A User Guide to Mexec

A Matlab-based Bespoke Processing Suite for Ship-based Oceanographic Data

Version 4 (subset of ocp_hydro_matlab)

Yvonne L. Firing, N. Penny Holliday, Alejandra L. Sanchez-Franks

Ocean Circulation and Processes Subgroup

Marine Physics and Ocean Climate

National Oceanography Centre

Southampton

1. Introduction

1.1 About this guide

This guide, and Mexec, are designed for those wishing to process CTD and underway data at sea. The underway portion is applicable to ships running SCS or TECHSAS systems. Basic familiarity with UNIX/Linux, shell scripting, and Matlab is assumed.

Throughout, > is used to indicate examples of steps run from the command line, including shell scripts, and >> for steps run from Matlab. Variables or parts of filenames that must be substituted with values are in italic (e.g. a reference to m_daily_proc(*day*) indicates that the day number should be substituted for *day*; opt_*cruise*.m refers to opt_dy113.m, opt_jc238.m, etc.; ctd_*cruise_nnn*_2db.nc refers to one of a set of files, with *cruise* replaced by the cruise designation and *nnn* replaced by the 3-digit form of the station number).

The following sections give instructions for setting up Mexec processing for a cruise (Section 2); processing CTD data (Section 3); and processing underway data (Section 4). A brief description of Mexec is given below, with a few more details in Appendix A, but in general this is meant to be a guide to a standard set of steps with limited variations. Example processing checklists are included in Appendix B.

Commented [FYL1]: links

This guide was first written by Penny Holliday and has been updated for v3 by Yvonne Firing and Alejandra Sanchez Franks and for v4 by Yvonne Firing.

1.2 What is Mexec?

Mexec is a system for processing, quality control, and integration of hydrographic data including CTD and water sample data and standard underway streams. It consists of Matlab and shell scripts, and interfaces with output of the SeaBird software (for collection and initial processing of CTD data), and with external libraries for processing of LADCP and VMADCP data. It saves its output as NetCDF files with a particular set of metadata and conventions (referred to here as Mstar files, see 1.2.3). The Matlab scripts are part of git repository git.noc.ac.uk/OCP/ocp_hydro_matlab, and the shell scripts are in git.noc.ac.uk/OCP/mexec_exec.

The Mexec libraries were developed over a number of years by scientists at the UK National Oceanography Centre, Southampton, principally Brian King and Yvonne Firing, with contributions from many others including D. Desbruyeres, G. Evans, C. Florindo Lopez, N.P. Holliday, L. Houpert, E. Kent, G. McCarthy, B. Moat, A. Sanchez-Franks, D. Smeed, Z. Szuts, and E. Woodward (?I think: efw***), as well as from external collaborators including E.P. Abrahamsen, K. Baumeister, S. Gary, and D. Ham, They use the Seawater (C. Morgan and L. Pender), GSW (SCOR/IAPSO WG127), and gamma_n (D. Jackett and T. McDougall) libraries in calculations. They interface with the LDEO IX LADCP processing software (M. Visbeck and A. Thurnherr), and read in underway data from SCS,, TechSAS, and RVDAS (NOC/NMF) as well as VMADCP data from CODAS (University of Hawaii Currents Group).

Commented [FYL2]: Say something about pstar?

Commented [FYL3]: Source/credit?

Commented [FYL4]: Source/credit?

Commented [FYL5]: Check source/credit

The code has undergone numerous revisions, more recently preserved as git commits (before 2018 preserved in each cruise's end-of-cruise backup of the processing workstation). Major changes since 2014 include:

JR15003 (2015/16): start of mexec_processing_scripts_v3, with introduction of cruise options files (see 1.2.2).

JC159 (2018): introduction of git for version control, using three repositories (now superseded and no longer maintained, see below):

git.noc.ac.uk/MEXEC/mexec_processing_scripts, git.noc.ac.uk/MEXEC/mexec, and git.noc.ac.uk/MEXEC/mexec_exec.

JC191 (2020): revision of netcdf file-interface code to use Matlab-native functions (replacing bespoke functions developed before the Matlab-native functions were available).

JC211 (2021): addition of interface to NMF RVDAS underway data acquisition system, as well as function for saving from Matlab workspace to Mstar files using Matlab save-like interface, and (enabled by this function) the start of simplification of processing steps and reduction in number of intermediate Mstar files

JC238 (2022): continued simplification of processing steps, merge of mexec and mexec_processing_scripts along with other related Matlab code into a single git repository, git.noc.ac.uk/OCP/ocp_hydro_matlab, and shifting of (maintained) mexec_exec repository from git.noc.ac.uk/MEXEC/ to git.noc.ac.uk/OCP/.

SD025 (2023): modularization of gridding and editing functionality; instrument serial numbers tracked; update of RVDAS interface for SDA.

