Woods Hole Oceanographic Institution

**Overturning of the Subpolar North Atlantic Program (OSNAP):
RAFOS Float Data Report
June 2014 - January 2019**

by

A. Ramsey, H. Furey, and A. Bower

December 2020

Technical Report

Funding was provided by the National Science Foundation under
Grant No. OCE-1259618 & OCE-1756361

Approved for public release; distribution unlimited.

WHOI-2020-06

**Overturning of the Subpolar North Atlantic Program (OSNAP):
RAFOS Float Data Report
June 2014 - January 2019**

by

A. Ramsey, H. Furey, and A. Bower

Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 02543

December 2020

Technical Report

Funding was provided by the National Science Foundation under
Grant No. OCE-1259618 & OCE-1756361

Reproduction in whole or in part is permitted for any purpose of the United States
Government. This report should be cited as Woods Hole Oceanographic Institution Technical
Report, WHOI-2020-06.

Approved for public release; distribution unlimited.

Approved for Distribution:

Amy Bower, Chair

Department of Physical Oceanography

TABLE OF CONTENTS

1 Abstract	4
2 Introduction	5
3 Float and Sound Source Deployments	7
4 Deployment and Recovery Cruises	10
5 RAFOS and Sound Source Arrays	12
6 Sound Source Performance and Signal Attenuation	21
7 Float Performance	27
8 Sound Source Drift	31
9 Float Data Processing	33
10 Acknowledgements	34
11 References	35
Appendix A: Float Summary Tables	36
Appendix B: Sound Source Drifts	46
Appendix C: Displacement and Spaghetti Plots	54
Appendix D: Postage Stamp Figures	60
Appendix E: Property and Trajectory Figures	75
Appendix F: Times-of-Arrivals	322
Appendix G: RAFOS Weight & Balance Investigation by E. Slate	446
Appendix H: Sound source Mooring Diagrams	495

TABLE OF FIGURES

Figure 1 The OSNAP line, sound source mooring and float deployment locations.....	5
Figure 2 Float deployment location by year and region.....	7
Figure 3 Yearly RAFOS float deployment locations by depth and distance along OSNAP line	8
Figure 4 Ballast offsets as a function of target pressure and year	18
Figure 5 Sound sources in the racks at sea before deployment in summer 2014	19
Figure 6 Float #1376 trajectory color coded by temperature and dates (MM/DD).....	22
Figure 7 Float #1376 times-of-arrival data	23
Figure 8 Float #1376 property data	24
Figure 9 Signal intensity by depth and distance through a section in the Iceland Basin..	25
Figure 10 Model results of signal attenuation as a function of surface roughness	26
Figure 11 Float displacements and trajectories.....	27
Figure 12 Float duration chart	28

TABLE OF TABLES

Table 1 Overview of 2014 deployment cruises	11
Table 2 Overview of 2015 deployment cruises	11
Table 3 Overview of 2016 deployment cruises	11
Table 4 Overview of 2017 deployment cruise	11
Table 5 Overview of 2018 sound source recovery cruises	11
Table 6 Initial ballasting performance results	14
Table 7 Float ballasting performance by target pressure and year.....	17
Table 8 Percentage of floats completing full missions as a function of target pressure	29
Table 9 Summary of data return for all OSNAP floats	30
Table 10 Sound source	32

1 ABSTRACT

The Overturning in the Subpolar North Atlantic Program (OSNAP) is an international effort started in 2014 dedicated to achieving a better understanding of the link between dense-water formation and the meridional overturning circulation in the high-latitude North Atlantic. Moorings, gliders, and subsurface acoustically-tracked RAFOS floats have been used to collect temperature, salinity, and current data across the Labrador Sea, Irminger Sea, Reykjanes Ridge, Iceland Basin, Rockall-Hatton Plateau, and Rockall Trough. The specific objective of the OSNAP float program is to gather information on the pathways of the dense overflow waters transported by the deep limb of the overturning circulation and assess the connection of those pathways with currents observed crossing the OSNAP mooring line. This data report details the observations collected by 148 floats that were deployed for OSNAP during the summers of 2014, 2015, 2016 and 2017. Deployment locations were in the Iceland Basin, Irminger Sea, and in the Charlie-Gibbs Fracture Zone. Mission lengths ranged from 540-730 days, and the floats were ballasted to passively drift at a fixed pressure of either 1800, 2000, 2200, 2500, or 2800 dbar to tag the deep overflow water masses of the subpolar North Atlantic (Iceland-Scotland and Denmark Strait Overflow Waters).

2 INTRODUCTION

Determining the pathways of the meridional overturning circulation in the subpolar North Atlantic is of vital importance to understanding the global climate system, as the latter effects heat and fresh water transport, as well as the carbon cycle. The Overturning in the Subpolar North Atlantic Program (OSNAP) is an international effort started in 2014 dedicated to achieving a better understanding of the link between dense-water formation in the subpolar North Atlantic and Nordic Seas and the meridional overturning circulation in the high-latitude North Atlantic. Moorings, gliders, and subsurface acoustically-tracked RAFOS floats have been used to collect temperature, salinity, and current data across the Labrador Sea, Irminger Sea, Reykjanes Ridge, Iceland Basin, Rockall-Hatton Plateau, and Rockall Trough (Figure 1). The so-called OSNAP Line (Figure 1) cuts across the entire subpolar North Atlantic from the east coast of Labrador to Scotland, and acts as a picket fence to measure water properties from the ocean surface to seafloor on a decade-long timescale. Moorings, hydrographic surveys, and gliders were all deployed along this survey line, and the RAFOS floats, for the most part, were deployed to begin drifting from this line.

Figure 1 The OSNAP line and sound source mooring locations. The thick black line denotes the OSNAP line, double circles indicate sound source locations, and dashed red lines mark float deployment locations. Sound source moorings are numbered 1-13, as referenced in text below. Bathymetry is shaded at 1000-m intervals. Geographic locations referred to in text are marked, with 'RR' for Reykjanes Ridge, 'MAR' for Mid-Atlantic Ridge, and 'CGFZ' for Charlie-Gibbs Fracture Zone.

OSNAP-Floats was proposed as a complementary Lagrangian component within the mooring-centric OSNAP for several reasons. First, the interpretation of the Eulerian OSNAP measurements would be significantly enhanced by an understanding of the flow pathways between arrays, particularly at the Overflow Water (OW) level ($\sigma-\theta \geq 27.80$, e.g., Dickson and Brown, 1994) where there have been no direct pathway (Lagrangian) observations. Second, as the OSNAP program sought to understand how OW impacts overturning in the basin and how climate signals from the Nordic Seas are propagated to lower latitudes, it is important to know OW pathways and transit times through the North Atlantic Subpolar Gyre and the mechanisms leading to pathway variability. Finally, deep pathway information would considerably aid the optimization of a long-term monitoring system by exposing where overflow transport is most geographically constrained.

The OSNAP-Floats goals were to provide the first ever direct observations of OW pathways in the subpolar North Atlantic, and along with a complementary numerical float study, address the following objectives:

1. Relate spatial and temporal changes in deep boundary current transport and property fluxes across the OSNAP line to OW pathways.
2. Identify preferred locations for boundary-interior exchange.
3. Determine the mechanisms that lead to boundary-interior exchange and investigate how these mechanisms impact transit times.
4. Investigate interannual and longer period variability in OW pathways and relate this to changes in source conditions and wind forcing.

This data report describes the observations collected by 148 RAFOS floats that were deployed for OSNAP-Floats during the summers of 2014, 2015, 2016 and 2017. Deployment locations, some repeated, were in the Iceland Basin, Irminger Sea, and in the Charlie-Gibbs Fracture Zone (CGFZ, Figure 1). Mission lengths ranged from 540-730 days, and the floats were programmed to passively drift at a fixed pressure of either 1800, 2000, 2200, 2500, or 2800 dbar to tag the deep overflow water masses of the subpolar North Atlantic (Iceland-Scotland and Denmark Strait Overflow Waters). Once every 24 hours, the floats would record acoustic signals emitted from a moored sound source array that was deployed specifically to track the floats for this project (Figure 1). The floats recorded the acoustic times-of-arrival (TOAs) and correlations (measure of signal quality) from the sound sources (used to reconstruct the float trajectories), along with water temperature and pressure. This report includes the instrument deployment strategy, cruise summaries, an overview of the RAFOS float processing, sound source drift calculations, a review of the lessons learned from the ballasting process, and details of acoustic signal quality relating to sound source and float performance. The Appendices contain trajectory, property (temperature/pressure/velocity), and acoustic signal plots for each float.

3 FLOAT AND SOUND SOURCE DEPLOYMENTS

Approximately 50 floats were deployed per year to seed overflow waters in the Iceland Basin, Irminger Sea, and in the CGFZ. Float deployment locations (Figure 2) were all located on the OSNAP Line except in 2014, when 10 floats were deployed in the CGFZ and in 2017, when 10 floats were deployed halfway between the Iceland Basin deployment location and the CGFZ, along a leg of the RREX hydrographic program (Branellec et al., 2018). RAFOS deployments were concentrated at the boundaries of the basins, where the deep overflow currents are banked up against the slopes of the east and west Reykjanes Ridge and Greenland, and funneled through the CGFZ. Deployments off the OSNAP Line were made when we had ships-of-opportunity available that passed over locations useful to studying the OW pathways, locations which would be illuminating to the goals of OSNAP-Floats.

Figure 2 Float deployment location by year and region. Red: east Reykjanes Ridge, Cyan: west Reykjanes Ridge, Purple: east Greenland, Blue: Charlie Gibbs Fracture Zone, Pink: southeast Reykjanes Ridge, Black: Monitor floats.

Figure 3 Yearly RAFOS float deployment locations (triangles) plotted by depth and distance on the OSNAP line. Shading shows the 30-day mean velocity for the month, where the along section velocity is normal to section, and positive velocity is poleward. The 0 cm/s isotach is drawn as a solid black line. Sigma- θ contours that cap the ISOW (27.80) and DSOW (27.88) are dashed lines. Floats that appear to be resting on the seafloor are in fact free floating – plotting is limited by the course bathymetry patch used to render these subplots.

The floats were all ballasted for depths from 1800-2800 m, which are denser than the traditional top of the OW layer ($\sigma\text{-}\theta = 27.80$) at the deployment sites. Float deployment locations in depth-space along each section are shown in Figure 3. Each seeding of float is shown plotted over the 30-day mean along-stream velocity data collected from the OSNAP mooring array, at the time of float deployment. All float deployment locations are below 27.80 $\sigma\text{-}\theta$, except along the west Reykjanes Ridge in August 2016, where the shallowest (1800-dbar) floats were located above the ISOW layer by 10-50 m. No CTD casts were made during the CGFZ float deployment, and the CTD data from the RREX section along the southeast Reykjanes Ridge is not shown. All OSNAP-floats, including those deployed in the CGFZ, that passed from the Iceland Basin through the CGFZ have been shown to be embedded well below 27.80 $\sigma\text{-}\theta$ (Zou et al., 2020). Judging from the placement

of floats in the ISOW layer in the Iceland Basin and at the CGFZ, and the float-measured temperature and pressure, we are confident that the floats deployed along the RREX line were also embedded in the ISOW layer.

Floats were deployed to blanket the continental slopes, with target depths chosen to keep the floats about 150 to 200-m above the seafloor. In some regions, floats with different target pressures were stacked at single stations. This was a particularly useful technique along the west Reykjanes Ridge, where very little was known about the flow field due to a lack of historical data. East of Greenland, floats were not stacked, as we presumed that floats from other regions would trace out the OW currents and ‘fill in the blanks’ as the floats transited through the subpolar ocean in the deep boundary current.

A subset of 18 floats, called either ‘monitor’ or ‘dual-purpose’ floats, were programmed with shorter mission lengths (12-317 days), and were used to assess the health of the sound source tracking array. The ten monitor floats (black symbols, Figure 2) were recycled from a previous float program: found washed up on shore by fishermen and beachcombers, shipped back to WHOI, refurbished, and deployed without re-ballasting at approximately 1000-m depth. The sole function of these monitor floats was to surface periodically during OSNAP-Floats and relay data in order to evaluate the acoustic signal quality of the source sound array. The eight dual-purpose floats were used as both monitor and “science” floats. These floats were also programmed to surface at various time steps throughout the program in order to further evaluate the functioning of the sound source array, allowing the opportunity to replace or add sound sources as needed. The dual-purpose floats were ballasted for the ISOW layer.

4 DEPLOYMENT AND RECOVERY CRUISES

The OSNAP-Floats deployments spanned a four-year period, with multiple cruises per year. Many of the participants were volunteers not directly involved with OSNAP-Floats. The RAFOS portion of OSNAP consisted of deploying 148 RAFOS floats (130 science, eight dual-purpose and 10 monitor) and 13 sound source moorings. Tables 1 through 5 list the cruise dates, region, cruise objectives, and ship's name for each year. In total, twelve different cruises were utilized to deploy and recover instruments related to this program. Science floats deployed in Year 1, 2, and 3 were programmed for mission lengths of 730 days, and in Year 4 for 540 days. The shorter mission length in Year 4 was dictated by the recovery of the sound sources in Summer 2018. Mission lengths ranged from 8-317 days for the monitor and dual-purpose floats. Details of float deployments can be found in Appendix A, Tables 12-15. Additional details of sound sources may be found in Chapters 5 and 6.

Ship	Region	Dates	Floats		Sound Source
			Science	Monitor	
<i>R/V Knorr</i> KN221-01	CGFZ	Jun 19 – Jul 02	9	3	4
<i>R/V Knorr</i> KN221-02	East Reykjanes Ridge	Jul 09 – Jul 23	9	3	3
<i>R/V Knorr</i> KN221-03	East Greenland	Aug 07 – Aug 26	17	3	4
<i>R/V Thalassa</i> (replacement vessel for MSM40)	Southeast Reykjanes Ridge	Aug 14	0	0	1
Total 2014			35	9	10

Table 1 Overview of 2014 deployment cruises. Date ranges are dates instruments were deployed. Dual-purpose are included with science floats.

Ship	Region	Dates	Floats	
			Science	Monitor
<i>R/V Pelagia</i> 64PE399	East Reykjanes Ridge	June 27-29	15	3
<i>R/V Pelagia</i> 64PE400	West Reykjanes Ridge	Jul 10-13	12	2
	East Greenland	Jul 17- 23	17	3
<i>CCGS Hudson</i>	Labrador Sea	Nov 12	0	1
Total 2015			44	9

Table 2 Overview of 2015 deployment cruises. Date ranges are dates instruments were deployed. Dual-purpose are included with science floats.

Ship	Region	Dates	Floats	Sound Source
<i>RRS Discovery</i> DY053	East Reykjanes Ridge	Jul 11-14	16	1
<i>RRS Discovery</i> DY054	East Greenland	Aug 1	14	
	West Reykjanes Ridge	Aug 6-9	11	1
<i>R/V Armstrong</i>	Sound Sources	Aug 5	-	1
Total 2016			41	3

Table 3 Overview of 2016 deployment cruises. Date ranges are dates instruments were deployed. Floats are all 'science'.

Ship	Region	Dates	Floats
<i>R/V Thalassa</i>	Southeast Reykjanes Ridge	Jul 20 – 21	10
Total 2017			10

Table 4 Overview of 2017 deployment cruise. Date ranges are dates instruments were deployed. Floats are all 'science'.

Ship	Cruise ID	Dates	Recovered
<i>R/V Maria S Merian</i>	MSM74	May 24 – Jun 26	2 (#1,#2)
<i>R/V Armstrong</i>	AR30-03	Jun 5 – 24	2 (#4, #11)
<i>R/V Armstrong</i>	AR30-04	Jul 1 – 30	3 (#5,#6,#7)
<i>R/V Armstrong</i>	AR30-05	Sep 7 – Oct 13	1 (#3)

Table 5 Overview of 2018 sound source recovery cruises. Identification numbers of sound sources recovered on each cruise are listed in parentheses. Date ranges are full span of cruise.

5 RAFOS AND SOUND SOURCE ARRAYS

5.1 RAFOS FLOAT SYSTEM

The Ranging and Fixing of Sound (RAFOS) float is an acoustically tracked subsurface Lagrangian drifter programmed to listen for signals from moored sound sources at scheduled intervals. The floats are designed to passively drift at a fixed pressure, recording hydrographic properties of temperature and pressure, and acoustic times-of-arrival (TOAs) and correlations from sound source signals at scheduled time intervals. They are equipped with a thermistor, pressure sensor, microprocessor, internal clock, hydrophone, and an Iridium antenna. Once the float's mission is complete, it is programmed to send an electrical signal through the external burn wire, causing the drop weight to release from the float. The float, suddenly positively buoyant, proceeds to rise to the surface. Once at the surface, it sends the data collected for the entire mission via the Iridium satellite system, to be processed and tracked. See Rossby et al. (1986) for a comprehensive description of the RAFOS system.

The 260 Hz sound sources used in this experiment emit an 80-second long swept frequency signal, from 259.38 Hz to 260.90 Hz, at the 'pong time' assigned to each of the sound sources. These pong times were scheduled to be ~15 minutes apart so that acoustic signals from the individual sources did not overlap during the float's listening window.

The floats were programmed to open a listening window every 24 hours at 00:00 UTC, and remaining open over the time span in which sound sources were scheduled to pong, recording the TOA and correlation value of the acoustic signal from the sound source as measured by the hydrophone. The correlation values indicate how well the 80-second swept signal received by the float matched the signal sent by the sound source. The floats were set to record the six best correlated TOAs during each listening window. At the end of the listening window, in this case 90 minutes, temperature and pressure were also recorded.

Upon deployment, the floats were set to activate after measuring 5 dbars pressure, or after 30 days, whichever came first. Float clocks were synchronized on initialization with a satellite GPS clock before deployment. Most often, the floats were turned on and the clocks were synchronized while the ship was still in port.

5.2 FLOAT OVERVIEW

All RAFOS floats used in this program were purchased from Seascan, Inc. in Falmouth, MA. Floats were RAFOS DLD II floats built with 90-mm OD x 7-mm wall thickness 'deep glass' for operational pressure of 3000 dbar, UHDP (ultra-high density plastic) insulators (that act like a separator

between the float and ballast weight), and no high-accuracy clock modules. MetOcean Telematics (formerly, JouBeh, Inc.) in Dartmouth, Nova Scotia, was the Iridium data provider.

5.3 FLOAT BALLASTING

All OSNAP floats were ballasted at the University of Rhode Island Graduate School of Oceanography (URI-GSO) ballasting facility. Since the floats were being deployed in the cold abyssal waters of the North Atlantic, they needed to be ballasted in a tank that could best emulate the cold, high pressure conditions. The URI-GSO ballasting tank cooled the water to ~5°C, close to the ambient temperature of the OSNAP float deployment sites, and had a working pressurization of 2200 psi (J. Fontaine, pers. comm. 3-Dec-2020). At the time, the WHOI tank did not have a chiller, and pressurized only to 1500 psi. Therefore, the compressibility data collected at URI-GSO, used to calculate ballast weight, would have greater accuracy, and less extrapolation would be needed when calculating the final weight to be added to the float to take it to the target pressure.

In practice, engineering concerns with the URI-GSO tank meant that was only pressurized to 1500 psi for the OSNAP project, but the chiller worked as expected and we were able to ballast floats close to the ambient temperature of the deployment sites. The OSNAP floats were ballasted over the course of three springs, from 2014 to 2016. The pressure strain gauge used in the URI-GSO tank began to degrade in late Spring 2016, when the 2016 and 2017 float compressibilities were measured in the tank.

Adjustments to float build and ballasting procedures occurred once the first set of floats surfaced (Table 6, OSNAP RAFOS Ballasting Assessment). Some of these first six floats surfaced as expected and completed their missions, while some surfaced early. Two issues were cause for concern: 4/6 of these floats seemed to surface after hitting the bottom, and 5/6 had settled at pressures greater than intended. Some variability in pressure (generally ± 50 dbars) is normal and expected, as seen in other float programs (e.g., Furey and Bower, 2009), but 4/6 floats were overpressure by more than 100 dbars, a cause for concern and remediation.

OSNAP RAFOS Ballasting Assessment										
S/N	launch region	programmed mission length (days)	actual mission length (days)	target pressure (dbar)	~initial settle pressure (dbar)	Δ dbar	initial settle depth (m)	water depth at launch (m)	initial height off bottom (m)	failure mode / good
1319	CGFZ	180	180	2200	2194	-6	2163	2485	322	good
<i>Float looks OK, advected in current travelling NE, entrained in anticyclone to east of current, then trapped in small canyon of west flank of MAR. TOAs blocked by bathy in canyon.</i>										
1330	East RR	180	180	2200	2258	58	2225	2300	75	good
<i>Float looks OK, travelled SSW along east RR. Reason for fade out of TOAs is unclear - winter?</i>										
1345	East GR	180	15	2800	2922	122	2875	2898	23	bottom hit
<i>Float is too deep, initially free-floating; hit bottom, went overpressure, and surfaced.</i>										
1342	CGFZ	730	152	2800	2923	123	2877	2944	67	leaker / bottom hit
<i>3.5 months good (possible hit at about 1 month), then leaks, overpressure and sfc. Fadeout of TOAs begins ~mid-September.</i>										
1347	East GR	730	157	2800	2960	160	2912	3000	88	bottom hit
<i>Float is too deep, initially free-floating; hit bottom, went overpressure, and surfaced.</i>										
1349	East GR	730	226	2800	2930	130	2882	2949	67	bottom hit
<i>Float is too deep, initially free-floating; hit bottom, went overpressure, and surfaced.</i>										

Table 6 Initial ballasting performance results. Floats included were the first six floats that surfaced which were ballasted to the OW pressures of 1800 dbar or greater. Acronyms: CGFZ, Charlie-Gibbs Fracture Zone; GR, Greenland; RR, Reykjanes Ridge; MAR, Mid-Atlantic Ridge; TOAs, times-of-arrival.

5.4 BOTTOM HITS AND TAILWIRES

The most likely reasons for a float to surface after interacting with the bottom are thought to be (1) losing their dropweight through interaction with the rough bathymetry or (2) picking up sediment when the float touches or drags along the bottom. In the latter case, the floats may become too heavy and sink past the bailout pressure, which was set to 3000 dbar for this experiment. Bailout is set to a depth where the float will register ‘overpressure’ and have enough time to send a command to activate the burnwire, detaching the dropweight, and surface before reaching crush depth for the glass housing. This ensures that if a float mission is cut short due to overpressure, the data collected up to that point is still retrieved.

The early bailouts associated with interactions with the sea floor led the team to add a ‘tailwire’ to some of the remaining OSNAP floats being deployed in 2015, 2016 and 2017. This was

particularly important for the floats with target ballast depths close to the bottom. A plastic-sheathed stainless steel tailwire was added to about half the floats (Appendix A: Tables 5-8). The two-meter long tailwire was attached to the float around an arm of the insulator with a crimp before ballasting. The tailwire was meant to hang vertically below the float in the water column, and be the first part of the float that would come in contact with the seafloor. Upon contact, the tail wire would rest on the seafloor, effectively lessening the overall weight of the float, allowing the float to become positively buoyant and lift off the bathymetry. Although we cannot conclusively determine if the tailwire was effective in preventing early risers, the percentage of floats that terminated their mission after interacting with bathymetry (so-called early risers) was reduced from 16% for the first year's deployment, to 0% for the remaining floats deployed once tailwires were employed. (These percentages do not include floats that did not surface at all, as we cannot determine probable cause of failure. There were 4/5/1/0 "did not surface" (DNS) floats in Years 1/2/3/4, respectively.)

5.5 OVER-BALLAST ASSESSMENT AND REMEDIATION

The second problem, where floats appeared to be ballasted 'too deep', initiated a detailed review of the ballasting process at both WHOI and URI, resulting in modification of how the compression results from the ballast tank were used to calculate the weight to be added. We hired an outside contractor, Ed Slate, to review the ballast procedure (see Slate Report in Appendix I). The floats were settling at pressures greater than expected because the calculated 'weights-to-be-added' were too great, resulting in a higher density instrument which settled deep. With the RAFOS float design used in this program, an error of 1 gram is approximately equal to 50-meters depth error, with the magnitude of this error dependent on local ocean properties at the site of deployment. The work done by Slate showed that the extrapolation of the compressibility data measured while the pressure tank cycles from 10 to 1500 PSI (~7 to 1034 dbars) to greater pressures leads to errors in weight-to-be-added (because the compressibility was assumed to be constant with depth). The OSNAP floats were ballasted to depths between 1800 and 2800 dbars, and the overballast increased with increasing pressure. This float program had target pressures much deeper than all recent (last 20 years) float programs. The central remediation was to modify the ballasting equations from linear extrapolation to a second order polynomial extrapolation, thus lessening the weight-to-be added with increasing pressure.

However, after a few more floats surfaced, this did not seem to solve the entire problem. After some initial Year 2 (2015) results from the short-mission deployments, the settle pressures of the floats were still too deep. As Year 3 (2016) floats had already been deployed, the final preparation of the floats for Year 4 (2017) included a 'fudge factor' of up to 1.0 grams less weight based on target pressure. The additional weight removed was based on ballast target pressure:

1800 dbar target had no additional weight removed, 2000 dbar had 0.5 grams weight removed, 2200 dbar had 1.0 grams weight removed.

A post-mortem was performed on these decisions and are summarized in Table 7 and in Figure 4, with performance data for each individual float presented in Appendix A: Tables 5-8. Figure 4 plots the differences between initial measured pressure and target pressure as a function of target pressure and color coded by year. If ballasting were perfect, we would expect to see tight clusters of floats about 0-dbar and at each target pressure. Instead, the 2014 floats show initial depths were all missing the target depths in the mean, and progressively worsen with depth from 24 dbar at 1800-dbar, to 131 dbar at 2800-dbar targets. This was due to the first- versus second-order extrapolation problem described above. The issue was solved in Years 2 and 3, where we can see the average differences are all about 20 dbars or less, and indistinguishable from zero within the standard deviation window, except at 1800-dbar in 2016, where the mean is 47 dbars too deep. The floats deployed in 2017, which included the fudge-factor of further weight removed, clearly show that this was not a good choice. Here the floats become progressively too light with depth, with mean pressures 45 to 88 dbars too light. Future float experiments should utilize the 2nd-order polynomial fit only. The additional fudge-factor weight does not seem to explain the total pressure differences. For example, the 1800-dbar float target were 45 dbars too deep in the mean, but had no additional weight removed. These floats were ballasted along with the 2016 floats, when the strain gauge was failing. We do not see any reason that these ten floats would have been different than the other 2016 floats – e.g., the 2017 floats were not all ballasted on the same day, or at the end of the ballasting session, when the strain gauge failure may have been exacerbated. We are left to suppose that the local ocean properties at the southeast Reykjanes Ridge site may have been different than the other sites, and explain the ~45 dbar offset at each target pressure.

The standard deviation of the float ballasting fit were larger in the mean in Years 3 and 4 compared to Years 1 and 2, 33 and 34 dbar versus 16 and 22 dbar respectively (Table 7). This may be due to the degrading pressure strain gauge in the URI-GSO tank that occurred in 2016 when the Years 3 and 4 float compressibilities were measured. Great pains were taking to obtain the best quality compressibility data possible for ballasting when the strain gauge was failing, including collecting data on a “standard” float between each OSNAP float pressurization.

Year	Pressure	Mean	Standard Deviation
2014			
2014	1800	24	11
2014	2000	40	11
2014	2200	43	23
2014	2500	80	24
2014	2800	131	12
	All	65	16
2015			
2015	1800	2	28
2015	2000	6	27
2015	2200	-17	19
2015	2500	-19	31
2015	2800	2	4
	All	-6	22
2016			
2016	1800	47	32
2016	2000	29	36
2016	2200	-2	52
2016	2500	-21	22
	All	18	35
2017			
2017	1800	-45	24
2017	2000	-79	23
2017	2200	-88	52
	All	-72	33

Table 7 Float ballasting performance by target pressure and year. Mean and standard deviation have been calculated for all floats with a specific pressure target and year, e.g. 1800-dbar target pressure in 2017. In addition, overall mean and standard deviation have been calculated for each year. Floats that are categorized as 'WP', or wrong pressure, with pressures greater than 190 dbar off target, are not included in the calculations. Data for each float may be found in Appendix A, Tables 16-19.

Figure 4 Ballast offsets (initial minus target pressure) as a function of target pressure and year. Data for each float may be found in Appendix A: Tables 5-8.

5.6 SOUND SOURCE ARRAY

Ten sound sources were funded as part of the OSNAP float program. Source sites were chosen to best ensonify three subpolar basins: Labrador, Irminger and Iceland, as well as the area around the southern terminus of the Reykjanes Ridge. A balance was struck between the number of sound source moorings that could be funded, and sound source signal coverage across the basins. The science objective of the program was to determine pathways of the OW within the subpolar North Atlantic, generally north of 45°N. It was understood before the moorings were placed that we would not be able to track the floats at the northern ends of each basin (Iceland, Irminger,

Labrador), or south of about 50°N. We knew if tracking were lost at the heads of each basin that the deep currents were contained within these areas, and the science objectives of this program would still be met.

