?
  - Run **GetOptodeConstants.py** to get the sg_calib_constants.m values for the optode from a selftest basepy

Basestation control files: *sg_calib_constants.m*

- Have a Sea-Bird CT sail? Include coefficients from the calibration sheets. These must also be included as parameters on the glider (**$SEABIRD_[C_G/ C_H/ C_I/ C_J/ T_G/ T_H/ T_I/ T_J]**). Example:

```
calibcomm = 'SBE s/n xxxx, calibration 12-Sep-2023;
t_g =  4.40953569e-0030260;
t_h =  6.45635007e-004 ;
t_i =  2.58444570e-005 ;
t_j =  3.22925756e-006 ;
c_g =  -9.78674202e+000 ;
c_h =  1.15519227e+000 ;
c_i =  -1.61005305e-003 ;
c_j =  1.98148889e-004 ;
cpcor = -9.57e-08 ;
ctcor =  3.25e-06 ;
sbe_cond_freq_C0 = 2914.46 ;
```



Sea-Bird Scientific
13431 NE 20th Street
Bellevue, WA 98005
USA

+1 425-643-9866
seabird@seabird.com
www.seabird.com

SENSOR SERIAL NUMBER: 0260
CALIBRATION DATE: 12-Sep-23

Glider APL CONDUCTIVITY CALIBRATION DATA
PSS 1978: C(35,15,0) = 4.2914 Siemens/meter

COEFFICIENTS:
g = -9.78674202e+000
h = 1.15519227e+000
i = -1.61005305e-003
j = 1.98148889e-004

CPcor = -9.5700e-008 (nominal)
CTcor = 3.2500e-006 (nominal)

BATH TEMP (° C)	BATH SAL (PSU)	BATH COND (S/m)	INSTRUMENT OUTPUT (kHz)	INSTRUMENT COND (S/m)	RESIDUAL (S/m)
22.0000	0.0000	0.00000	2.91446	0.00000	0.00000
1.0000	34.2918	2.93537	5.82755	2.93538	0.00001
4.5000	34.2924	3.24012	6.04975	3.24011	-0.00001

Basestation control files: *.pagers* (pre-basestation3)

The pagers file (*.pagers*) is used to transmit the most recent Seaglider surface position to shore via email:

Format:

```
email_address,notification1[,notification2,notification3]
```

Many cell phone carriers have email to SMS gateway services so that SMS can be sent via an email address, e.g. (in the US):

Verizon: `phone_number@vtext.com`

AT&T: `phone_number@mms.att.net`

T-mobile: `phone_number@tmomail.net`

Iridium also provides the capability to send an SMS to an Iridium handset:

```
8816nnnnnnnn@msg.iridium.com
```

In our experience, reliability of email to SMS gateways is deteriorating. There are other apps that may be more reliable (e.g., **ntfy**), available via the pagers.yml mechanism.

Basestation control files: *.pagers* notification types

`gps` *	GPS position (short message); GPS in DDMM.MM format
`recov` *	notification that the glider is in recovery for any reason, includes GPS fix in DDMM.MM format
`critical` *	notification if the glider is in a non-quit recovery, has rebooted, or has uploaded a capture file with any errors
`alerts`	notifications of any problems or issues that occurred during basestation processing; non-critical but important conditions the glider has encountered; capture files that contain critical errors. Pilots should always subscribe to alerts, as it is the primary way for the basestation to issue errors
`comp`	notification of completion of processing and a list of resulting files

`gpsdd|recovdd|criticaldd`	*same as above but GPS fix formatted in DD.DDDD format
`gpsddmm|recovddmm|criticalddmm`	*same as above - GPS fix formatted in DD MM.MM format
`gpsddmmss|recovddmmss|criticalddmmss`	*same as above but GPS fix formatted in DD MM SS.SS format

Basestation control files: *pagers.yml* (basestation3)

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/pagers.yml`)
- 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)

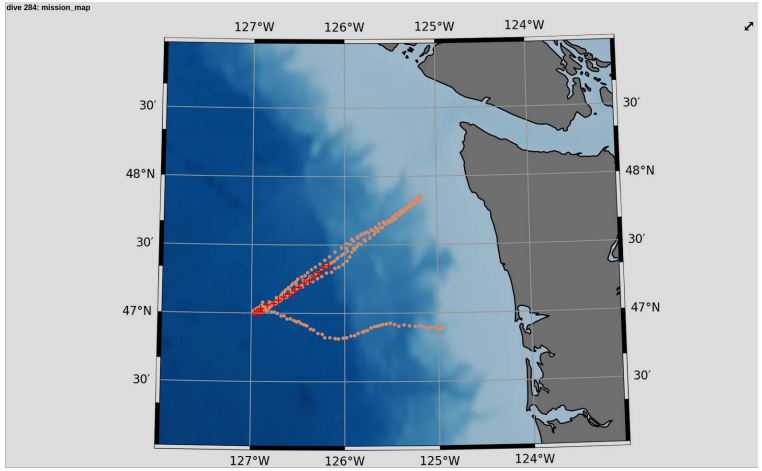
```
universal:
  status: on
  mattermost: [ { filters: [gps],  hook: https://server.og/mattermost/hooks/token1 },
                { filters: [alerts], hook: https://server.org/mattermost/hooks/token2 },
                { filters: [critical], hook: https://server.org/mattermost/hooks/token3 }]

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]
```

sections.yml - time-depth section data plots

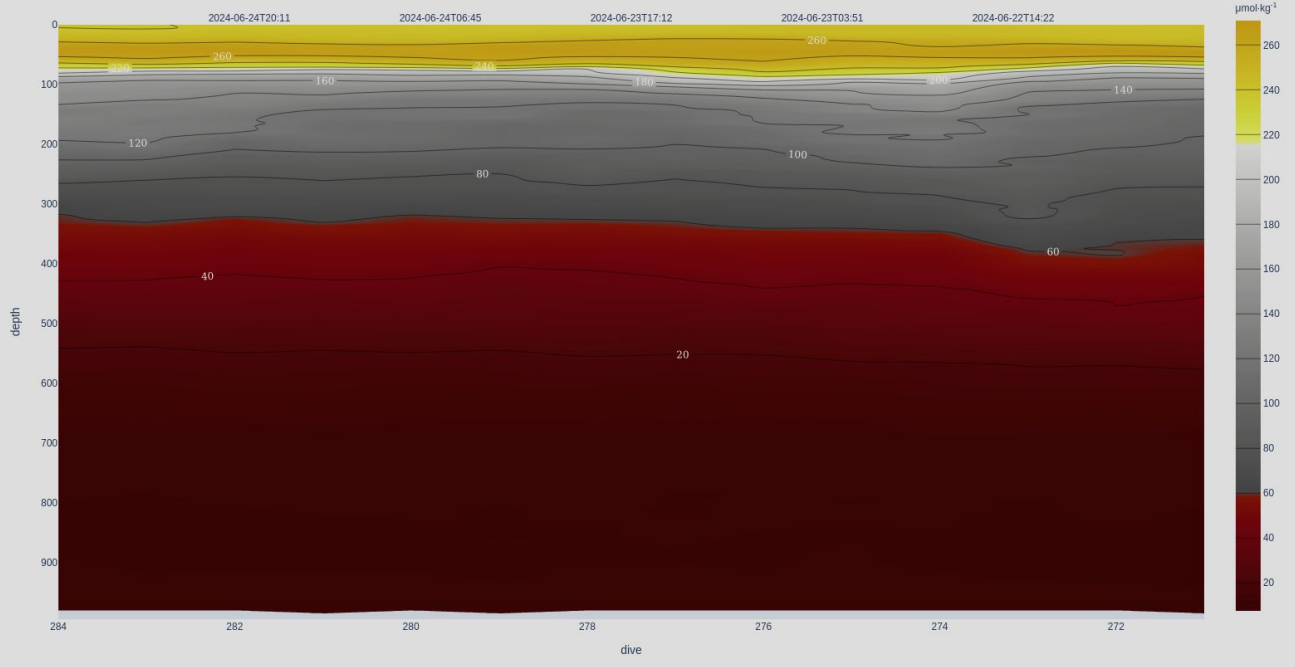
```
---
defaults:
  top: 0
  bottom: 990
  bin: 5    ← bin size (meters)
  step: 1   ← dive increment
  which: 4 ← which profiles (1=dive, 2=climb, 3=both, 4=combine, default is 4)

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
```



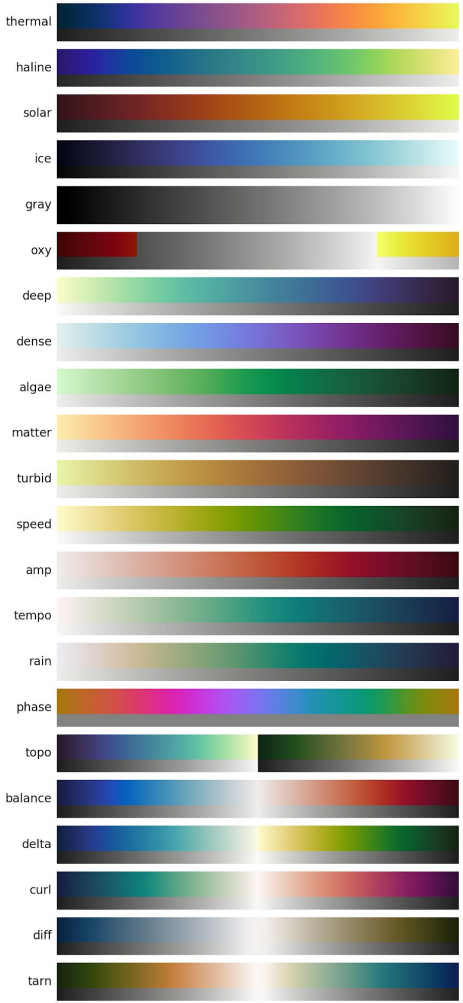
cmocean colormaps: https://matplotlib.org/cmocean/

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

Basestion log files

Communications log file	***comm.log***
Basestation log files	*baselog_login_YYMMDDHHMMSS* *glider_early_gps_YYMMDDHHMMSS* *baselog_YYMMDDHHMMSS* ***baselog.log***
Processed dives file	*processed_dives.cache*
Alerts	*alert_message.html.ddd.ccc*

Basestation log files: *comm.log*

The comm.log is a running record of all calls from the glider to the basestation. It is written in realtime, so it is useful for pilots to monitor this file continuously, e.g. from the basestation using the command (it is also updated in the pilot view on vis)

```
tail -f comm.log
```

Example of a message in the comm.log:

```
Connected at 2024-08-05T22:24:24Z (sg249)
logged in
437:0:3:1:2:37:0:428:2290:501:-44.03:1.17:9.30:15.00:15.13:8.56:50.35 GPS,050824,221233,4712.773,-12638.181,25,1.5,28,15.7
ver=67.01,rev=7036M,frag=8,launch=110424:161929
Iridium bars: 4 geolocation: 4713.150391,-12633.806641,050824,221551
2024-08-05T22:24:36Z [sg249] Sending 318 bytes of cmdfile
2024-08-05T22:24:36Z [sg249] Sent 318 bytes of cmdfile
Parsed GO from cmdfile
2024-08-05T22:24:43Z [sg249] ready to receive 1 files
2024-08-05T22:24:45Z [sg249] Receiving 7168 bytes of sc0436bg.x01
2024-08-05T22:25:06Z [sg249] Received 7168 bytes of sc0436bg.x01 (334.6 Bps)
2024-08-05T22:25:06Z [sg249] OK
437:0:3:1:2:37:1 logout
Disconnected at 2024-08-05T22:25:09Z (normal)
```

See next slide

Info about motherboard and code versions

Iridium strength and geolocation position

cmdfile is always the first file sent

Directive in the cmdfile

List of file fragments sent and their size. Pilot can verify that size sent = size received.

OK signals end of sending file

Logout message
Time and status of disconnect.
`(hangup)` indicates call was dropped.