EN705 (2023): update for reading NetCDF SCS underway data files.

DY174 (2024): CF-compliant time units.

Some attempts at backwards compatibility have been made, with the goal that newer code could be used to reprocess older cruises' data (without starting completely from scratch), for instance if new bottle samples analysed ashore become available, or calibrations are reconsidered. Where file name patterns (e.g. for merged intermediate files) have changed, the aim is for code to be compatible with both old and new versions. Backwards compatibility is also maintained for some variable names but not in every case. When cruise options files were introduced on JR15003, processing choices from previous cruises were captured, and earlier cruises' options files were generally kept up to date with changes to code through DY113. Changes from JC211 on (e.g. to option names, how they are specified, or in which scripts they are used) have been only partially propagated back to earlier cruises' options files as of this point, so reprocessing earlier cruises' data may require more editing of the options files.

1.2.1 Metadata, history files and data file version control for multiple people processing data

Each step that goes into a Mstar file is recorded both in the header comment field and in a history file for the processing stream (so, although there are multiple ctd Mstar files for each station, there will be a single ctd history file for each station). The history files are found in mexec_housekeeping/history/. The comments in the header of the data file are set by the code writing it and, if the code is run on a Unix/Linux system, include the user and latest commit of the code used at a given step.

To prevent conflicts over modifying Mstar files, editing one also sets a lock file, in mexec_housekeeping/version/, which will normally be reset in closing the program. If a program is interrupted mid-run, however, the flag may have to be reset manually using mreset.m.

1.2.2 Processing options and parameters

A number of processing parameters and variables (e.g. which CTD is primary, the calibration functions for conductivity and other parameters, etc.) change from cruise to cruise. From Mexec v3 on, rather than editing the various scripts that may contain these parameters, they are set centrally, with modifications contained in one script (per cruise), as follows: processing scripts and functions call get_cropt.m, which

1) calls the four setdef_cropt_*.m scripts to set any defaults using switch/case on two (string) variables: scriptname (generally the name of the calling script) and oopt;
2) calls opt_*cruise*.m to set cruise-specific options, again using switch/case on scriptname and oopt; and
3) calls check_cropt.m to (for some cases) check and warn about invalid or unset options.

The defaults are split, purely for convenience/readability, across four scripts: set_mexec_defaults.m contains defaults relating to CTD casts, setdef_cropt_sam.m those relating to sample data, setdef_cropt_uway.m those relating to underway streams, and setdef_cropt_other.m contains defaults relating to everything else (csv output files, multi-cast gridding, etc.). These setdef*.m scripts contain explanations for each set of options and therefore may be useful to examine, but general users should not edit them. Instead, make changes to opt_*cruise*.m for your cruise – it is recommended that you not duplicate (unchanged) default settings in this file -- and raise an issue (see 1.3) if you think the defaults themselves should change.

A guide to finding out about cruise-specific options is in Appendix A.

To determine processing choices that went into a given data file, in addition to the comment field, the metadata includes the commit active when the data file was most recently modified; the setdef_cropt_*.m and opt_*cruise*.m files in that commit thus contain a record of corresponding processing parameters. If you add parameters (e.g. a list of salinity sample flags to modify) not explicitly in opt_*cruise*.m but by having opt_*cruise*.m read in another file (e.g. salflags.txt), you should place that file in your data directory (e.g. in ctd/BOTTLE_SAL) so it will be backed up along with the original input "data" files (e.g. sal_jc238_crate1.csv, etc.).

1.2.3 Mexec conventions

Mexec uses global variables for passing arguments to functions. This means if you clear the workspace you will have to re-run m_common or m_global before using any other Mexec programs.

The reason for this setup is that many of the original functions that performed calculations and did file i/o (now in ocp_hydro_matlab/file_tools/mexec/) were designed to be run in an interactive mode, prompting for successive inputs on the command line. To call these functions within other scripts or functions, therefore, a global variable (MEXEC_A.MARGS_IN) is used and queried for the inputs. Because the list of inputs depends on the previous choices, the best way to figure out how to use one of these functions is to call it in interactive mode (with empty or missing MEXEC_A.MARGS_IN) and follow the prompts.

Partially in v3 and increasingly in v4, these functions are being superseded, first by increased calculations in the workspace, and second by functions that take input arguments in a more Matlab-standard format. This guide thus assumes users will interact with Mexec by running the wrapper scripts (ocp_hydro_matlab/mexec_processing_scripts and subdirectories, described below) and modifying the cruise options file, ocp_hydro_matlab/mexec_processing_scripts/cruise_options/opt_*cruise*.m, and will not need to directly call the ocp_hydro_matlab/file_tools/mexec/ functions (or edit them except for some specifying underway file information, detailed in Section 4).