The original 10 sound sources were all built by J. Valdes at WHOI, using the URI-WHOI design (Figure 5). Moorings were deployed and recovered as described in Chapter 4 (above), and mooring diagrams may be found in Appendix B.

Figure 5 URI-WHOI designed sound sources built by J. Valdes in the racks at sea before deployment in summer 2014.

We chose 1200-meter depth for all sources, deeper than the existing sound channels in these regions (we note here that there is only a half sound channel in the subpolar North Atlantic for much of the year due to cold near-surface temperatures), because the floats are to be ballasted for 1800-meter depth or deeper. We would have chosen a deeper depth for the sound sources (as was done in the BOUNCE program – where sources were moored at 1500 m), but the pressure rating of the sound source transducer used in this study (as opposed to BOUNCE) was rated to nominal 1250-meters depth, providing a limit for instrument placement in the water column. (An aside: In discussions prior to designing the source moorings, T. Rossby (pers. comm.) stated he

did not feel it mattered much for deep signal strength to place the sources deeper than 1200 meters.)

After the programmed short mission floats (180, 270-day missions) surfaced, it became apparent that additional sound source moorings were needed, as the signals attenuated much more than anticipated, especially in winter. This is described in detail in the next section. Three additional sources were put together from inventory and donated to the OSNAP program and deployed in Summer 2016. One of the sources (#11) was a Webb design source.

Sound sources 1-7 and 11 were deployed on or near the OSNAP line, and therefore were equipped with acoustic releases so that they could be recovered during OSNAP current meter mooring service cruises at the end of the OSNAP-Floats program. Sound sources 8-10 and 12 and 13 were deployed far from the OSNAP line and without releases, as the ship time costs to recover them were greater than the cost to replace the sources and moorings. All sources with acoustic releases were successfully recovered. The transducer tube for sound source #4 was significantly pocked with corrosion, all zincs were depleted, and this resonator tube had to be disposed. The other sources with similar builds (and number of zincs) did not suffer the same damage.

6 SOUND SOURCE PERFORMANCE AND SIGNAL ATTENUATION

All sound sources performed well. There were no source failures, and all source moorings equipped with acoustic releases were recovered. The signal quality was weaker than expected, though, so three additional source moorings were deployed in the northern Irminger Sea, near the southwest terminus of both the Reykjanes Ridge and Rockall Plateau. These additional sources significantly improved float tracking where float tracking was necessary to meet project goals. However, the regions where no tracking was expected (heads of each basin and south of 50°N) remained the same – untrackable. Below is an explanation of time-of-arrival signal attenuation due to distance from source, and from surface roughness and how this effected the data return.

6.1 EFFECTS OF SIGNAL ATTENUATION

Time-of-arrival signal weakening looks like a fading of a signal over time and distance. The trajectory will have regions where no tracking is possible, due to floats being out of range of two or more sound sources (the minimum needed to track), and due to decreased seasonal ranges of the sources. Figure 6 through Figure 8 illustrate this using a single float (#1376) as an example. The missing trajectory segments are due to missing times-of-arrival data, shown in Figure 7 for the same float. In the resultant float properties (Figure 8) velocity data is missing when the float is not able to be tracked, but pressure and temperature are continuously measured.

Figure 6 Float trajectory color coded by temperature, and dated MM/DD at open circles on trajectory. Inset shows float displacement between launch and surface, with sound source positions marked as black circles. Different regions of float trajectory quality are annotated.

Figure 7 Float #1376 times-of-arrival data collected during the floats two-year mission. Each signal is attributed to a different sound source mooring, and can be identified at the start and end of the float mission, when location determined by GPS launch and surface locations. Blue and pink shaded regions correspond to similarly shaded annotations in Figure 6, winter (blue) and extended distance from sources (pink).

OSNAP1376

Figure 8 Float #1376 property data collected during the floats two-year mission. U- and v-velocity subplots show where tracking lost due to float being out of range of sources, while properties of pressure and temperature are continuously recorded. Blue and pink shaded regions correspond to similarly shaded annotations in Figure 6 and Figure 7, winter (blue) and extended distance from sound sources (pink).

6.2 CAUSES OF SIGNAL ATTENUATION

We worked with Tim Duda and Art Newhall in WHOI's AOP&E Department to determine the cause of the signal attenuation seen in these float data (Figure 6). We checked to see if the TOAs were degraded with increasing float pressure. Although this was true for the deepest floats (not shown), Duda/Newhall felt that this was due to the floats being too close to the bottom and the signals being affected by bottom topography roughness, rather than depth. Acoustic modeling of sound signal intensity over a 600-km section in the Iceland Basin using local climatological sound speed and hydrographic profile data were done using sound sources placed at different depths (Figure 9). Positioning sources at 400-, 1200-, or 2000-m depth did not qualitatively alter signal intensity at the deepest float depths. At 2000-m (but not at the other two depths tested), there is increased signal intensity when near (within about 50 meters) the depth that the sound source is moored. The primary reason for signal degradation at extended distances from the sources was the range limitation due to 'open' or near-open sound source channels in this ocean region. With sound speed profiles providing no cap on the sound channel for many months in the year, the acoustic signal is not refracted in the sound speed channel as it is in subtropical latitudes. This effect was anticipated, and significantly shorter ranges were considered when designing the sound source array.

Figure 9 Sections of signal intensity in decibels by depth and distance on a section through the Iceland Basin. From left to right, sources were positioned at 400-, 1200-, and 2000-m depths, and sound sources are placed on the left hand side of each plot. Courtesy of T. Duda and A. Newhall, WHOI.

A second concern about signal attenuation was the dramatic loss of signal in winter. Duda and Newhall provided modelling results of signal loss that increased with increasing sea surface roughness. With the sound speed profile open in the subpolar region, the signals that refract upward hit the sea surface. The flatter the surface, the better the signal can reflect back downward and retain intensity. With increasing wind speed, the sea surface becomes increasingly rough, scattering the sound signal and diminishing acoustic ranges.

This is shown in Figure 10, where wind speeds are almost an order of magnitude stronger in winter than summer in the mean. The ‘Erbe-Farmer’ loss per bounce for 1- and 3-m waves (Figure 10) shows the resulting sound degradation due to winter surface roughness (e.g., Erbe and Farmer, 2000). We did not anticipate this severe degradation due to sea surface roughness in winter when designing the tracking array, resulting in less data being collected during winter.

Figure 10 Objective analysis model mean 2000-2012 wind speeds and vectors for January and July, and signal attenuation as a function of surface roughness. Courtesy of T. Duda and A. Newhall, WHOI.

7 FLOAT PERFORMANCE

The OSNAP-Floats study has generated a comprehensive amount of Lagrangian data using RAFOS floats. Of the 148 floats deployed, 13 were monitor floats, and 135 were science or dual-purpose floats. Data from 123 (out of 135 possible) floats were collected between June 2014 and January 2019 (Figure 11). The results discussed here include the full mission and dual mission floats.

Figure 11 Float displacements (left) and trajectories (right) collected as part of the OSNAP-Floats program. Floats are color-coded by initial density measured by shipboard CTD and moorings at the time of deployment, with green indicating the float was deployed in ISOW, and blue indicating the float as deployed in DSOW. Float displacements and trajectories represent different mission lengths, with the majority of floats measuring over 1.5 years.

7.1 FLOAT MISSION LENGTHS

Float mission lengths for OSNAP ranged between 12 and 730 days. Full mission floats for Years 1, 2, and 3 were 730 days, and 540 days for year 4. Floats programmed with short missions (less than 540 days) were used to evaluate the sound source signal strength. Float mission length as a function of time are shown in Figure 12. This presentation shows the improved float retention after Year 1 and 2, with fewer early-release (ER) and DNS floats, when the 2800-dbar float depth was abandoned and tailwires were added to the floats deployed closest to the seafloor. Monitor floats were only used in Years 1 and 2, when it was possible to still replace or deploy a sound source mooring and have it be a cost effective benefit to the program.

Figure 12 Float duration chart. Float serial numbers are noted to left, and codes are as follows: DNS- did not surface, M – monitor float, ER – early riser.

7.2 ABANDONING THE 2800-DBAR PRESSURE TARGET

The 2800-dbar pressure target was abandoned for Years 3 and 4 of the float program. The success rate of the 2800-dbar floats in Years 1 and 2, defined as completing a full mission, was only 58%, as compared to the 89-97% success rates of the floats at the shallower depth levels (Table 8). This pressure target was considered too risky to the success of the program, and float assets were re-directed to 2500-dbar (still in the DSOW) and shallower.

Target Pressure	Deployed	Completed Full Mission	% Success	Early Riser	Did Not Surface
1800	39	38	97%	0	1
2000	30	27	90%	1	2
2200	47	42	89%	0	5
2500	19	17	89%	0	2
2800	12	7	58%	4	1

Table 8 Percentage of floats completing full missions, and counts of ER and DNS floats, as a function of target pressure.

7.3 DATA RETURN

Of 135 science and dual-purpose floats deployed, 87% (117/135, Table 9) surfaced and completed the full programmed missions. The failures were largely from the floats deployed at the East Greenland site, where only 78% (40/51) of floats returned data as expected. Of the floats deployed at the remaining sites, 92% (77/84) surfaced as expected, a rate in keeping with other recent float programs. As shown in Table 8, above, the percentage of floats completing full missions was 89% or higher for all target pressures except the 2800-dbar target, which was only 58%. We collected 238 float years' worth of good data out of an expected 258 years (84,737 days collected, 94,040 days expected). Total success for the program in terms of expected versus collected data was 90%.

	Deployed	Fill Mission	Early Riser	Did Not Surface	Bad Data
Monitor					
Year 1	6	5	1	0	0
Year 2	7	3	1	3	0
Year 3	-	-	-	-	-
Year 4	-	-	-	-	-
Total	13	8	2	3	0
East Reykjanes Ridge					
Year 1	10	9	0	1	0
Year 2	15	13	0	1	1
Year 3	16	16	0	0	0
Year 4	-	-	-	-	-
Total	41	38	0	2	1
West Reykjanes Ridge					
Year 1	-	-	-	-	-
Year 2	12	11	0	1	0
Year 3	11	10	1	0	0
Year 4	-	-	-	-	-
Total	23	21	1	1	0
East Greenland					
Year 1	18	12	3	2	1
Year 2	19	15	1	3	0
Year 3	14	13	0	1	0
Year 4	-	-	-	-	-
Total	51	40	4	6	1
Southeast Reykjanes Ridge					
Year 1	-	-	-	-	-
Year 2	-	-	-	-	-
Year 3	-	-	-	-	-
Year 4	10	10	0	0	0
Total	10	10	0	0	0
Charlie Gibbs Fracture Zone					
Year 1	10	8	1	1	-
Year 2	-	-	-	-	-
Year 3	-	-	-	-	-
Year 4	-	-	-	-	-
Total	10	8	1	1	0
All Floats	148	125	8	13	2

Table 9 Summary of data return for all OSNAP floats. Full mission means the float completed the programmed mission length, which includes the short-mission dual purpose floats.

8 SOUND SOURCE DRIFT

Typically, a sound source clock drifts between a few seconds to 100 seconds (or more in rare cases) over the course of a multi-year deployment. To correct for this offset, for each sound source, we calculated the sound source clock drift and removed it from the sound source TOA signal before tracking. The sound source clock drifts were calculated both from observed offsets at deployment and (for sound sources 1-7 and 11) at recovery, and/or by the inferred offsets derived from the TOAs recorded by the floats. Both methods are described below. In practice, we used the inferred drifts when tracking the floats.

Measured Sound Source Clock Drift

First, the measured drift was calculated using the pre-deployment and post-deployment times for sound source clock and ship's clock (GMT). The sound source clocks were checked and recorded on deck just before deployment and were found to be synchronized with GMT. Upon recovery, the sound source clocks were immediately checked again and recorded. Observed drifts were derived by subtracting the sound source clock time at recovery from the ship's clock time at recovery and dividing by the length of time of the deployment (assumes source clock drift is linear) (Table 10).

$$\text{Measured Drift (seconds/day)} = \frac{(\text{Ship Time}_{\text{recover}} - \text{Sound Source Time}_{\text{recover}})}{\text{Length of deployment}}$$

Inferred Sound Source Clock Drift

The inferred sound source clock drift was calculated using the difference between the expected and observed signal arrival time at the launch and surface of each float's mission. We used only data values where float launch (surface) and first (last) recorded TOA were within 24 hours, and float had launch (surface) satellite-derived position data.

Estimated TOAs at the beginning and end of the float mission were established using the distance between the recorded GPS position of the float and the location of the sound source and dividing by sound speed representative of the mean at the sound channel axis.

$$\text{Estimated TOA}_{\text{initial}} = \frac{(\text{Sound Source Position} - \text{Float GPS Position}_{\text{initial}})}{\text{Sound Speed}}$$

$$\text{Estimated TOA}_{\text{end}} = \frac{(\text{Sound Source Position} - \text{Float GPS Position}_{\text{end}})}{\text{Sound Speed}}$$

The recorded TOAs needed to be adjusted for the pong time. Each sound source has a unique time at which they pong after the float's listening window opens. This allows us to identify which sound source the TOA came from. The true TOA value was found by subtracting the unique pong time from the recorded TOA values (Table 10).

$$True\ TOA_{initial} = RecordedTOA_{initial} - Sound\ Source\ Pong\ Time$$

$$True\ TOA_{end} = RecordedTOA_{end} - Sound\ Source\ Pong\ Time$$

The inferred drift was found by subtracting the Estimated TOA from the True TOA at the beginning and end of the mission and calculating a linear trend.

$$Inferred\ Offset_{initial} = True\ TOA_{initial} - Estimated\ TOA_{initial}$$

$$Inferred\ Offset_{end} = True\ TOA_{end} - Estimated\ TOA_{end}$$

$$Inferred\ Drift_{final} = linear\ regression(Recorded\ Offset\ Times, Observed\ Offsets)$$

In practice, inferred drifts are used in float tracking since the drifts are derived from several points and various instruments. For these data, the initial sound source clock offset (at deployment) used was zero and the inferred sound source drifts calculated above were applied to the final float tracks during processing. Details of the sound source positions, offsets, and drifts in graphical format may be found in Appendix B. Table 10 summarizes the sound source drifts used in final float data processing.

Name	Position		Depth (m)	Deploy	Recover	Reference (‘Pong’) Time (HH MM)	Measured Clock Offset (MM:SS)		Empirical Drift (sec/day)
	Lat	Lon					Initial	Final	
OS1	54.991	-49.503	1200	2014 08 25	2018 06 01	00 05	00:00	-00:19	-0.014
OS2	54.529	-53.993	1200	2014 08 14	2018 06 03	00 15	00:00	-00:77	-0.056
OS3	58.947	-47.460	1200	2014 08 23	2020 08 23	00 30	00:00	-	-0.040
OS4	59.754	-40.606	1200	2014 08 10	2020 08 10	00 15	00:00	-00:07	-0.005
OS5	59.039	-34.236	1200	2014 07 09	2018 07 17	00 45	00:00	-00:36	-0.025
OS6	58.021	-27.817	1200	2014 07 14	2018 07 23	00 05	00:00	-00:24	-0.017
OS7	58.0013	-22.999	1200	2014 07 15	2020 07 14	00 15	00:00	-00:07	-0.005
OS8	53.004	-29.991	1200	2014 06 27	2020 06 27	00 30	00:00	NP	0.009
OS9	50.493	-38.512	1200	2014 06 25	2020 06 25	00 15	00:00	NP	-0.038
OS10	52.496	-43.000	1200	2014 06 24	2020 06 24	01 00	00:00	NP	0.000
OS11	61.679	-33.898	1700	2016 08 05	2020 08 05	00 30	00:00	00:00	0.000
OS12	56.238	-25.984	1200	2016 08 11	2020 08 11	01 00	00:00	NP	-0.138
OS13	56.601	-40.998	1200	2016 07 08	2020 07 08	00 05	00:00	NP	-0.047
exB	53.496	-37.003	1200	2014 06 27	2020 07 16	01 00	00:00	NP	0.226

Table 10 Sound source initial and final measured offsets are reported in ‘Clock Offset’ columns, but drifts used in tracking were derived empirically from the difference between expected and measured times-of-arrival for each possible source and for each float combination at launch and at surface. ‘NNNP’ stands for ‘not possible’, as sound source mooring was not recovered. The final clock offset for sound source OS3 was not measured when the source was recovered. All sources were the URI-WHOI design, built by J. Valdes, except for #11 (a Webb design) and #2 (a URI-Rossby design).

9 FLOAT DATA PROCESSING

9.1 HEXADECIMAL TO DECIMAL DATA

When the float surfaced at the end of its mission, the stored data was transmitted, via Iridium cellphone technology, and received as emails with Short Burst Data (SBD) message files attached (one SBD message for each data point). Using the software package 'Python2RFB', these messages were imported, converted from hexadecimal to decimal format, and combined into four processed files: `Data_Drfit.txt`, `Data_Reciever.txt`, `Mission_Engineering.txt`, and `Surface_Engineering.txt`.

MATLAB functions were used to create RFB files from the four processed files above. The RFB file contained all data and information needed to track the floats using the ARTOA3 Float Tracking Software: float programming meta data, launch and surface position and times, float offset times, TOAs, correlation coefficients, temperature, pressure, and window times.

9.2 TRACKING

Using the ARTOA3 Float Tracking Software in MATLAB, the RFB and sound source file were loaded, valid pressure and temperature data were selected, and TOA values were assigned to a specific sound source. Once all data had been selected and assigned, the TOAs were corrected for the Doppler shift and difference in transmission times, and interpolated using a variable (usually 10-day) cubic spline fitting routine. Tracking included using a least-squares method if more than two TOAs were selected. A more detailed description of the float tracking method may be found in Wooding et al. (2005).

9.3 FINAL CALIBRATION AND QUALITY CONTROL

Outliers in the velocity data with greater than three standard deviations were replaced with NaNs. The final calibration of the float data required application of the final estimated sound source drifts. Once all tracking was completed and the quality of the data had been checked, all float data was combined into a single .mat MATLAB file and included data for 123 floats.

10 ACKNOWLEDGEMENTS

The authors would like to thank the captains, crews, and science parties of the *R/V Knorr*, *R/V Pelagia*, *R/V Armstrong*, *RRS Discovery*, *R/V Thalassa*, *R/V Maria S Merian*, and *CCGS Hudson*, for their able assistance in the deployment and recovery of the RAFOS sound source moorings and RAFOS floats. Jim Valdes and Keith Bradley of the WHOI RAFOS Float Operations Group are gratefully acknowledged for expert fabrication of the sound sources and preparation and ballasting of the floats. The WHOI Mooring and Rigging Group was responsible for fabrication of the sound source moorings. The field work and data processing were supported under U.S. National Science Foundation Grants OCE-1259618 and OCE-1756361.

11 REFERENCES

Branellec, P., and V. Thierry, (2018). RREX 2017. CTD-O2 Data report. Rap. Int. LOPS/18-04. <https://doi.org/10.13155/58074>

Dickson, R. R., and J. Brown (1994), The production of North Atlantic Deep Water: Sources, rates, and pathways, *J. Geophys. Res.*, 99(C6), 12,319–12,341, doi:10.1029/94JC00530.

Erbe C, Farmer DE (2000) Zones of impact around icebreakers affecting beluga whales in the Beaufort Sea. *Journal of the Acoustical Society of America* 108: 1332-1340.

Furey, H. H., and A. S. Bower, 2009. Export Pathways from the Subpolar North Atlantic: DLD2 RAFOS Float Data Report July 2003--November 2008. *WHOI Technical Report WHOI-2009-06*, 166 pp.

Lozier, M. S., S. Bacon, A.S. Bower, S.A. Cunningham, M.F. de Jong, L. de Steur, B. DeYoung, J. Fischer, S.F. Gary, B.J.W. Greenan, P. Heimback, N.P. Holliday, L. Houpert, M.E. Inall, W.E. Johns, H.L. Johnson, J. Karstensen, F. Li, X. Lin, N. Mackay, D.P. Marshall, H. Mercier, P.G. Myers, R.S. Pickart, H.R. Pillar, F. Straneo, V. Thierry, R.A. Weller, R.G. Williams, C. Wilson, J. Yang, J. Zhao, and J.D. Zika, 2017. Overturning in the Subpolar North Atlantic Program: A new international ocean observing system. *Bulletin of the American Meteorological Society*, doi: <http://dx.doi.org/10.1175/BAMS-D-16-0057.1>

Rossby, T., D. Dorson, and J. Fontaine, 1986. The RAFOS system. *J. Atmos. Oceanic Technol.*, **3**, 672-679.

Wooding, C., H. Furey, and M. Pacheco, 2005. RAFOS Float Processing at the Woods Hole Oceanographic Institution. *Woods Hole Oceanographic Institution Technical Report*, WHOI-2005-02, 35 pp.

Zou, S., A. Bower, H. Furey, S. Lozier, and X. Xu, 2020. Redrawing the Iceland-Scotland Overflow Water Pathways in the North Atlantic. *Nature Communications*, **11**(1), pp.1-8.

APPENDIX A: FLOAT SUMMARY TABLES

The tables in Appendix A list the programmed mission length, target ballast pressure, launch time and location, surface time and location, and status code for each float. Launch and surface information taken from GSP satellite data. There are four tables, one for each deployment year. Within the table, the floats are grouped by monitor float or deployment area: Charlie Gibbs Fracture Zone, East Greenland, East Reykjanes Ridge, Southeast Reykjanes Ridge, or West Reykjanes Ridge. The Status Codes are: **F** (full mission science), **S** (short mission), **M** (monitor), **DNS** (did not surface), **BAD** (bad data), **MD** (missing data), **ER** (early risers), **WP** (wrong pressure). **Full mission** science floats were programmed for either 540 or 730 days and returned a complete data set. The **short mission** floats were programmed with shorter mission lengths (180-270 days) and were used to monitor sound source array functioning while also collecting science data. **Monitor** floats were deployed at about 1000 dbar, programmed with very short mission lengths, and used to report back on health of sound source array. All the monitor floats used were recycled from a previous float program. **Early risers** refer to floats that surfaced before they completed the programmed mission length. Floats that **did not surface** never transmitted data at the end of their mission. Data designated as **bad** returned data that was not usable; float 1315 was on the surface during the entire mission and float 1356 float rested on the seafloor the entire mission. Float 1428 returned null pressure and temperature data, but tracking was good, and is designated with **missing data** status code. Several floats were mis-ballasted and are designated WP, for **wrong pressure**. Mis-ballasted is defined as greater than 190 dbar from target pressure.

Float	Mission Length (days)	Ballast Pressure (dbar)	Launch			Surface			Status Code
			YYYYMMDD	Latitude (N)	Longitude (E)	YYYYMMDD	Longitude (N)	Latitude (E)	
Monitor Floats									
1028	90	1000	20140826	56.522	-49.008	20141102	57.032	44.114	ER
1046	14	1000	20140721	59.964	-23.154	20140804	57.717	24.283	M
1081	12	1000	20140624	52.492	-42.997	20140706	52.815	42.565	M
1168	12	1000	20140624	52.492	-42.997	20140706	52.815	42.565	M
1177	270	1000	20140726	59.213	-37.478	20150511	62.961	31.580	M
1222	12	1000	20140722	57.962	-27.009	20140803	58.004	27.200	M
East Reykjanes Ridge									
1310	730	1800	20140723	58.620	-29.983	20160722	58.412	35.665	F
1311	730	1800	20140723	58.625	-29.934	20160722	59.486	37.788	F
1312	730	1800	20140723	58.571	-29.801	20160722	53.857	33.437	F
1313	730	2000	20140723	58.294	-29.321	-	-	-	DNS
1314	730	2000	20140723	58.344	-29.118	20160722	58.063	24.534	F
1316	730	2000	20140723	58.529	-28.588	20160722	50.851	35.472	F
1325	730	2200	20140722	57.993	-28.040	20160721	55.319	28.087	F
1326	730	2200	20140722	57.981	-27.926	20160721	50.270	32.071	F
1329	730	2200	20140723	58.082	-28.404	20160722	52.718	38.114	F
1330	180	2200	20140723	58.142	-28.542	20150119	54.719	32.381	S
East Greenland									
1307	730	1800	20140810	59.919	-41.311	20160809	59.664	51.540	F
1308	730	1800	20140810	59.909	-41.220	20160809	49.163	43.293	F
1309	730	1800	20140810	59.909	-41.219	20160809	51.424	45.723	F
1315	730	2000	20140809	59.894	-40.997	20160808	66.885	23.582	BAD, WP
1317	730	2000	20140809	59.894	-40.997	20160808	62.719	34.601	F
1318	730	2200	20140809	59.876	-40.900	20160808	61.677	37.172	F
1327	730	2200	20140809	59.875	-40.871	20160808	49.960	43.659	F
1328	730	2200	20140809	59.875	-40.876	20160808	43.991	41.116	F
1331	730	2200	20140809	59.884	-40.833	20160808	62.883	37.751	F
1332	730	2200	20140809	59.881	-40.872	-	-	-	DNS
1335	730	2500	20140809	59.722	-39.998	20160809	49.716	45.785	F
1336	730	2500	20140809	59.789	-39.998	20160808	58.425	42.578	F
1339	730	2500	20140809	59.782	-39.872	-	-	-	DNS
1340	730	2500	20140809	59.783	-39.872	20160808	60.848	33.098	F
1345	180	2800	20140808	59.694	-39.020	20140823	58.756	41.576	ER
1346	730	2800	20140808	59.664	-38.764	20160807	59.637	39.724	F
1347	730	2800	20140807	59.640	-38.499	20150111	58.903	40.872	ER
1349	730	2800	20140814	59.665	-38.766	20150328	59.454	39.905	ER
Charlie Gibbs Fracture Zone									
1319	180	2200	20140628	52.878	-38.428	20141225	53.352	36.042	S
1320	730	2200	20140628	52.878	-35.428	-	-	-	DNS
1321	730	2200	20140628	52.878	-35.428	20160628	50.995	40.280	F
1322	730	2200	20140628	52.878	-35.429	20160628	55.788	44.729	F
1333	730	2500	20140628	52.857	-35.413	20160628	51.029	31.522	F
1334	730	2500	20140628	52.856	-35.413	20160628	52.339	39.595	F
1338	730	2500	20140628	52.856	-35.413	20160629	55.519	48.182	F
1341	730	2800	20140628	52.782	-35.347	20160628	52.174	41.191	F
1342	730	2800	20140614	52.782	-35.346	20141127	54.088	-0.522	ER
1344	730	2800	20140628	52.782	-35.347	20160628	51.300	34.183	F

Table 11 Floats deployed in 2014. Status Codes: F (full mission science), S (short dual mission), M (monitor), DNS (did not surface), BAD (returned bad data), MD (missing data), ER (early risers), WP (wrong pressure).

Float	Mission Length (days)	Ballast Pressure (dbar)	Launch			Surface			Status Code
			YYYYMMDD	Latitude (N)	Longitude (E)	YYYYMMDD	Longitude (N)	Latitude (E)	
Monitor Floats									
1045	8	1000	20150710	59.015	-32.771	-	-	-	DNS
1082	85	1000	20151112	50.170	-48.470	20160205	46.381	-43.492	M
1084	10	1000	20150627	57.971	-26.993	-	-	-	DNS
1108	100	1000	20150627	57.962	-27.009	20151005	57.062	-33.852	M
1117	317	1000	20150722	59.757	-39.606	20160517	62.086	-60.455	ER
1131	80	1000	20150715	59.147	-34.236	20151003	58.186	-35.124	M
1135	317	1000	20150627	57.971	-26.993	-	-	-	DNS
East Reykjanes Ridge									
1351	730	1800	20150629	58.571	-29.801	20170628	49.124	-36.931	F
1354	730	1800	20150629	58.623	-29.945	20170628	61.406	-36.586	F
1355	730	1800	20150629	58.620	-29.981	20170628	53.772	-39.973	F
1356	730	2000	20150628	58.344	-29.117	20170627	58.288	-29.279	BAD, WP
1357	730	2000	20150628	58.490	-29.534	20170627	54.300	-39.068	F
1358	730	1800	20150628	58.490	-29.534	20170627	56.273	-41.532	F
1359	730	1800	20150628	58.529	-29.588	20170627	60.208	-35.942	F
1360	730	2000	20150628	58.528	-29.587	20170627	55.473	-27.181	F
1365	730	2000	20150628	58.082	-28.404	20170627	49.793	-30.199	F
1366	730	2000	20150628	57.991	-28.049	20170627	55.999	-30.106	F
1369	730	2200	20150628	58.143	-28.545	20170627	52.306	-31.524	F
1370	730	2200	20150628	57.991	-28.047	-	-	-	DNS
1373	730	2200	20150628	58.112	-28.474	20170627	53.536	-33.047	F
1378	730	2200	20150628	58.082	-28.403	20170627	47.672	-28.838	F
1379	730	2200	20150627	57.938	-27.885	20170626	53.895	-47.382	F
West Reykjanes Ridge									
1352	730	1800	20150713	59.023	-32.978	20170712	59.945	-57.024	F
1353	730	1800	20150712	59.075	-33.494	20170711	58.585	-54.131	F
1361	730	2000	20150712	59.075	-33.494	20170711	60.313	-30.909	F
1362	730	1800	20150712	59.127	-34.020	20170711	60.788	-59.966	F
1363	730	1800	20150712	59.183	-34.563	20170711	62.983	-52.812	F
1364	730	2000	20150713	59.023	-32.978	20170712	60.928	-50.229	F
1367	730	2200	20150712	59.128	-34.020	20170711	63.134	-29.640	F, WP
1368	730	2000	20150712	59.183	-34.563	20170711	59.454	-41.596	F
1371	730	2200	20150712	59.128	-34.020	20170711	60.752	-58.075	F
1372	730	2200	20150712	59.183	-34.563	20170711	56.001	-53.264	F
1380	730	2200	20150711	59.243	-35.113	-	-	-	DNS
1385	730	2500	20150711	59.243	-35.113	20170710	56.037	-49.639	F

Table 12 Floats deployed in 2015. Status Codes: F (full mission science), S (short dual mission), M (monitor), DNS (did not surface), BAD (returned bad data), MD (missing data), ER (early risers), WP (wrong pressure). Continues onto next page.