Basestation log files: *comm.log*

The line after "`logged in`" contains useful information about the position of the glider and its motors. Exact formatting of the "colon" portion of the line has changed slightly over time. In the most recent code, the format is as follows:

diveNum, callCycle, callsMade, NoComm, missionNum, bootCount, lastFileError, pitchAD, rollAD, vbdAD, angle, depth, temperature, v10, v24, intPress, rh

E.g., from the previous slide:

```
437:0:3:1:2:37:0:428:2290:501:-44.03:1.17:9.30:15.00:15.13:8.56:50.35
```

The most recent fix from the GPS follows the status information:

GPS,date (DDMMYY),time (HHMMSS),lat (DDMM.MMM),lon (DDMM.MMM),timeToFirstValidFix,HDOP,timeToFinalFix,magneticDeclination

E.g.:

```
GPS,050824,221233,4712.773,-12638.181,25,1.5,28,15.7
```

Example of glider mission directory after dive 5

```
.                       cmdfile.3       p2640003.pvt            pt2640003.dat     sc0002ag.x      sc0004bg.x      sg0002lz.r            st0001kz.r
..                      cmdfile.4       p2640004.asc            pt2640003.eng     sc0002at.r      sc0004bt.r      sg0002lz.x            st0001kz.x00
.extensions             cmdfile.5       p2640004.dat            pt2640003.log     sc0002b         sc0005a         sg0003du.r            st0001kz.x01
.ftp                    comm.log        p2640004.eng            pt2640003.pvt     sc0002bg.1a.x   sc0005ag.1a.x   sg0003dz.r            st0001kz.x02
.mailer                 flight          p2640004.log            pt2640003.pvtst   sc0002bg.r      sc0005ag.r      sg0003dz.x            st0001kz.x03
.pagers                 p2640000.log    p2640004.nc             pt2640004.asc     sc0002bg.x      sc0005ag.x      sg0003lz.r            st0001lz.r
.post_dive              p2640000.prm    p2640004.pvt            pt2640004.cap     sc0002bt.r      sc0005at.r      sg0003lz.x            st0001lz.x
.post_mission           p2640000.pvt    p2640005.asc            pt2640004.dat     sc0003a         sc0005b         sg0004du.r            st0003du.r
.pre_extensions         p2640001.asc    p2640005.dat            pt2640004.eng     sc0003ag.1a.x   sc0005bg.1a.x   sg0004dz.r            st0003dz.r
.pre_login              p2640001.dat    p2640005.eng            pt2640004.log     sc0003ag.r      sc0005bg.r      sg0004dz.x            st0003dz.x
.urls                   p2640001.eng    p2640005.log            pt2640004.pvt     sc0003ag.x      sc0005bg.x      sg0004lz.r            st0003kz.r
alert_message.html.1.1  p2640001.log    p2640005.nc             pt2640004.pvtst   sc0003at.r      sc0005bt.r      sg0004lz.x            st0003kz.x
alert_message.html.1.2  p2640001.nc     p2640005.pvt            sc0001a           sc0003b         sections.yml    sg0005du.r            st0003lz.r
alert_message.html.1.5  p2640001.pvt    pagers.yml              sc0001ag.1a.x     sc0003bg.1a.x   sg0000kl.r      sg0005dz.r            st0003lz.x
alert_message.html.3    p2640002.asc    plots                   sc0001ag.r        sc0003bg.r      sg0000kl.x      sg0005dz.x            st0004du.r
alert_message.html.5    p2640002.dat    processed_files.cache   sc0001ag.x        sc0003bg.x      sg0000lz.r      sg0005lz.r            st0004dz.r
cmdfile                 p2640002.eng    pt2640001.asc           sc0001at.r        sc0003bt.r      sg0000lz.x      sg0005lz.x            st0004dz.x
cmdfile.0               p2640002.log    pt2640001.cap           sc0001b           sc0004a         sg0001du.r      sg264.db              st0004kz.r
cmdfile.0.5             p2640002.nc     pt2640001.dat           sc0001bg.1a.x     sc0004ag.1a.x   sg0001dz.r      sg264.kmz             st0004kz.x
cmdfile.1               p2640002.pvt    pt2640001.eng           sc0001bg.r        sc0004ag.r      sg0001dz.x      sg_calib_constants.m  st0004lz.r
cmdfile.1.1             p2640003.asc    pt2640001.log           sc0001bg.x        sc0004ag.x      sg0001lz.r      sg_plot_constants.m   st0004lz.x
cmdfile.1.2             p2640003.dat    pt2640001.pvt           sc0001bt.r        sc0004at.r      sg0001lz.x      sms_messages.log      tcm2mat.cal
cmdfile.1.5             p2640003.eng    pt2640001.pvtst         sc0002a           sc0004b         sg0002du.r      st0001du.r            tcm2mat.cal.4
cmdfile.2.1             p2640003.log    pt2640003.asc           sc0002ag.1a.x     sc0004bg.1a.x   sg0002dz.r      st0001dz.r            tcm2mat.cal.5
cmdfile.2.7             p2640003.nc     pt2640003.cap           sc0002ag.r        sc0004bg.r      sg0002dz.x      st0001dz.x
iopbase3%
```

Mission planning and preparation

Overview of a Seaglider mission

1. Prepare the basestation
2. Prepare the glider
3. Run a self-test and review the results
4. Put glider into “sea launch”
5. Deploy the glider
6. Review the data
7. Recover the glider

Field team responsibilities

- Check the weather and tides
- Communicate with pilot team
- Install antenna, rudder and wings in the field
- Conduct the selftest and (when instructed) Sea Launch
- Deploy and recover the vehicle
- Disassemble, clean and stow the vehicle

1. Prepare the basestation (pilot team)

- If previous mission data are in the glider home directory, run **MoveData.py**
- Run **NewMission.py** to set up a new mission directory
- Check basestation files:
  - *sg_calib_constants.m*
  - *targets*
  - *science*
  - *.pagers/pagers.yml*
  - *cmdfile*
  - If glider has scicon: *scicon.ins, scicon.att, scicon.sch*
- Set up glider and paths in *missions.yml*

Prepare the basestation: 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 missions.new` (copy missions.yml to a temporary file called missions.new)
  - Edit missions.new
  - Run **vis.py** in test mode to check for syntax errors:

    ```
    basepy vis.py -r ./ -f missions.new -t
    ```

  - If the changes are ok, `cp missions.new missions.yml` to copy the temporary file 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.

missions.yml example

```
---
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} ]
```

2. Prepare the glider (field team)

- Attach wings and rudder
- Remove sensor covers
- Remove hatch cover and check that bulkhead connectors are finger tight (antenna connector is pliers tight)
- Ensure antenna is pointed up and has a clear view of the sky
- Go into direct communication with glider:
  - Connect comms cable from connector at base of antenna to laptop
  - Start terminal program on laptop with session log/capture turned on
  - Connect at 9600 baud (RevB motherboard) or 115200 baud (RevE motherboard)
- Turn on the glider (magnetic wand or shorting plug/cable)

3. Run a selftest

- A selftest sequentially checks out each subsystem of the glider, and sends the result to the basestation via an Iridium call
- Field team runs a selftest with `launch/autotest` (or `launch/selftest` for interactive selftest). If comms fail, field team can resend the selftest results with `launch/uploadst`
- *cmdfile*, *science* and *targets* files are not uploaded to the glider prior to the start of the test, so parameter, science and targets values presently residing on the glider are used for the test
- *cmdfile*, *science* and *targets* are uploaded to the glider at the conclusion of the test when the gliders calls in the test results
- Selftest results are found in file *ptGGG0nnn.cap* , where *nnn* is the selftest number that increments by one with each test
- Only when all issues in the selftest have been resolved should launch continue

Vis 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 on the basestation (set in the *~/.bashrc* file) so you can use vim, emacs, joe, pico, nano, etc.

4. Sea Launch

If selftest results are fine (or problems discovered in selftest have been resolved) and the pilot team agrees, the glider is ready to be deployed:

- Pilot team instructs field team to begin Sea Launch
- Field team begins Sea Launch: `launch/sea`

* Sea Launch will delete all data files in the root directory on the glider's memory card - make sure they are archived (copied to the basestation or moved to the glider's archive directory (from pdos: `mkdir archive` followed by `mv dv0* archive`) BEFORE getting to this stage.

Sea Launch generates a *.prm* file that is a special version of a capture file containing a dump of all glider parameters, which is sent to the basestation

- Typically, this is the last chance to ensure that all parameters are correct before dive 1
- The bottom of the selftest review tool on vis compares the *.prm* to canonical values
- Prior to deployment, pilot team reviews the *.prm* file to ensure all parameters are correct, puts any needed parameter changes into the cmdfile
- If review is favorable, pilot team instructs field team to continue Sea Launch by pressing "Enter" key

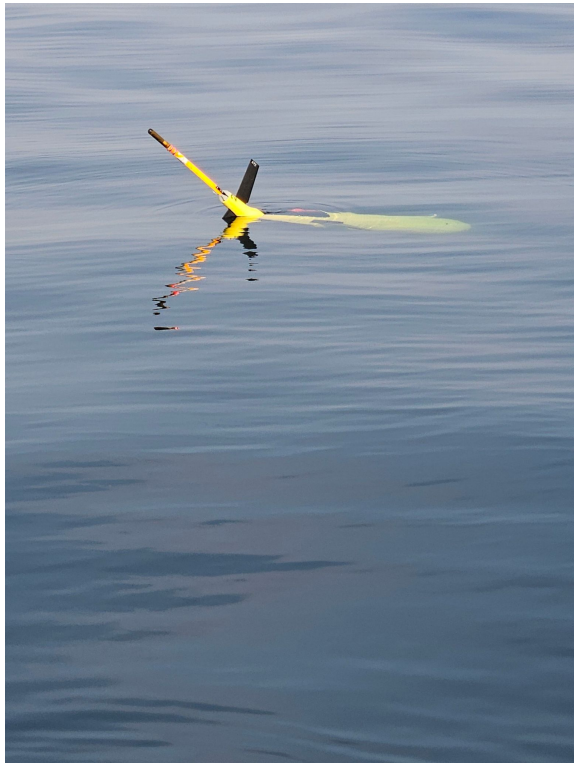
Canonical parameters

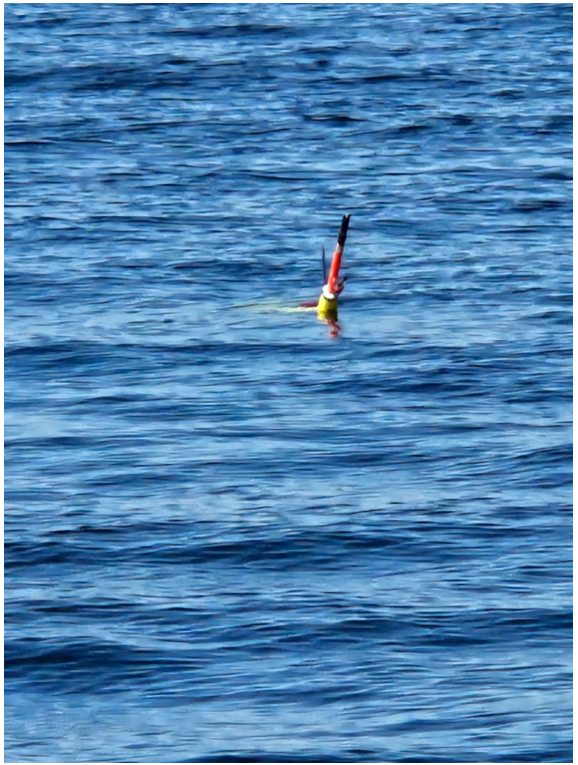
- The selftest review tools and the compare.py basestation script compare parameter values in selftest *ptGGGnnnn.cap* and *pGGG0000.prm* files against typical values contained in *canonical* files on the basestation
- Available versions are in `/usr/local/basestation3/canonicals`
- Or create your own to define your favorites for dive 1 **$D_TGT** and **$T_DIVE**, etc.
  - Specify which file to use for your comparison by creating a symlink in the mission directory:
    `ln -s /usr/local/basestation3/canonicals/canon_RevE.log .canonicals`

5. Deploy glider

- When .prm file passes review, pilot team instructs field team to deploy the glider
- Field team confirms that the glider is in a launched state and is in recovery mode, waiting for a **$RESUME** directive before it will dive.
- This is the final handoff of control between field team and pilot. Once the cable is disconnected while Seaglider is running, it is under pilot control
- Field team disconnects comms cable and installs dummy plug, lowers glider into the water on a tag-line
- Field team verifies that the glider is floating and has a surface angle of ~30-60° (for best communications). Angles are reported in the comm.log on each call before the RESUME is sent to command the glider to begin diving.
- Field team interrogates transponder and verifies acoustic response. Interrogate and reply frequencies are reported in the selftest. Any acoustic release or transponder deckbox that operates in the 10-15kHz band should be capable of ranging to the glider.

Surface angle





Pilot monitors glider before its first dive

- Monitor Seaglider comms prior to first dive. Check Iridium and GPS performance
- Be sure D_TGT, T_DIVE, and T_MISSION are properly set - typically shallow, e.g.

  $D_TGT,45 **$T_DIVE**,15 **$T_MISSION**,30.

- Replace **$QUIT** with **$RESUME** in the cmdfile to initiate the first dive.
- After Seaglider takes up the cmdfile with **$RESUME** directive (monitor the comm.log), edit the cmdfile to replace **$RESUME** with **$QUIT** so that the glider will not resume diving once it surfaces
- For subsequent dives, increment **$N_DIVES** by 1

Trimming the glider

- Trimming glider to fly efficiently and smoothly can take a while - start with first dive (shallow).
- Use both intuition + suggested parameters on pitch, roll and vertical velocity plots to trim the glider
- Trim pitch (**$C_PITCH**, **$PITCH_GAIN**) first to make glider fly, not stall, through dive, apogee, and climb phases.
- Trim buoyancy (**$C_VBD**) next to make vertical velocity cross zero where buoyancy does.
- Adjust roll last to make fly straight (**$C_ROLL_DIVE**, **$C_ROLL_CLIMB**) - start by making approximately flat (roll deg. vs. control). Roll should only be trimmed once the compass calibration is reasonably accurate (see slides on compass cal)
- Do all the above over a progression of deeper dives - use the **$GO** directive (rather than **$RESUME**) after the first deep dive: the glider will continue diving after each surface phase. (Do not leave the **$RESUME** directive in place, as it will cause the glider to dive even if it has gone into recovery due to a problem)
- As long as the initial ballast is ok, you won't lose the glider with bad trim - the glider just won't be efficient and you may not get good data.