East Greenland									
1337	730	1800	20150723	59.909	-41.220	20170722	58.445	-40.491	F
1350	730	1800	20150723	59.920	-41.311	-	-	-	DNS
1375	730	2200	20150722	59.884	-40.834	20170721	57.512	-54.830	F
1376	730	2000	20150722	59.893	-40.997	20170721	50.155	-47.970	F
1377	730	2000	20150722	59.893	-40.997	-	-	-	DNS
1381	730	2200	20150722	59.881	-40.872	20170721	44.829	-35.588	F
1382	730	2200	20150722	59.881	-40.872	20170721	53.215	-46.473	F
1383	730	2200	20150722	59.876	-40.901	20170721	55.964	-55.508	F
1384	730	2200	20150722	59.884	-40.834	20170721	49.208	-48.743	F
1386	250	2500	20150721	59.790	-39.999	20160327	54.711	-51.949	S
1387	730	2500	20150721	59.783	-39.874	20170720	61.043	-49.909	F
1388	730	2500	20150721	59.783	-39.873	20170720	48.525	-34.734	F
1389	730	2800	20150717	59.640	-38.499	20170716	56.702	-49.286	F
1390	730	2500	20150721	59.790	-40.000	20170720	52.676	-44.634	F
1391	730	2500	20150721	59.790	-40.000	20170720	56.017	-48.279	F
1392	730	2800	20150717	59.640	-38.500	-	-	-	DNS
1393	730	2800	20150717	59.665	-38.766	20170716	62.309	-52.579	F
1394	730	2800	20150717	59.665	-38.766	20170716	58.522	-49.423	F
1395	270	2800	20150717	59.692	-39.017	20160323	61.306	-50.534	ER

Table 13 Continued Floats deployed in 2015. Status Codes: F (full mission science), S (short dual mission), M (monitor), DNS (did not surface), BAD (returned bad data), MD (missing data), ER (early risers), WP (wrong pressure).

Float	Mission Length (days)	Ballast Pressure (dbar)	Launch			Surface			Status Code
			YYYYMMDD	Latitude (N)	Longitude (E)	YYYYMMDD	Latitude (N)	Longitude (E)	
East Reykjanes Ridge									
1396	730	1800	20160714	58.571	29.801	20180714	61.023	-36.381	F
1397	730	1800	20160714	58.627	29.935	20180714	55.466	-32.461	F
1398	730	2000	20160712	57.993	28.040	20180713	54.414	-30.097	F
1399	730	2000	20160713	58.144	28.545	20180713	59.623	-25.360	F
1400	730	1800	20160714	58.622	29.975	20180714	52.589	-40.569	F, WP
1401	730	2000	20160713	58.343	29.115	20180713	52.881	-33.146	F, WP
1402	730	2000	20160714	58.492	29.539	20180714	47.590	-30.590	F
1403	730	2000	20160714	58.529	29.592	20180714	51.495	-32.427	F
1406	730	2200	20160711	58.019	27.920	20180711	52.651	-29.767	F
1410	730	2200	20160712	58.010	26.966	20180712	53.384	-29.364	F
1411	730	2200	20160712	58.011	27.386	20180712	57.134	-28.220	F, WP
1412	730	2200	20160713	58.115	28.477	20180713	53.899	-29.398	F
1413	730	2200	20160713	58.144	28.545	20180713	51.267	-37.639	F
1436	730	2500	20160712	58.010	26.963	20180712	49.769	-24.030	F
1440	730	1800	20160714	58.492	29.539	20180714	50.384	-23.620	F
1441	730	1800	20160714	58.529	29.593	20180715	60.711	-35.154	F
West Reykjanes Ridge									
1404	730	2000	20160809	59.070	-33.491	20180107	59.384	-33.843	ER, WP
1405	730	2000	20160808	59.022	-39.974	20180808	61.790	-31.862	F
1407	730	1800	20160808	59.022	-39.974	20180808	62.441	-38.359	F
1408	730	1800	20160809	59.070	-33.492	20180809	61.777	-32.490	F
1409	730	1800	20160807	59.124	-34.020	20180807	51.102	-48.855	F
1414	730	2200	20160807	59.178	-34.565	20180807	59.309	-41.356	F
1415	730	2200	20160807	59.124	-34.020	20180807	63.130	-30.256	F
1416	730	1800	20160807	59.178	-34.565	20180807	62.028	-54.504	F
1417	730	2000	20160807	59.214	-34.020	20180807	59.095	-57.346	F
1418	730	2000	20160807	59.178	-34.565	20180807	56.895	-56.190	F
1419	730	2200	20160807	59.216	-35.114	20180806	57.402	-45.996	F
East Greenland									
1348	730	1800	20160801	59.877	-41.217	20180801	43.779	-46.864	F
1420	730	1800	20160801	59.877	-41.217	20180801	46.348	-43.929	F
1421	730	2000	20160801	59.843	-40.997	20180803	58.482	-35.536	F
1422	730	2500	20160801	59.780	-39.873	-	-	-	DNS
1423	730	2500	20160801	59.780	-39.873	20180801	51.911	-48.560	F
1424	730	2000	20160801	59.843	40.997	20180801	52.637	-37.554	F
1425	730	2200	20160801	59.825	-40.813	20180801	58.371	-53.151	F
1426	730	2200	20160801	59.825	-40.813	20180801	64.037	-32.895	F
1427	730	2200	20160801	59.823	-40.810	20180801	49.020	-44.038	F
1428	730	1800	20160801	59.885	-41.258	20180806	54.060	-46.261	MD, F
1429	730	2200	20160801	59.830	-40.872	20180801	50.542	-45.967	F
1432	730	2200	20160801	59.823	-40.810	20180801	55.454	-53.221	F
1433	730	2500	20160801	59.768	-39.958	20180801	51.155	-41.987	F
1437	730	2500	20160801	59.768	-39.958	20180801	52.242	-42.272	F

Table 14 Floats deployed in 2016. Status Codes: F (full mission science), S (short dual mission), M (monitor), DNS (did not surface), BAD (returned bad data), MD (missing data), ER (early risers), WP (wrong pressure).

Float	Mission Length (days)	Ballast Pressure (dbar)	Launch			Surface			Status Code
			YYYYMMDD	Latitude (N)	Longitude (E)	YYYYMMDD	Latitude (N)	Longitude (E)	
Southeast Reykjanes Ridge									
1430	540	1800	20170720	56.243	-33.687	20190111	59.747	-35.156	F
1431	540	1800	20170720	56.243	-33.687	20190111	57.923	-38.356	F
1434	540	1800	20170721	55.998	-33.152	20190112	53.647	-37.680	F
1435	540	2000	20170721	55.998	-33.152	20190112	52.942	-36.175	F
1438	540	2000	20170721	55.998	-33.152	20190112	53.821	-30.578	F
1439	540	2000	20170721	55.772	-32.666	20190112	52.928	-33.815	F
1442	540	2200	20170721	55.619	-32.340	20190112	52.981	-16.258	F
1443	540	2200	20170721	55.619	-32.340	20190112	51.525	-27.437	F
1444	540	2200	20170721	55.772	-32.666	20190112	53.764	-23.976	F
1445	540	2200	20170721	55.772	-32.666	20190112	50.023	-21.387	F

Table 15 Floats deployed in 2017. Status Codes: F (full mission science), S (short dual mission), M (monitor), DNS (did not surface), BAD (returned bad data), MD (missing data), ER (early risers), WP (wrong pressure).

Float	Mission Length (days)		Ballast pressure (dbar)			Float Clock Offset (sec)		Wire Designation	Status Code
	Target	Actual	Target	Initial	Delta	Initial	Final		
East Reykjanes Ridge									
1310	730	730	1800	1841	41	0	-44	-	F
1311	730	730	1800	1834	34	0	-27	-	F
1312	730	730	1800	1816	16	0	-34	-	F
1313	730	730	2000	-	-	-	-	-	DNS
1314	730	730	2000	2045	45	0	-38	-	F
1316	730	730	2000	2048	48	0	-21	-	F
1325	730	730	2200	2260	60	0	-57	-	F
1326	730	730	2200	2257	57	0	-51	-	F
1329	730	180	2200	2255	55	0	-47	-	F
1330	180	730	2200	2258	58	0	-4	-	S
East Greenland									
1307	730	730	1800	1812	12	0	-37	-	F
1308	730	730	1800	1826	26	0	-5	-	F
1309	730	730	1800	1816	16	0	-50	-	F
1315	730	730	2000	4	-1996	0	-48	-	BAD, WP
1317	730	730	2000	2027	27	0	2	-	F
1318	730	730	2200	2221	21	0	-100	-	F
1327	730	730	2200	2227	27	0	-84	-	F
1328	730	730	2200	2231	31	0	-57	-	F
1331	730	730	2200	2235	35	0	-56	-	F
1332	730	0	2200	-	-	-	-	-	DNS
1335	730	731	2500	2554	54	0	-37	-	F
1336	730	730	2500	2559	59	0	-40	-	F
1339	730	0	2500	-	-	-	-	-	DNS
1340	730	730	2500	2563	63	0	-19	-	F
1345	180	15	2800	2922	122	0	0	-	ER
1346	730	730	2800	2940	140	0	-10	-	F
1347	730	157	2800	2954	154	0	25	-	ER
1349	730	226	2800	2932	132	0	-6	-	ER
Charlie Gibbs Fracture Zone									
1319	180	180	2200	2194	-6	0	-5	-	S
1320	730	0	2200	-	-	-	-	-	DNS
1321	730	731	2200	2271	71	0	-39	-	F
1322	730	731	2200	2263	63	0	-46	-	F
1333	730	731	2500	2605	105	0	-35	-	F
1334	730	731	2500	2601	101	0	-43	-	F
1338	730	732	2500	2601	101	0	-51	-	F
1341	730	731	2800	2929	129	0	-26	-	F
1342	730	166	2800	2923	123	0	-6	-	ER
1344	730	731	2800	2919	119	0	-38	-	F

Table 16 Performance of 2014 floats. If float deployed after 22:00, the first measured pressure was taken just after midnight, and the float was still in decent. In these cases, the second pressure measurement was used, after the float had reached equilibrium. Status Codes: F (full mission science), S (short dual mission), M (monitor), DNS (did not surface), BAD (returned bad data), MD (missing data), ER (early risers), WP (wrong pressure).

Float	Mission Length (days)		Ballast Pressure (dbar)			Float Clock Offset (sec)		Wire Designation	Status Code
	Target	Actual	Target	Initial	Delta	Initial	Final		
East Reykjanes Ridge									
1351	730	730	1800	1810	10	0	-72	Y	F
1354	730	730	1800	1748	-52	0	-56	Y	F
1355	730	730	1800	1810	10	0	-68	Y	F
1356	730	730	2000	2301	301	0	-28	Y	BAD, WP
1357	730	730	2000	2004	4	0	5	Y	F
1358	730	730	1800	1758	-42	0	-3	-	F
1359	730	730	1800	1817	17	0	18	-	F
1360	730	730	2000	1985	-15	0	-8	Y	F
1365	730	730	2000	2010	10	0	-26	-	F
1366	730	730	2000	2003	3	0	-42	-	F
1369	730	730	2200	2182	-18	0	-46	Y	F
1370	730	0	2200	-	-	-	-	Y	DNS
1373	730	730	2200	2191	-9	0	-9	Y	F
1378	730	730	2200	2192	-8	0	-39	-	F
1379	730	730	2200	2185	-15	0	-36	-	F
West Reykjanes Ridge									
1352	730	730	1800	1833	33	0	-39	-	F
1353	730	730	1800	1810	10	0	-50	-	F
1361	730	730	2000	2005	5	0	-3	Y	F
1362	730	730	1800	1819	19	0	-26	-	F
1363	730	730	1800	1817	17	0	0	-	F
1364	730	730	2000	2060	60	0	-38	Y	F
1367	730	730	2200	2009	-191	0	-12	-	F, WP
1368	730	730	2000	2012	12	0	-31	-	F
1371	730	730	2200	2207	7	0	-13	Y	F
1372	730	730	2200	2202	2	0	-28	Y	F
1380	730	0	2200	-	-	-	-	-	DNS
1385	730	730	2500	2537	37	0	-3	-	F
East Greenland									
1337	730	730	1800	1803	3	0	-47	-	F
1350	730	0	1800	-	-	-	-	Y	DNS
1375	730	730	2200	2154	-46	0	-27	Y	F
1376	730	730	2000	1968	-33	0	-29	-	F
1377	730	0	2000	-	-	-	-	-	DNS
1381	730	730	2200	2164	-36	0	-29	-	F
1382	730	730	2200	2209	9	0	-14	-	F
1383	730	730	2200	2157	-43	0	-26	Y	F
1384	730	730	2200	2169	-31	0	-42	-	F
1386	250	250	2500	2466	-34	0	-17	-	S
1387	730	730	2500	2477	-23	-21	-40	-	F
1388	730	730	2500	2490	-10	0	24	-	F
1389	730	730	2800	2804	4	0	-2	-	F
1390	730	730	2500	2448	-52	0	-11	Y	F
1391	730	730	2500	2467	-34	0	-41	Y	F
1392	730	0	2800	-	-	-	-	-	DNS
1393	730	730	2800	2801	1	0	-48	-	F
1394	730	730	2800	2806	6	0	-25	Y	F
1395	270	250	2800	2797	-3	0	-11	Y	ER

Table 17 Performance of 2015 floats. If float deployed after 22:00, the first measured pressure was taken just after midnight, and the float was still in decent. In these cases, the second pressure measurement was used, after the float had reached equilibrium. Status Codes: F (full mission science), S (short dual mission), M (monitor), DNS (did not surface), BAD (returned bad data), MD (missing data), ER (early risers), WP (wrong pressure).

Float	Mission Length (Days)		Ballast Pressure (dbar)			Float Clock Offset (sec)		Wire Designation	Status Code
	Target	Actual	Target	Initial	Delta	Initial	Final		
East Reykjanes Ridge									
1396	730	730	1800	1824	24	0	-48	Y	F
1397	730	730	1800	1801	1	0	-29	Y	F
1398	730	731	2000	2071	71	0	-12	-	F
1399	730	730	2000	2088	88	0	-25	-	F
1400	730	730	1800	1170	-630	0	-25	Y	F, WP
1401	730	730	2000	1593	-407	0	-62	Y	F, WP
1402	730	730	2000	1999	-1	0	-58	Y	F
1403	730	730	2000	2006	6	0	-12	Y	F
1406	730	730	2200	2336	136	0	-46	-	F
1410	730	730	2200	2155	-45	0	-20	Y	F
1411	730	730	2200	1869	-331	0	-51	Y	F, WP
1412	730	730	2200	2142	-58	0	167	Y	F
1413	730	730	2200	2170	-30	0	-43	Y	F
1436	730	730	2500	2497	-4	0	-56	Y	F
1440	730	730	1800	1867	67	0	-9	-	F
1441	730	731	1800	1868	68	0	-27	-	F
West Reykjanes Ridge									
1404	730	516	2000	2329	329	0	-30	Y	ER, WP
1405	730	730	2000	2044	44	0	-51	Y	F
1407	730	730	1800	1910	110	3	-8	-	F
1408	730	730	1800	1872	72	0	-35	-	F
1409	730	730	1800	1831	31	0	-20	-	F
1414	730	730	2200	2168	-32	0	-54	Y	F
1415	730	730	2200	2185	-15	0	-8	Y	F
1416	730	730	1800	1841	41	0	-56	-	F
1417	730	730	2000	2027	27	0	-56	-	F
1418	730	730	2000	2040	40	0	-54	-	F
1419	730	729	2200	2234	34	0	-48	-	F
East Greenland									
1348	730	730	1800	1820	20	0	-31	-	F
1420	730	730	1800	1836	36	0	-55	-	F
1421	730	732	2000	2010	10	0	-39	-	F
1422	730	0	2500	-	-	-	-	-	DNS
1423	730	730	2500	2475	-25	0	-47	-	F
1424	730	730	2000	1977	-23	0	-38	-	F
1425	730	730	2200	2171	-29	0	-5	-	F
1426	730	730	2200	2189	-11	0	-21	-	F
1427	730	730	2200	2190	-10	0	-30	-	F
1428	730	735	1800	null	null	0	-35	Y	F
1429	730	730	2200	2201	1	0	-43	Y	F
1432	730	730	2200	2235	35	0	-33	-	F
1433	730	730	2500	2450	-50	0	-98	Y	F
1437	730	730	2500	2496	-4	0	62	Y	F

Table 18 Performance of 2016 floats. If float deployed after 22:00, the first measured pressure was taken just after midnight, and the float was still in decent. In these cases, the second pressure measurement was used, after the float had reached equilibrium. Status Codes: F (full mission science), S (short dual mission), M (monitor), DNS (did not surface), BAD (returned bad data), MD (missing data), ER (early risers), WP (wrong pressure).

Float	Mission Length (days)		Ballast Pressure (dbar)			Float Clock Offset (sec)		Wire Designation	Status Code
	Target	Actual	Target	Initial	Delta	Initial	Final		
Southeast Reykjanes Ridge									
1430	540	540	1800	1762	-38	0	-25	Y	F
1431	540	540	1800	1776	-25	0	-28	Y	F
1434	540	540	1800	1729	-71	0	-8	Y	F
1435	540	540	2000	1894	-106	0	-28	Y	F
1438	540	540	2000	1938	-62	0	-34	-	F
1439	540	540	2000	1931	-70	0	-49	-	F
1442	540	540	2200	2105	-95	0	-39	-	F
1443	540	540	2200	2049	-151	0	-61	-	F
1444	540	540	2200	2176	-24	0	-69	Y	F
1445	540	540	2200	2119	-81	0	-26	Y	F

Table 19 Performance of 2017 floats. If float deployed after 22:00, the first measured pressure was taken just after midnight, and the float was still in decent. In these cases, the second pressure measurement was used, after the float had reached equilibrium. Status Codes: F (full mission science), S (short dual mission), M (monitor), DNS (did not surface), BAD (returned bad data), MD (missing data), ER (early risers), WP (wrong pressure).

APPENDIX B: SOUND SOURCE DRIFTS

This appendix contains plots of inferred (i.e., float-observed) drifts for a set of sound source clocks used in tracking the OSNAP RAFOS float data. Drift estimates were made for sound sources where measured final drift was not possible, either because mooring was not recovered (sound sources #8, 9 10, 12 13) or final clock measurement was not reliable (sound sources #2, 3). The difference between observed minus expected time-of-arrival (TOA) data points pulled from float observations are plotted as blue circles. A first order (linear) fit to the difference data points is the inferred drift (blue line). Float serial numbers mark each data point source, at the right of each asterisk.

Figure 1 is a plot of delta TOA (seconds) versus time (year, month). The y-axis ranges from -100 to 100 seconds, and the x-axis shows time from June 2014 to June 2016. A vertical dashed line indicates the start of the SS (Superconducting System) at approximately September 2014. The plot shows a linear trend of decreasing delta TOA over time, starting from 0 seconds in late 2014 and reaching approximately -50 seconds by June 2016. The inferred drift is -0.0455 sec/day, and the inferred drift intercept is 0.0048 sec. Data points are labeled with TOA numbers: 1307, 1352, 1387, 1388, 1389, 1393, and 1395.

47

Figure 14 Inferred drift for OSNAP sound source #3

Figure 15 Inferred drift for OSNAP sound source #8

Sound Source OS9: Inferred Drift

Figure 16 Inferred drift for OSNAP sound source #9

Figure 17 Inferred drift for OSNAP sound source #10

Figure 18 Inferred drift for OSNAP sound source #12

Sound Source OS13: Inferred Drift

Figure 19 Inferred drift for OSNAP sound source #13

APPENDIX C: DISPLACEMENT AND SPAGHETTI PLOTS

Appendix C includes displacement and spaghetti plots for each seeding of floats. The bathymetry is drawn every 1000-m for both plots. A circle indicates launch location; triangle indicates surface location. Displacement figure: The float displacement is connected by a black line. Spaghetti figure: Each year is drawn in a different color: Cyan (2014), Purple (2015), Magenta (2016), Red (2017). Dashed lines connect track segments and indicate where tracking was not possible due to lack of sound source data.

Figure 20 Float displacements for all deployments.

Figure 21 Float trajectories by seeding.

Figure 22 Float trajectories color coded by in situ temperature. Float trajectories are nominally two years in length, and temperature is recorded daily. Dashed lines connect trajectory segments within the same float mission, where tracking was lost due to float being out of range of sound source signals. Floats have been divided into four groups based on launch location region. Multiple releases occurred in each grouping except for 'Southeastern RR and CGFZ Launch' group. 'RR' stands for Reykjanes Ridge, 'GR' stands for Greenland, and 'CGFZ' stands for Charlie-Gibbs Fracture Zone. Black dots mark launch locations. Bathymetry is contoured at 1000-, 2000-, and 3000-meters depth.

Figure 23 Float trajectories color coded by speed. Float trajectories are nominally two years in length, and position is recorded daily. Dashed lines connect trajectory segments within the same float mission, where tracking was lost due to float being out of range of sound source signals. Floats have been divided into four groups based on launch location region. Multiple releases occurred in each grouping except for 'Southeastern RR and CGFZ Launch' group. 'RR' stands for Reykjanes Ridge, 'GR' stands for Greenland, and 'CGFZ' stands for Charlie-Gibbs Fracture Zone. Black dots mark launch locations. Bathymetry is contoured at 1000-, 2000-, and 3000-meters depth.

Figure 24 Float trajectories color coded by elapsed day. Float trajectories are nominally two years in length, and position is recorded daily. Dashed lines connect trajectory segments within the same float mission, where tracking was lost due to float being out of range of sound source signals. Floats have been divided into four groups based on launch location region. Multiple releases occurred in each grouping except for 'Southeastern RR and CGFZ Launch' group. 'RR' stands for Reykjanes Ridge, 'GR' stands for Greenland, and 'CGFZ' stands for Charlie-Gibbs Fracture Zone. Black dots mark launch locations. Bathymetry is contoured at 1000-, 2000-, and 3000-meters depth.

APPENDIX D: POSTAGE STAMP FIGURES

Appendix D includes individual trajectory plots for each float, organized by serial number. Each year is drawn in a different color: Cyan (2014), Purple (2015), Magenta (2016), Red (2017). The deployment location is marked as a green circle; surface location is a red triangle. Bathymetry is drawn every 1000-m. Dashed line connects track segments where tracking was not possible due to lack of sound source data.

APPENDIX E: PROPERTY AND TRAJECTORY FIGURES

Appendix E contains individual trajectory and property plots for each float, organized by float serial number. On the trajectory plots, the launch position is marked with a green circle; the surface location is marked with a red triangle. Bathymetry is drawn every 1000-m. Dashed line connects track segments and indicates where tracking was not possible due to lack of sound source data. The companion plot contains in-situ temperature, pressure, u-velocity, v-velocity (float velocity is computed as a centered difference between positions 48 hours apart), and vector velocity for the floats underwater mission length. Blank space indicates that the float was programmed to do a longer mission than the data were recorded.

Float 1307 (1800-dbar) Aug 2014 to Aug 2016

Float 1307 (1800-dbar) Aug 2014 to Aug 2016

Float 1308 (1800-dbar) Aug 2014 to Aug 2016

Float 1308 (1800-dbar) Aug 2014 to Aug 2016

Float 1309 (1800-dbar) Aug 2014 to Aug 2016

Float 1309 (1800-dbar) Aug 2014 to Aug 2016

Float 1310 (1800-dbar) Jul 2014 to Jul 2016

Float 1310 (1800-dbar) Jul 2014 to Jul 2016

Float 1311 (1800-dbar) Jul 2014 to Jul 2016

Float 1311 (1800-dbar) Jul 2014 to Jul 2016

Float 1312 (1800-dbar) Jul 2014 to Jul 2016

Float 1312 (1800-dbar) Jul 2014 to Jul 2016

Float 1314 (2000-dbar) Jul 2014 to Jul 2016

Float 1314 (2000-dbar) Jul 2014 to Jul 2016

Float 1316 (2000-dbar) Jul 2014 to Jul 2016

Float 1316 (2000-dbar) Jul 2014 to Jul 2016

Float 1317 (2000-dbar) Aug 2014 to Aug 2016

Float 1317 (2000-dbar) Aug 2014 to Aug 2016

Float 1318 (2200-dbar) Aug 2014 to Aug 2016

Float 1318 (2200-dbar) Aug 2014 to Aug 2016

Float 1319 (2200-dbar) Jun 2014 to Dec 2014

Float 1319 (2200-dbar) Jun 2014 to Dec 2014

Float 1321 (2200-dbar) Jun 2014 to Jun 2016

Float 1321 (2200-dbar) Jun 2014 to Jun 2016

Float 1322 (2200-dbar) Jun 2014 to Jun 2016

Float 1322 (2200-dbar) Jun 2014 to Jun 2016

Float 1325 (2200-dbar) Jul 2014 to Jul 2016

Float 1325 (2200-dbar) Jul 2014 to Jul 2016

Float 1326 (2200-dbar) Jul 2014 to Jul 2016

Float 1326 (2200-dbar) Jul 2014 to Jul 2016

Float 1327 (2200-dbar) Aug 2014 to Aug 2016

Float 1327 (2200-dbar) Aug 2014 to Aug 2016

Float 1328 (2200-dbar) Aug 2014 to Aug 2016

Float 1328 (2200-dbar) Aug 2014 to Aug 2016

Float 1329 (2200-dbar) Jul 2014 to Jul 2016

Float 1329 (2200-dbar) Jul 2014 to Jul 2016

Float 1330 (2200-dbar) Jul 2014 to Jan 2015

Float 1330 (2200-dbar) Jul 2014 to Jan 2015

Float 1331 (2200-dbar) Aug 2014 to Aug 2016

Float 1331 (2200-dbar) Aug 2014 to Aug 2016

Float 1333 (2500-dbar) Jun 2014 to Jun 2016

Float 1333 (2500-dbar) Jun 2014 to Jun 2016

Float 1334 (2500-dbar) Jun 2014 to Jun 2016

Float 1334 (2500-dbar) Jun 2014 to Jun 2016

Float 1335 (2500-dbar) Aug 2014 to Aug 2016

Float 1335 (2500-dbar) Aug 2014 to Aug 2016

Float 1336 (2500-dbar) Aug 2014 to Aug 2016

Float 1336 (2500-dbar) Aug 2014 to Aug 2016

Float 1337 (1800-dbar) Jul 2015 to Jul 2017

Float 1337 (1800-dbar) Jul 2015 to Jul 2017

Float 1338 (2500-dbar) Jun 2014 to Jun 2016

Float 1338 (2500-dbar) Jun 2014 to Jun 2016

Float 1340 (2500-dbar) Aug 2014 to Aug 2016

Float 1340 (2500-dbar) Aug 2014 to Aug 2016

Float 1341 (2800-dbar) Jun 2014 to Jun 2016

Float 1341 (2800-dbar) Jun 2014 to Jun 2016

Float 1342 (2800-dbar) Jun 2014 to Nov 2014

Float 1342 (2800-dbar) Jun 2014 to Nov 2014

Float 1344 (2800-dbar) Jun 2014 to Jun 2016

Float 1344 (2800-dbar) Jun 2014 to Jun 2016

Float 1345-2800dbar: Aug2014-Aug2014

Float 1345 (2800-dbar) Aug 2014 to Aug 2014

Float 1346 (2800-dbar) Aug 2014 to Aug 2016

Float 1346 (2800-dbar) Aug 2014 to Aug 2016

Float 1347 (2800-dbar) Aug 2014 to Jan 2015

Aug-2014

Oct-2014

Dec-2014

Feb-2015

Float 1347 (2800-dbar) Aug 2014 to Jan 2015

Float 1348 (1800-dbar) Aug 2016 to Aug 2018

Float 1348 (1800-dbar) Aug 2016 to Aug 2018

Float 1349 (2800-dbar) Aug 2014 to Mar 2015

Float 1349 (2800-dbar) Aug 2014 to Mar 2015

Float 1351 (1800-dbar) Jun 2015 to Jun 2017

Float 1351 (1800-dbar) Jun 2015 to Jun 2017

Float 1352 (1800-dbar) Jul 2015 to Jul 2017

Float 1353 (1800-dbar) Jul 2015 to Jul 2017

Float 1353 (1800-dbar) Jul 2015 to Jul 2017

Float 1354 (1800-dbar) Jun 2015 to Jun 2017

Float 1354 (1800-dbar) Jun 2015 to Jun 2017

Float 1355 (1800-dbar) Jun 2015 to Jun 2017

Float 1355 (1800-dbar) Jun 2015 to Jun 2017

Float 1357 (2000-dbar) Jun 2015 to Jun 2017

Float 1357 (2000-dbar) Jun 2015 to Jun 2017

Float 1358 (1800-dbar) Jun 2015 to Jun 2017

Float 1358 (1800-dbar) Jun 2015 to Jun 2017

Float 1359 (1800-dbar) Jun 2015 to Jun 2017

Float 1359 (1800-dbar) Jun 2015 to Jun 2017

Float 1360 (2000-dbar) Jun 2015 to Jun 2017

Float 1360 (2000-dbar) Jun 2015 to Jun 2017

Float 1361 (2000-dbar) Jul 2015 to Jul 2017

Float 1361 (2000-dbar) Jul 2015 to Jul 2017

Float 1362 (1800-dbar) Jul 2015 to Jul 2017

Float 1362 (1800-dbar) Jul 2015 to Jul 2017

Float 1363 (1800-dbar) Jul 2015 to Jul 2017

Float 1363 (1800-dbar) Jul 2015 to Jul 2017

Float 1364 (2000-dbar) Jul 2015 to Jul 2017

Float 1364 (2000-dbar) Jul 2015 to Jul 2017

Float 1365 (2000-dbar) Jun 2015 to Jun 2017

Float 1365 (2000-dbar) Jun 2015 to Jun 2017

Float 1366 (2000-dbar) Jun 2015 to Jun 2017

Float 1366 (2000-dbar) Jun 2015 to Jun 2017

Float 1367 (2200-dbar) Jul 2015 to Jul 2017

Float 1367 (2200-dbar) Jul 2015 to Jul 2017

Float 1368 (2000-dbar) Jul 2015 to Jul 2017

Float 1368 (2000-dbar) Jul 2015 to Jul 2017

Float 1369 (2200-dbar) Jun 2015 to Jun 2017

Float 1369 (2200-dbar) Jun 2015 to Jun 2017

Float 1371 (2200-dbar) Jul 2015 to Jul 2017

Float 1372 (2200-dbar) Jul 2015 to Jul 2017

Float 1372 (2200-dbar) Jul 2015 to Jul 2017

Float 1373 (2200-dbar) Jun 2015 to Jun 2017

Float 1373 (2200-dbar) Jun 2015 to Jun 2017

Float 1375 (2200-dbar) Jul 2015 to Jul 2017

Float 1375 (2200-dbar) Jul 2015 to Jul 2017

Float 1376 (2000-dbar) Jul 2015 to Jul 2017

Float 1376 (2000-dbar) Jul 2015 to Jul 2017

Float 1378 (2200-dbar) Jun 2015 to Jun 2017

Float 1378 (2200-dbar) Jun 2015 to Jun 2017

Float 1379 (2200-dbar) Jun 2015 to Jun 2017

Float 1379 (2200-dbar) Jun 2015 to Jun 2017

Float 1381 (2200-dbar) Jul 2015 to Jul 2017

Float 1381 (2200-dbar) Jul 2015 to Jul 2017

Float 1382 (2200-dbar) Jul 2015 to Jul 2017

Float 1382 (2200-dbar) Jul 2015 to Jul 2017

Float 1383 (2200-dbar) Jul 2015 to Jul 2017

Float 1383 (2200-dbar) Jul 2015 to Jul 2017

Float 1384 (2200-dbar) Jul 2015 to Jul 2017

Float 1384 (2200-dbar) Jul 2015 to Jul 2017

Float 1385 (2500-dbar) Jul 2015 to Jul 2017

Float 1385 (2500-dbar) Jul 2015 to Jul 2017

Float 1386 (2500-dbar) Jul 2015 to Mar 2016

Float 1386 (2500-dbar) Jul 2015 to Mar 2016

Float 1387 (2500-dbar) Jul 2015 to Jul 2017

Float 1387 (2500-dbar) Jul 2015 to Jul 2017

Float 1388 (2500-dbar) Jul 2015 to Jul 2017

Float 1388 (2500-dbar) Jul 2015 to Jul 2017

Float 1389 (2800-dbar) Jul 2015 to Jul 2017

Float 1389 (2800-dbar) Jul 2015 to Jul 2017

Float 1390 (2500-dbar) Jul 2015 to Jul 2017

Float 1390 (2500-dbar) Jul 2015 to Jul 2017

Float 1391 (2500-dbar) Jul 2015 to Jul 2017

Float 1391 (2500-dbar) Jul 2015 to Jul 2017

Float 1393 (2800-dbar) Jul 2015 to Jul 2017

Float 1393 (2800-dbar) Jul 2015 to Jul 2017

Float 1394 (2800-dbar) Jul 2015 to Jul 2017

Float 1394 (2800-dbar) Jul 2015 to Jul 2017

Float 1395 (2800-dbar) Jul 2015 to Mar 2016

Float 1395 (2800-dbar) Jul 2015 to Mar 2016

Float 1396 (1800-dbar) Jul 2016 to Jul 2018

Float 1396 (1800-dbar) Jul 2016 to Jul 2018

Float 1397 (1800-dbar) Jul 2016 to Jul 2018

Float 1397 (1800-dbar) Jul 2016 to Jul 2018

Float 1398 (2000-dbar) Jul 2016 to Jul 2018

Float 1398 (2000-dbar) Jul 2016 to Jul 2018

Float 1399 (2000-dbar) Jul 2016 to Jul 2018

Float 1399 (2000-dbar) Jul 2016 to Jul 2018

Float 1400 (1800-dbar) Jul 2016 to Jul 2018

Float 1400 (1800-dbar) Jul 2016 to Jul 2018

Float 1401 (2000-dbar) Jul 2016 to Jul 2018

Float 1401 (2000-dbar) Jul 2016 to Jul 2018

Float 1402 (2000-dbar) Jul 2016 to Jul 2018

Float 1402 (2000-dbar) Jul 2016 to Jul 2018

Float 1403 (2000-dbar) Jul 2016 to Jul 2018

Float 1403 (2000-dbar) Jul 2016 to Jul 2018

Float 1404 (2000-dbar) Aug 2016 to Jan 2018

Float 1404 (2000-dbar) Aug 2016 to Jan 2018

Float 1405 (2000-dbar) Aug 2016 to Aug 2018

Float 1405 (2000-dbar) Aug 2016 to Aug 2018

Float 1406 (2200-dbar) Jul 2016 to Jul 2018

Float 1406 (2200-dbar) Jul 2016 to Jul 2018

Float 1407 (1800-dbar) Aug 2016 to Aug 2018

Float 1407 (1800-dbar) Aug 2016 to Aug 2018

Float 1408 (1800-dbar) Aug 2016 to Aug 2018

Float 1408 (1800-dbar) Aug 2016 to Aug 2018

Float 1409 (1800-dbar) Aug 2016 to Aug 2018

Float 1409 (1800-dbar) Aug 2016 to Aug 2018

Float 1410 (2200-dbar) Jul 2016 to Jul 2018

Float 1410 (2200-dbar) Jul 2016 to Jul 2018

Float 1411 (2200-dbar) Jul 2016 to Jul 2018

Float 1411 (2200-dbar) Jul 2016 to Jul 2018

Float 1412 (2200-dbar) Jul 2016 to Jul 2018

Float 1412 (2200-dbar) Jul 2016 to Jul 2018

Float 1413 (2200-dbar) Jul 2016 to Jul 2018

Float 1413 (2200-dbar) Jul 2016 to Jul 2018

Float 1414 (2200-dbar) Aug 2016 to Aug 2018

Float 1414 (2200-dbar) Aug 2016 to Aug 2018

Float 1415 (2200-dbar) Aug 2016 to Aug 2018

Float 1415 (2200-dbar) Aug 2016 to Aug 2018

Float 1416 (1800-dbar) Aug 2016 to Aug 2018

Float 1416 (1800-dbar) Aug 2016 to Aug 2018

Float 1417 (2000-dbar) Aug 2016 to Aug 2018

Float 1417 (2000-dbar) Aug 2016 to Aug 2018

Float 1418 (2000-dbar) Aug 2016 to Aug 2018

Float 1418 (2000-dbar) Aug 2016 to Aug 2018

Float 1419 (2200-dbar) Aug 2016 to Aug 2018

Float 1419 (2200-dbar) Aug 2016 to Aug 2018

Float 1420 (1800-dbar) Aug 2016 to Aug 2018

Float 1420 (1800-dbar) Aug 2016 to Aug 2018

Float 1421 (2000-dbar) Aug 2016 to Aug 2018

Float 1421 (2000-dbar) Aug 2016 to Aug 2018

Float 1423 (2500-dbar) Aug 2016 to Aug 2018

Float 1423 (2500-dbar) Aug 2016 to Aug 2018

Float 1424 (2000-dbar) Aug 2016 to Aug 2018

Float 1424 (2000-dbar) Aug 2016 to Aug 2018

Float 1425 (2200-dbar) Aug 2016 to Aug 2018

Float 1426 (2200-dbar) Aug 2016 to Aug 2018

Float 1426 (2200-dbar) Aug 2016 to Aug 2018

Float 1427 (2200-dbar) Aug 2016 to Aug 2018

Float 1427 (2200-dbar) Aug 2016 to Aug 2018

Float 1428 (1800-dbar) Aug 2016 to Aug 2018

Float 1428 (1800-dbar) Aug 2016 to Aug 2018

Float 1429 (2200-dbar) Aug 2016 to Aug 2018

Float 1429 (2200-dbar) Aug 2016 to Aug 2018

Float 1430 (1800-dbar) Jul 2017 to Jan 2019

Float 1430 (1800-dbar) Jul 2017 to Jan 2019

Float 1431 (1800-dbar) Jul 2017 to Jan 2019

Map of the North Atlantic showing the cruise track of the R/V Atlantis from 2017 to 2019. The track is marked with black dots and lines, with specific dates labeled at various points. A red dot indicates the start of the cruise on 10/18, and a green dot indicates the end on 08/17. The map includes latitude and longitude coordinates and a scale bar.

Float 1432 (2200-dbar) Aug 2016 to Aug 2018

Float 1432 (2200-dbar) Aug 2016 to Aug 2018

Float 1433 (2500-dbar) Aug 2016 to Aug 2018

Float 1433 (2500-dbar) Aug 2016 to Aug 2018

Float 1434 (1800-dbar) Jul 2017 to Jan 2019

Float 1434 (1800-dbar) Jul 2017 to Jan 2019

Float 1435 (2000-dbar) Jul 2017 to Jan 2019

Float 1435 (2000-dbar) Jul 2017 to Jan 2019

Float 1436 (2500-dbar) Jul 2016 to Jul 2018

Float 1436 (2500-dbar) Jul 2016 to Jul 2018

Float 1437 (2500-dbar) Aug 2016 to Aug 2018

Float 1437 (2500-dbar) Aug 2016 to Aug 2018

Float 1438 (2000-dbar) Jul 2017 to Jan 2019

Float 1438 (2000-dbar) Jul 2017 to Jan 2019

Float 1439 (2000-dbar) Jul 2017 to Jan 2019

Float 1439 (2000-dbar) Jul 2017 to Jan 2019

Float 1440 (1800-dbar) Jul 2016 to Jul 2018

Float 1440 (1800-dbar) Jul 2016 to Jul 2018

Float 1441 (1800-dbar) Jul 2016 to Jul 2018

Float 1441 (1800-dbar) Jul 2016 to Jul 2018

Float 1442 (2200-dbar) Jul 2017 to Jan 2019

Float 1442 (2200-dbar) Jul 2017 to Jan 2019

Float 1443 (2200-dbar) Jul 2017 to Jan 2019

Float 1443 (2200-dbar) Jul 2017 to Jan 2019

Float 1444 (2200-dbar) Jul 2017 to Jan 2019

Float 1444 (2200-dbar) Jul 2017 to Jan 2019

Float 1445 (2200-dbar) Jul 2017 to Jan 2019

Float 1445 (2200-dbar) Jul 2017 to Jan 2019

APPENDIX F: TIMES-OF-ARRIVALS

This appendix includes times-of-arrival data for each float tracked in this experiment. The data are plotted as single dots for each TOA recorded by the float, and those TOAS with strong correlations are circled. The sound source that is the source of each signal is drawn on the right and left hand side of each plot, and the signal position is derived from the floats surface location. Time is along the x-axis, and signal arrival time is along the y-axis. Pressure is plotted separately at the bottom, and can indicate when a float was grounded.

Float 1307 Times of Arrival (seconds)

Float 1308 Times of Arrival (seconds)

Float 1309 Times of Arrival (seconds)

Float 1310 Times of Arrival (seconds)

Float 1311 Times of Arrival (seconds)

Float 1312 Times of Arrival (seconds)

Float 1314 Times of Arrival (seconds)

Float 1316 Times of Arrival (seconds)

Float 1317 Times of Arrival (seconds)

Float 1318 Times of Arrival (seconds)

Float 1319 Times of Arrival (seconds)

Float 1321 Times of Arrival (seconds)

Float 1322 Times of Arrival (seconds)

Float 1325 Times of Arrival (seconds)

Float 1326 Times of Arrival (seconds)

Float 1327 Times of Arrival (seconds)

Float 1328 Times of Arrival (seconds)

Float 1329 Times of Arrival (seconds)

Float 1330 Times of Arrival (seconds)

Float 1331 Times of Arrival (seconds)

Float 1333 Times of Arrival (seconds)

Float 1334 Times of Arrival (seconds)

Float 1335 Times of Arrival (seconds)

Float 1336 Times of Arrival (seconds)

Float 1337 Times of Arrival (seconds)

Float 1338 Times of Arrival (seconds)

Float 1340 Times of Arrival (seconds)

Float 1341 Times of Arrival (seconds)

Float 1342 Times of Arrival (seconds)

Float 1344 Times of Arrival (seconds)

Float 1345 Times of Arrival (seconds)

Float 1346 Times of Arrival (seconds)

Float 1347 Times of Arrival (seconds)

Float 1348 Times of Arrival (seconds)

Float 1349 Times of Arrival (seconds)

Float 1351 Times of Arrival (seconds)

Float 1352 Times of Arrival (seconds)

Float 1353 Times of Arrival (seconds)

Float 1354 Times of Arrival (seconds)

Float 1355 Times of Arrival (seconds)

Float 1357 Times of Arrival (seconds)

Float 1358 Times of Arrival (seconds)

Float 1359 Times of Arrival (seconds)

Float 1360 Times of Arrival (seconds)

Float 1361 Times of Arrival (seconds)

Float 1362 Times of Arrival (seconds)

Float 1363 Times of Arrival (seconds)

Float 1364 Times of Arrival (seconds)

Float 1365 Times of Arrival (seconds)

Float 1366 Times of Arrival (seconds)

Float 1367 Times of Arrival (seconds)

Float 1368 Times of Arrival (seconds)

Float 1369 Times of Arrival (seconds)

Float 1371 Times of Arrival (seconds)

Float 1372 Times of Arrival (seconds)

Float 1373 Times of Arrival (seconds)

Float 1375 Times of Arrival (seconds)

Float 1376 Times of Arrival (seconds)

Float 1378 Times of Arrival (seconds)

Float 1379 Times of Arrival (seconds)

Float 1381 Times of Arrival (seconds)

Float 1382 Times of Arrival (seconds)

Float 1383 Times of Arrival (seconds)

Float 1384 Times of Arrival (seconds)

Float 1385 Times of Arrival (seconds)

Float 1386 Times of Arrival (seconds)

Float 1387 Times of Arrival (seconds)

Float 1388 Times of Arrival (seconds)

Float 1389 Times of Arrival (seconds)

Float 1390 Times of Arrival (seconds)

Float 1391 Times of Arrival (seconds)

Float 1393 Times of Arrival (seconds)

Float 1394 Times of Arrival (seconds)

Float 1395 Times of Arrival (seconds)

Float 1396 Times of Arrival (seconds)

Float 1397 Times of Arrival (seconds)

Float 1398 Times of Arrival (seconds)

Float 1399 Times of Arrival (seconds)

Float 1400 Times of Arrival (seconds)

Float 1401 Times of Arrival (seconds)

Float 1402 Times of Arrival (seconds)

Float 1403 Times of Arrival (seconds)

Float 1404 Times of Arrival (seconds)

Float 1405 Times of Arrival (seconds)

Float 1406 Times of Arrival (seconds)

Float 1407 Times of Arrival (seconds)

Float 1408 Times of Arrival (seconds)

Float 1409 Times of Arrival (seconds)

Float 1410 Times of Arrival (seconds)

Float 1411 Times of Arrival (seconds)

Float 1412 Times of Arrival (seconds)

Float 1413 Times of Arrival (seconds)

Float 1414 Times of Arrival (seconds)

Float 1415 Times of Arrival (seconds)

Float 1416 Times of Arrival (seconds)

Float 1417 Times of Arrival (seconds)

Float 1418 Times of Arrival (seconds)

Float 1419 Times of Arrival (seconds)

Float 1420 Times of Arrival (seconds)

Float 1421 Times of Arrival (seconds)

Float 1423 Times of Arrival (seconds)

Float 1424 Times of Arrival (seconds)

Float 1425 Times of Arrival (seconds)

Float 1426 Times of Arrival (seconds)

Float 1427 Times of Arrival (seconds)

Float 1428 Times of Arrival (seconds)

Float 1429 Times of Arrival (seconds)

Float 1430 Times of Arrival (seconds)

Float 1431 Times of Arrival (seconds)

Float 1432 Times of Arrival (seconds)

Float 1433 Times of Arrival (seconds)

Float 1434 Times of Arrival (seconds)

Float 1435 Times of Arrival (seconds)

Float 1436 Times of Arrival (seconds)

Float 1437 Times of Arrival (seconds)

Float 1438 Times of Arrival (seconds)

Float 1439 Times of Arrival (seconds)

Float 1440 Times of Arrival (seconds)

Float 1441 Times of Arrival (seconds)

Float 1443 Times of Arrival (seconds)

Float 1444 Times of Arrival (seconds)

Float 1445 Times of Arrival (seconds)

APPENDIX G: RAFOS WEIGHT & BALANCE INVESTIGATION BY E. SLATE

Rafos W&B Investigation
Rev. 2015/05/01
E. Slate

Description:

This document investigates the potential sources and associated magnitudes of errors for the Rafos float ballast evaluation using the URI/GSO ballast process and test facility. This investigation is motivated by the “suspicion” that floats are being ballasted slightly heavy, as evidenced by recent field data for float serial numbers 1319, 1330, 1342, 1345, 1347, and 1349.

Quantifying the magnitude of the ballasting error:

Recent field data indicates ballasting of 2800 m floats (sn 1342, 1345, 1347 and 1349) exhibited initial pressure errors of 122 to 160 dbar. The 2200 m floats ballasted in the same batch (sn 1319 and 1330) exhibited closer pressure match; initial pressure errors ranged from -6 to 58 dbar for the 2200 m floats.

The approximate magnitude of ballast error yielding depth (pressure) errors observed in the field can be evaluated by examining the sensitivity of ballast as a function of pressure. The ballast.m code for a specific float was used to derive the net ballast as a function of pressure with all other conditions held constant.

Using float 1347 ballast data, the ballast.m code was run for a range of pressures from 2700 m to 3000m. The target salinity ($S_{\text{target}} = 34.887$ psu) and temperature ($T_{\text{target}} = 1.375$ degC) were held constant. The resulting net ballast as a function of pressure was plotted; the resulting linear relationship between ballast and pressure exhibited a slope of 0.0377 g/dbar. Using this relationship, a ballast error of approximately 3.77 g results in a 100 dbar depth error (at 2800 dbar operating point).

Ballast error sources:

The following list of potential ballast error sources was derived from a brainstorm session *without consideration of the potential magnitude or likelihood of the individual errors*.

1. Measurement errors
 - 1.1. Dry weights
 - 1.1.1. Float
Type of error: constant offset
 - 1.1.2. Trim weights and ball cage
Type of error: constant offset

1.2. Wet weights

1.2.1. Float

The wet weight of the float at tank conditions is measured by the strain-gauge beam. Thus, errors in float wet weight can be attributed to strain gauge beam instrumentation errors (see item 2.1) or errors in the extrapolation of the measured wet weight from tank to target conditions (see item 4.1).

1.2.2. Trim weights and ball cage

The wet weight of the ballast trim balls is computed from the trim material density (fixed in code at 8.0 g/cm^3), tank water density at target pressure, and the measured dry weight of the balls. Errors in the trim ball wet weight can be attributed to errors in ball dry weight measurement (item 1.1.2), specification of trim material density, and/or estimation of tank water density at target pressure.

The ball cage wet weight is measured.

Type of error: constant offset

2. Instrumentation errors

2.1. Beam strain-gauge scale

2.2. Tank temperature

2.3. Tank pressure

3. Property errors

3.1. Tank properties

3.1.1. Salinity (S_{tank})

3.1.2. Temperature (T_{tank})

3.1.3. Density

3.2. In-situ properties

3.2.1. Salinity (S_{target})

3.2.2. Temperature (T_{target})

3.2.3. Density

The density calculation at specified conditions (salinity, temperature and pressure) is performed using the CSIRO Matlab toolset that is based on the UNESCO 1983 polynomial (EOS 80). The valid range of input parameters is described in the Unesco 1983 publication "Algorithms for computation of fundamental properties of seawater", by A.P. Foffonoff and R. C Millard, Jr., page 19:

"The EO80 is valid for $S = 0$ to 42; $t = -2$ to 40°C ; $p = 0$ to 10000 decibars."

- 3.3. Float properties
 - 3.3.1. Dry weight
 - 3.3.2. Float volume
- 4. Computation errors
 - 4.1. Extrapolation of float wet weight from tank conditions to target conditions
 - 4.1.1. Pressure
 - 4.1.2. Salinity
 - 4.1.3. Temperature
 - 4.2. Float volume

Float Ballast Sensitivity Study:

A sensitivity study was conducted to evaluate the relative magnitudes of potential error sources. The weight and balance (W&B) equations incorporated by the GSO ballast procedure to derive the net ballast weight at specified target conditions was evaluated using a perturbation analysis around a fixed operating point.

1. GSO ballast code

The data from a ballast test conducted at GSO is processed by a toolset implemented as a collection of Matlab scripts, referred to here as “ballast.m”. A high level flowchart of the ballast.m code (version 1.15, dated 2006/10/26) is included in appendix A for reference. The ballast.m code produces several output products (files), including a primary output that documents the raw test data and processed results. A sample copy of this primary output product is included in Appendix B. The internal variables of ballast.m associated with the various output data items are identified in blue text adjacent to each of the output values (see Appendix B). In Appendix C, brief descriptions of the key variables used by the ballast.m toolset are documented. This documentation includes references to the variable names and associated line numbers in ballast.m code where the variables are created and/or referenced. Appendix D contains a printout of the ballast.m code for reference.

2. Description of sensitivity study

A sensitivity analysis was conducted on the ballast equations implemented by the ballast.m toolset. The net ballast to be added to a float based on a GSO test run is computed in ballast.m via the following equations. (note: variable names from ballast.m are used in this text)

$$\text{total_correction} = \text{neutral_weight} + \text{thermal_correction} + \text{seawater_correction} \quad (1)$$

$$\text{neutral_weight} = \text{trim_total_wet} - \text{trim_surplus_target_wet} \quad (2)$$

$$\text{thermal_correction} = \text{volume_float} * \text{float_thermal_coeff} * (\text{T_target} - \text{Tsbe_mean}) * \text{density_target} \quad (3)$$

$$\text{seawater_correction} = \text{volume_float} * \text{density_target} - \text{density_tank} \quad (4)$$

The three components (equations 2,3,4) of the net ballast equation (equation 1) are evaluated individually to simplify the process.

The general approach used for the sensitivity analysis is to generate a linearized approximation to the non-linear equation set, evaluate the linearized approximation at a specified (nominal) operating point, then conduct a perturbation study on the linearized model. To linearize the non-linear system of equations describing the ballast process, a Taylor's series expansion is performed, and higher order terms are ignored.

$$f(x_1, x_2, \dots) = f(x_{1o}, x_{2o}, \dots) + \left. \frac{\partial f}{\partial x_1} \right|_o (x_1 - x_{1o}) + \left. \frac{\partial f}{\partial x_2} \right|_o (x_2 - x_{2o}) + \dots + \text{HOT} \quad (5)$$

Where:

o means "evaluated at the operating point"
HOT means "higher order terms"

Letting:

$$\begin{aligned} \Delta f &= f(x_1, x_2, \dots) - f(x_{1o}, x_{2o}, \dots) \\ \Delta x_1 &= x_1 - x_{1o} \\ \Delta x_2 &= x_2 - x_{2o} \\ \text{etc.} \end{aligned}$$

Equation 5 can then be re-written as:

$$\Delta f = \left. \frac{\partial f}{\partial x_1} \right|_o \Delta x_1 + \left. \frac{\partial f}{\partial x_2} \right|_o \Delta x_2 + \dots \quad (6)$$

Equation 6 can now be used to evaluate each of the components of the net ballast relationship (equation 1). The various parameters ($\Delta x_1, \Delta x_2, \text{etc.}$) that comprise the ballast relationship can be perturbed to estimate the impact on net ballast (Δf). In some cases, the partial derivatives of equation 6 must be evaluated using the chain rule. For example, the trim_total_wet term of equation 2 is described by the following equation:

$$\text{trim_total_wet} = \text{ball_cage_wet} + \text{trim_dry} * (1 - \text{density_target_tank}/\text{trim_density})$$

To evaluate the derivative of trim_total_wet with respect to tank temperature, the following would be used:

Let $f = \text{trim_total_wet}$, then:

$$\left. \frac{\partial f}{\partial T_{\text{tank}}} \right|_o = \frac{\partial f}{\partial \text{density_target_tank}} \frac{\partial \text{density_target_tank}}{\partial T_{\text{tank}}} \bigg|_o$$

The detailed expression of equations 1 – 4 using a Taylor's series expansion is documented (via handwritten notes) in Appendix E.

3. Overview of the sensitivity study results

The sensitivity study is evaluated for a single float (float 1345) in an Excel spreadsheet (file: Rafos_WB_2015-04-13.xlsx, worksheet: Float 1345 Sensitivity study). An overview of the results is documented in this section. The net ballast deviation is tabulated in Table 1 as a function of the perturbed parameter and the magnitude of perturbation for each parameter. The table entries are grouped by the three components (neutral_weight, thermal_correction, seawater_correction) that make up the net ballast (total_correction).

The magnitudes of the perturbations used in the sensitivity study were derived as follows:

- For measured parameters, the perturbation magnitude was based on estimated or specified measurement accuracy. Variables falling into this category include:
 - Ball_cage_wet: ball cage wet weight
 - Trim_dry: trim ball(s) dry weight
 - T_tank: test tank water temperature. The tank temperature measurement that is used for float ballast calculations is the mean temperature reported by the Seabird SBE-38 Oceanographic Thermometer. The device specification sheets indicate an initial accuracy of 0.001 deg C, and a long-term stability of 0.001 deg C. A tank temperature perturbation value of 0.01 deg C was used in the sensitivity study (an order of magnitude greater than the specified accuracy).
- For derived or computed parameters, the perturbation magnitude was based on an assumed level of accuracy (e.g., 1%). Variables falling into this category include:
 - Trim_density: trim ball(s) material density

Ballast_slope: derived slope of weight vs pressure data
from ballast test

Y_intercept: derived y intercept of weight vs pressure
data from ballast test

Volume_float: derived float volume (see discussion
below)

Float_thermal_coeff: float coefficient of thermal
expansion

- For target parameters, the perturbation magnitude was based on an estimated in-situ property variance based on data from the Hydrobase database. Variables falling into this category include:

S_target: target salinity

T_target: target temperature

The volume_float parameter in Table 1 incorporates a number of variables itself, specifically:

$$\text{volume_float} = (\text{float_total_dry} + \text{neutral_weight}) / \text{density_target_tank} \quad (7)$$

While these parameters could have been incorporated in the ballast sensitivity study directly, it was decided to present the results in terms of volume perturbation for the benefit of simplification. A similar linearized perturbation study was performed independently on the volume equation, with results tabulated in Table 2 below. The delta volume results of Table 2 were used to develop the magnitude of volume perturbation for the float ballast sensitivity analysis summarized in table 1. Based on the volume perturbation study, the 0.05% volume perturbation (7.45 cm³) used in the ballast sensitivity study represents a very conservative upper limit on the volume.