Dive progression

Beginning of mission:

- Field team remains on site until released by pilot team
- Shallow dives (50-150 m)
- Pilot checks for vehicle function. If a significant problem is detected, the pilot may request that the field team recover the glider
- Pilot trims the glider, focusing on pitch then VBD
- Review sensor data to ensure all sensors are reporting and reasonable
- Once pilot team has determined that there are no major problems and that the glider can be trimmed, they can release the field team.

Deeper dives:

- Progress toward the desired mission depth
- Pilot team makes minor adjustments to pitch and VBD, and begins trimming roll. Compass calibration may be needed first.

Recovery:

- Pilot team shortens shortens dives in anticipation of recovery
- Pilot team parks the glider on the surface with the **$QUIT** command when the field team is at the recovery site. Glider will send a GPS fix approximately every **$T_RSLEEP** minutes while on the surface
- Field team notifies the pilot once the glider has been recovered

Vis and analyzing Seaglider dive data

Accessing vis

- Running as a publicly available server on the internet:
  - https://iopbase3.apl.washington.edu/
  - https://basestation.server.org/GGG
  - https://seaglider.pub/aoml/610
- Running standalone for a single pilot via ssh tunnel
  - On local machine:

```
ssh -L 20001:localhost:20001  myname@basestation.server.org
```

  - On basestation:

```
basepy vis.py -m sg610:/home/sg610/current -p 20001
```

  - In browser: https://localhost:20001/610
- Plots are also available in the **plots/** directory in the glider's mission directory for use with any image viewer

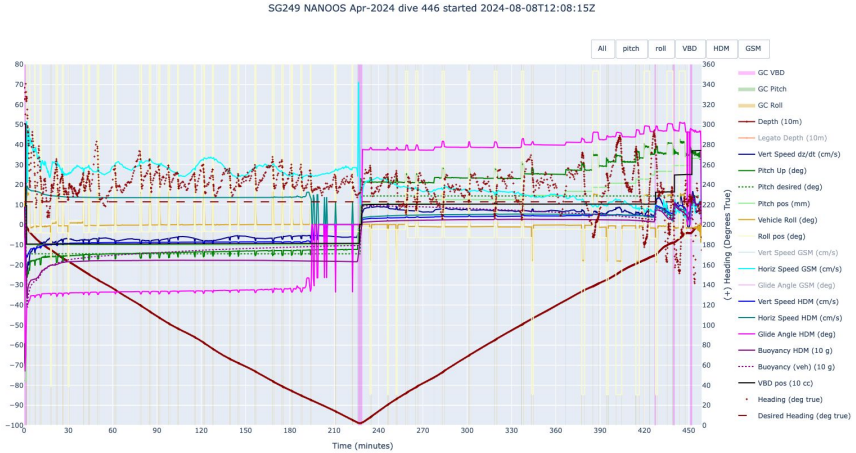
Using vis to help trim during initial dives

- 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 degress of freedom 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.

Later dives

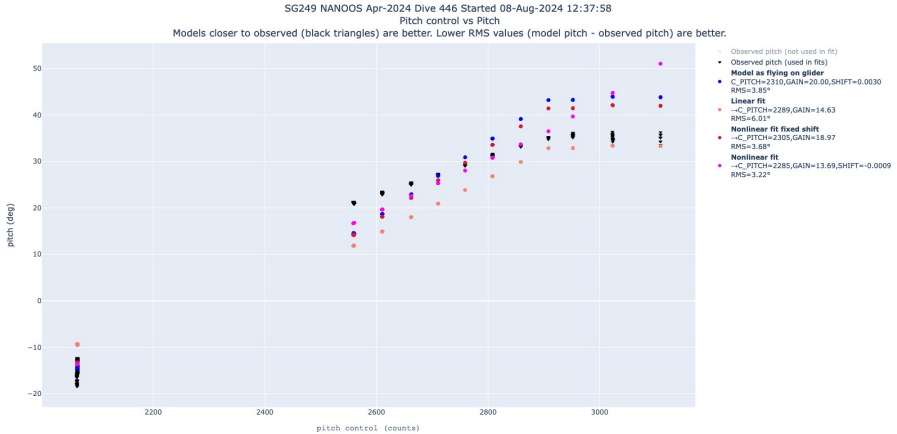
- With 10+ dives or once diving to typical apogee depth, review mission energy consumption
  - Rev E gliders present both model based and fuel-gauge based results
    - Model uses hardcoded current draws or values from CURRENTS file along with measured on time (phone and motors are exceptions, those 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

Dive plot



- All-in-one overview of the dive
- Assess dive symmetry
- Understand pitch/roll behavior and relationship to G&C moves
- Identify breakdowns in flight model

Pitch plot and trimming pitch



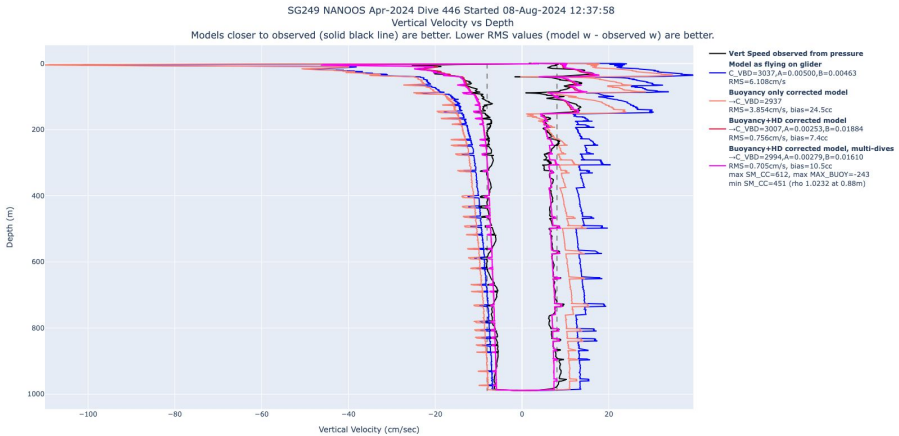
Black: observed pitch (degrees vs AD counts)

Blue: model with current **$C_PITCH**

Orange/red/pink: model adjusted for **$C_PITCH**, **$PITCH_GAIN**, and **$PITCH_VBD_SHIFT**

- Legend has guidance for changing pitch parameters
- Look for model with improved agreement to observations (smaller RMS value)

Vertical velocity (w) plot and trimming **$C_VBD**



Black: observed w

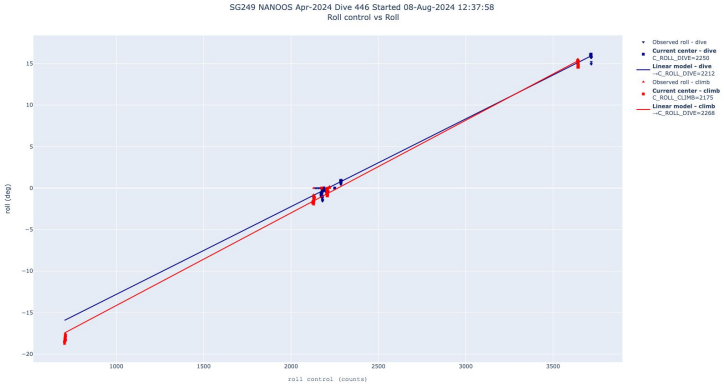
Blue: model with current parameters

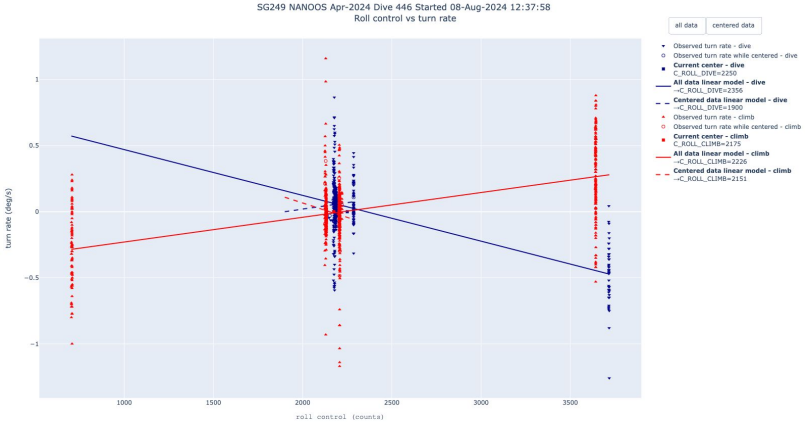
Orange/red/pink: model adjusted for buoyancy and hydrodynamic model from one or more dives

Dashed lines: target w

- Legend has guidance for changing **$C_VBD**
- Look for model with improved agreement to observations (smaller RMS value)
- Making **$C_VBD** smaller will make the glider more buoyant - slower dive, faster climb.

Roll plots and trimming roll





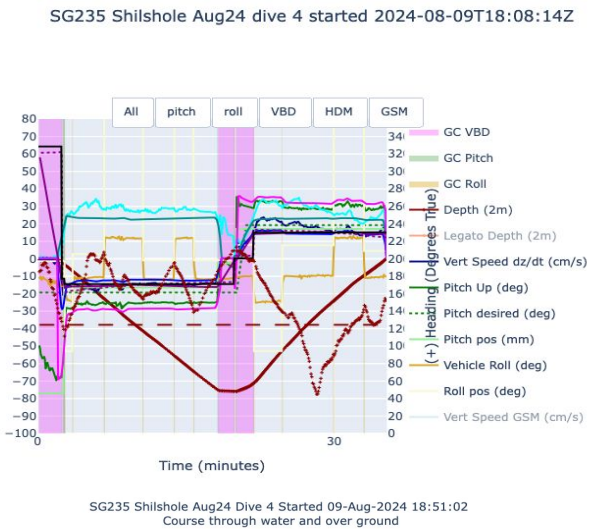
Two roll plots:

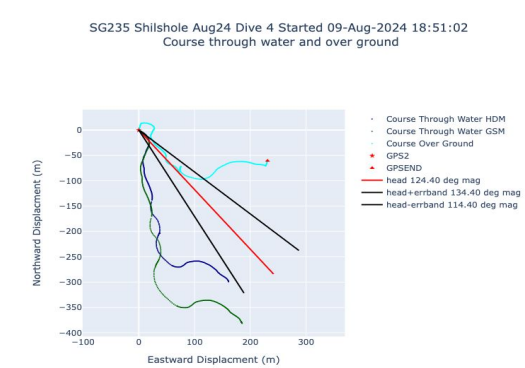
(1) Roll (deg) vs roll control (AD counts)

(2) Turn rate (deg/sec) vs coll control (AD counts)

- tends to provide better suggestions for roll parameters

Roll plots and trimming roll





The dive plot and course plots are also useful for understanding and trimming roll