Component	Perturbed variable	Perturbation magnitude	Delta ballast (g)
neutral_weight	ball_cage_wet	0.01 g	0.010
	trim_dry	0.1 g	0.087
	trim_density	1% (0.08 g/cc)	0.279
	T_tank	0.01 C	0.000
	ballast_slope	1%	1.142
	y_intercept	1%	0.835
thermal_correction	volume_float	0.05% (7.45 cc)	-0.0005
	float_thermal_coeff	10%	-0.119
	S_target	0.5% (0.174 psu)	-1.558*10 ⁻⁴
	T_target	5% (0.069C)	0.021
	T_tank	0.01 C	0.003
seawater_correction	volume_float	0.05% (7.45 cc)	0.205
	S_target	0.5% (0.174 psu)	2.032
	T_target	5% (0.069 C)	-0.149
	T_tank	0.01 C	0.016

Table 1: Sensitivity Analysis Results

Component	Perturbed variable	Perturbation magnitude	Delta volume (cm ³)
volume_float	float_total_dry	0.1 g	0.099
	neutral_weight	1 g	-0.987
	T_tank	0.01 C	0.015

Table 2: Float volume perturbation analysis results

The “delta ballast” column contained in table 1 can be used to identify candidates for further evaluation. Specifically, the items identified in red text represent the largest ballast error sources (based on this study) that will be investigated further in the following section.

Float Ballast Error Candidates:

This section further investigates the prime candidates contributing to ballast error based on the sensitivity study conducted in the previous section.

1. Target Salinity

The sensitivity study indicates that an error of 0.5% (0.174 psu for float 1345) resulted in a potential ballast error of 2.03 g. Examining the HydroBase database of measured (and interpolated?) water

properties at the lat-lon regions for each of the floats under evaluation indicates a much smaller standard deviation of target salinity than the “proxy” 0.5% level used for the sensitivity study. For each of the regions of the deployed floats, the predicted worst-case deviation of salinity at the deployed depth was at least one to two orders of magnitude less than the 0.5% value used here. Assuming the HydroBase predictions/measurements are reasonably accurate, the potential ballast error due to errors in target salinity can be ignored.

2. Ballast data curve fit (ballast slope and y intercept)

The current ballast.m code fits a 1st order (linear) relationship to the weight versus pressure data from the test tank to calculate the trim surplus at the following conditions: tank temperature, tank salinity and target pressure. Two different aspects of the data fit were examined in an effort to improve the estimation of trim surplus for pressures that exceed the tank’s maximum pressure rating (approximately 1450 dbar).

- Selection of the test data used for the polynomial curve fit: currently, most of the floats had the first two and last two data points eliminated from the data set used for polynomial generation. In an effort to evaluate the impact of the data set used for the curve fit, four different cases of data "filtering" were examined:
 - Case 1: All data accepted and used for the data fit
 - Case 2: First 2 and last 2 data points removed (“standard” for most of the floats examined here)
 - Case 3: First pressure increase cycle removed
 - Case 4: Second full pressure cycle used
- The degree of the polynomial curve fit to the data: currently, a 1st order (straight line) fit is used to determine the data model. A second order curve fit was examined in addition to the first order fit.

The metrics for the basis of comparison of results was rms weight error between the tank measurements and predicted weight using the polynomial curve fit to the data.

The evaluation is conducted for each float in the Excel spreadsheet Rafos_WB_2015-04-13.xlsx, worksheets: “float_1345_Datafit” through “float_1349_Datafit”. The results are summarized in the worksheet “Data Fit Study” and reproduced in Table 3 below.

RMS Weight error (g)		Float					
		1319	1330	1342	1345	1347	1349
case 1:	1st order poly fit	0.550	0.678	0.533	0.645	0.415	0.575
	2nd order poly fit	0.370	0.370	0.389	0.434	0.283	0.389
case 2:	1st order poly fit	0.057	0.057	0.063	0.063	0.061	0.080
	2nd order poly fit	0.034	0.054	0.043	0.039	0.023	0.049
case 3:	1st order poly fit	0.058	0.067	0.066	0.066	0.062	0.084
	2nd order poly fit	0.034	0.048	0.040	0.029	0.019	0.040
case 4:	1st order poly fit	0.079	0.079	0.091	0.080	0.078	0.082
	2nd order poly fit	0.044	0.038	0.056	0.042	0.033	0.043

neutral_weight (g)		Float					
		1319	1330	1342	1345	1347	1349
case 1:	1st order poly fit	195.551	195.156	225.188	220.429	219.905	217.610
	2nd order poly fit	200.051	200.914	233.424	231.211	226.793	227.128
case 2:	1st order poly fit	197.169	197.216	227.471	223.345	221.810	220.207
	2nd order poly fit	196.384	196.600	225.728	221.437	219.640	217.756
case 3:	1st order poly fit	197.198	197.194	227.476	223.331	221.846	220.210
	2nd order poly fit	196.292	196.296	225.270	220.828	219.326	217.117
case 4:	1st order poly fit	197.203	197.059	227.454	223.225	221.788	220.243
	2nd order poly fit	196.167	196.148	224.937	220.798	219.241	217.781

Delta neutral weight (g) - 2nd order poly relative to 1st order poly		Float					
		1319	1330	1342	1345	1347	1349
case 1:		4.500	5.759	8.236	10.782	6.888	9.519
case 2:		-0.785	-0.616	-1.743	-1.908	-2.169	-2.451
case 3:		-0.906	-0.898	-2.205	-2.504	-2.520	-3.093
case 4:		-1.037	-0.911	-2.517	-2.427	-2.547	-2.462

Table 3: Float Ballast Data Curve Fit Results

Table 3 summarizes the following items for each of the data filtering cases, and for 1st and 2nd order curve fits:

- Rms weight error (predicted vs actual weight at the actual pressure test points).
- Total computed neutral weight (in g) at tank temperature, tank salinity and target pressure.
- Difference in neutral weight using the 1st order and 2nd order (e.g., $\text{neutral_weight}_{2\text{nd_order}} - \text{neutral_weight}_{1\text{st_order}}$).

Several conclusions can be drawn from the results summarized in Table 3:

- a. Filtering the "end points" from the data set results in a much better fit for both the 1st order and 2nd order poly curve fits.
- b. Results from the first order poly curve fit when removing the first 2 and last 2 data points (case 2) matches the ballast.m results perfectly (as expected... this was done as a check).
- c. For all cases of data filtering except the case where all data is used, the second order poly results in much improved rms weight error, a sign that the higher order poly yields a better match to the actual data.
- d. For all cases of data filtering except the case where all data is used, the second order poly results in a 2.5 to 3 g reduction in the weight required to establish neutral buoyancy at tank conditions (salinity and temperature) and target pressure for the 2800 m floats, and about 0.75 to 1 g reduction for the 2800 m floats.

3. Trim density

The density value of the trim balls used in the test tank is set to a constant value of 8.0 g/cm³ in the ballast.m code. While this is a typical density specification for the grade of stainless used (316), it does not account for slight variation arising from the different grades of 316 available. According to the following materials database (<http://www.azom.com/properties.aspx?ArticleID=863>), the density of grade 316 stainless can vary from 7.87 to 8.07 g/cm³ (mean = 7.97 g/cc). This represents a +0.875 to -1.625% variation in density from the constant value used by the ballast.m code.

The trim balls are mechanically gauged (diameter and weight) to a high accuracy as part of the qualification process. It would make good sense to modify the ballast.m code to use the measured properties for the trim balls instead of the typical value from published physical properties.

4. Float volume

The sensitivity study indicates that a 0.05% (7.45 cm³) error in estimation of float volume results in a 0.205 g error in the float ballast. However, as Table 2 indicates, the variables that make up the float volume estimate would have to exhibit exceptionally large errors to contribute a volume error of this magnitude. This implies that the estimate of the volume perturbation used in the sensitivity study (0.05%) was high by upwards of an order of magnitude. Based on these results, it is estimated that the ballast error resulting from float volume inaccuracy is negligible.

Summary Comments:

1. Effect of compressibility of individual components:

The overall weight versus pressure relationship for the float assembly is the net sum of the weight versus pressure relationships for the individual components that make up the assembly. Incorporating a “highly” compressible element of the float assembly (e.g., UHMW insulator) is acceptable as long as two conditions are met:

- a. The material behaves in a predictable manner with respect to pressure. If using a linear curve fit to the test tank data, then this implies that the material must behave in a linear fashion with respect to pressure in order to minimize ballast errors. Alternatively, if the material does not operate in a linear fashion, then using a higher order curve fit may provide a more predictable behavior at pressures beyond the tank test range.
- b. The material does not alter the compressibility of the net assembly to the point that it is “unstable” at the ballasted pressure. For high natural stability, the stiffness of the float assembly must be higher than that of the surrounding water. Based on observations of the UHMW insulators in the GSO test tank on 4/30/15, the plastic material stiffness is roughly equivalent to the water in which it was tested. Given this result, and the volume of the material relative to the total float volume, it is not expected that the insulator will negatively impact the float natural stability in a meaningful way.

2. The presence of compressible and non-linear materials is accounted for in the ballast process. Errors caused by such materials can be attributed to poor model fit to the test tank data, and the extrapolation of inaccurate models to pressures beyond the tank’s capabilities.

3. Effect of non-linear behavior of float materials:

The presence of float materials that exhibit non-linear behavior at high pressures can be acceptable if the material remains stable with time, and if the non-linearity is accounted for in the ballast calculations.

If the material exhibits long-term creep, resulting in a loss of volume over time, then the float would demonstrate ballast errors that increase with time. However, assuming the float is properly ballasted for target conditions, the float should initially demonstrate proper operation at target depth. Based on the observations of the six OSNAP float data, it appears that these floats demonstrated static ballast errors from the start of operation.

The impact of non-linear material behavior can be mitigated in two ways:

- a. Extend the pressure regime of testing to cover the target conditions to minimize the extrapolation-induced errors.
- b. Apply a better model to the test data to reduce the extrapolation error. The data fit study documented in this report indicates that the

data model for the range of pressures tested in the tank can be improved by using a higher order polynomial than the one currently employed. In all six floats, the use of a second order model resulted in a better data fit than the first order model currently used. In addition, the model may be improved further by careful selection of the data set used for the model characterization.

4. Tank strain beam induced errors:

The strain beam in the GSO test tank is calibrated daily using a range of weight standards that span the full measurement range of the device. The calibration is done at ambient pressure. I'm not aware if any testing has been performed on the beam to investigate the effect of pressure on the beam calibration. During float ballast testing at GSO on 4/30/15, Jim Fontaine suggested a simple test using a weight "standard" attached to the beam that could be used to investigate beam behavior at elevated pressures. If this has not been done already, I think it's a very worthwhile test to run to characterize the beam behavior at pressure and rule it out as a potential contributor to ballast errors.

Appendix A

Ballast.m (version 1.15) High Level Flowchart

function call: ballast(tank_filename, param_filename, mode)

Appendix B

Ballast.m (version 1.15) Output Product

Ballast Summary: Float 1319.01 : "OSNAP"

Data File : "RTC1319-01.txt"
Param File : "param_OSNAP_2200dbar_20140527.txt"
Data Time: 05/27/2014 18:15 -- Plot Time: 04/08/2015 14:42

Dry Weights	
float	= 13153.10 grams
compressee	= 0.90 grams
drop weight	= 1759.90 grams
miscellaneous	= 0.00 grams
total float	= 14913.00 grams
ball cage	= 66.60 grams
balls (2:0:2)	= 153.58 grams
trim density	= 8.0000 grams/cc
trim (total)	= 220.18 grams
trim (surplus)	= -5.13 grams
Wet Weights	
ball cage	= 58.50 grams
balls (2:0:2)	= 134.32 grams
trim (total)	= 192.68 grams
trim (surplus)	= -4.48 grams
Tank Properties	
TEMPERATURE	= 5.799 degC
Tdiff top,bot	= -0.72, 0.22 degC
conductivity	= 1.447e-05 S
sigma t	= -0.051 kg/m ³
rho (target)	= 1.01046 grams/cc
Target Properties	
salinity	= 34.947 psu
temperature	= 2.857 degC
pressure	= 2200.0 dbar
sigma t	= 27.856 kg/m ³
rho	= 1.03793 grams/cc
Compressibilities	
float volume	= 14953.79 cc
target	= -4.307e-06 1/dbars
tank	= -4.614e-06 1/dbars
float	= -1.934e-06 1/dbars
float error	= 0.004e-06 1/dbars
Ballast Corrections	
for neutral	= 197.17 grams
for seawater	= 410.75 grams
thermal coeff	= 2.000e-05 1/degC
for thermal	= -0.91 grams
total weight	= 607.00 grams

Total Additional Ballast Weight
INTERNAL 607.00 grams

ballast version 1.15 - 2006/10/26

3.txt
xt
xt
l.txt
27_quad.txt

Appendix C

Description of Ballast.m Variables

GSO Ballast Code Notes
Rev. 2105/04/10

Target and Tank Conditions

- Target salinity, temperature and pressure conditions are read in from the parameter file (see ballast.m line 260):
 - S_target (psu)
 - T_target (deg C)
 - P_target (dbar)
- density_target:
 - Target water density, computed via (line 1070):
 $\text{density_target} = \text{sw_dens}(\text{S_target}, \text{T_target}, \text{P_target}) / 1000$
- Tank salinity and temperature conditions are specified as:
 - S_tank = 0(psu): hardcoded (line 1094)
 - T_tank (deg C): computed from the SBE temperature measurements contained in the tank file (line 1075)
- density_target_tank:
 - Density of the tank at tank salinity/temperature and target pressure, computed via (line 1098):
 $\text{density_target_tank} = \text{sw_dens}(\text{S_tank}, \text{T_tank}, \text{P_target}) / 1000$

Dry Weights

- float_dry:
 - Float dry weight, entered from tank file (line 436). This parameter is measured using a *TBD* scale.
- comp_dry:
 - Compressed dry weight, entered from tank file (line 437). This parameter is measured using a *TBD* scale. Note: this parameter is 0.0 in rafos tank files.
- drop_dry:
 - Drop weight dry weight, entered from tank file (line 438). This parameter is measured using a *TBD* scale.
- misc_dry:
 - Misc dry weight, entered from tank file (line 439). This parameter is measured using a *TBD* scale. Note: this parameter is 0.0 in rafos tank files.
- float_total_dry:
 - Total float dry weight, entered from tank file (line 1110):
 $\text{float_total_dry} = \text{float_dry} + \text{drop_dry} + \text{comp_dry} + \text{misc_dry}$
- trim_density:
 - Density of the stainless steel trim ball material. The trim_density variable is fixed (hardcoded) to a value of 8.00 g/cm³ (in the proper range for typical grades of stainless steel).
- trim_dry:

Total dry weight of the trim ball material. This number is based on the specified ball dry weights and ball configuration, e.g., # A balls, B balls and C balls. The ball dry weights are hard coded (line 195). The values entered are measured values. Using ball geometry and specified density yields slightly different ball weight: (difference less than 1%)

Ball	wt from ballast.m (g)	wt from geometry (g)
A	8.50	8.58
B	28.79	28.96
C	68.29	68.642

- trim_wet:
Total wet weight of the trim ball material at tank conditions and target pressure. This number is based on the specified ball wet weight and ball configuration. The total trim wet weight value computed from the hardcoded ball wet weights are printed in the output page under the “Wet Weights” section, “balls (###)” line (variable trim_wet). ***This value is not used in the weight and balance computation (see trim_total_wet).***
- ball_cage_wet:
Wet weight of the trim ball cage, read in from tank file (line 456).
- trim_total_wet:
Total wet weight of the trim ball material and associated ball cage. The trim ball wet weight is computed from the trim dry weight, ball density and tank target density (line 1120). The trim ball wet weight computation assumes the material is incompressible.
$$\text{Trim_total_wet} = \text{ball_cage_wet} + \text{trim_dry} * (\text{trim_density} - \text{density_target_tank}) / \text{trim_density}$$

Wet Weights

- trim_wet:
Total wet weight of the trim ball material at tank conditions and target pressure. This number is based on the specified ball wet weight and ball configuration. The total trim wet weight value computed from the hardcoded ball wet weights are printed in the output page under the “Wet Weights” section, “balls (###)” line (variable trim_wet). ***This value is not used in the weight and balance computation (see trim_total_wet).***
- ball_cage_wet:
Wet weight of the trim ball cage, read in from tank file (line 456).
- trim_total_wet:
Total wet weight of the trim ball material and associated ball cage. The trim ball wet weight is computed from the trim dry weight, ball density and tank target density (line 1120). The trim ball wet weight

computation assumes the material is incompressible. ***This value is used in the weight and balance computation.***

$$\text{trim_total_wet} = \text{ball_cage_wet} + \text{trim_dry} * (\text{trim_density} - \text{density_target_tank}) / \text{trim_density}$$

- **trim_surplus_target_wet:**
Total wet weight of the surplus trim. This value represents the “excess” trim ballast at the specified tank salinity and temperature and target pressure. This is the weight value that the beam transducer “indicates” (target pressure within tank range capability) or “would indicate” (target pressure beyond tank range capability) at target pressure. For target pressures outside of the maximum range capability of the tank, the surplus trim weight is computed by extrapolating the weight versus pressure relationship to the target pressure. The extrapolation is calculated using a first order polynomial fit to the tank test data via least squares method, i.e., $W_t = \text{ballast_slope} * \text{pressure} + y_intercept$

$$\text{trim_surplus_target_wet} = y_intercept + \text{ballast_slope} * P_target$$

Float Volume

- **volume_float:**
Represents the float volume at tank salinity and temperature, and target pressure (line 1135).

$$\text{volume_float} = (\text{float_total_dry} + \text{trim_total_wet} - \text{trim_surplus_target_wet}) / \text{density_target_tank}$$

Ballast Computations

- **neutral_weight:**
Represents the ballast required for neutral stability at tank salinity and temperature, and target pressure (line 1179).
- $$\text{neutral_weight} = \text{trim_total_wet} - \text{trim_surplus_target_wet}$$
- **thermal_correction:**
Represents the ballast required to correct for the thermal expansion of the float due to differences in tank and target temperatures (line 1192).

$$\text{thermal_correction} = \text{density_target} * \text{volume_float} * \text{float_thermal_coeff} * (T_target - T_{sbe_mean})$$

where:

float_thermal_coeff is the float’s overall thermal coefficient of expansion (l/deg C), entered from the parameter file.

- **seawater_correction:**
Represents the ballast required to correct for the water density differences in tank and target due to different salinity and temperature conditions (line 1199).
$$\text{seawater_correction} = \text{volume_float} * (\text{density_target} - \text{density_target_tank})$$
- **total_correction:**
Represents the total ballast required to achieve neutral stability at target conditions (line 1204).
$$\text{total_correction} = \text{neutral_weight} + \text{thermal_correction} + \text{seawater_correction}$$

Appendix D

Printout of Ballast.m Code

10/26/06
15:18:24

ballast.m

```
1: function ballast( file_tank, file_param, mode )
2: %
3: % BALLAST( FILE_TANK, FILE_PARAM, MODE )
4: %
5: % This routine will
6: %
7: % (1) Read in the strain vs. pressure data (and ancillary
8: % information) from a file created by the RTC (RAPOS Tank
9: % Controller),
10: %
11: % (2) Read various parameter and calibration files,
12: %
13: % (3) Optionally ask the user for information, and then
14: %
15: % (4) Compute the float volume, compressibility, and the amount
16: % of weight to add to the float in order to reach the target
17: % pressure or density surface.
18: %
19: % The file FILE_TANK contains the RTC data, and will be unique
20: % for each ballasting of each float.
21: %
22: % The file FILE_PARAM will contain the parameters for running
23: % the BALLAST routine. The contents will include:
24: %
25: % * The target salinity, temperature, and pressure.
26: %
27: % * The float's coefficient of thermal expansion (determined
28: % experimentally).
29: %
30: % * The density of the ballast weights (if they are to be
31: % carried external to the float).
32: %
33: % * The limiting temperature difference to determine if the
34: % tank is too stratified
35: %
36: % * A reference frequency of the Data Acquisition Card, to be
37: % used to monitor the stability of the DAQ and, if
38: % necessary, modify the sensor frequencies measured by the
39: % RTC.
40: %
41: % * Relative or complete path names of files containing
42: % calibration coefficients for all the various tank
43: % measurements:
44: %
45: % -- tank pressure ( frequency to psi )
46: % -- tank temperature (top) ( frequency to degC )
47: % -- tank temperature (bottom) ( frequency to degC )
48: % -- strain beam pressure effect ( psi to frequency )
49: % -- strain beam temperature effect ( degC to frequency )
50: % -- strain beam weight ( frequency to grams )
51: %
52: % Most of the contents of FILE_PARAM will likely be unchanged
53: % throughout a series of ballasting runs.
54: %
55: % The routine can be run in an automatic mode with no user
56: % intervention (MODE = 0) or with the user prompted to evaluate
57: % data quality and to modify the seawater parameters for the
58: % target surface (MODE = 1).
59: %
60: % NOTES: The density of fresh water and sea water is computed
61: % using a subset of the Matlab toolkit SEAWATER (version 2.0.1
62: % - 1998) written by Phillip Morgan. The standard reference is:
63: %
64: % Morgan, Phillip P., 1994: SEAWATER: A Library of MATLAB
65: % Computational Routines for the Properties of Sea Water.
```

```
66: %
67: % The files can be obtained at:
68: %
69: % <http://www.marine.csiro.au/datacentre/processing.htm>
70: %
71: % Mark Prater - GSO/URI
72: %
73: % 2002/09/25 : started
74: %
75: % 2002/11/07 : version 1.0 beta
76: %
77: % 2002/12/08 : version 1.0
78: %
79: % (1) Added parameter file option "multi_option" to choose how
80: % to handle multiple data samples at the same pressure
81: % level
82: %
83: % (2) Added parameter file option "pressure_range" to specify
84: % a range of pressure values to use, ignoring all values
85: % outside this range.
86: %
87: % (3) Changed ballage weights
88: %
89: % 2002/12/11 : version 1.01
90: %
91: % (1) Fixed errors found in review by Dave Hebert and Tom
92: % Rossby
93: %
94: % (2) Implemented a tank stratification warning
95: %
96: % 2002/12/12 : version 1.02
97: %
98: % (1) Allow "stddev_limit" to be set regardless of MODE
99: %
100: % 2002/12/13 : version 1.03
101: %
102: % (1) Fixed placement of title in edit figure
103: %
104: % (2) Allowed user to select mode = 1 but not do any editing
105: %
106: % (3) Changed wording of query to change target properties
107: %
108: % 2002/12/13 : version 1.04
109: %
110: % (1) Properly handle case where RTC terminates before
111: % collecting data at all nominal pressure steps
112: %
113: % (2) Automatically produce a "eps2" postscript file of the
114: % summary plot under the name "tank_file.ps" (removing the
115: % suffix "txt" if it exists)
116: %
117: % (3) Remove the suffix "txt" (if it exists) in the name of
118: % the summary text file
119: %
120: % (4) Print pressure range in both psi and dbars
121: %
122: % 2002/12/16 : version 1.05
123: %
124: % (1) Corrected the sign of how the beam pressure and
125: % temperature corrections are applied
126: %
127: % 2002/12/19 : version 1.06
128: %
129: % (1) Changed the name of the RTC datafile keyword from
130: % '60_sec_conductivity_siemens' to
131: % '90_sec_conductivity_siemens'. This is a hack that had
132: % to be made because the present version of RTC software
133: % does not yet comply with the universally accepted
134: % "Prater's law of keywords".
135: %
136: % (2) Changed the plot domain so that the figure has about a
137: % 1-inch margin on the top (so titles are not punctured by
138: % three-hole punches.
139: %
140: % (3) Added tank stratification checks Ttop_diff and Tbot_diff
141: % (Ttop-tsbe and Ttsbe-Tbot) to right-hand text panel of
142: % the ballast summary plot, and in the text summary.
143: %
144: % 2002/12/19 : version 1.07
145: %
146: % (1) Added the beam serial number to the summary plot and to
147: % the text summary.
148: %
149: % 2003/01/06 : version 1.08
150: %
151: % (1) Fix bug that prevented the Pmax/Pmin range parameters
152: % from being used in the user-editing mode (MODE=1)
153: %
154: % (2) Change user-editing so that if multiple points are "very
155: % close" to an edited point, all will be removed - the
156: % reasoning being that the points "very close" to a bad
157: % point are bad as well. This required slight
158: % modifications in the format of the summary listing of
159: % the selected points.
```

10/26/06
15:18:24

2

ballast.m

```
131: % (3) Fixed parameter file application of temperature
132: % calibrations
133: % (4) Allow the keyword '60_sec.conductivity.siemens' to be
134: % read from the data file if the keyword
135: % '90_sec.conductivity.siemens' does not exist. This
136: % enables the user to read older RNC files (pre
137: % mid-December, 2002) with this and newer versions of the
138: % ballast software.
139: % 2003/01/07 : version 1.09
140: % (1) Increased "delta_Pstep" from 30 to 50 psi.
141: % 2004/05/24 : version 1.10
142: % (1) Keyword for conductivity is now "conductivity_siemens"
143: % 2004/06/07 : version 1.11
144: % (1) Specification of "delta_Pstep" is now made in the
145: % parameter file.
146: % Modified the Y-axis of the plot domain so that the plot
147: % legend is less likely to cover parts of the ballast
148: % data.
149: % 2004/06/07 : version 1.12
150: % (1) Plot target compressibility line on the ballast plot as
151: % an aid for comparing the float compressibility and
152: % determining the ballasting error as the isopycnal
153: % surface changes pressure.
154: % 2004/06/10 : version 1.13
155: % (1) List tank pressures in psi as well as dbars, and number
156: % the pressure level sequence (step 1, 2, 3, etc.)
157: % (2) Allow the user to specify, in the parameter file, the
158: % sequence number of the pressure steps to omit from the
159: % ballast calculation. This eliminates repetitive mouse-
160: % clicking to manually edit the pressure steps during each
161: % run of the ballast routine.
162: % (3) Changed format of times in plot title.
163: % 2004/07/27 : version 1.14
164: % (1) Changed plot legend label from "target" to "target
165: % compressibility".
166: % (2) changed the width of the filled box to fit text.
167: % 2006/10/26 : version 1.15
168: % (1) Purely cosmetic changes so that the program can be
169: % viewed cleanly when printed. All lines are limited to
170: % 65 characters, so that (especially when printing
171: % two to a side), no lines are wrapped.
172: % Change the coding slightly so that the Matlab style
173: % police (in Version 7.3) are satisfied. This primarily
174: % involved commenting out useful (but unused) statements,
175: % removing unnecessary brackets, and incorporating
176: % sprintf formatting syntax directly into error calls. The
177: % only remaining warnings are for arrays that are growing
178: % within loops. That is purely a speed issue, and changes
179: % would increase the complexity of the program.
180: % (3) Specified that the output plot be a postscript file (ps)
181: % rather than an encapsulated postscript file (eps). This
182: % choice might end up as a parameter file entry.
183: % -----
184: % -- set "universal" constants
185:
186: version = 'ballast version 1.15 - 2006/10/26' ;
187:
188: ball_type = 'ABCDE' ; % -- ball indentifiers
189:
190: % -- we have 3 stainless steel balls in use:
191: % A -> 0.50 inch diameter,
192: % B -> 0.75 inch diameter,
193: % C -> 1.00 inch diameter
194:
195: ball_dry(1) = 8.50 ; % -- ball A weight in air [grams]
```

```
196: ball_dry(2) = 28.79 ; % -- ball B weight in air [grams]
197: ball_dry(3) = 68.29 ; % -- ball C weight in air [grams]
198: ball_dry(4) = 0.00 ; % -- ball D, not used
199: ball_dry(5) = 0.00 ; % -- ball E, not used
200: ball_dry(6) = 0.00 ; % -- ball F, not used
201:
202: ball_wet(1) = 7.44 ; % -- ball A weight in fresh water [grams]
203: ball_wet(2) = 25.20 ; % -- ball B weight in fresh water [grams]
204: ball_wet(3) = 59.72 ; % -- ball C weight in fresh water [grams]
205: ball_wet(4) = 0.00 ; % -- ball D, not used
206: ball_wet(5) = 0.00 ; % -- ball E, not used
207: ball_wet(6) = 0.00 ; % -- ball F, not used
208:
209: ball_dim = length(ball_dry) ;
210:
211: trim_density = 8.00 ; % -- [grams/cc]
212:
213: psi2dbars = 0.690456 ; % -- conversion from psi to dbars
214:
215: xedit_fuzz = 5.0 ; % -- x range for data edit (dbars)
216: yedit_fuzz = 0.10 ; % -- y range for data edit (grams)
217:
218: norder = 1 ; % -- use this degree of polynomial for
219: % the fit to the ballast data
220: d_press = 10 ; % -- change in pressure to compute
221: % compressibility from
222: % equation-of-state
223:
224: x_tank_min = 0 ; % -- minimum tank pressure
225: x_tank_max = 3000 ; % -- maximum tank pressure
226: y_tank_min = 0 ; % -- minimum strain weight
227: y_tank_max = 150 ; % -- maximum strain weight
228:
229: % -- define valid ranges for target water properties, used
230: % to check for gross errors
231:
232: S_target_min = 30.0 ; % -- [psu]
233: S_target_max = 38.0 ;
234: T_target_min = -2.0 ; % -- [degC]
235: T_target_max = 35.0 ;
236: P_target_min = 0.0 ; % -- [dbar]
237: P_target_max = 5000.0 ;
238:
239: % -- YSI Resistance-to-Temperature calibration coefficients
240:
241: RtoT = [ -8.86789e-2 4.21019 -81.8005 517.998 ] ;
242:
243: % ----- check command line
244: %
245: if( nargin ~= 3 )
246: fprintf( 'usage: ballast( file_tank, file_param, mode )\n' ) ;
247: return
248: end
249:
250:
251: % -----
252: % -- access the parameter file
253:
254: fid = fopen( file_param, 'r' ) ;
255:
256: if( fid == -1 )
257: error( 'ballast: cannot open file_param "%s"', file_param )
258: end
259:
260: % -- get specifications of target water
```

10/26/06
15:18:24

3

ballast.m

```
261: S_target = getparam( fid, 'S_target', 'n' ) ;
262: T_target = getparam( fid, 'T_target', 'n' ) ;
263: P_target = getparam( fid, 'P_target', 'n' ) ;
264: P_target = getparam( fid, 'P_target', 'n' ) ;
265:
266: % -- specify range of tank pressures to use for ballasting
267: % calculations,
268:
269: pressure_range = getparam( fid, 'pressure_range', 'n' ) ;
270: Pmax_psi_range = max( pressure_range ) ;
271: Pmin_psi_range = min( pressure_range ) ;
272: Pmax_range = psi2dbars * Pmax_psi_range ;
273: Pmin_range = psi2dbars * Pmin_psi_range ;
274:
275: % -- select the ballast cycle sequence values to omit
276:
277: omit_steps = getparam( fid, 'omit_steps', 'n' ) ;
278:
279: % -- specify maximum allowable difference between a nominal and
280: % an actual pressure step for the calculation of mean step
281: % values
282:
283: delta_Pstep = getparam( fid, 'delta_Pstep', 'n' ) ;
284:
285: if( isnan(delta_Pstep) )
286:     delta_Pstep = 50 ; % -- psi ;
287: end
288:
289: % -- specify how to handle multiple data samples at a single
290: % pressure level. the valid options are : (0) take first
291: % value, (1) take last value, and (2) take average value
292:
293: multi_option = getparam( fid, 'multi_option', 'n' ) ;
294:
295: % -- get limit of standard deviations for excluding bad data (not
296: % really needed if user is going to manually edit the data)
297:
298: stddev_factor = getparam( fid, 'stddev_factor', 'n' ) ;
299:
300: % -- the reference frequency of the Data Acquisition Card [Hz],
301: % and the maximum allowable deviation to the reference.
302:
303: DAQ_reference_freq = getparam( fid, ...
304:     'DAQ_reference_freq', 'n' ) ;
305: DAQ_deviation_limit = getparam( fid, ...
306:     'DAQ_deviation_limit', 'n' ) ;
307:
308: % -- criteria for warning about excessive tank temperature
309: % stratification
310:
311: temp_diff_warning = getparam( fid, 'temp_diff_warning', 'n' ) ;
312:
313: % -- get float's experimentally determined coefficient of thermal
314: % expansion
315:
316: float_thermal_coeff = getparam( fid, ...
317:     'float_thermal_coeff', 'n' ) ;
318:
319: % -- information about the drop weight
320:
321: ballast_location = getparam( fid, 'ballast_location', 's' ) ;
322:
323: if( strcmp( ballast_location, 'external' ) )
324:     ballast_flag = 1 ;
325:     ballast_density = getparam( fid, 'ballast_density', 'n' ) ;
```

```
326: elseif( strcmp( ballast_location, 'internal' ) )
327:     ballast_flag = 0 ;
328: else
329:     error( [ 'ballast: Error specifying "%s" as location ', ...
330:         'of ballast weight.\n', ...
331:         'The choices are ', ...
332:         '## ballast_location internal' or '\n', ...
333:         ' ', ...
334:         '## ballast_location external' ], ...
335:         ballast_location )
336: end
337:
338: % -- get tick intervals for plotting
339:
340: tick_weight = getparam( fid, 'tick_weight', 'n' ) ; % [grams]
341: tick_press = getparam( fid, 'tick_press', 'n' ) ; % [dbars]
342:
343: % -- get the names of all the files that contain tank
344: % calibration coefficients
345:
346: file_directory = ... 'file_directory', 's' ;
347:
348: file_P_tank_coeff = ... 'file_P_tank_coeff', 's' ;
349:
350: file_Tbot_tank_coeff = ... 'file_Tbot_tank_coeff', 's' ;
351:
352: file_Ttop_tank_coeff = ... 'file_Ttop_tank_coeff', 's' ;
353:
354: file_beam_press_coeff = ... 'file_beam_press_coeff', 's' ;
355:
356: file_beam_temp_coeff = ... 'file_beam_temp_coeff', 's' ;
357:
358: file_beam_weight_coeff = ... 'file_beam_weight_coeff', 's' ;
359:
360:
361: % -- access and load tank calibration coefficients
362:
363: if( ~strcmp( file_P_tank_coeff, 'NaN' ) )
364:     P_tank_cal = load( sprintf( '%s%s', file_directory, ...
365:         file_P_tank_coeff ) ) ;
366:
367: else
368:     P_tank_cal = 0 ;
369: end
370:
371: if( ~strcmp( file_Ttop_tank_coeff, 'NaN' ) )
372:     Ttop_tank_cal = load( sprintf( '%s%s', file_directory, ...
373:         file_Ttop_tank_coeff ) ) ;
374:
375: else
376:     Ttop_tank_cal = 0 ;
377: end
378:
379: if( ~strcmp( file_Tbot_tank_coeff, 'NaN' ) )
380:     Tbot_tank_cal = load( sprintf( '%s%s', ...
381:         file_directory, ...
382:         file_Tbot_tank_coeff ) ) ;
383:
384: else
385:     Tbot_tank_cal = 0 ;
386: end
387:
388: if( ~strcmp( file_beam_press_coeff, 'NaN' ) )
389:     beam_press_cal = load( sprintf( '%s%s', ...
390:         beam_press_coeff = load( sprintf( '%s%s', ...
```


10/26/06
15:18:24

4

ballast.m

```
391: file_directory, ...
392: file_beam_press_coeff ) ;
393: else
394: beam_press_cal = 0 ;
395: end
396:
397: if( ~strcmp( file_beam_temp_coeff, 'NaN' ) )
398: beam_temp_cal = 1 ;
399: beam_temp_coeff = load( sprintf( '%s%s', ...
400: file_directory, ...
401: file_beam_temp_coeff ) ) ;
402: else
403: beam_temp_cal = 0 ;
404: end
405:
406: if( ~strcmp( file_beam_weight_coeff, 'NaN' ) )
407: beam_weight_cal = 1 ;
408: beam_weight_coeff = load( sprintf( '%s%s', ...
409: file_directory, ...
410: file_beam_weight_coeff ) ) ;
411: else
412: beam_weight_cal = 0 ;
413: end
414:
415: fclose(fid) ;
416:
417: % -----
418: % -- access the tank file
419:
420: fid = fopen( file_tank, 'r' ) ;
421:
422: if( fid == -1 )
423: error( 'ballast: cannot open file_tank "%s"', file_tank )
424: end
425:
426: % -- general information about this ballast run
427:
428: float_name = getparam( fid, 'Float', 's' ) ;
429: project = getparam( fid, 'Project', 's' ) ;
430: start_time = getparam( fid, 'start_time', 's' ) ;
431: start_time = start_time(1:(length(start_time)-3)) ;
432: beam_number = getparam( fid, 'beam_s/n', 's' ) ;
433:
434: % -- get the weights of the float and components
435:
436: float_dry = getparam( fid, 'dryweight_float', 'n' ) ;
437: comp_dry = getparam( fid, 'dryweight_comressee', 'n' ) ;
438: drop_dry = getparam( fid, 'dryweight_drop', 'n' ) ;
439: misc_dry = getparam( fid, 'dryweight_misc', 'n' ) ;
440:
441: % -- get the trim mass (balls and ball cage)
442:
443: trim_weights = getparam( fid, 'trim_weights', 's' ) ;
444:
445: nballs = NaN*ones( size(ball_dry) ) ;
446:
447: for k=1:ball_dim
448: i = 1 + findstr( trim_weights, ball_type(k) ) ;
449: nballs(k) = str2double( trim_weights(i) ) ;
450: end
451:
452: trim_dry = sum( nballs.*ball_dry ) ;
453: trim_wet = sum( nballs.*ball_wet ) ;
454:
455: ball_cage_dry = getparam( fid, 'dry_ballcage_wt_grams', 'n' ) ;
456: ball_cage_wet = getparam( fid, 'wet_ballcage_wt_grams', 'n' ) ;
457:
458: % -- access the DAQ timebase values
459:
460: DAQ_timebase = getmultiples( fid, 'DAQ_1Hz_timebase' ) ;
461:
462: % -- remove the occasional bad value
463:
464: bad_DAO = abs( DAQ_timebase - DAQ_reference_freq ) ...
465: > DAQ_deviation_limit ;
466:
467: DAQ_timebase(bad_DAO) = [] ;
468:
469: if( ~isempty(DAQ_timebase) )
470:
471: % -- if there are acceptable values remaining, determine
472: % the frequency correction factor
473:
474: DAQ_timebase_mean = mean(DAQ_timebase) ;
475: DAQ_frequency_ratio = DAQ_reference_freq /DAQ_timebase_mean ;
476: DAQ_warning = 0 ;
477: else
478:
479: % -- otherwise, set the factor to 1 and post a warning
480:
481: DAQ_timebase_mean = NaN ;
482: DAQ_frequency_ratio = 1.0 ;
483: DAQ_warning = 1 ;
484: end
485:
486: % -- get pressure steps used
487:
488: pressure_steps = getparam( fid, 'pressure_steps_PSI', 'n' ) ;
489:
490: % -- get conductivity
491:
492: conductivity = getparam( fid, 'conductivity_siemens', 'n' ) ;
493:
494: if( isnan(conductivity) ) % -- try an old version
495: conductivity = ...
496: getparam( fid, '90_sec_conductivity_siemens', 'n' ) ;
497:
498: if( isnan(conductivity) ) % -- or yet an even older version
499: conductivity = ...
500: getparam( fid, '60_sec_conductivity_siemens', 'n' ) ;
501: end
502: end
503:
504: % -----
505: % -- Parse out the RTC tank data. Note that in BALLAST we will
506: % only use the "raw" (i.e. unprocessed) frequency data, and
507: % then apply the best/newest calibration coefficients. The
508: % current sequence of data variables is :
509: %
510: % (1) P tank Hz (2) P tank psi (3) weight_beam_Hz
511: % (4) weight_beam (5) Tbot tank Hz (6) Tbot tank
512: % (7) Ttop_tank_Hz (8) Ttop_tank (9) Tabe_tank
513: %
514: % NOTE: Although these names differ slightly from those given
515: % in the "# data_sequence" header in the RTC data files, their
516: % meanings are the same. Also, although the raw pressure,
517: % weight, and temperature frequencies have already been
518: % processed with calibration coefficients stored in the RTC
519: % software, whenever possible the frequencies will be
520: % reprocessed in this program with better calibration
```

ballast.m

```

521: % coefficients specified by the user.
522: % -----
523:
524: [ type, date, time, data ] = getballast(fid) ;
525:
526: % -- use only the "norm" measurements for ballasting
527:
528: knorm = find( type == 0 ) ;
529: % kauto = find( type == 1 ) ;
530: % ksamp = find( type == 2 ) ;
531:
532: % datanorm = data(knorm,:) ;
533:
534: % -- get pressure and (if needed) apply calibrations
535:
536: if( P_tank_cal )
537:     P_tank_Hz = DAO_frequency_ratio * data(knorm,1) ;
538:     P_tank_psi = polyval( P_tank_coeff, P_tank_Hz ) ;
539: else
540:     P_tank_psi = data(knorm,2) ; % -- use RTC calibrations
541: end
542:
543: % -- get bottom temperature and (if needed) apply calibrations
544:
545: if( Tbot_tank_cal )
546:     Tbot_tank_Hz = DAO_frequency_ratio * data(knorm,5) ;
547:     % -- Hz to R
548:     Rbot_tank = polyval( Tbot_tank_coeff, Tbot_tank_Hz ) ;
549:     % -- R to T
550:     Tbot_tank = polyval( Rbot_tank ) ;
551: else
552:     Tbot_tank = data(knorm,6) ; % -- use RTC calibrations
553: end
554:
555: % -- get top (beam) temperature and (if needed) apply
556: % calibrations
557:
558: if( Ttop_tank_cal )
559:     Ttop_tank_Hz = DAO_frequency_ratio * data(knorm,7) ;
560:     % -- Hz to R
561:     Rtop_tank = polyval( Ttop_tank_coeff, Ttop_tank_Hz ) ;
562:     % -- R to T
563:     Ttop_tank = polyval( Rtop_tank ) ;
564: else
565:     Ttop_tank = data(knorm,8) ; % -- use RTC calibrations
566: end
567:
568: % -- get Sea-Bird temperature (no calibrations required)
569:
570: Tsbe_tank = data(knorm,9) ; % -- use default calibrations
571:
572: % -- get weight from strain beam and (if needed) apply
573: % calibrations
574:
575: if( beam_press_cal || beam_weight_cal )
576:
577:     weight_beam_Hz = DAO_frequency_ratio * data(knorm,3) ;
578:
579:     % -- correct the frequency for affects due to pressure
580:
581:     if( beam_press_cal )
582:         beam_press_corr = polyval( beam_press_coeff, P_tank_psi ) ;
583:     else
584:         beam_press_corr = zeros( size(weight_beam_Hz) ) ;
585:     end

```

```

586: % -- correct the frequency for affects due to temperature
587:
588: if( beam_temp_cal )
589:     beam_temp_corr = polyval( beam_temp_coeff, Ttop_tank ) ;
590: else
591:     beam_temp_corr = zeros( size(weight_beam_Hz) ) ;
592: end
593:
594: % -- finally, apply calibrations to the strain beam
595: % measurements
596:
597: weight_beam_Hz = weight_beam_Hz - beam_press_corr ...
598:
599: weight_beam = polyval( beam_weight_coeff, ...
600:     weight_beam_Hz ) ;
601:
602: else
603:     weight_beam = data(knorm,4) ;
604: end
605:
606: % -----
607: % -- handle multiple samples at single pressure
608:
609: npsteps = length(pressure_steps) ;
610: nval = 1 ;
611:
612: P_tmp = NaN*ones(size(pressure_steps)) ;
613: beam_tmp = NaN*ones(size(pressure_steps)) ;
614: Ttop_tmp = NaN*ones(size(pressure_steps)) ;
615: Tbot_tmp = NaN*ones(size(pressure_steps)) ;
616: Tsbe_tmp = NaN*ones(size(pressure_steps)) ;
617:
618: P_tank_psi = [ P_tank_psi, -999 ]' ;
619: weight_beam = [ weight_beam, NaN ]' ;
620: Ttop_tank = [ Ttop_tank, NaN ]' ;
621: Tbot_tank = [ Tbot_tank, NaN ]' ;
622: Tsbe_tank = [ Tsbe_tank, NaN ]' ;
623:
624: for ncnt = 1:npsteps
625:
626:     step_done = 0 ;
627:
628:     P_lew = [] ;
629:     beam_lew = [] ;
630:     Ttop_lew = [] ;
631:     Tbot_lew = [] ;
632:     Tsbe_lew = [] ;
633:
634:     % -- extract the values for this step
635:     while( ~step_done )
636:
637:         if( abs( P_tank_psi(nval) - pressure_steps(ncnt) ) ...
638:             < delta_pstep )
639:
640:             % -- this is on the same level
641:
642:             P_lew = [ P_lew, P_tank_psi(nval) ] ;
643:             beam_lew = [ beam_lew, weight_beam(nval) ] ;
644:             Ttop_lew = [ Ttop_lew, Ttop_tank(nval) ] ;
645:             Tbot_lew = [ Tbot_lew, Tbot_tank(nval) ] ;
646:             Tsbe_lew = [ Tsbe_lew, Tsbe_tank(nval) ] ;
647:
648:             nval = nval + 1 ;
649:
650:

```

ballast.m

```

651: else
652:
653: % -- we have finished our level, now use specified
654: % "multi-option" to determine how to select one value
655: % from this level.
656:
657: step_done = 1 ;
658:
659: if ( isempty(P_lev) )
660:
661: % -- data exists in this pressure_step
662:
663: if( multi_option == 0 )
664:
665: % -- pick the first value
666:
667: P_tmp(ncnt) = P_lev(1) ;
668: beam_tmp(ncnt) = beam_lev(1) ;
669: Ttop_tmp(ncnt) = Ttop_lev(1) ;
670: Tbot_tmp(ncnt) = Tbot_lev(1) ;
671: Tsbe_tmp(ncnt) = Tsbe_lev(1) ;
672:
673: elseif( multi_option == 1 )
674:
675: % -- pick the first value
676:
677: nlast = length(P_lev) ;
678:
679: P_tmp(ncnt) = P_lev(nlast) ;
680: beam_tmp(ncnt) = beam_lev(nlast) ;
681: Ttop_tmp(ncnt) = Ttop_lev(nlast) ;
682: Tbot_tmp(ncnt) = Tbot_lev(nlast) ;
683: Tsbe_tmp(ncnt) = Tsbe_lev(nlast) ;
684:
685: elseif( multi_option == 2 )
686:
687: % -- pick the average
688:
689: P_tmp(ncnt) = mean(P_lev) ;
690: beam_tmp(ncnt) = mean(beam_lev) ;
691: Ttop_tmp(ncnt) = mean(Ttop_lev) ;
692: Tbot_tmp(ncnt) = mean(Tbot_lev) ;
693: Tsbe_tmp(ncnt) = mean(Tsbe_lev) ;
694:
695: end
696:
697: end % --- end of data exists in this pressure step
698:
699: end
700:
701: end % -- end of gathering data for this step while loop
702: end % -- end of multiple sample loop
703:
704: % -- remove all NaNs, and return data arrays to their original
705: % names
706:
707: good = find( ~isnan(P_tmp) ) ;
708:
709: P_tank_psi = P_tmp(good) ;
710: weight_beam = beam_tmp(good) ;
711: Ttop_tank = Ttop_tmp(good) ;
712: Tbot_tank = Tbot_tmp(good) ;
713: Tsbe_tank = Tsbe_tmp(good) ;
714:
715: % -- convert pressure from psi to dbars
716:
717: P_tank = psi2dbars * P_tank_psi ;
718: P_direction = diff(P_tank) ;
719:
720: % -----
721: % -- initialize figure
722:
723: figure(1)
724: orient landscape
725: clf
726:
727: HA = 'HorizontalAlignment' ;
728: VA = 'VerticalAlignment' ;
729: IP = 'Interpreter' ;
730: FS = 'FontSize' ;
731: FN = 'FontName' ;
732: MS = 'MarkerSize' ;
733: LW = 'LineWidth' ;
734:
735: % -----
736: % -- plot strain weight (grams) vs. pressure (dbars)
737:
738: axes( 'Position', [ 0.20 0.20 0.65 0.70 ] )
739:
740: hold on
741:
742: for k=1:length(P_direction)
743:
744: if( P_direction(k) > 0 ) % -- pressure increasing - red
745:
746: plot( [ P_tank(k) P_tank(k+1) ], ...
747: [ weight_beam(k) weight_beam(k+1) ], 'r' )
748:
749: plot( P_tank(k), weight_beam(k), 'ro', MS, 4 )
750: plot( P_tank(k+1), weight_beam(k+1), 'ro', MS, 4 )
751:
752: elseif( P_direction(k) < 0 ) % -- pressure decreasing - blue
753:
754: plot( [ P_tank(k) P_tank(k+1) ], ...
755: [ weight_beam(k) weight_beam(k+1) ], 'b' )
756:
757: plot( P_tank(k), weight_beam(k), 'bo', MS, 12 )
758: plot( P_tank(k+1), weight_beam(k+1), 'bo', MS, 12 )
759:
760: end
761: end
762:
763: % -- set plot limits
764:
765: min_press = 0 ;
766: max_press = tick_press * ceil( max(P_tank) / tick_press ) ;
767: min_weight = tick_weight * ... floor( min(weight_beam) / tick_weight ) ;
768:
769: max_weight = tick_weight * ...
770: cell( max(weight_beam) / tick_weight ) ;
771:
772: % xrange = max_press - min_press ;
773: % yrange = max_weight - min_weight ;
774:
775: set( gca, 'YLim', [ min_weight max_weight ] )
776: set( gca, 'XLim', [ min_press max_press ] ) ;
777:
778: ylabel( 'Strain Beam Weight (grams)', FS, 12, VA, 'bottom' ) ;
779: xlabel( 'Tank Pressure (dbars)', FS, 12, VA, 'top' ) ;
780:

```

10/26/06
15:18:24

ballast.m

```
781: % ytl = max_weight + 0.09*( max_weight - min_weight ) ;
782: % xt = max_press + 0.50*( max_press - min_press ) ;
783:
784: title( sprintf( [ 'Edit plot : Float "%s" \n Project "%s"', ' , ...
785: 'Time "%s"', ], float_name, project, ...
786: 'start_time', FS, 12 ) )
787:
788: grid on
789: set( gca, 'Box', 'on' )
790:
791: % -----
792: % -- options for editing out bad data
793:
794: % -- first, mark data according to whether the pressure falls in
795: % (1, good) or out (0, bad) of the specified range
796:
797: data_status = ( P_tank >= Pmin_range & P_tank <= Pmax_range ) ;
798:
799: if( ~isnan( omit_steps ) )
800:     data_status(omit_steps) = 0 ;
801: end
802:
803: P_tank_good = P_tank(logical(data_status)) ;
804: weight_beam_good = weight_beam(logical(data_status)) ;
805: ndata_good = length(weight_beam_good) ;
806:
807: if( mode == 1 )
808:
809:     % -- for MODE == 1, the user will manually select the points
810:     % to be removed before the ballast weight calculations.
811:
812:     edit_data = 1 ;
813:
814:     while( edit_data )
815:
816:         % -- initialize editing arrays
817:
818:         data_status = ( P_tank >= Pmin_range & ...
819:             P_tank <= Pmax_range ) ;
820:
821:         if( ~isnan( omit_steps ) )
822:             data_status(omit_steps) = 0 ;
823:         end
824:
825:         fprintf( '\n' ) ;
826:         response = query( [ 'Do you wish to edit the ', ...
827:             'ballast data [y/n]? ', ] ) ;
828:
829:         if( ~response )
830:
831:             edit_data = 0 ;
832:
833:         else
834:
835:             fprintf( '\n' ) ;
836:             fprintf( ' Click on the points to be discarded.\n' ) ;
837:             fprintf( ' When finished, press return.\n' ) ;
838:
839:             % -- select bad points with mouse
840:
841:             [ x_bad, y_bad ] = ginput ;
842:
843:             num_select = length(x_bad) ;
844:
845:             % -- since it is impossible to click EXACTLY on the
```

```
846: % measured data point, we will accept a click within
847: % a small rectangular window around the point
848:
849: j_index = 0 ; % -- index for edit handles
850: k_index = [] ;
851:
852: for k=1:num_select
853:
854:     idx = find( abs( P_tank - x_bad(k) ) < ...
855:         xedit_fuzz & ...
856:         abs( weight_beam - y_bad(k) ) < ...
857:         yedit_fuzz ) ;
858:
859:     if( isempty(idx) )
860:
861:         fprintf( [ 'warning: The point indicated ', ...
862:             'for editing,\n', ...
863:             ' (pressure = %6.1f dbar, ', ...
864:             'weight = %6.2f grams),\n', ...
865:             ' did not match any measured ', ...
866:             'data point.\n' ], ...
867:             x_bad(k), y_bad(k) ) ;
868:
869:     else
870:
871:         j_index = [ j_index idx ] ;
872:         k_index = [ k_index k*ones(size(idx)) ] ;
873:
874:         % -- there is a loop here in case multiple points
875:         % were within the selected window - the
876:         % reasoning is that all points "very close" to
877:         % a bad point are also bad.
878:
879:         for m=1:length(idx) % -- loop through all the
880:
881:             j = j + 1 ;
882:
883:             data_status(idx(m)) = 0 ;
884:
885:             edit_handle(2*j-1) = ...
886:                 plot( P_tank(idx(m)), ...
887:                     weight_beam(idx(m)), ...
888:                     'w.', MS, 12 ) ;
889:
890:             edit_handle(2*j) = ...
891:                 text( P_tank(idx(m)), ...
892:                     weight_beam(idx(m)), ...
893:                     'bad', FS, 12, HA, 'center' ) ;
894:
895:             end
896:
897:         end
898:
899:     end % -- end of select loop
900:
901:     % -- write out a summary listing of the selected points
902:     fprintf( '\n' ) ;
903:     fprintf( [ ' The following %d points were ', ...
904:         length(k_index), num_select ] ;
905:
906:     fprintf( [ ' click point', ...
907:         ' click actual\n' ] ) ;
908:
909:     for k=1:length(k_index)
910:         fprintf( [ ' %2d ', ...
```

ballast.m

```

911:         '%7.1f %7.1f dbar, ', ...
912:         '%7.2f %7.2f gms\n' ], ...
913:         k_index(k), k, ...
914:         x_bad(k_index(k)), P_tank(j_index(k)), ...
915:         y_bad(k_index(k)), weight_beam(j_index(k)) );
916:     end
917:
918: fprintf( '\n' );
919: response = query( ['Do you accept this selection ', ...
920:                   'of data points? [y/n]? ' ] );
921:
922: if( response )
923:
924:     % -- accept the editing, set data variables and
925:     % arrays to new values for the next processing
926:     % step
927:
928:     P_tank_good = P_tank(logical(data_status));
929:     weight_beam_good = ...
930:         weight_beam(logical(data_status));
931:
932:     ndata_good = length(weight_beam_good);
933:     edit_data = 0;
934:
935:     else
936:
937:     % -- before cycling through this section again, set
938:     % data plot back to original state
939:
940:     delete( edit_handle )
941:
942:     end
943:
944:     end % -- end of this cycle of user editing
945:
946: end
947:
948: elseif( mode == 0 )
949:
950:     % -- if MODE == 0, an automated editing of ballast data is
951:     % performed to remove all points that exceed
952:     % "stddev factor" number of standard deviations from the
953:     % fitted line.
954:
955:     notdone = 1;
956:
957:     while( notdone )
958:
959:         P_tank_good = P_tank(logical(data_status));
960:         weight_beam_good = weight_beam(logical(data_status));
961:
962:         ndata_good = length(weight_beam_good);
963:
964:         reg_coeff = polyfit( P_tank_good, weight_beam_good, ...
965:                               norder );
966:         weight_diff = weight_beam_good -
967:             polyval( reg_coeff, P_tank_good );
968:
969:         stddev = std( weight_diff );
970:         num_select = find( abs( weight_diff ) >
971:                           stddev_factor*stddev );
972:
973:         if( isempty(num_select) )
974:             data_status(num_select) = 0;
975:         else
976:             notdone = 'notdone';
977:         end
978:     end
979:
980: end
981:
982: % -----
983: % -- perform a least-squared fit to the data points
984:
985: reg_coeff = polyfit( P_tank_good, weight_beam_good, norder);
986: press_fit = 0 : max_press;
987: weight_fit = polyval( reg_coeff, press_fit );
988: weight_diff = weight_beam_good - ...
989:     polyval( reg_coeff, P_tank_good );
990:
991: Y_intercept = reg_coeff( norder + 1 );
992: ballast_slope = reg_coeff( norder );
993:
994: % -- plot the fitted function
995:
996: plot( press_fit, weight_fit, 'k' );
997:
998: % -- calculate the error statistics of the fit using section 14.2
999: % of Press et al., "Numerical Recipes", 1986.
1000:
1001: Sx = sum( P_tank_good );
1002: Sxx = sum( P_tank_good .* P_tank_good );
1003: delta = ndata_good*Sxx - Sx*Sx;
1004:
1005: chi_sq = sum( weight_diff .* weight_diff );
1006: chi_fact = sqrt( chi_sq / ( ndata_good - 2 ) );
1007: err_slope = chi_fact * sqrt( ndata_good / delta );
1008: err_intercept = chi_fact * sqrt( Sxx / delta );
1009:
1010: rms_fit = sqrt( chi_sq / ndata_good );
1011:
1012: % -----
1013: % -- option for modifying the target water properties
1014:
1015: sigma_target = sw_dens( S_target, T_target, P_target ) - 1000;
1016: sigmaT_target = sw_dens( S_target, T_target, 0.0 ) - 1000;
1017:
1018: if( mode == 1 )
1019:
1020:     change_target = 1;
1021:
1022:     while( change_target )
1023:
1024:         fprintf( '\n' );
1025:         fprintf( ['ballast: Target water properties ', ...
1026:                   'are currently: \n' ] );
1027:
1028:         fprintf( ' : S = %6.3f psu \n', S_target );
1029:         fprintf( ' : T = %6.3f degC \n', T_target );
1030:         fprintf( ' : P = %6.1f dbar \n', P_target );
1031:         fprintf( ' : sigma(S,T,P) = %6.3f \n', ...
1032:                 sigma_target );
1033:         fprintf( ' : sigma(S,T,0) = %6.3f \n', ...
1034:                 sigmaT_target );
1035:
1036:         response = query( [ 'Do you wish to modify these ', ...
1037:                             'target properties [y/n]? ' ] );
1038:
1039:         if( ~response )
1040:

```

10/26/06
15:18:24

ballast.m

```
1041: change_target = 0 ;
1042:
1043: else
1044:     fprintf( [ 'ballast: Please enter the new ', ...
1045:               'property values. \n' ] ) ;
1046:
1047: S_target = obtain( ...
1048:     '          ', new target Salinity    [psu] : ', ...
1049:     '          ', S_target_min, S_target_max ) ;
1050:
1051: T_target = obtain( ...
1052:     '          ', new target Temperature [degC] : ', ...
1053:     '          ', T_target_min, T_target_max ) ;
1054:
1055: P_target = obtain( ...
1056:     '          ', new target Pressure    [dbars] : ', ...
1057:     '          ', P_target_min, P_target_max ) ;
1058:
1059: sigma_target = sw_dens( S_target, T_target, ...
1060:     P_target ) - 1000 ;
1061:
1062: sigmaT_target = sw_dens( S_target, T_target, ...
1063:     0.0 ) - 1000 ;
1064:
1065: end
1066:
1067: % -- target density
1068: density_target = sw_dens( S_target, T_target, P_target ) / 1000 ;
1069:
1070: % -- average tank temperature
1071: Ttop_mean = mean( Ttop_tank ) ;
1072: Tbot_mean = mean( Tbot_tank ) ;
1073:
1074: Ttop_mean = mean( Ttop_tank ) ;
1075: Tbot_mean = mean( Tbot_tank ) ;
1076:
1077: Ttop_diff = Ttop_mean - Tsbe_mean ;
1078: Tbot_diff = Tsbe_mean - Tbot_mean ;
1079:
1080: Ttop_warning = 0 ;
1081: Tbot_warning = 0 ;
1082:
1083: if( abs( Ttop_diff ) > temp_diff_warning )
1084:     Ttop_warning = 1 ;
1085:
1086: end
1087:
1088: if( abs( Tbot_diff ) > temp_diff_warning )
1089:     Tbot_warning = 1 ;
1090:
1091: end
1092:
1093: % -- tank salinity
1094: S_tank = 0.00 ;
1095:
1096: % -- density in tank [grams/cc] at target pressure
1097: density_target_tank = sw_dens( S_tank, Tsbe_mean, P_target ) ...
1098: / 1000 ;
1099:
1100: sigmaT_tank = sw_dens( S_tank, Tsbe_mean, 0 ) - 1000 ;
1101:
1102: % -- total dry weight of float, drop weight and/or compressesee,
1103: % miscellaneous additional weights, as well as the ballage
1104: % and trim weights. Note that this sum is not the weight to
1105: % -----
1106: % achieve neutral buoyancy, since the strain beam should be
1107: % recording this total as about 40-to-60 grams negative
1108: % buoyancy.
1109:
1110: float_total_dry = float_dry + drop_dry + comp_dry + misc_dry ;
1111:
1112: % -- total weight added in tank
1113:
1114: trim_wet2dry = trim_density / ...
1115:     ( trim_density - density_target_tank ) ;
1116:
1117: trim_dry2wet = ( trim_density - density_target_tank ) / ...
1118:     trim_density ;
1119:
1120: trim_total_dry = ball_cage_dry + trim_dry ;
1121:
1122: trim_total_wet = ball_cage_wet + ...
1123:     ( trim_dry2wet * trim_dry ) ;
1124:
1125: % -- total surplus weight added in tank beyond neutral buoyancy
1126:
1127: % trim_surplus_wet = Y_intercept ;
1128:
1129: trim_surplus_target_wet = Y_intercept + ...
1130:     ballast_slope * P_target ;
1131:
1132: % Can now calculate float volume at target pressure
1133:
1134: volume_float = ( float_total_dry + trim_total_wet ...
1135:     - trim_surplus_target_wet ) / ...
1136:     density_target_tank ;
1137:
1138: % -----
1139: % -- determine the compressibilities of the target, the tank
1140: % water at target pressure, and the float at target pressure.
1141: % a simple centered difference is used.
1142:
1143: kappa = ( 1 / vol ) * ( d_vol / d_press )
1144:
1145: or, using vol = mass/rho,
1146:
1147: = - ( 1 / rho ) * ( d_rho / d_press )
1148:
1149: % -----
1150:
1151: kappa_target = - ( sw_dens( S_target, T_target, ...
1152:     P_target + d_press ) - ...
1153:     sw_dens( S_target, T_target, ...
1154:     P_target - d_press ) ) / ...
1155:     ( 1000 * 2 * density_target * d_press ) ;
1156:
1157: kappa_tank = - ( sw_dens( S_tank, Tsbe_mean, ...
1158:     P_target + d_press ) - ...
1159:     sw_dens( S_tank, Tsbe_mean, ...
1160:     P_target - d_press ) ) / ...
1161:     ( 1000 * 2 * density_target_tank * d_press ) ;
1162:
1163: kappa_float = kappa_tank - ( ballast_slope / ...
1164:     ( density_target_tank * volume_float ) ) ;
1165:
1166: kappa_error = err_slope / ...
1167:     ( density_target_tank * volume_float ) ;
1168:
1169: % -----
1170: % -----
```

10/26/06
15:18:24

ballast.m

10

```
1171: % -- compute the additional weight needed for the float to
1172: % achieve neutral buoyancy in the specified target waters due
1173: % to ...
1174: % -----
1175: %
1176: % -- (1) trim weights AND compressibility to target pressure
1177: %
1178: weight_target = Y_intercept + ballast_slope * P_target ;
1179: neutral_weight = trim_total_wet - weight_target ;
1180: %
1181: % -- (2) thermal expansion of the float due to differences
1182: % between the tank and target temperatures.
1183: %
1184: % NOTE: the coefficient of thermal expansion for the
1185: % float can be determined through two methods: (1) derive
1186: % the value theoretically based on handbook values and the
1187: % composition of the float, or (2) derive the value
1188: % experimentally by ballasting the float at two (or more)
1189: % different temperatures and computing the changes in the
1190: % float volume as a function of temperature.)
1191: %
1192: thermal_correction = volume_float * float_thermal_coeff * ...
1193: % ( T_target - Tsbe_mean ) * density_target ;
1194: %
1195: % -- (3) differences between the tank and target densities (this
1196: % is only salinity and temperature, since the pressure has
1197: % been accounted for.
1198: %
1199: seawater_correction = volume_float * ...
1200: % ( density_target - density_target_tank ) ;
1201: %
1202: % -- total ballast correction
1203: %
1204: total_correction = seawater_correction + thermal_correction ...
1205: %
1206: %
1207: % -- the weight require to achieve neutral buoyancy at the target
1208: % water can be distributed in two primary ways: internal or
1209: % external
1210: %
1211: if( ballast_flag == 1 ) % -- external
1212: %
1213: ballast_factor = ballast_density / ...
1214: % ( ballast_density - density_target ) ;
1215: ballast_addition = ballast_factor * total_correction ;
1216: %
1217: elseif( ballast_flag == 0 ) % -- internal
1218: %
1219: ballast_addition = total_correction ;
1220: end
1221: %
1222: % -----
1223: % -- plot a summary page of all the ballast data and results
1224: %
1225: % -----
1226: %
1227: %
1228: clf
1229: figure(1)
1230: orient landscape
1231: clf
1232: %
1233: % -----
1234: % -- panel 1a - text - ballast data
1235: %
```

```
1236: axes( 'Position', [ 0.035 0.35 0.15 0.55 ] )
1237: %
1238: xt = 0.00 ;
1239: yt = 0.99 ;
1240: dy = 0.024 ;
1241: %
1242: text( xt, yt, ' Ballast Data', HA, 'left', FS, 12 ) ;
1243: yt = yt - 1.5*dy ;
1244: text( xt, yt, sprintf( ' Beam ID = "%s"', beam_number ), ...
1245: % HA, 'left', FS, 8, FN, 'courier' ) ;
1246: yt = yt - 1.5*dy ;
1247: %
1248: xt = xt - 0.15 ;
1249: %
1250: text( xt, yt, ' k Press Press Weight', ...
1251: % HA, 'left', FS, 8, FN, 'courier' ) ;
1252: yt = yt - dy ;
1253: text( xt, yt, ' (psi) (dbars) (grams)', ...
1254: % HA, 'left', FS, 8, FN, 'courier' ) ;
1255: yt = yt - 2*dy ;
1256: %
1257: for k=1:length(P_tank)
1258: % text( xt, yt, sprintf( '%2d %6.1f %6.2f', ...
1259: % k, P_tank(k)/psi2dbars, P_tank(k), ...
1260: % weight_beam(k) ), ...
1261: % HA, 'left', FS, 8, FN, 'courier' ) ;
1262: if( ~data_status(k) )
1263: % text( xt, yt, ' x', ...
1264: % HA, 'left', FS, 8, FN, 'courier' ) ;
1265: end
1266: %
1267: yt = yt - dy ;
1268: %
1269: end
1270: %
1271: % -----
1272: %
1273: axis( [ 0 1 0 1 ] )
1274: axis( 'off' )
1275: %
1276: % -----
1277: % -- panel 1b - text - parameter listing
1278: %
1279: axes( 'Position', [ 0.035 0.07 0.15 0.25 ] )
1280: %
1281: xt = 0.00 ;
1282: yt = 0.99 ;
1283: dy = 0.060 ;
1284: %
1285: text( xt, yt, sprintf( 'Multi Option = %d', multi_option), ...
1286: % yt = yt - dy ;
1287: %
1288: %
1289: text( xt, yt, sprintf( 'Pressure Range:' ), ...
1290: % HA, 'left', FS, 8, FN, 'courier' ) ;
1291: yt = yt - dy ;
1292: %
1293: text( xt, yt, sprintf( ' %4.0f to %4.0f psi', ...
1294: % Pmin_psi_range, Pmax_psi_range ), ...
1295: % HA, 'left', FS, 8, FN, 'courier' ) ;
1296: yt = yt - dy ;
1297: %
1298: text( xt, yt, sprintf( ' %4.0f to %4.0f dbar', ...
1299: % Pmin_range, Pmax_range ), ...
1300: % HA, 'left', FS, 8, FN, 'courier' ) ;
```

```

1301:      yt = yt - 2*dy ;
1302:
1303: % -- calibration coefficients
1304: text( xt, yt, 'Calibration Coeff', HA, 'left', FS, 10 ) ;
1305: yt = yt - 1.5*dy ;
1306:
1307: % -- tank pressure
1308:
1309: text( xt, yt, sprintf( 'Tank Pressure' ), ...
1310:      HA, 'left', FS, 8, FN, 'courier' ) ;
1311:
1312: yt = yt - dy ;
1313:
1314: if( P_tank_cal )
1315:     ns = 1 ;
1316:     n = findstr( file_P_tank_coeff, '/' ) ;
1317:     if( isempty(n) ) ; ns = max(n) + 1 ; end
1318:
1319:     text( xt, yt, sprintf( ' %s',
1320:         file_P_tank_coeff(ns:ne) ), ...
1321:         HA, 'left', FS, 8, FN, 'courier', IP, 'none' ) ;
1322: else
1323:     text( xt, yt, sprintf( ' : RTC values' ), ...
1324:         HA, 'left', FS, 8, FN, 'courier' ) ;
1325: end
1326:
1327: yt = yt - 2*dy ;
1328:
1329: % -- tank temperature
1330:
1331: text( xt, yt, sprintf( 'Tank Temperature' ), ...
1332:      HA, 'left', FS, 8, FN, 'courier' ) ;
1333:
1334: yt = yt - dy ;
1335:
1336: if( T_top_tank_cal )
1337:     ns = 1 ;
1338:     n = findstr( file_Ttop_tank_coeff, '/' ) ;
1339:     if( isempty(n) ) ; ns = max(n) + 1 ; end
1340:
1341:     text( xt, yt, sprintf( ' %s',
1342:         file_Ttop_tank_coeff(ns:ne) ), ...
1343:         HA, 'left', FS, 8, FN, 'courier', IP, 'none' ) ;
1344: else
1345:     text( xt, yt, sprintf( ' top : RTC values' ), ...
1346:         HA, 'left', FS, 8, FN, 'courier' ) ;
1347: end
1348:
1349: yt = yt - dy ;
1350:
1351: if( Tbot_tank_cal )
1352:     ns = 1 ;
1353:     n = findstr( file_Tbot_tank_coeff, '/' ) ;
1354:     if( isempty(n) ) ; ns = max(n) + 1 ; end
1355:
1356:     text( xt, yt, sprintf( ' %s',
1357:         file_Tbot_tank_coeff(ns:ne) ), ...
1358:         HA, 'left', FS, 8, FN, 'courier', IP, 'none' ) ;
1359: else
1360:     text( xt, yt, sprintf( ' bottom : RTC values' ), ...
1361:         HA, 'left', FS, 8, FN, 'courier' ) ;
1362: end
1363:
1364: yt = yt - 2*dy ;
1365:
1366: % -- strain beam
1367:
1368: text( xt, yt, sprintf( 'Strain Beam' ), ...
1369:      HA, 'left', FS, 8, FN, 'courier' ) ;
1370:
1371: yt = yt - dy ;
1372:
1373: if( beam_press_cal )
1374:     ns = 1 ;
1375:     n = findstr( file_beam_press_coeff, '/' ) ;
1376:     if( isempty(n) ) ; ns = max(n) + 1 ; end
1377:
1378:     text( xt, yt, sprintf( ' %s',
1379:         file_beam_press_coeff(ns:ne) ), ...
1380:         HA, 'left', FS, 8, FN, 'courier', IP, 'none' ) ;
1381: else
1382:     text( xt, yt, sprintf( ' press : RTC values' ), ...
1383:         HA, 'left', FS, 8, FN, 'courier' ) ;
1384: end
1385:
1386: yt = yt - dy ;
1387:
1388: if( beam_temp_cal )
1389:     ns = 1 ;
1390:     n = findstr( file_beam_temp_coeff, '/' ) ;
1391:     if( isempty(n) ) ; ns = max(n) + 1 ; end
1392:
1393:     text( xt, yt, sprintf( ' %s',
1394:         file_beam_temp_coeff(ns:ne) ), ...
1395:         HA, 'left', FS, 8, FN, 'courier', IP, 'none' ) ;
1396: else
1397:     text( xt, yt, sprintf( ' temp : RTC values' ), ...
1398:         HA, 'left', FS, 8, FN, 'courier' ) ;
1399: end
1400:
1401: yt = yt - dy ;
1402:
1403: if( beam_weight_cal )
1404:     ns = 1 ;
1405:     n = findstr( file_beam_weight_coeff, '/' ) ;
1406:     if( isempty(n) ) ; ns = max(n) + 1 ; end
1407:
1408:     text( xt, yt, sprintf( ' %s',
1409:         file_beam_weight_coeff(ns:ne) ), ...
1410:         HA, 'left', FS, 8, FN, 'courier', IP, 'none' ) ;
1411: else
1412:     text( xt, yt, sprintf( ' weight : RTC values' ), ...
1413:         HA, 'left', FS, 8, FN, 'courier' ) ;
1414: end
1415:
1416: % -----
1417:
1418: axis( [ 0 0 1 ] )
1419: axis( [ 0 0 1 ] )
1420: axis( 'off' )
1421:
1422: % -----
1423: % -- panel 2 - plot - ballast data and line fit
1424:
1425: axes( 'Position', [ 0.25 0.21 0.40 0.60 ] )
1426:
1427: hold on
1428:
1429: plot( press_fit, weight_fit, 'Color', 0.65*[ 1 1 1 ], LW, 2 ) ;
1430:

```


10/26/06
15:18:24

12

ballast.m

```
1431: % --- plot target compressibility
1432:
1433: weight_fit_target = weight_target + ...
1434: density_target_tank * ...
1435: volume_float * ...
1436: ( kappa_tank - kappa_target ) * ...
1437: ( press_fit - P_target ) ;
1438:
1439: plot( press_fit, weight_fit_target, '-.-', ...
1440:       'Color', 0.65*[ 1 1 1 ], LW, 2 ) ;
1441:
1442: % -- plot data points and lines
1443:
1444: for k=1:length(P_direction)
1445:
1446:     if ( P_direction(k) > 0 ) % -- pressure increasing - red
1447:
1448:         plot( [ P_tank(k) P_tank(k+1) ], 'r', LW, 0.2 )
1449:         [ weight_beam(k) weight_beam(k+1) ], 'r', LW, 0.2 )
1450:
1451:         plot( P_tank(k), weight_beam(k), 'ro', MS, 4 )
1452:         plot( P_tank(k+1), weight_beam(k+1), 'ro', MS, 4 )
1453:
1454:     elseif ( P_direction(k) < 0 ) % -- pressure decreasing - blue
1455:
1456:         plot( [ P_tank(k) P_tank(k+1) ], 'b', LW, 0.2 )
1457:         [ weight_beam(k) weight_beam(k+1) ], 'b', LW, 0.2 )
1458:
1459:         plot( P_tank(k), weight_beam(k), 'bo', MS, 12 )
1460:         plot( P_tank(k+1), weight_beam(k+1), 'bo', MS, 12 )
1461:     end
1462:
1463: end
1464:
1465: % -- mask bad data
1466:
1467: for k=1:length(P_tank)
1468:
1469:     if ( ~data_status(k) )
1470:         plot( P_tank(k), weight_beam(k), 'w.', MS, 12 ) ;
1471:         text( P_tank(k), weight_beam(k), 'x', FS, 12, ...
1472:              HA, 'center' ) ;
1473:     end
1474:
1475: end
1476:
1477: % -- target pressure and weight
1478:
1479: plot( [ P_target P_target ], [ y_tank_min y_tank_max ], 'k' ) ;
1480: plot( [ x_tank_min x_tank_max ], ...
1481:       [ weight_target weight_target ], 'k' ) ;
1482:
1483: % -- set plot limits
1484:
1485: min_press = 0 ;
1486: max_press = tick_press * ceil( max(P_tank) / tick_press ) ;
1487:
1488: min_weight = tick_weight * floor( ...
1489:               min( [ weight_target min(weight_beam) ] ) ...
1490:               / tick_weight ) ;
1491: weight_range = max(weight_beam) - min_weight ;
1492: max_weight = max(weight_beam) + 0.20*weight_range ;
1493: max_weight = max( [ max_weight, weight_fit(1), ...
1494:                   weight_target ] ) ;
1495: max_weight = tick_weight * ceil( max_weight / tick_weight ) ;
1496:
1497: set( gca, 'YLim', [ min_weight max_weight ] )
1498: set( gca, 'XLim', [ min_press max_press ] ) ;
1499:
1500: set( gca, 'XTick', ( min_press : tick_press : max_press ) )
1501: set( gca, 'YTick', ( min_weight : tick_weight : max_weight ) )
1502:
1503: labform( gca, 'Y', '$5.1f', 1 )
1504:
1505: ylabel( 'Strain Beam Weight (grams)', FS, 12, VA, 'bottom' ) ;
1506: xlabel( 'Tank Pressure (dbars)', FS, 12, VA, 'top' ) ;
1507:
1508: plot_time = [ datestr(now, 6), '/' , ...
1509:               datestr(now, 10), ' ', ...
1510:               datestr(now, 15) ] ;
1511:
1512: xt = min_press + 0.50*( min_press + max_press ) ;
1513: yt = max_weight + 0.14*( max_weight - min_weight ) ;
1514:
1515: text( xt, yt, ...
1516:       sprintf( 'Ballast Summary: Float %s : "%s"', ...
1517:               float_name, project ), ...
1518:       FS, 14, HA, 'center', IP, 'none' )
1519:
1520: title( sprintf( 'Data File : "%s" \n', ...
1521:                'Param File : "%s" \n', ...
1522:                'Data Time: %s -- Plot Time: %s', ], ...
1523:        file_tank, file_param, ...
1524:        start_time, plot_time ), ...
1525:        FS, 10, IP, 'none' )
1526:
1527: % -----
1528: % -- within figure legends
1529:
1530: % -- statistics for fitted line
1531:
1532: xfl = 0.30*max_press ;
1533: xfr = 0.98*max_press ;
1534: yfb = min_weight + 0.785*( max_weight - min_weight ) ;
1535: yft = min_weight + 0.985*( max_weight - min_weight ) ;
1536:
1537: fill( [ xfl xfr xfr xfl ], [ yfb yfb yft yft ], [ 1 1 1 ] ) ;
1538:
1539: xt = xfl + 0.030*max_press ;
1540: yt = yft - 0.035*( max_weight - min_weight ) ;
1541: dyt = 0.028*( max_weight - min_weight ) ;
1542:
1543: text( xt, yt, ...
1544:       sprintf( 'intercept = %6.2f gms \n', ...
1545:               y_intercept ), ...
1546:       HA, 'left', FS, 10, FN, 'courier' ) ;
1547:
1548: yt = yt - dyt ;
1549:
1550: text( xt, yt, ...
1551:       sprintf( 'slope = %9.5f gms/db \n', ...
1552:               ballast_slope ), ...
1553:       HA, 'left', FS, 10, FN, 'courier' ) ;
1554:
1555: yt = yt - dyt ;
1556:
1557: text( xt, yt, ...
1558:       sprintf( 'target weight = %6.2f gms \n', ...
1559:               weight_target ), ...
1560:       HA, 'left', FS, 10, FN, 'courier' ) ;
1561:
1562: yt = yt - 1.5*dyt ;
```

10/26/06
15:18:24

ballast.m

13

```
1561: text( xt, yt, ...
1562:   sprintf( 'fit rms'           = %6.2f gms \n',
1563:     rms_fit ),
1564:   HA, 'left', FS, 10, FN, 'courier' ) ;
1565:   yt = yt - dyt ;
1566:
1567: text( xt, yt, ...
1568:   sprintf( 'intercept error = %6.2f gms \n',
1569:     err_intercept ),
1570:   HA, 'left', FS, 10, FN, 'courier' ) ;
1571:   yt = yt - dyt ;
1572:
1573: text( xt, yt, ...
1574:   sprintf( 'slope error = %9.5f gms/db \n',
1575:     err_slope ),
1576:   HA, 'left', FS, 10, FN, 'courier' ) ;
1577:
1578: % -- plot symbols
1579:
1580: xfl = 0.02*max_press ;
1581: % xfr = 0.48*max_press ;
1582: xfr = 0.55*max_press ;
1583: yfb = min_weight + 0.080*( max_weight - min_weight ) ;
1584: yft = min_weight + 0.225*( max_weight - min_weight ) ;
1585:
1586: fill( [ xfl xfr xfl ], [ yfb yfb yft ], [ 1 1 1 ] ) ;
1587:
1588: xp = xfl + 0.035*max_press ;
1589: xpl = xfl + 0.015*max_press ;
1590: xpr = xfl + 0.055*max_press ;
1591: xt = xfl + 0.070*max_press ;
1592:
1593: yt = yft - 0.025*( max_weight - min_weight ) ;
1594: dyt = 0.028*( max_weight - min_weight ) ;
1595:
1596: plot( [ xpl xpr ], [ yt yt ], 'x', LW, 1 )
1597: plot( xp, yt, 'ro', 'MarkerSize', 6 )
1598: text( xt, yt, 'increasing pressure',
1599:   HA, 'left', FS, 10, FN, 'courier' ) ;
1600:
1601: yt = yt - dyt ;
1602:
1603: plot( [ xpl xpr ], [ yt yt ], 'b', LW, 1 )
1604: plot( xp, yt, 'b', 'MarkerSize', 18 )
1605: text( xt, yt, 'decreasing pressure',
1606:   HA, 'left', FS, 10, FN, 'courier' ) ;
1607:
1608: yt = yt - dyt ;
1609:
1610: plot( [ xpl xpr ], [ yt yt ], 'Color', 0.65*[ 1 1 1 ], LW, 2 ) ;
1611: text( xt, yt, 'straight line fit',
1612:   HA, 'left', FS, 10, FN, 'courier' ) ;
1613:
1614: yt = yt - dyt ;
1615: xpl1 = xfl + 0.015*max_press ;
1616: xpl2 = xfl + 0.028*max_press ;
1617: xpr1 = xfl + 0.042*max_press ;
1618: xpr2 = xfl + 0.055*max_press ;
1619:
1620: plot( [ xpl1 xpl2 ], [ yt yt ], 'Color', 0.85*[ 1 1 1 ], LW, 2 ) ;
1621: plot( [ xpr1 xpr2 ], [ yt yt ], 'Color', 0.65*[ 1 1 1 ], LW, 2 ) ;
1622: text( xt, yt, 'target compressibility',
1623:   HA, 'left', FS, 10, FN, 'courier' ) ;
1624:
1625: if( length( find( data_status == 0 ) ) > 0 )
1626: % -- some data was removed in editing
1627:
1628:   rms_fit = 0.020*max_press ;
1629:   xfl = 0.625*max_press ;
1630:   xfr = 0.625*max_press ;
1631:   yfb = min_weight + 0.015*( max_weight - min_weight ) ;
1632:   yft = min_weight + 0.065*( max_weight - min_weight ) ;
1633:   fill( [ xfl xfr xfl ], [ yfb yfb yft ], [ 1 1 1 ] ) ;
1634:
1635:   xt = xfl + 0.020*max_press ;
1636:   yt = yft - 0.025*( max_weight - min_weight ) ;
1637:
1638:   text( xt, yt, 'x" denotes data edited out',
1639:     HA, 'left', FS, 10, FN, 'courier' ) ;
1640:   ...
1641: end
1642:
1643: % -----
1644:
1645: grid on
1646: set( gca, 'Box', 'on' )
1647:
1648: % -----
1649: % -- panel 3 - text - float characteristics
1650:
1651: axes( 'Position', [ 0.68 0.20 0.30 0.70 ] )
1652:
1653: hold on
1654:
1655: xt = 0.00 ;
1656: yt = 0.99 ;
1657: dy = 0.026 ;
1658: dyp = 0.010 ;
1659:
1660: % -----
1661: % --- dry weights
1662: text( xt, yt, '   Dry Weights', ...
1663:   VA, 'base', HA, 'left', FS, 10 ) ;
1664:   yt = yt - dy ;
1665:
1666:
1667: yt = yt - dyp ; % -- partial spacer
1668:
1669: text( xt, yt, ...
1670:   sprintf( 'float'           = %8.2f grams',
1671:     float_dry ),
1672:   VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1673:   yt = yt - dy ;
1674:
1675: text( xt, yt, ...
1676:   sprintf( 'compressee'      = %8.2f grams',
1677:     comp_dry ),
1678:   VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1679:   yt = yt - dy ;
1680:
1681: text( xt, yt, ...
1682:   sprintf( 'drop weight'     = %8.2f grams',
1683:     drop_dry ),
1684:   VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1685:   yt = yt - dy ;
1686:
1687: text( xt, yt, ...
1688:   sprintf( 'miscellaneous'   = %8.2f grams',
1689:     misc_dry ),
1690:   VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
```

10/26/06
15:18:24

ballast.m

```
1691:      yt = yt - dy ;
1692:
1693:      yt = yt - dyp ;      % -- partial spacer
1694:
1695:      text( xt, yt, ...
1696:      sprintf( 'total float = %8.2f grams',
1697:      float_total_dry ), ...
1698:      VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1699:      yt = yt - dy ;
1700:
1701:      yt = yt - dyp ;      % -- partial spacer
1702:
1703:      text( xt, yt, ...
1704:      sprintf( 'ball cage = %8.2f grams',
1705:      ball_cage_dry ), ...
1706:      VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1707:      yt = yt - dy ;
1708:
1709:      text( xt, yt, ...
1710:      sprintf( 'balls (%d:%d) = %8.2f grams',
1711:      nballs(1:3), trim_dry ), ...
1712:      VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1713:      yt = yt - dy ;
1714:
1715:      text( xt, yt, ...
1716:      sprintf( 'trim density = %8.4f grams/cc',
1717:      trim_density ), ...
1718:      VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1719:      yt = yt - dy ;
1720:
1721:      text( xt, yt, ...
1722:      sprintf( 'trim (total) = %8.2f grams',
1723:      trim_total_dry ), ...
1724:      VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1725:      yt = yt - dy ;
1726:
1727:      text( xt, yt, ...
1728:      sprintf( 'trim (surplus) = %8.2f grams',
1729:      trim_surplus_target_dry ), ...
1730:      VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1731:      yt = yt - dy ;
1732:
1733:      % -----
1734:      % --- wet weights
1735:
1736:      yt = yt - dyp ;      % -- spacer
1737:
1738:      text( xt, yt, ...
1739:      sprintf( 'Wet Weights', ...
1740:      VA, 'base', HA, 'left', FS, 10 ) ;
1741:      yt = yt - dy ;
1742:      yt = yt - dyp ;      % -- partial spacer
1743:
1744:      text( xt, yt, ...
1745:      sprintf( 'ball cage = %8.2f grams',
1746:      ball_cage_wet ), ...
1747:      VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1748:      yt = yt - dy ;
1749:
1750:      text( xt, yt, ...
1751:      sprintf( 'balls (%d:%d) = %8.2f grams',
1752:      nballs(1:3), trim_wet ), ...
1753:      VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1754:      yt = yt - dy ;
1755:
1756:      text( xt, yt, ...
1757:      sprintf( 'trim (total) = %8.2f grams',
1758:      trim_total_wet ), ...
1759:      VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1760:      yt = yt - dy ;
1761:
1762:      text( xt, yt, ...
1763:      sprintf( 'trim (surplus) = %8.2f grams',
1764:      trim_surplus_target_wet ), ...
1765:      VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1766:      yt = yt - dy ;
1767:
1768:      % -----
1769:      % --- tank properties
1770:
1771:      yt = yt - dyp ;      % -- spacer
1772:
1773:      text( xt, yt, ...
1774:      sprintf( 'Tank Properties', ...
1775:      VA, 'base', HA, 'left', FS, 10 ) ;      yt = yt - dy ;
1776:      yt = yt - dyp ;      % -- partial spacer
1777:
1778:      if ( ~top_warning && ~tbot_warning )
1779:      text( xt, yt, ...
1780:      sprintf( 'temperature = %8.3f degC',
1781:      Tsbe_mean ), ...
1782:      VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1783:      yt = yt - dy ;
1784:      else
1785:      text( xt, yt, ...
1786:      sprintf( 'TEMPERATURE = %8.3f degC',
1787:      Tsbe_mean ), ...
1788:      VA, 'base', HA, 'left', FS, 10, ...
1789:      FN, 'courier', 'COLOR', [ 1 0 0 ] ) ;
1790:      yt = yt - dy ;
1791:      end
1792:
1793:      text( xt, yt, ...
1794:      sprintf( 'Tdiff top,bot = %4.2f, %4.2f degC',
1795:      Ttop_diff, Tbot_diff ), ...
1796:      VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1797:      yt = yt - dy ;
1798:
1799:      text( xt, yt, ...
1800:      sprintf( 'conductivity = %10.3e S',
1801:      conductivity ), ...
1802:      VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1803:      yt = yt - dy ;
1804:
1805:      text( xt, yt, ...
1806:      sprintf( 'sigma_t = %8.3f kg/m^3',
1807:      sigma_tank ), ...
1808:      VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1809:      yt = yt - dy ;
1810:
1811:      text( xt, yt, ...
1812:      sprintf( 'rho (target) = %8.5f grams/cc',
1813:      density_target_tank ), ...
1814:      VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1815:      yt = yt - dy ;
1816:
1817:      % -----
1818:      % --- target properties
1819:
1820:      yt = yt - dyp ;      % -- spacer
```