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

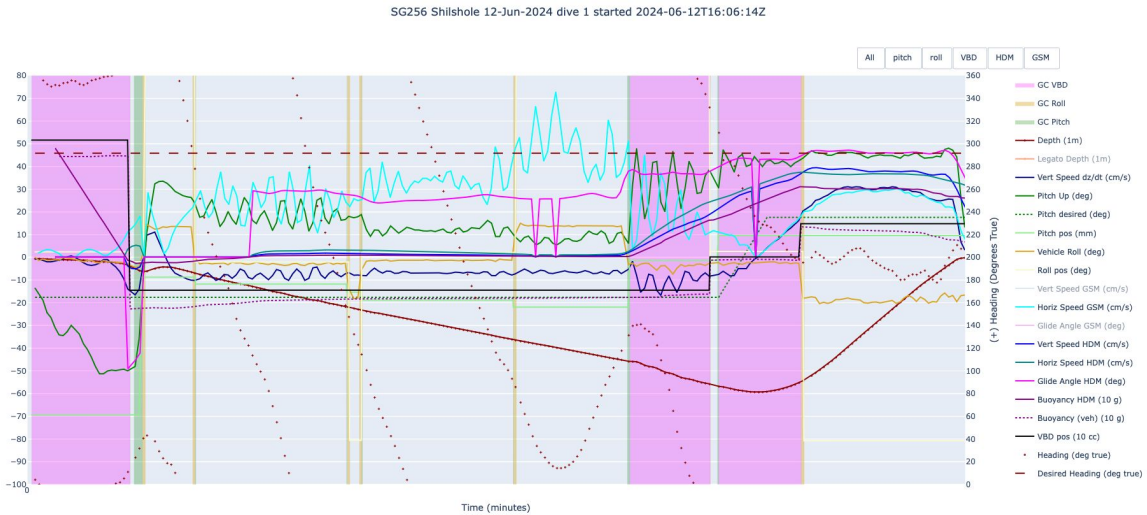
Climbing

heading ↑ then $C_ROLL_CLIMB ↓

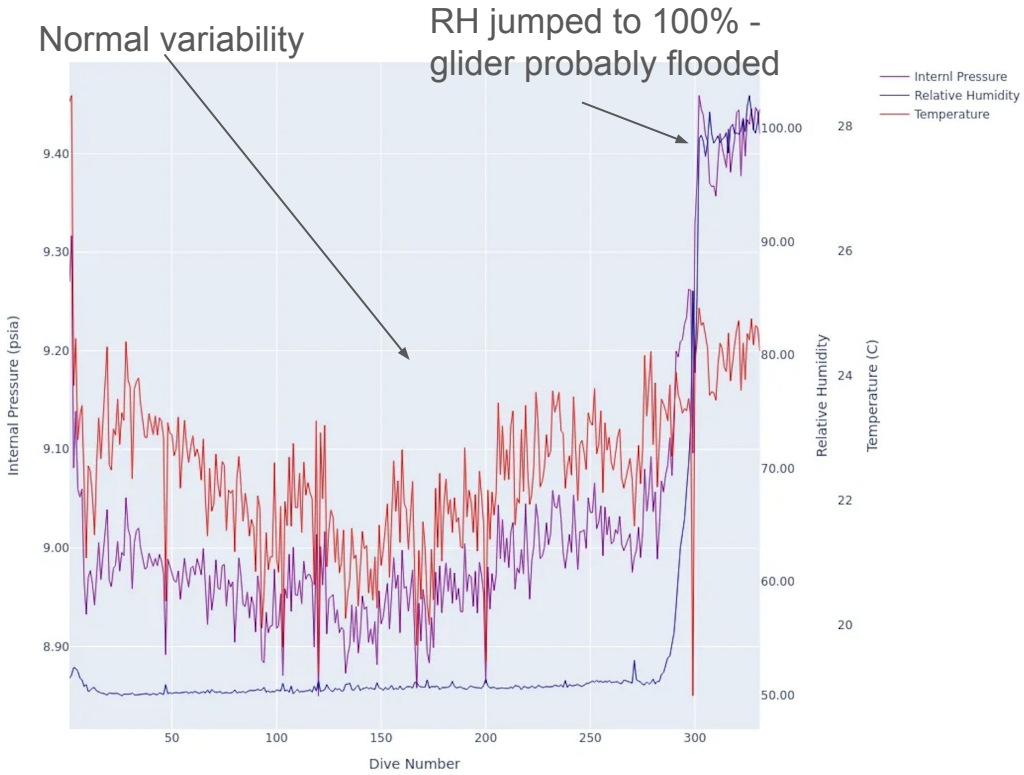
heading ↓ then $C_ROLL_CLIMB ↑

Bad trim example

Dive 1 - https://iopbase3.apl.washington.edu/256?mission=Shilshole_Jun24



Engineering plots (entire mission): Look for trends or sudden changes. Example:



Internal pressure
Relative humidity
Internal temperature

Useful vis API endpoints on vis (URL queries)

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

- /data/nc/GGG/DDD - e.g., http://iopbase3.apl.washington.edu/data/nc/527/001
  - 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

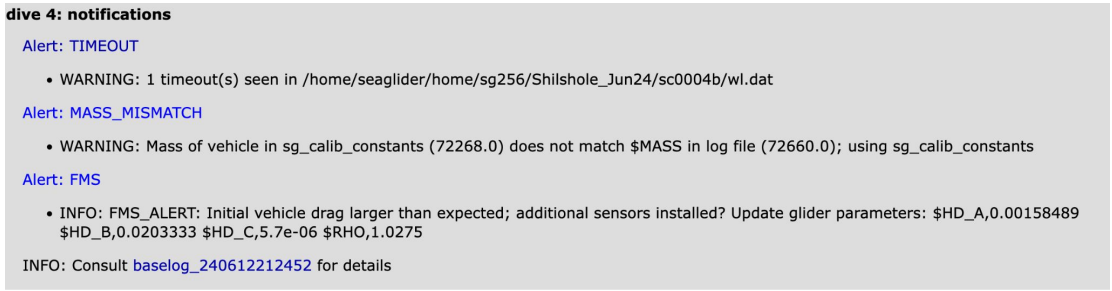
GGG=glider number, DDD=dive number

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

Problems and troubleshooting

Vis offers tools to help with troubleshooting

The alerts bell indicates errors, alerts, and warnings and offers a link to the relevant baselog file



dive 4: notifications

Alert: TIMEOUT

- WARNING: 1 timeout(s) seen in /home/seaglider/home/sg256/Shilshole_Jun24/sc0004b/wl.dat

Alert: MASS_MISMATCH

- WARNING: Mass of vehicle in sg_calib_constants (72268.0) does not match $MASS in log file (72660.0); using sg_calib_constants

Alert: FMS

- INFO: FMS_ALERT: Initial vehicle drag larger than expected; additional sensors installed? Update glider parameters: $HD_A,0.00158489 $HD_B,0.0203333 $HD_C,5.7e-06 $RHO,1.0275

INFO: Consult baselog_240612212452 for details

The dives table indicates dives with alerts (**a**) and for which there are capture files (**c**) and capture files containing critical errors (**c**)

306	24/08/02 00:19	194	503(500)	1.10	256.3
305**a**	24/08/01 21:01	190	504(500)	1.38	252.1
304	24/08/01 17:43	193	503(500)	1.08	290.1
303**a**	24/08/01 14:25	197	503(500)	0.72	268.7
302**c**	24/08/01 07:58	191	504(500)	1.61	256.7

Look at the baselog and comm.log

Basestation log files help you understand errors/alerts

Search for lines with errors, warnings, critical, e.g.,

```
grep ERROR baselog_yymmddHHMMSS
```

Look at the selftest, historical selftests and historical missions

Is this a new problem or has it shown up before?

Is the behavior getting worse?

Bring back a capture file from the glider

- Capture files contain detailed information about everything the glider does, so they can be helpful for understanding strange behavior
- *pdoscmds.bat* file to get a capture file from dive DDDD:

```
resend_dive /c DDDD
```

E.g., Glider is diving and calling but not going where you want

- Trim could be bad
- Glider could be in shallow water
- Glider could be in strong currents
  - Currents tend to be strong in the upper ocean; if the glider is diving shallow, consider deeper dives (if possible) until it is through the strong currents
  - Flying 90 degrees to the depth-averaged-current (DAC) can be a good strategy
  - Decreasing **$T_DIVE** and increasing **$MAX_BUOY** shortens the dive (glider flies faster) and improves the glider's ability to make horizontal velocity
- Something could be physically broken
  - Get a capture file (via a pdoscmds.bat file with `resend_dive /c divenum`) and look for unusual values, e.g. unusual AD counts or currents during G&C moves

E.g., The glider is calling but not diving - in recovery mode due to some problem

- Look at the recovery code reported in the comm.log (and in the alert message received by the pilot). This will change to NO_RECOVERY_REASON if a **$QUIT** directive is in place so you might need to scroll back to see the original reason.
- Get a capture file if it has not already been uploaded. Look through it carefully.

E.g., The glider just stops calling

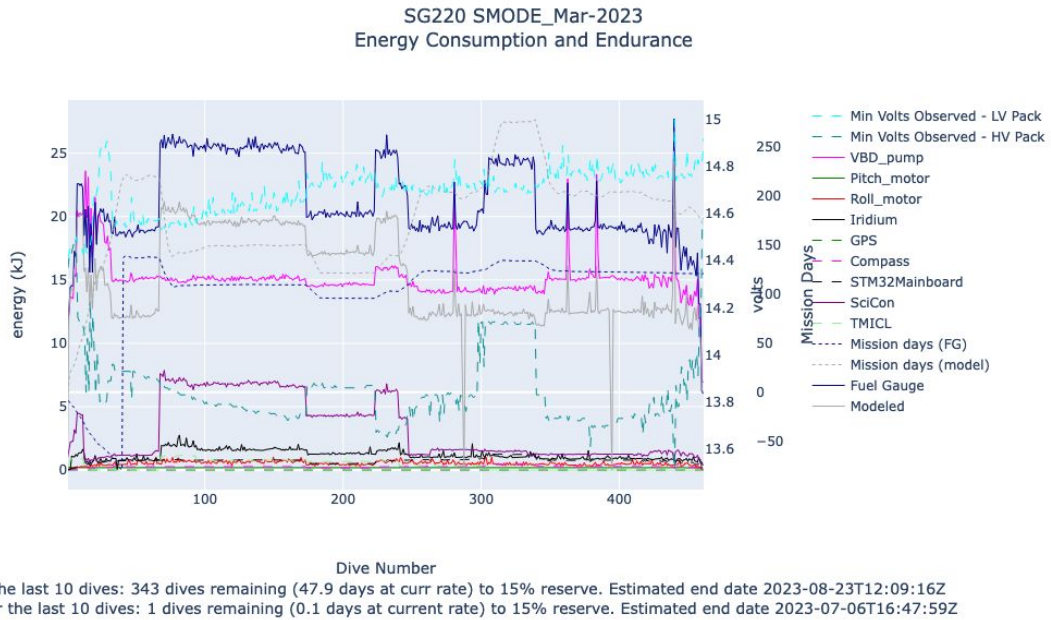
- Wait and hope
- But there are a couple of things you can have in place when setting up the mission to give some extra hope:
  - Set SMSemail. Sometimes the glider can get an SMS through even when it cannot make RUDICS calls. This can be done with a *pdoscmds.bat* file containing

    ```
    writenv SMSemail name@email.address
    ```

  - Set **$NOCOMM_ACTION** and **$N_NOCOMM** appropriately so that you know what behavior to expect from the glider if it is still diving but not calling.

Piloting to save power

Piloting to save power



Monitor the energy consumption and endurance plot

- Estimate the total mission days / mission end date
- Identify what systems are using the most energy
- Identify jumps or spikes in energy consumption
- Monitor voltage

Piloting to save power

- Loitering
- Minimize unnecessary VBD pumping:
  - Lowering **$MAX_BUOY** reduces the amount of pumping. However, if **$MAX_BUOY** too low, the glider will stall and lose control
  - Lowering **$SM_CC** means less pumping during the surface maneuver. However, if **$SM_CC** is too low and the glider can't surface, it will pump to max (which uses a lot of power). If glider misses a call, glider will pump to max and set a new **$SM_CC**
  - Dive as deep as possible/practical
- Reduce science sampling (if possible), e.g., sample less frequently in depth ranges that are scientifically less interesting; sample only on dive or climb; turn off a sensor entirely

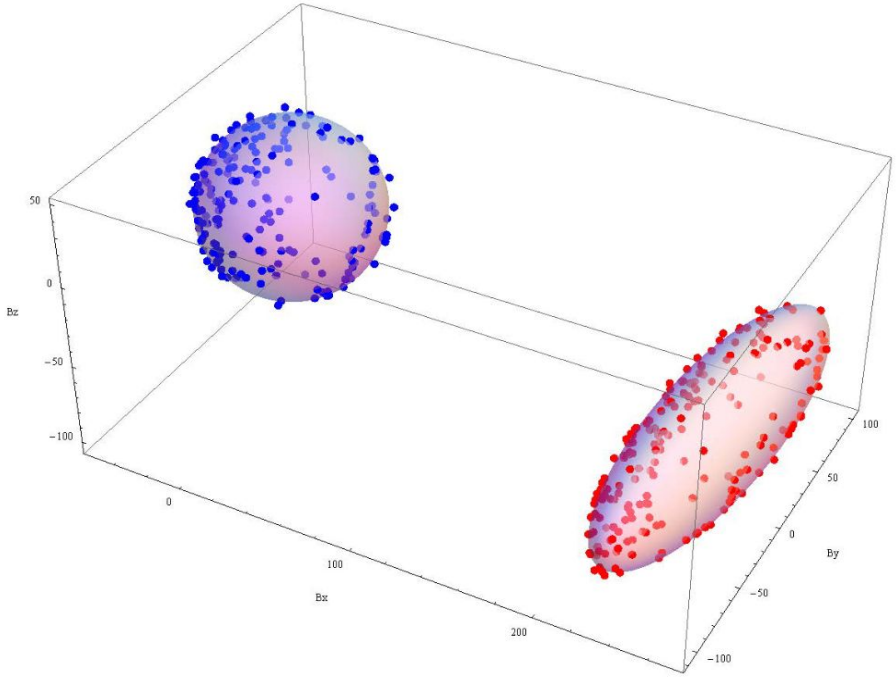
Compass calibration

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

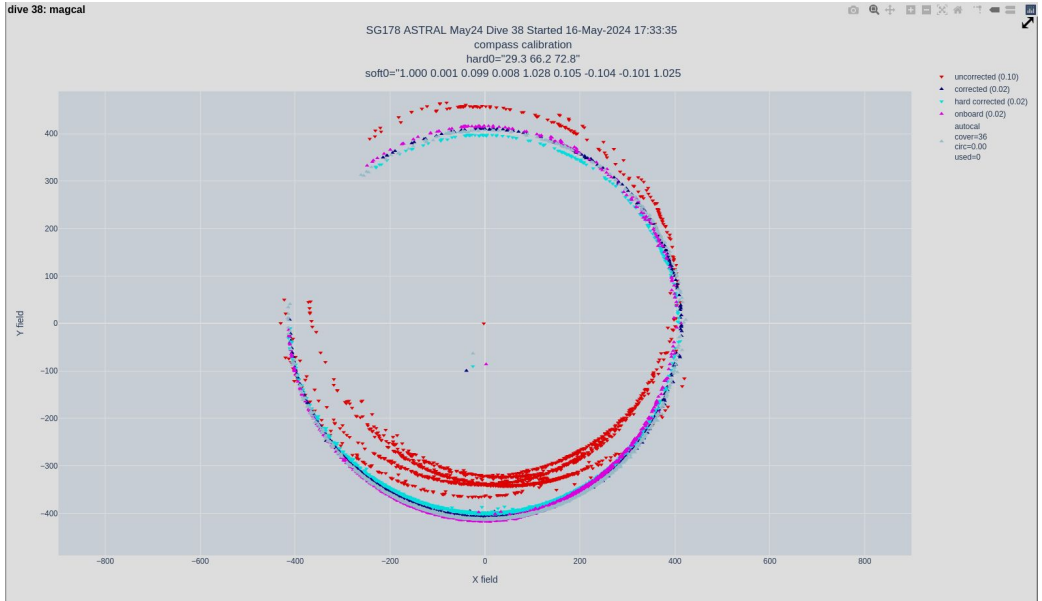
- **$COMPASS_USE,4** (at least four) critical to send back magnetometer data
- Multiple ways for a pilot to compute a compass calibration:
  - vis computes a per-dive calibration result and plot in the plot ribbon
  - **Magcal.py** can be run from the command line
  - Multi-dive calibration (same as 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)?

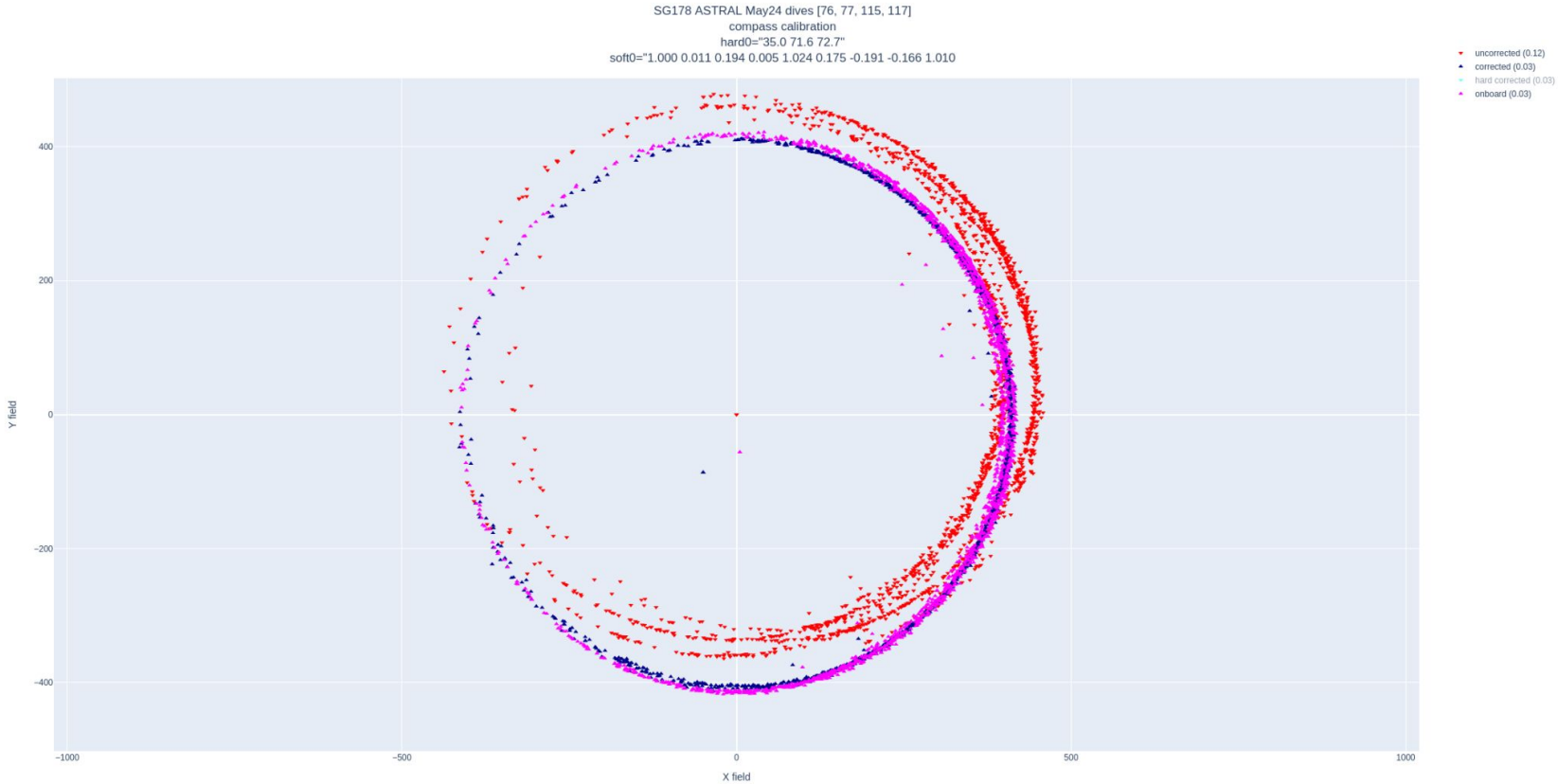


uncorrected = glider's measured magnetometer data
corrected = field corrected for hard and soft iron
hard corrected = field corrected by hard iron only
onboard = corrected field calculated for this dive (using tcm2mat.cal/tcm2mat.aut)
autocal = autocalibration calculated onboard glider (used=0 means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

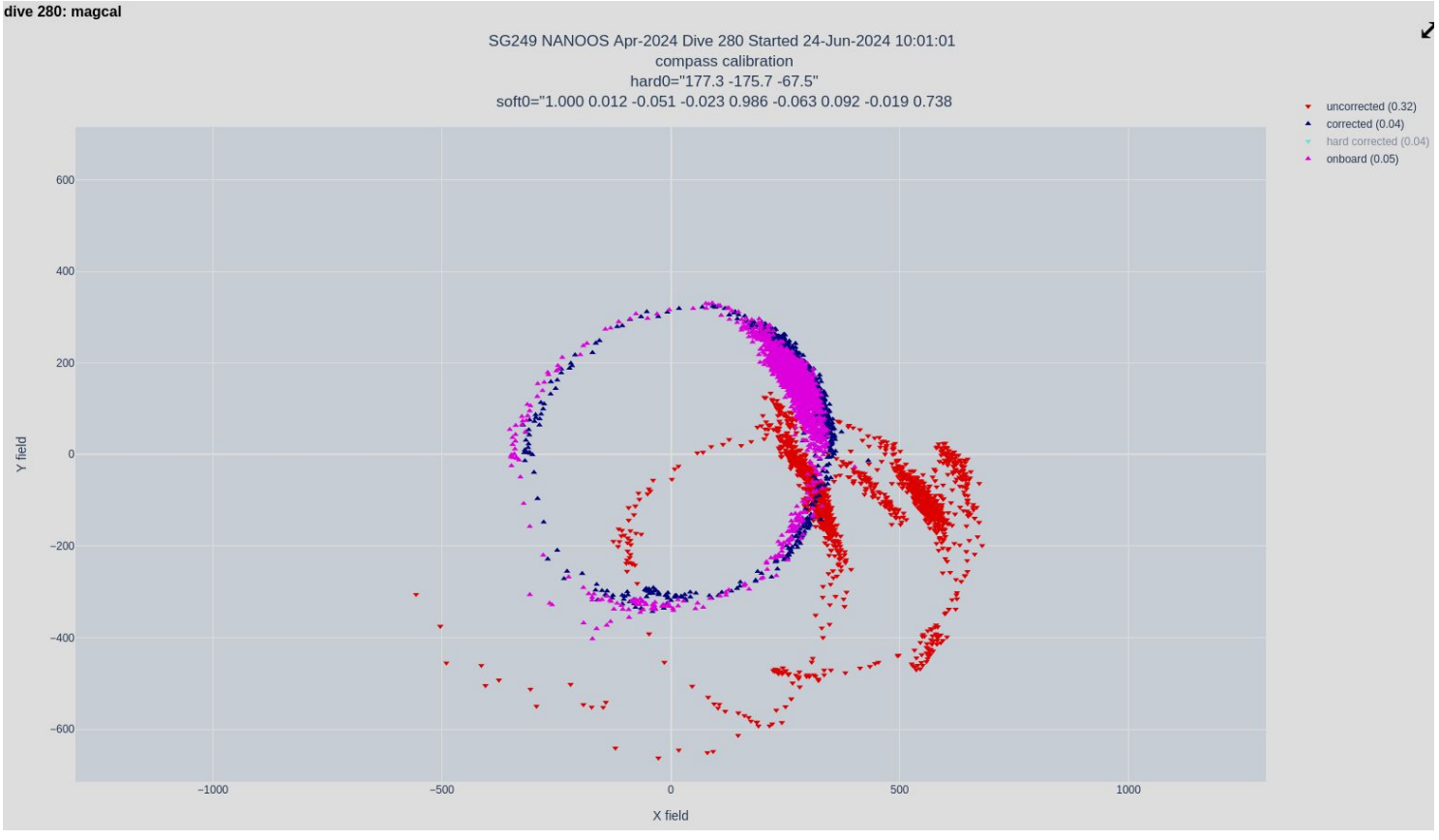
(numbers in the legend are RMS deviation from perfect circle - smaller is better)

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

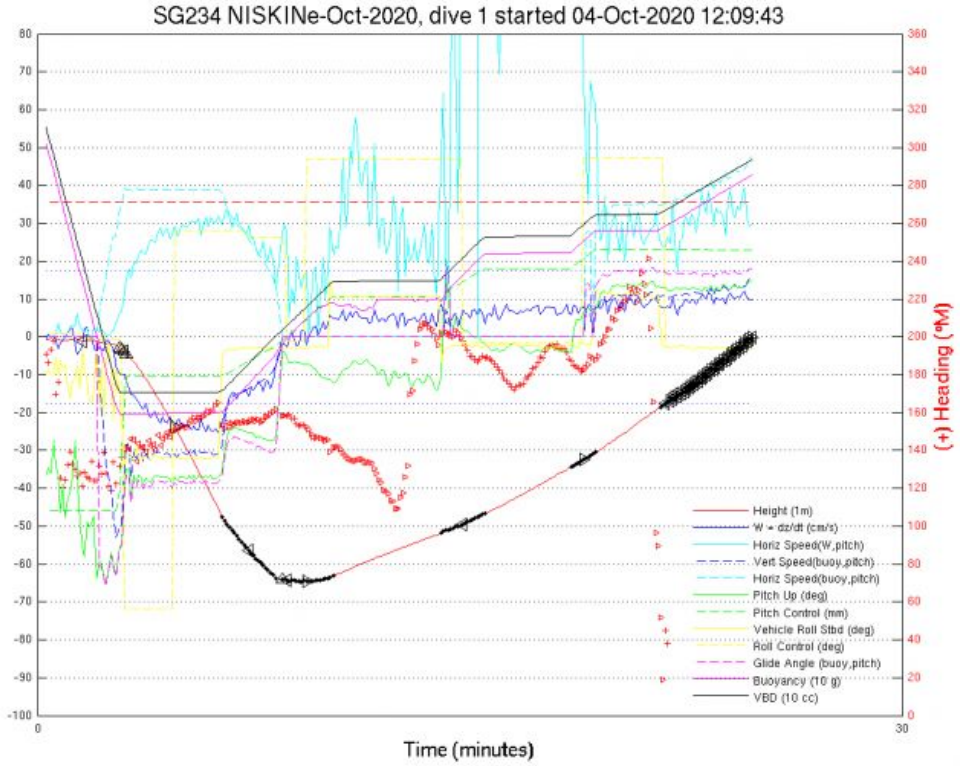
hard0="35.0 71.6 72.7"
soft0="1.000 0.011 0.194 0.005 1.024 0.175-0.191 -0.166 1.010"

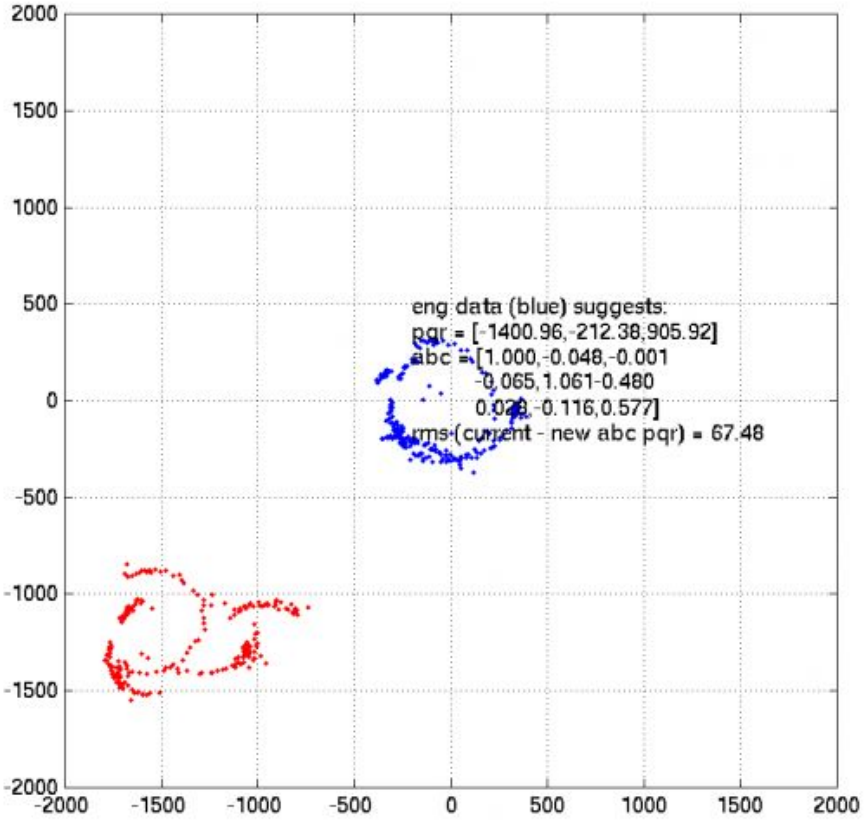


Example: Good coverage, but spare. Coverage strongly biased in one direction. The “circle” is not very circular and the 3rd diagonal soft iron term is suspiciously low.