14

10/26/06
15:18:24

ballast.m

15

```
1821: text( xt, yt, ' Target Properties', ...
1822: VA, 'base', HA, 'left', FS, 10 );
1823: yt = yt - dy ;
1824:
1825: yt = yt - dyp ; % -- partial spacer
1826:
1827: text( xt, yt, ...
1828: sprintf( 'salinity = %8.3f psu ', ...
1829: S_target ), ...
1830: VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1831:
1832: yt = yt - dy ;
1833:
1834: text( xt, yt, ...
1835: sprintf( 'temperature = %8.3f degC ', ...
1836: T_target ), ...
1837: VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1838:
1839: yt = yt - dy ;
1840:
1841: text( xt, yt, ...
1842: sprintf( 'pressure = %8.1f dbar ', ...
1843: P_target ), ...
1844: VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1845:
1846: yt = yt - dy ;
1847:
1848: text( xt, yt, ...
1849: sprintf( 'sigma_t = %8.3f kg/m^3 ', ...
1850: sigma_target ), ...
1851: VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1852:
1853: yt = yt - dy ;
1854:
1855: text( xt, yt, ...
1856: sprintf( 'rho = %8.5f grams/cc ', ...
1857: density_target ), ...
1858: VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1859:
1860: yt = yt - dy ;
1861:
1862: text( xt, yt, ...
1863: sprintf( 'compressibilities', ...
1864: VA, 'base', HA, 'left', FS, 10 );
1865:
1866: yt = yt - dy ;
1867:
1868: yt = yt - dyp ; % -- partial spacer
1869:
1870: text( xt, yt, ...
1871: sprintf( 'float volume = %8.2f cc ', ...
1872: volume_float ), ...
1873: VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1874:
1875: yt = yt - dyp ; % -- partial spacer
1876:
1877: text( xt, yt, ...
1878: sprintf( 'target = %10.3e 1/dbars ', ...
1879: kappa_target ), ...
1880: VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1881:
1882: yt = yt - dy ;
1883:
1884: text( xt, yt, ...
1885: sprintf( 'tank = %10.3e 1/dbars ', ...
1886: kappa_tank ), ...
1886: VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1887:
1888: yt = yt - dy ;
1889:
1890: text( xt, yt, ...
1891: sprintf( 'float = %10.3e 1/dbars ', ...
1892: kappa_float ), ...
1893: VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1894:
1895: yt = yt - dy ;
1896:
1897: text( xt, yt, ...
1898: sprintf( 'float error = %6.3fe-06 1/dbars ', ...
1899: kappa_error ), ...
1900: VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1901:
1902: yt = yt - dy ;
1903:
1904: text( xt, yt, ...
1905: sprintf( 'ballast weight components', ...
1906: kappa_error / ( 1.0e-6 ) );
1907:
1908: yt = yt - dyp ; % -- partial spacer
1909:
1910: text( xt, yt, ...
1911: sprintf( 'for neutral = %8.2f grams ', ...
1912: neutral_weight ), ...
1913: VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1914:
1915: yt = yt - dy ;
1916:
1917: text( xt, yt, ...
1918: sprintf( 'for seawater = %8.2f grams ', ...
1919: seawater_correction ), ...
1920: VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1921:
1922: yt = yt - dy ;
1923:
1924: text( xt, yt, ...
1925: sprintf( 'thermal coeff = %10.3e 1/degC ', ...
1926: float_thermal_coeff ), ...
1927: VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1928:
1929: yt = yt - dy ;
1930:
1931: text( xt, yt, ...
1932: sprintf( 'thermal correction = %8.2f grams ', ...
1933: thermal_correction ), ...
1934: VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1935:
1936: yt = yt - dyp ; % -- partial spacer
1937:
1938: text( xt, yt, ...
1939: sprintf( 'total weight = %8.2f grams ', ...
1940: total_correction ), ...
1941: VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1942:
1943: yt = yt - dy ;
1944:
1945: if( ballast_flag == 1 ) % -- external ballast weights
1946:
1947: yt = yt - dyp ; % -- partial spacer
1948:
1949: text( xt, yt, ...
1950:
```

10/26/06
15:18:24

ballast.m

16

```
1951: sprintf( 'ballast weight density = %6.4f ', ...
1952: ballast_density ), ...
1953: VA, 'base', HA, 'left', FS, 10, FN, 'courier' ) ;
1954: yt = yt - dy ;
1955:
1956: text( xt, yt, ...
1957: sprintf( 'external ballast factor = %6.4f ', ...
1958: ballast_factor ), VA, 'base', HA, 'left', ...
1959: FS, 10, FN, 'courier' ) ;
1960: end
1961:
1962: axis( [ 0 1 0 1 ] )
1963: axis( 'off' )
1964:
1965: % -----
1966: % -- panel 4 - text - final ballast weights
1967:
1968: axes( 'Position', [ 0.30 0.06 0.30 0.08 ] )
1969:
1970: hold on
1971:
1972: plot( [ 0 1 0 0 ], [ 0 0 1 1 0 ], 'k' )
1973:
1974: text( 0.5, 0.75, 'Total Additional Ballast Weight', ...
1975: HA, 'center', FS, 14 ) ;
1976:
1977: if( ballast_flag == 1 ) % -- external
1978: text( 0.5, 0.20, ...
1979: sprintf( 'EXTERNAL %8.2f grams', ballast_addition ), ...
1980: VA, 'base', HA, 'center', FS, 14 ) ;
1981:
1982: elseif( ballast_flag == 0 ) % -- internal
1983:
1984: text( 0.5, 0.20, ...
1985: sprintf( 'INTERNAL %8.2f grams', ballast_addition ), ...
1986: VA, 'base', HA, 'center', FS, 14 ) ;
1987: end
1988:
1989: text( 0.5, -0.30, version, VA, 'base', HA, 'center', FS, 10 ) ;
1990:
1991: axis( [ 0 1 0 1 ] )
1992: axis( 'off' )
1993:
1994: % -----
1995: % -- append results to the ballast summary file
1996:
1997: % -----
1998: % -----
1999:
2000: % -- open summary file for appending (create if necessary)
2001:
2002: nlen = length(file_tank) ;
2003: nm3 = nlen - 3 ;
2004: nm4 = nlen - 4 ;
2005:
2006: if( strcmp( file_tank(nm3:nlen), '.txt' ) )
2007: file_sum = sprintf( '%s.sum', file_tank(1:nm4) ) ;
2008: else
2009: file_sum = sprintf( '%s.sum', file_tank ) ;
2010: end
2011:
2012: fsum = fopen( file_sum, 'a' ) ;
2013:
2014: fprintf( fsum, '\n' ) ;
2015: fprintf( fsum, [ '-----', ...
2016: 'Summary of BALLAST Calculations \n' ] ) ;
2017: fprintf( fsum, 'RTC Data File : "%s" \n', file_tank ) ;
2018: fprintf( fsum, [ 'RTC Data Time : %s ', ...
2019: 'Calculation Time : %s \n' ], ...
2020: start_time, plot_time ) ;
2021:
2022: fprintf( fsum, '-- Ballast Software: "%s" \n', version ) ;
2023: fprintf( fsum, [ '-----', ...
2024: '-----\n' ] ) ;
2025:
2026: fprintf( fsum, '\n-- Parameter and Calibration Data \n\n' ) ;
2027:
2028: fprintf( fsum, 'file_param = "%s" \n', ...
2029: file_param ) ;
2030: fprintf( fsum, 'file_directory = "%s" \n', ...
2031: file_directory ) ;
2032: fprintf( fsum, 'beam serial number = "%s" \n', ...
2033: beam_number ) ;
2034: fprintf( fsum, '\n' ) ;
2035: fprintf( fsum, 'file_P_tank_coeff = "%s" \n', ...
2036: file_P_tank_coeff ) ;
2037:
2038: if( ~strcmp( file_P_tank_coeff, 'NaN' ) )
2039: fprintf( fsum, 'P_tank_coeff = ' ) ;
2040: fprintf( fsum, '%12.5e', P_tank_coeff ) ;
2041: fprintf( fsum, '\n' ) ;
2042: end
2043:
2044: fprintf( fsum, 'file_Tbot_tank_coeff = "%s" \n', ...
2045: file_Tbot_tank_coeff ) ;
2046:
2047: if( ~strcmp( file_Tbot_tank_coeff, 'NaN' ) )
2048: fprintf( fsum, 'Tbot_tank_coeff = ' ) ;
2049: fprintf( fsum, '%12.5e', Tbot_tank_coeff ) ;
2050: fprintf( fsum, '\n' ) ;
2051: end
2052:
2053: fprintf( fsum, 'file_Ttop_tank_coeff = "%s" \n', ...
2054: file_Ttop_tank_coeff ) ;
2055:
2056: if( ~strcmp( file_Ttop_tank_coeff, 'NaN' ) )
2057: fprintf( fsum, 'Ttop_tank_coeff = ' ) ;
2058: fprintf( fsum, '%12.5e', Ttop_tank_coeff ) ;
2059: fprintf( fsum, '\n' ) ;
2060: end
2061:
2062: fprintf( fsum, 'file_beam_press_coeff = "%s" \n', ...
2063: file_beam_press_coeff ) ;
2064:
2065: if( ~strcmp( file_beam_press_coeff, 'NaN' ) )
2066: fprintf( fsum, 'beam_press_coeff = ' ) ;
2067: fprintf( fsum, '%12.5e', beam_press_coeff ) ;
2068: fprintf( fsum, '\n' ) ;
2069: end
2070:
2071: fprintf( fsum, 'file_beam_temp_coeff = "%s" \n', ...
2072: file_beam_temp_coeff ) ;
2073:
2074: if( ~strcmp( file_beam_temp_coeff, 'NaN' ) )
2075: fprintf( fsum, 'beam_temp_coeff = ' ) ;
2076: fprintf( fsum, '%12.5e', beam_temp_coeff ) ;
2077: fprintf( fsum, '\n' ) ;
2078: end
2079:
2080: fprintf( fsum, 'file_beam_weight_coeff = "%s" \n', ...
```

10/26/06
15:18:24

ballast.m

```
2081: file_beam_weight_coeff ) ;
2082: if( ~strcmp( file_beam_weight_coeff, 'NaN' ) )
2083:     fprintf( fsum, 'file_beam_weight_coeff = %s\n' );
2084:     fprintf( fsum, 'beam_weight_coeff = %s\n' );
2085:     fprintf( fsum, '%12.5e', beam_weight_coeff );
2086:     fprintf( fsum, '\n' );
2087: end
2088:
2089: fprintf( fsum, '\n' );
2090:
2091: fprintf( fsum, 'float thermal_coeff = %10.3e 1/degC\n', ...
2092:     float_thermal_coeff );
2093: fprintf( fsum, 'ballast_location = "%s"\n', ...
2094:     ballast_location );
2095:
2096: if( strcmp( ballast_location, 'external' ) )
2097:     fprintf( fsum, [ 'ballast_density = %s',
2098:         '%7.4f g/cc (external)\n' ], ...
2099:         ballast_density );
2100: end
2101:
2102: fprintf( fsum, '\n-- Float Dry Weights\n\n' );
2103:
2104: fprintf( fsum, 'float
2105:     = %8.2f grams\n', ...
2106:     float_dry );
2107: fprintf( fsum, 'float_dry
2108:     = %8.2f grams\n', ...
2109:     float_dry );
2110: fprintf( fsum, 'float_dry
2111:     = %8.2f grams\n', ...
2112:     float_dry );
2113: fprintf( fsum, 'float_dry
2114:     = %8.2f grams\n', ...
2115:     float_dry );
2116: fprintf( fsum, 'float_dry
2117:     = %8.2f grams\n', ...
2118:     float_dry );
2119: fprintf( fsum, 'float_dry
2120:     = %8.2f grams\n', ...
2121:     float_dry );
2122: fprintf( fsum, 'float_dry
2123:     = %8.2f grams\n', ...
2124:     float_dry );
2125: fprintf( fsum, 'float_dry
2126:     = %8.2f grams\n', ...
2127:     float_dry );
2128: fprintf( fsum, 'float_dry
2129:     = %8.2f grams\n', ...
2130:     float_dry );
2131: fprintf( fsum, 'float_dry
2132:     = %8.2f grams\n', ...
2133:     float_dry );
2134: fprintf( fsum, 'float_dry
2135:     = %8.2f grams\n', ...
2136:     float_dry );
2137: fprintf( fsum, 'float_dry
2138:     = %8.2f grams\n', ...
2139:     float_dry );
2140: fprintf( fsum, 'float_dry
2141:     = %8.2f grams\n', ...
2142:     float_dry );
2143: fprintf( fsum, 'float_dry
2144:     = %8.2f grams\n', ...
2145:     float_dry );
2146:
2147: fprintf( fsum, 'strat. limit
2148:     = %8.3f degC\n', ...
2149:     temp_diff_warning );
2150:
2151: msg = '
2152:     Ttop diff = %8.3f degC\n';
2153: if( Ttop_warning ); msg = [ msg, '(warning)\n' ]; end
2154: fprintf( fsum, [ msg, '\n' ], Ttop_diff );
2155:
2156: msg = '
2157:     Tbot diff = %8.3f degC\n';
2158: if( Tbot_warning ); msg = [ msg, '(warning)\n' ]; end
2159: fprintf( fsum, [ msg, '\n' ], Tbot_diff );
2160:
2161: fprintf( fsum, 'conductivity
2162:     = %10.3e S\n', ...
2163:     conductivity );
2164: fprintf( fsum, 'sigma_t
2165:     = %8.3f kg/m^3\n', ...
2166:     sigma_t );
2167: fprintf( fsum, 'rho(target)
2168:     = %8.5f grams/cc\n', ...
2169:     density_target );
2170:
2171: fprintf( fsum, '\n-- Target Properties\n\n' );
2172:
2173: fprintf( fsum, 'salinity
2174:     = %8.3f psu\n', ...
2175:     S_target );
2176: fprintf( fsum, 'temperature
2177:     = %8.3f degC\n', ...
2178:     T_target );
2179: fprintf( fsum, 'pressure
2180:     = %8.1f dbar\n', ...
2181:     P_target );
2182: fprintf( fsum, 'sigma_t
2183:     = %8.3f kg/m^3\n', ...
2184:     sigma_t );
2185: fprintf( fsum, 'rho
2186:     = %8.5f grams/cc\n', ...
2187:     density_target );
2188:
2189: fprintf( fsum, '\n-- Ballast Data\n\n' );
2190:
2191: fprintf( fsum, [ 'Pressure Weight ("x" denotes data', ...
2192:     ', edited out)\n' ] );
2193: fprintf( fsum, ' (psi) (grams)\n\n' );
2194:
2195: for k=1:length(P_tank)
2196:     fprintf( fsum, '%6.1f %6.2f', ...
2197:         P_tank(k), weight_beam(k) );
2198:     if( ~data_status(k) ); fprintf( fsum, 'x' ); end
2199:     fprintf( fsum, '\n' );
2200: end
2201:
2202: fprintf( fsum, '\n-- Ballast Parameters\n\n' );
2203:
2204: fprintf( fsum, 'min pressure = %4.0f psi (%4.0f dbar)\n', ...
2205:     Pmin_psi_range, Pmin_psi );
2206: fprintf( fsum, 'max pressure = %4.0f psi (%4.0f dbar)\n', ...
2207:     Pmax_psi_range, Pmax_psi );
2208: fprintf( fsum, 'multi_option = %2d\n', ...
2209:     multi_option );
2210: fprintf( fsum, 'stddev_factor = %4.1f grams\n', ...
2211:     stddev_factor );
2212:
2213: fprintf( fsum, 'DAQ reference_freq = %13.10f Hz\n', ...
2214:     DAQ_reference_freq );
2215: fprintf( fsum, 'DAQ deviation_limit = %9.6f Hz\n', ...
2216:     DAQ_deviation_limit );
2217: fprintf( fsum, 'DAQ timebase_mean = %13.10f Hz\n', ...
2218:     DAQ_timebase_mean );
2219: fprintf( fsum, 'DAQ frequency_ratio = %9.6f\n', ...
2220:     DAQ_frequency_ratio );
2221: fprintf( fsum, 'DAQ frequency_ratio = %2d\n', ...
2222:     DAQ_frequency_ratio );
```

17

2276: 8

```

2276: # -----
2277: nlen = length(file_tank) ;
2278: nlen = nlen - 3 ;
2279: nm3 = nlen - 4 ;
2280: nm4 = nlen - 4 ;
2281:
2282: if ( strcmp( file_tank(nm3:nlen), 'txt' ) )
2283:     file_plot = sprintf( '%s.ps', file_tank(1:nm4) ) ;
2284: else
2285:     file_plot = sprintf( '%s.ps', file_tank ) ;
2286: end
2287:
2288: eval( [ 'print -dpsc2 ', file_plot ] )
2289:
2290: return

```

Appendix E

Details of Linearized Model Used For Sensitivity Study

①

Sensitivity Analysis

- Uses variable names from Matlab code ballast.m
- Weight & Balance

$$\text{total_correction} = \text{neutral_weight} + \text{thermal_correction} + \text{seawater_correction} \quad (1)$$

- General approach to sensitivity study:

- linearize the W&B equation (1) with respect to parameters of interest, and evaluate around a specified operating point.
- linearization via Taylor's Series

$$f(x_1, x_2, \dots) = f(x_{10}, x_{20}, \dots) + \left. \frac{\partial f}{\partial x_1} \right|_0 (x - x_1) + \left. \frac{\partial f}{\partial x_2} \right|_0 (x - x_2) + \dots + \text{HOT} \quad (2)$$

where: ① x_{10}, x_{20} are operating point conditions

② $\left|_0 \right.$ implies evaluated at the operating pt.

③ HOT $\equiv$ higher order terms

$$\text{letting: } \Delta f(x_1, x_2, \dots) = f(x_1, x_2, \dots) - f(x_{10}, x_{20}, \dots) \quad (2)$$

$$\Delta x_1 = x - x_1$$

$$\Delta x_2 = x - x_2$$

Ignoring the higher order terms, eqn (2) can be rewritten as:-

$$\Delta f(x_1, x_2, \dots) = \left. \frac{\partial f}{\partial x_1} \right|_0 \Delta x_1 + \left. \frac{\partial f}{\partial x_2} \right|_0 \Delta x_2 + \dots$$

To simplify the evaluation, the W&B equation

(1) will be evaluated by the three

Components: neutral-weight
thermal correction
seawater correction

1. Neutral-weight

Description: this is the ballast required to achieve neutral density at tank temp & Salinity and target pressure.

$$\text{neutral-weight} = \text{trim-total-wet} - \text{trim-surplus-target-wet} \quad (3)$$

where:

$$\text{trim-total-wet} = \text{ballage-wet} + \text{trim-dry} \left(1 - \frac{\text{density-target-tank}}{\text{trim-density}} \right) \quad (4)$$

③

$$\text{trim_surplus_wet} = \text{ballast_slope} * P_target + y_intercept \quad (5)$$

where:

ball-cage-wet $\equiv$ measured wet wt of ball cage

trim-dry $\equiv$ measured dry wt of trim balls

trim-density $\equiv$ ball material density (ss)

from material property specs

density-target-tank $\equiv$ estimated tank water density at tank temp., salinity and target pressure, (from unesco water properties functions in Matlab).

④ Evaluate equation 4:

$$f(x) = \text{trim_total_wet}$$

$$\Delta \text{trim_total_wet} = \frac{\partial f}{\partial (\text{ball_cage_wet})} \bigg|_0 \Delta \text{ball_cage_wet}$$

$$+ \frac{\partial f}{\partial (\text{trim_dry})} \bigg|_0 \Delta \text{trim_dry}$$

$$+ \frac{\partial f}{\partial (\text{trim_density})} \bigg|_0 \Delta \text{trim_density}$$

$$+ \frac{\partial f}{\partial (\text{density_target_tank})} \frac{\partial (\text{density_target_tank})}{\partial S_{\text{tank}}} \bigg|_0 \Delta S_{\text{tank}}$$

$$+ \frac{\partial f}{\partial (\text{density_target_tank})} \frac{\partial (\text{density_target_tank})}{\partial T_{\text{tank}}} \bigg|_0 \Delta T_{\text{tank}}$$

$$\Delta \text{trim_total_wet} = \Delta \text{ball_cage_wet} \quad (6)$$

(4)

$$+ \left(1 - \frac{\text{density_target_tank}_0}{\text{trim_density}_0} \right) \Delta \text{trim_dry}$$

$$+ (\text{trim_dry}_0 \cdot \frac{\text{density_target_tank}_0}{\text{trim_density}_0^2}) \Delta \text{trim_density}$$

$$+ \left(- \frac{\text{trim_dry}_0}{\text{trim_density}_0} \right) \left(\frac{\partial \text{density_target_tank}}{\partial S_{\text{tank}}} \right) \Big|_0 \Delta S_{\text{tank}}$$

$$+ \left(- \frac{\text{trim_dry}_0}{\text{trim_density}_0} \right) \left(\frac{\partial \text{density_target_tank}}{\partial T_{\text{tank}}} \right) \Big|_0 \Delta T_{\text{tank}}$$

Reference Excel spreadsheet for evaluation of equation (6) to specified levels of perturbation of $\Delta \text{ball_cage_wet}$, $\Delta \text{trim_dry}$, $\Delta \text{trim_density}$, ΔS_{tank} and ΔT_{tank} .

⑥ Evaluate eqn(5):

(7)

$$\Delta \text{trim_surplus_target_wet} = P_{\text{target}_0} \cdot \Delta \text{ballast_slope} + \Delta Y\text{-intercept}$$

Ref. excel spreadsheet for sensitivity evaluation of eqn(7).

(5)

2) Thermal correction

$$\text{thermal_correction} = \text{volume_float} \cdot \text{float_thermal_coeff} \cdot$$

$$(T_{\text{target}} - T_{\text{sbmean}}) \cdot \text{density_target}$$

Rather than expand float volume in terms of weight and density, will evaluate the thermal correction in terms of ΔVolume .

$$\Delta \text{thermal_correction} =$$

(8)

$$\text{float_thermal_coeff}_0 \cdot (T_{\text{target}_0} - T_{\text{sbmean}_0}) \cdot \text{density_target}_0 \cdot \Delta \text{Volume_float}$$

$$+ \text{volume_float}_0 \cdot \text{float_thermal_coeff}_0 (T_{\text{target}_0} - T_{\text{sbmean}_0}) \cdot$$

$$\frac{\partial \text{density_target}}{\partial S_{\text{target}}} \cdot \Delta S_{\text{target}}$$

$$+ \text{Volume_float}_0 \cdot \text{float_thermal_coeff}_0 (T_{\text{target}_0} - T_{\text{sbmean}_0}) \cdot$$

$$\frac{\partial \text{density_target}}{\partial T_{\text{tank}}} \cdot \Delta T_{\text{tank}}$$

$$+ \text{Volume_float}_0 \cdot \text{float_thermal_coeff}_0 \cdot \text{density_target}_0 \cdot \Delta T_{\text{target}}$$

$$- \text{Volume_float}_0 \cdot \text{float_thermal_coeff}_0 \cdot \text{density_target}_0 \cdot \Delta T_{\text{tank}}$$

$$+ \text{Volume_float}_0 \cdot (T_{\text{target}_0} - T_{\text{sbmean}_0}) \cdot \text{density_target}_0 \cdot$$

$$\Delta \text{float_thermal_coeff}$$

Ref. excel spreadsheet for sensitivity evaluation of eqn (8).

⑥

3) Saltwater correction:

$$\text{seawater_correction} = \text{volume_float} \cdot (\text{density_target} - \text{density_tank})$$

$$\Delta \text{seawater_correction} = \quad (9)$$

$$\begin{aligned} & (\text{density_target}_0 - \text{density_tank}_0) \cdot \Delta \text{volume_float} \\ & + \text{volume_float}_0 \left. \frac{\partial \text{density_target}}{\partial S_{\text{target}}} \right|_0 \Delta S_{\text{target}} \\ & + \text{volume_float}_0 \left. \frac{\partial \text{density_target}}{\partial T_{\text{target}}} \right|_0 \Delta T_{\text{target}} \\ & - \text{volume_float}_0 \left. \frac{\partial \text{density_target_tank}}{\partial T_{\text{tank}}} \right|_0 \Delta T_{\text{tank}} \end{aligned}$$

Ref. excel spreadsheet for sensitivity evaluation of eqn (9).

APPENDIX H: SOUND SOURCE MOORING DIAGRAMS

REPORT DOCUMENTATION PAGE	1. REPORT NO. WHOI-2020-06	2.	3. Recipient's Accession No.
4. Title and Subtitle Overturning of the Subpolar North Atlantic Program (OSNAP): RAFOS Float Data Report June 2014 - January 2019			5. Report Date December 2020
7. Author(s) A. Ramsey, H. Furey, and A. Bower			6.
9. Performing Organization Name and Address Woods Hole Oceanographic Institution Woods Hole, Massachusetts 02543			8. Performing Organization Rept. No.
			10. Project/Task/Work Unit No.
			11. Contract(C) or Grant(G) No. (C) DCE-1259618 (G) DCE-1756361
12. Sponsoring Organization Name and Address National Science Foundation			13. Type of Report & Period Covered Technical Report
			14.
15. Supplementary Notes This report should be cited as: Woods Hole Oceanographic Institution Technical Report, WHOI-2020-06.			
16. Abstract (Limit: 200 words) The Overturning in the Subpolar North Atlantic Program (OSNAP) is an international effort started in 2014 dedicated to achieving a better understanding of the link between dense-water formation and the meridional overturning circulation in the high-latitude North Atlantic. Moorings, gliders, and subsurface acoustically-tracked RAFOS floats have been used to collect temperature, pressure, and current data across the Labrador Sea, Irminger Sea, Reykjanes Ridge, Iceland Basin, Rockall-Hatton Plateau, and Rockall Trough. The specific objective of the OSNAP float program is to gather information on the pathways of the dense overflow waters transported by the deep limb of the overturning circulation and assess the connection of those pathways with currents observed crossing the OSNAP mooring line. This data report details the observations collected by 148 floats that were deployed for OSNAP during the summers of 2014, 2015, 2016 and 2017. Deployment locations were in the Iceland Basin, Irminger Sea, Labrador Sea, and in the Charlie-Gibbs Fracture Zone. Mission lengths ranged from 540-730 days, and the floats were programmed to passively drift at a fixed pressure of either 1800, 2000, 2200, 2500, or 2800 dbar to tag the deep overflow water masses of the subpolar North Atlantic (Iceland-Scotland and Denmark Strait Overflow Waters).			
17. Document Analysis			
a. Descriptors RAFOS float subpolar North Atlantic circulation deep boundary currents			
b. Identifiers/Open-Ended Terms			
c. COSATI Field/Group			
18. Availability Statement Approved for public release; distribution unlimited.	19. Security Class (This Report) UNCLASSIFIED		21. No. of Pages 510
	20. Security Class (This Page)		22. Price