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.





Ballasting

Recap of definitions

Volmax: maximum volume of the glider (i.e., volume when all moveable oil is in the external bladder), in cc's. Getting an accurate estimate of volmax is key to ballasting. We get an initial estimate of volmax in a tank, and a more refined estimate during the test deployment. Volmax is a constant for a given glider configuration (external payload and VBD settings).

Neutral volume: This is the volume of the glider including external bladder position at which the glider is neutrally buoyant at the desired apogee density. By definitions below, the bladder position at this point is 0cc and C_VBD AD counts.

Thrust: Net buoyancy in grams. Negative thrust means the glider is "heavy". Sometimes also described in cc relative to neutral volume at **$C_VBD**.

Target thrust ($MAX_BUOY): This is the desired amount of negative thrust that we want to have at apogee so that we can maintain speed throughout the entire dive.

$C_VBD: The position of the oil reservoir in AD counts that defines where the glider thinks it has zero thrust at apogee density.

Apogee density ($RHO): Reference density in sigma-theta units. Typically set to the most dense water expected to be seen.

$VBD_MIN, $VBD_MAX: AD limits of the VBD system (VBD_MIN is pumped to max, VBD_MAX is bled to minimum)

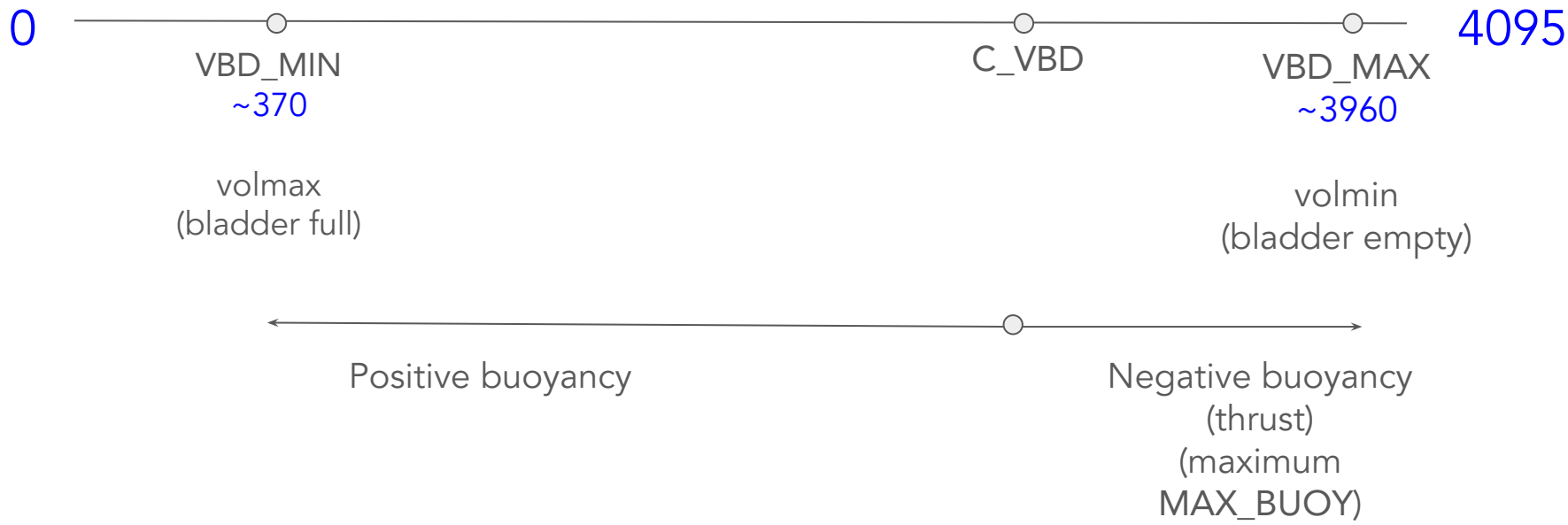
Goal of ballasting

Determine the total mass of the glider (i.e., how much lead to add/remove) needed to achieve the desired thrust at a given density while preserving enough reserve positive buoyancy for the glider to surface in the widest possible range of surface densities.

The glider is ballasted according to:

- Target density (at max depth, typically 1000 m) in the area of operation.
- Thrust – typically 150 to 250 cc. Larger thrusts are needed to overcome strong currents.
- Reserve buoyancy – e.g., in areas with a very fresh surface layer, it can be crucial to have enough reserve buoyancy so the glider can surface

Ballasting strategy



Ballasting strategy

1. Weigh glider to get scale mass
2. Obtain an estimate of volmax (e.g., in tank or from test dives)
3. Compute the target mass needed for a given target density & thrust:
   a. Target thrust fixes C_VBD because we never need to get heavier than that thrust, so we can set C_VBD such that when bled to VBD_MAX (minimum buoyancy) we will be at target thrust:
      C_VBD = VBD_MAX – (thrust/RHO)/VBD_CNV
   b. When we know volmax, we also know volmin: it's the difference between volume with all oil outside (VBD at VBD_MAX) and volume with all oil inside (VBD at VBD_MAX):
      volmin = volmax + (VBD_MAX – VBD_MIN)*VBD_CNV
   c. We can now calculate the desired mass based on desired thrust:
      new mass = (RHO*(volmin - thrust/RHO))
4. Difference between scale mass and new mass = amount of lead to add or remove, with a small adjustment for the volume of the changed lead:
   a. dmass = (RHO*(volmin - thrust/RHO) - MASS)/(1 - RHO/leadDensity) (leadDensity = 11.296g/cc)
5. Double check that at new Volmax, glider will be able to raise antenna (~150cc) in lowest density surface water expected:
   a. RHO_min = Mass / (Volmax - 150)

*VBD_CNV = cc's of oil per AD count = -0.2453 (constant, same for SG and SGX)

Worked example - SG526, Shilshole 14-August-2024

1. Scale mass = 53322 grams
2. Volmax (from multi-dive regression plot after dive 5) = 52759 cc
3. Compute the target mass needed for a given target density & thrust:
   a. Target thrust fixes C_VBD because we never need to get heavier than that thrust, so we can set C_VBD such that when bled to VBD_MAX (minimum buoyancy) we will be at target thrust:
      C_VBD = 3960 – (-250/1.0275)/-0.2453 = 2968
   b. When we know volmax, we also know volmin: it's the difference between volume with all oil outside (VBD at VBD_MAX) and volume with all oil inside (VBD at VBD_MAX):
      volmin = 52759 + (3960 - 500)*-0.2453 = 51910
4. Difference between scale mass and target mass = amount of lead to add or remove:
   dmass = (1.0275*(51910 - -250/1.0275) - 53322)/(1 - 1.0275/11.296) = 292 grams

*VBD_CNV = cc's of oil per AD count = -0.2453 (constant, same for SG and SGX)

Ballast worksheet

Access from the Tools menu on the basestation3 visualization:

Ballast worksheet: generic tool to compute the target mass based on scale mass, volmax, etc.
– we use this when we already have an estimate of volmax from the ballast tank

VBD regression: uses dive data to estimate the glider's volmax by regressing observed vertical velocity (w) against w from the glider flight model.
– we use this after Puget Sound test dives to get a more accurate estimate of volmax
– that volmax can then be input in the ballast worksheet to get final ballast numbers prior to deployment

Ballast worksheet

VBD min counts	648
VBD max counts	3962
scale mass as flown in field (g)	51657
target thrust for final ballast (g)	-150
target density for final ballast (g/cc)	1.023
antenna volume loss (cc)	150
volmax from regression of field test data (cc)	51097.06696

calculate

Automatically populated from the most recent log file
← actual mass of glider

Typical values we use for testing in Puget Sound
For open ocean, typical target values are -250 g thrust, 1.0275 g/cc density

← Volume that is out of the water when the glider is at the surface. Antenna is always 150cc.

← volmax - from tank or from regression to field data

← This button calculates the final ballast results from the above parameters

Deepglider parameters (leave blank for SG/SGX):

abs compress (model or fixed) (m^3/dbar)	
therm expan (model or fixed) (m^3/°C)	
ref temperature (°C)	
apogee temperature (°C)	
apogee pressure (dbar)	

calculate Deepglider

Ignore for SG/SGX

comments

Optional box for adding comments
(for record keeping purposes)

Final ballast results

lead to add for final ballast (g)	-72.926385
target mass for final ballast (g)	51584.07361
final scale mass after adjustments (g)	
predicted volmax at final ballast (cc)	51090.61101
predicted C_VBD at final ballast	3364.25207
minimum surface density (g/cc)	1.012631623
maximum buoyancy (cc)	666.296634
neutral stroke (%)	81.96294734

Calculated final ballast and lead to add/remove

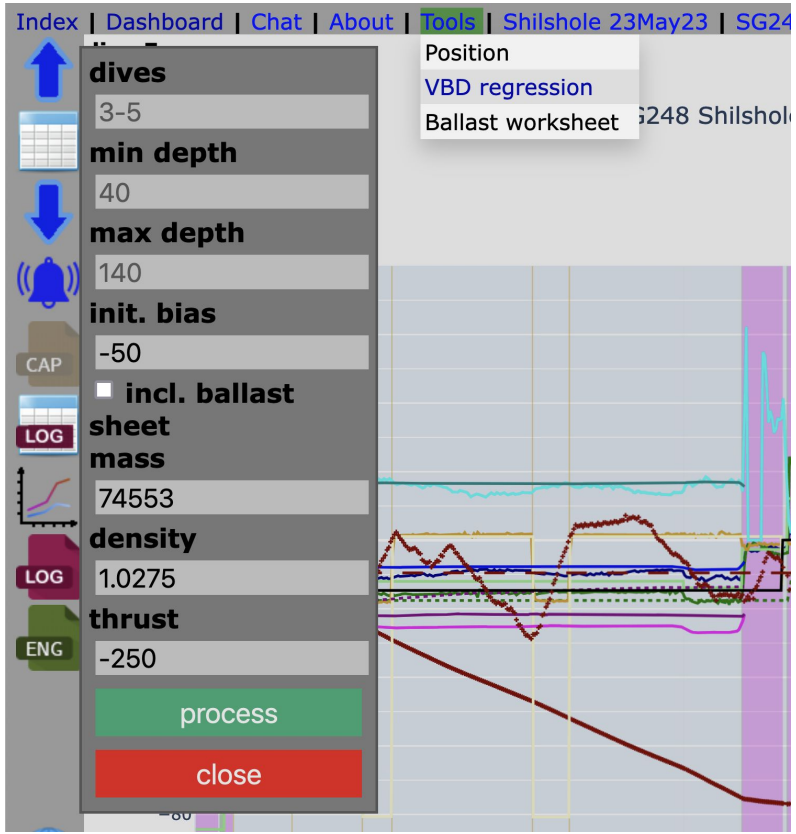
← Optional - enter the actual final mass (for record keeping)

← Good starting place for trimming the glider (typically ~200 counts higher than the actual C_VBD if ballasted on ~150m flights for open ocean operating at 1000m)

← The glider will struggle to surface if the surface is less dense than this

VBD regression tool (basestation3)

Iterative regression of observed vs. modeled vertical velocity, varying flight coefficients and VBD bias. The resulting best fit gives us our best estimate of volmax.



Dives over which to compute the regression- only use dives for which the glider flew well. We use at least 3 dives to 150 m to get best regression results.

Depth range over which to compute the regression (avoid the upper ~40 m and lowest ~10 m, when the glider model isn't accurate)

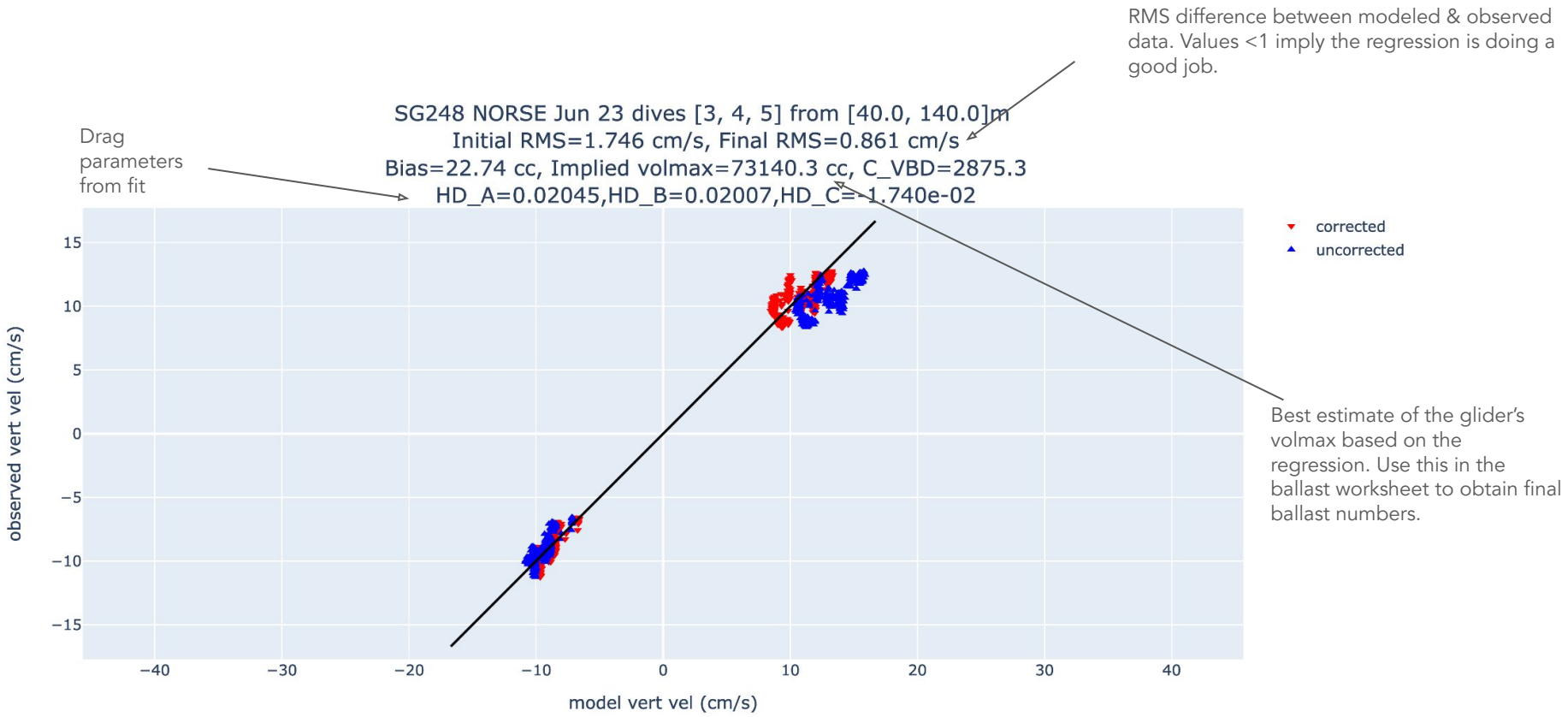
First guess of bias for the iterative regression process

Include the ballast sheet tool in the regression output

Scale mass - populated from the log file

Target density and thrust - only used for the ballast estimate

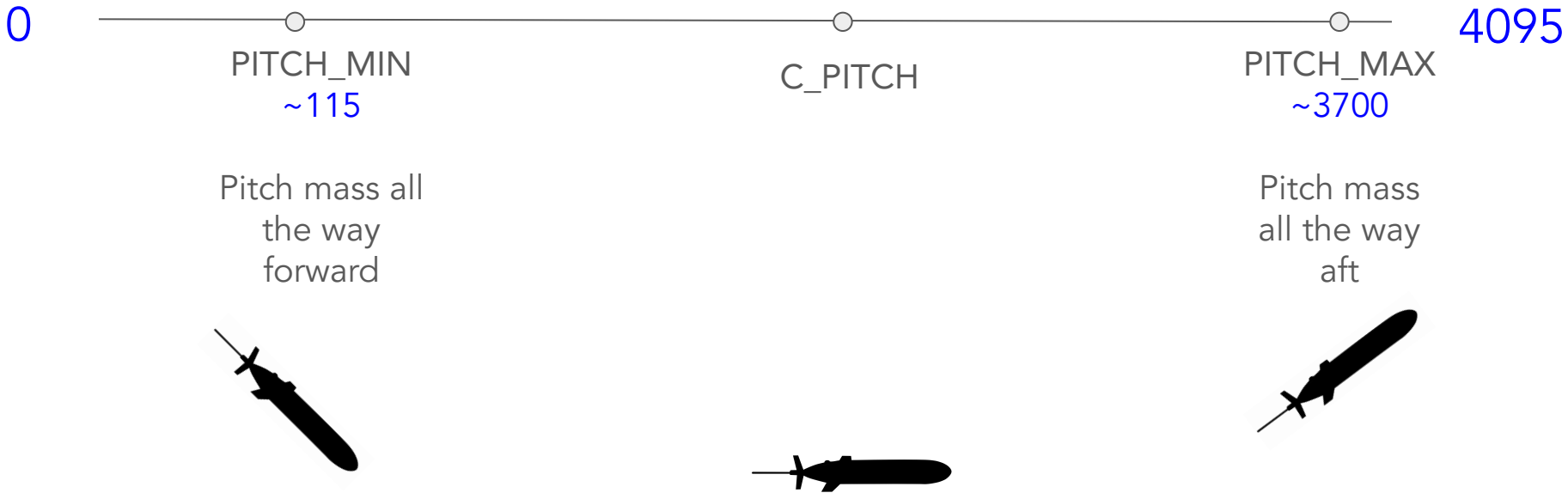
Output of VBD regression tool



Ballasting: Lead placement

Following a trip to the tank (if available) or a field test, adjust the lead position as needed to achieve the desired trim.

Ballasting: pitch



Ballasting: lead placement to affect pitch

We aim *approximately* for –60° pitch when the glider is at the surface and ±18° when the glider is diving/climbing

- If pitch is too flat/steep, lead can be placed forward/aft

We aim *very roughly* for $**C_PITCH**=2400 (± a few hundred AD counts)

- If $**C_PITCH** is much lower than this (i.e., pitch center is too far forward, so the glider can’t pitch down enough and surface angle is low), place lead forward

Additional topics

Bit-field parameters

Several parameters on Seaglider are bit-fields: **$COMPASS_USE**, **$NOCOMM_ACTION**, **$OPTIONS**, **$PROTOCOL**, **$COMM_SEQ**, etc. A bit-field is a summation of powers of 2. They allow us to pack many on/off type conditions into a single number and be able to unpack them with no ambiguity.

For example, for a given menu of choices: **a = 1**, **b = 2**, **c = 4**, **d = 8**, **e = 16**, etc. we can choose **a** and **d** by specifying 9 (1 + 8). There is no other combination of choices that sums to 9. This would not work if a=1, b=2, c=3, d=4, etc. because 3 could mean either c or a + b.

When calculating the value for a bit-field parameter, consult the parameter reference manual (**https://iop-apl-uw.github.io/basestation3/html/Parameter_Reference_Manual.html**) to see what values to add to turn on the options that you want to use for that parameter.

Deck dives / sim dives

Deck dives are a valuable tool to test glider configuration, basestation data pipeline, and new sensor integrations.

For deck dives, configure the glider and basestation just as you would for a real mission, except set the **$SIM_W** parameter (usually to 0.1 for 10 cm/s dives), and use `launch/test` instead of `launch/sea` from the glider menu.

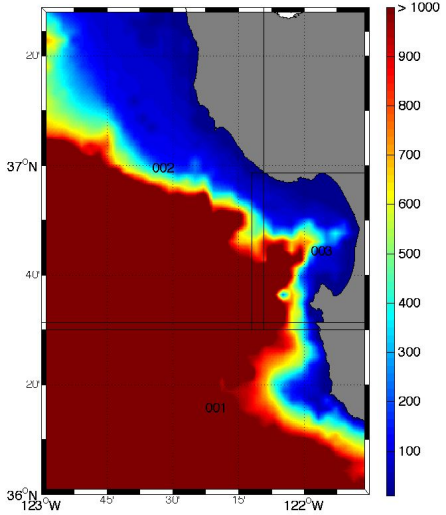
You can stay connected on the serial console to monitor the dives or disconnect as normal and rely on Iridium for all feedback.

If you do not have sky access, stay connected and set **$N_NO_SURFACE** to a large negative number so the glider skips Iridium and GPS. In this case you will need to monitor the dives via the serial console and retrieve any data files you need for analysis post-test via the serial console.

Make sure to start a new mission on the basestation when launching for real.

Bathymaps

- When operating in water shallower than 1000m, the glider can refer to bathymetric maps loaded on the memory card to automatically adjust dive depth.
- Maps are loaded on the card as files named bathymap.bbb, where bbb is an integer number.
- When in use a map is loaded into memory. This imposes a practical limit of about 200kB on each individual map. Thus the need for multiple maps to cover a large area or to provide high resolution.
- Specify which map to use with **$USE_BATHY**.
  - 0 turns off the use of maps
  - Positive values indicate the map number to use
  - Negative values (typical case) indicate the map number currently in use and the glider will automatically search for a map matching its current location. Set to -1 initially if unsure of which map to start on.
- Glider determines **$D_GRID** from the current bathymap and present position. If **$D_GRID** is less than **$D_TGT** then **$D_GRID** is used as the apogee depth.
- If maps are in use (**$USE_BATHY** != 0), but there is no map for the present position then glider sets **$D_GRID** to **$D_OFFGRID**.
- Create new maps using legacy matlab tool.



Seaglider batteries

- Each glider has 2 (SG) or 3 (SGX) lithium primary battery packs: a large main pack, a smaller forward pack (SG and SGX), and small aft pack (SGX only)
- The packs are comprised of numerous DD cells that are secured with shrink-wrap
- Main battery pack rests in a metal cage frame
- Battery packs are fused:
  - Fuse opens circuit of the cell when it receives a sustained 5 amp current overload
  - Fuses can be replaced if needed
- Avoid physical, electrical, and thermal abuse of the batteries and glider
- Seaglider is equipped with a pressure relief valve (PRV) which will automatically vent should pressure build up inside the pressure hull

Battery safety

If you have any reason to suspect water intrusion into the glider (e.g., if there is a visible hull penetration, history of very rough handling, or unusually high internal pressure and/or relative humidity), proceed with extreme caution as the lithium batteries may be compromised:

- Wear PPE (eye protection, gloves)
- Store and open the glider outdoors
- Keep your face and body away from the glider as you are opening it
- Have Lith-X or a lithium fire extinguisher available
- Have vermiculite and baking soda available for spills / battery storage

Shipping Seaglider batteries

- Lithium battery shipments are controlled by the Department of Transportation (DOT), International Civil Aviation Organization (ICAO), and the International Air Transport Association (IATA).
- Hazmat shipping certification is required for shipping lithium batteries
- Regulations are different for shipping gliders (“Lithium metal batteries contained in equipment”, UN3091) vs shipping batteries by themselves (“Lithium metal batteries”, UN3090)
- Proper shipping labels and documentation is needed, e.g.:
  - **Dangerous goods declaration** (signed by hazmat certified person)
  - **Labels** on shipping crates
  - **Safety data sheets** for batteries
  - **Transport certificates** for batteries
  - **Weight** of battery packs (12 kg for SG, 20 kg for SGX)
- Regulations change frequently; it is becoming more difficult to ship lithium both by air and by sea. Plan on shipping taking longer than planned.

Resources and getting help

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

- Documentation and manuals
- Piloting webinars
- Technical notes

Email: iopsg@uw.edu

- Monitored by all of us in IOP

Sending us tarballs - to send us a whole mission directory (if you are not on seaglider.pub):

- `cd ~sgNNN; find . -maxdepth 1 -iname '[a-z][a-z][0-9][0-9][0-9][0-9][a-z][a-z].x*' -o -name comm.log -o -name baselog.log -o -name sg_calib_constants.m -o -name glider_early_gps.log -o -name history.log | tar -cvzf /tmp/tarNNN_problem_description.tgz -T -`
- Contact us for details on how to upload this to us

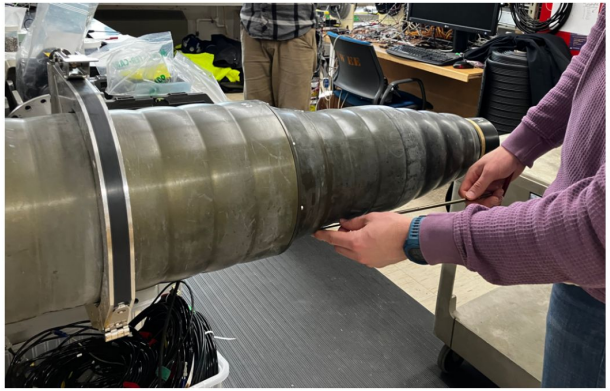
Removing batteries

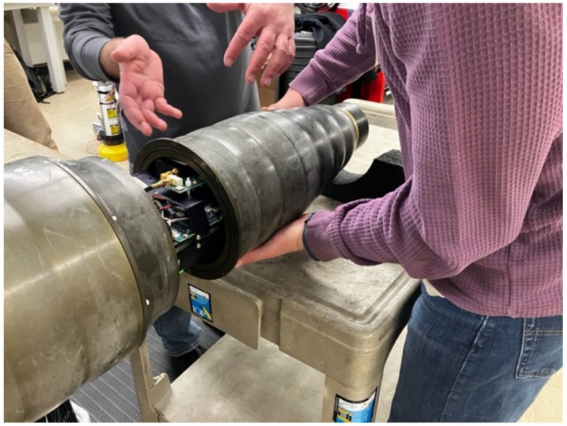
Glider disassembly for battery removal

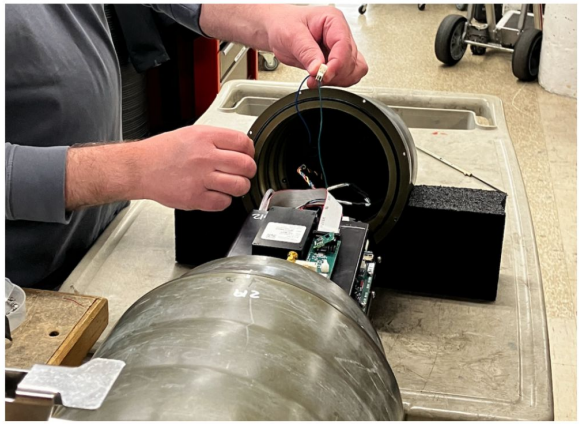
With the glider powered off, disassemble it section by section. Reassembly will be easier if you take lots of photos during disassembly.

1. remove forward fairing and affix the glider in jig
2. disconnect and remove sensors and hatch cover
    a. -put dust cap on the antenna cable and bulkhead to avoid losing the o-ring
3. remove aft fairing
4. with the glider horizontal in the jig, use the PRV tool to release the vacuum
    a. listen for the hissing sound
    b. screw the PRV back in, ensuring that the o-ring is still in place
5. remove the forward hull section:
    a. remove fasteners and gently slide the nose straight out and onto a cart/table
    b. disconnect grounding strap, compass, and transducer wires, then set nose aside

Glider disassembly for battery removal



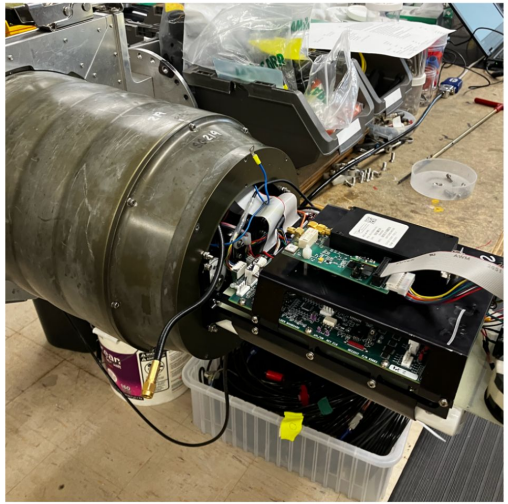




Glider disassembly II

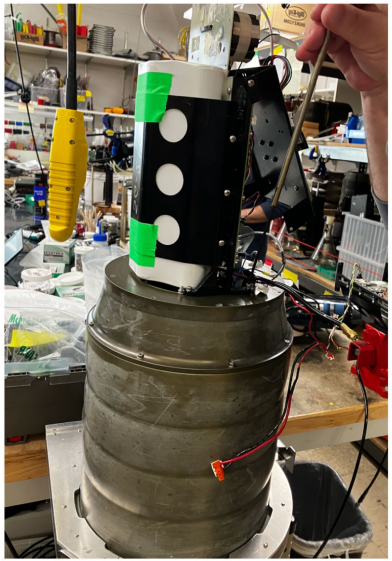
6. disconnect electronics:

a. disconnect the coax cable and grounding strap from the motherboard
b. tip back the phone bridge and disconnect the ribbon cables, batteries, and potentiometers
c. motherboard can typically be left in place for a standard refurb; if necessary, the electronics can be dropped by removing the 4 fasteners connecting the board to the rails (do this with the glider vertical)



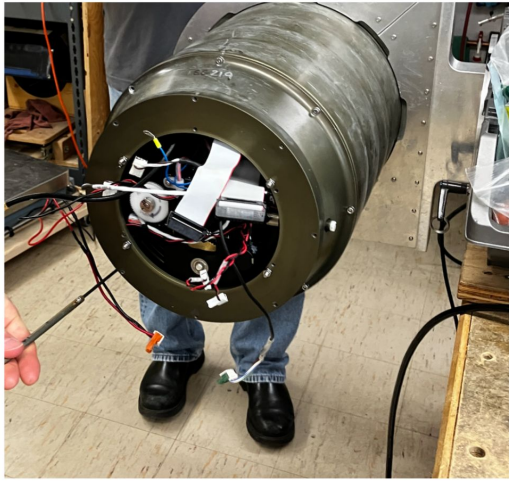
Glider disassembly III

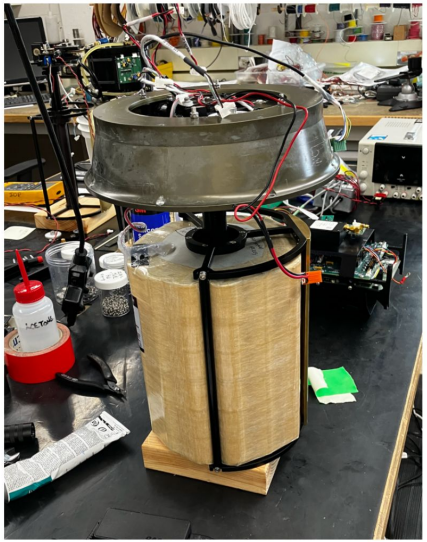
7. tilt the glider vertical and unfasten/remove the entire electronics/forward battery assembly: untape and remove the forward battery pack from the assembly. It should just slide out, but loosen fasteners if needed



Glider disassembly IV

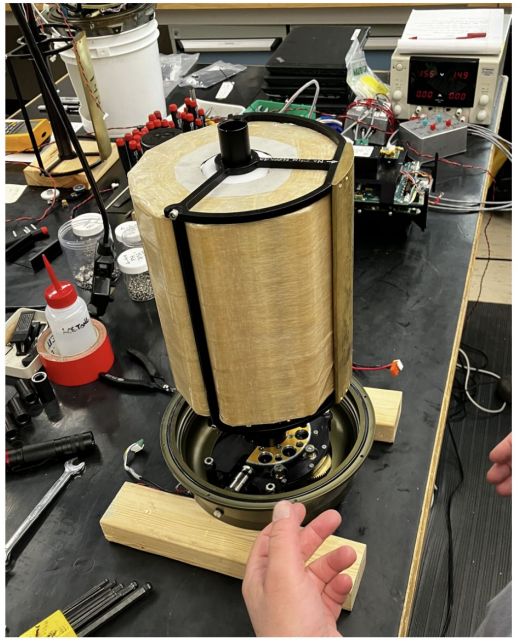
8. with the glider horizontal, unfasten and remove the bulkhead section. Slide the mass shifter out, being careful that the cables don’t get snagged. Rest the mass shifter assembly, battery pack down, supported on a block with a hole in the center or across two blocks





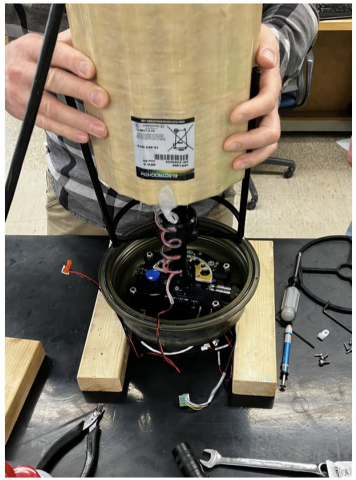
Glider disassembly V

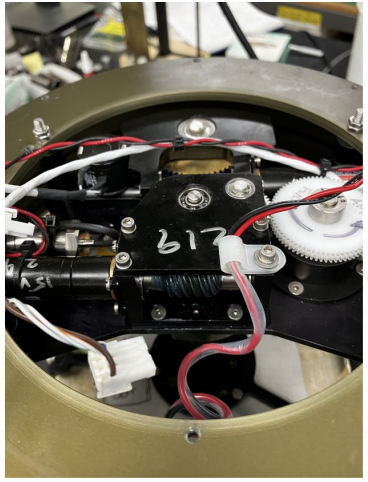
9. Disconnect the battery cable. With the battery up, remove the plate on top of the battery. Loosen the vertical rails holding the battery in place and remove the battery from the assembly

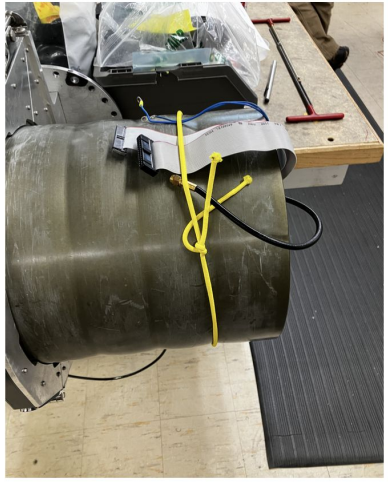


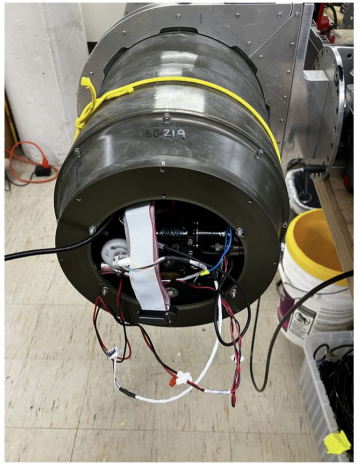
Glider reassembly: main battery pack

1. Prepare batteries:
    a. Check the voltage: new batteries should be ~15.7 V
    b. Run the battery through a degausser to minimize the hard-iron magnetic field (this is best practice but not critical)
2. Slide the battery onto the mass shifter: align the flat side of the battery with the brass piece and the grooves in the battery with the vertical supports: This can take some force as not all battery packs are built perfectly
3. Reattach the plate on top of the battery back
4. Feed the cable through the flanged hull section
5. Attach the battery pigtail to the gear cover: it can go on either fastener, so choose the position that fits best (e.g., wire not touching the worm gear). Don’t forget to use loctite.
6. Carefully clean the o-ring grooves and faces and replace all o-rings between the hull sections. Ensure the o-ring is greased and is fully seated in the groove
7. Slide the mass shifter-battery assembly into the hull section, brass plate facing down. Make sure the cables are out of the way
8. When the two sections are a couple inches apart, feed all of the cables through, being careful not to tangle with the battery leads
9. Fasten the mass shifter hull section to the main hull section





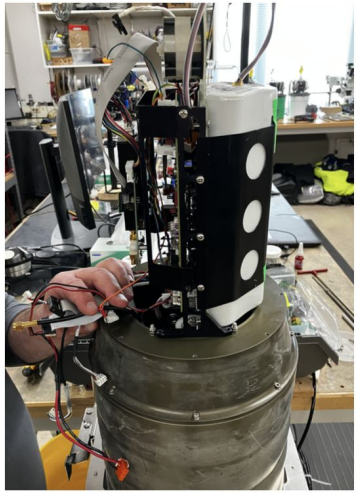


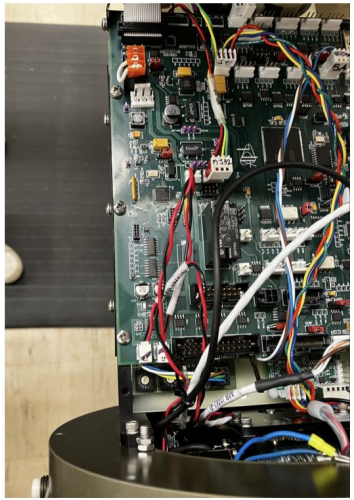


Glider reassembly: electronics/forward battery pack

1. Slide the forward battery pack into the electronics assembly. It is a tight fit.
2. Apply tape to ensure that the battery doesn’t slide out
3. With the forward end of the glider up, fasten the electronics rails onto the hull section then tilt the glider horizontal
4. Plug in the ribbon cables, pressure sensor, pitch/roll potentiometers, and battery leads. Make sure the ribbon cables go on straight so the pins don’t get bent


SG-219





Glider reassembly: nose section

1. Slide the nose section partway on
2. Connect the nose cables to the main board: ground strap, transducer, compass
3. BEFORE closing up the glider, connect to the glider with the comms cable and run through all items in the hardware menu again:
    a. During pitch and roll moves, look into the glider with a flashlight to see how the cables respond. If cables are getting snagged or squished, adjust them as needed. This may require disassembling the glider again to adjust the cable routing
    b. While running through the hardware menu, verify that motor rates/currents haven't changed significantly; that all sensors and electronics (GPS, compass, etc) are attached and producing reasonable outputs
4. Verify that the sensors are all plugged in and behaving correctly - Once everything looks good, fasten the nose section

Glider reassembly: final steps

1. Pull a vacuum:
    a. connect to the glider comms and run an internal pressure selftest, which will continuously output the internal pressure (`hw/intpress/selftest`)
    b. use the pressure relief valve (PRV) tool to fully tighten the PRV, then unscrew 6 turns
2. Connect the vacuum to the PRV and run the vacuum until the internal pressure reads ~8.5 PSI
3. Fully tighten the PRV, then unscrew 4 turns.
4. Verify that the internal pressure reading holds steady. Check it again the next day.
5. Attach the forward fairing
    a. Ensure that it is pushed all the way up, with no gap between the fairing sections
    b. Use tef-gel on all fasteners and fairing screws
6. Run a selftest to test all hardware, sensors, electronics, and comms.