1.2.4 Mstar file format

Mexec generates NetCDF files, with a particular format (set of attributes), referred to here as Mstar files. While these NetCDF files can be read with ncdisp.m and ncread.m (or equivalent programs in other languages), there are specific Mexec scripts for interfacing with them. There are two notable idiosyncracies that may trip up when interacting with Mstar files: a) all data must be numeric; b) before DY174, their specification of time base and units is non-standard: the time units, specified as variable attributes, are seconds or sometimes days, and the origin is given as global attribute data_time_origin ([yyyy mm dd HH MM SS]). From DY174, CF-compliant units (e.g. 'seconds since 2024-01-01 00:00:00') are used in time variable units attribute, and the data_time_origin global attribute can be ignored.

A quick guide to interacting with Mstar files is given in Appendix A. For python users loading older (pre-DY174) Mstar data into xarray, you may need to force the time variable *not* to be read not as a datetime, and then convert it.

1.3 Changes and bugs

We encourage you to create a pull request or send us your scripts at the end of your cruise (ideally with context about your changes or additions as an issue or by email to yvonne.firing@noc.ac.uk), so that other users can benefit from your improvements and examples. To make it easier for us to track and integrate changes, please use ocp_hydro_matlab/ only for scripts/functions that you think should be maintained in the code going forward (e.g. a bug fix to an existing script, or a new script to read in a new type of data that may be used again); use a different directory for working code just for your cruise (e.g. quick code making plots of data specifically for your cruise).

2. Setting up a new cruise and using Mexec to process data

2.1 Before the cruise

1) Get the ocp_hydro_matlab git repository.
   If you have git, clone the main branch:
   > cd ${PROGDIR} #this is wherever you choose to keep the software
   > git clone git@github.com/NOC-OCP/ocp_hydro_matlab.git #if you have a github login
   OR
   > git clone https://github.com/NOC-OCP/ocp_hydro_matlab.git
   Generally the main branch is the version you should use, but in special cases you may instead want to work from a different cruise branch:
   > cd ${PROGDIR}/ocp_hydro_matlab
   > git branch *other_branch*
   > git fetch origin *other_branch*
   > git checkout origin *other_branch*
   > git pull origin *other_branch*
   Otherwise, download the desired branch (probably the master branch) from http://github.com/NOC-OCP/ocp_hydro_matlab and unpack in ${PROGDIR}.

   1b) If you intend to use the shell scripts for syncing data from a server to your Linux/MacOS working machine, also clone or download (as above) the mexec_exec git repository from git.noc.ac.uk/OCP/mexec_exec.

2) Determine your cruise designation, typically 2-3 letters denoting the ship and 3-5 letters denoting the cruise number, e.g. dy113, jr18002, nbp0705. This will be used in various places and is the variable meant by (italic) *cruise* through the rest of this document.

   If you are using git, for each of the repositories, create and switch to a new branch for your cruise:
   > cd ${PROGDIR}/ocp_hydro_matlab
   > git branch *cruise*
   > git checkout *cruise*
   If you are using mexec_exec also:
   > cd ${PROGDIR}/mexec_exec
   > git branch *cruise*
   > git checkout *cruise*

   3) Add the full path to ocp_hydro_matlab to your Matlab startup path.

   4) If using mexec_exec:
      a. Add it and its subdirectories to your shell path (in .bashrc or equivalent).
      b. Edit the cruise name and processing base directory (e.g. /local/users/pstar/dy113/mcruise/) in ${PROGDIR}/mexec_exec/conf_scripts/conf_script_mexec. You may also need to edit the underway data system and a flag for whether

LADCP data are acquired. Run to configure the cruise directory structure:
> conf_script_mexec
This will set up a processing directory structure, including relative symbolic links so that the directory structure can be copied elsewhere (e.g. at the end of a cruise) and processing continued.

5) Configure ocp_hydro_matlab:
   a. Add the full path to ocp_hydro_matlab/ to your Matlab startup path
   b. Edit ocp_hydro_matlab/mexec_processing_scripts/m_setup.m. Most things you (may) need to modify are near the top of the file, including *cruise* (stored as MEXEC_G.MSCRIPT_CRUISE_STRING), and the paths to other software used in processing (e.g. seawater toolboxes). You can alternately specify the *cruise* and a few other variables in an input argument to m_setup.
   c. Generate an empty cruise-specific options file, ocp_hydro_matlab/mexec_processing_scripts/cruise_options/opt_*cruise*.m, replacing *cruise* with your cruise reference (in the same form as you set in m_setup.m, e.g. dy113). See below, as well as other files in that directory, for examples.

2.2 On the ship

6) Remotely mount data filesystems, and if necessary set up symbolic links, e.g.

> ln -s /mnt/Data/current_cruise ~/mounts/mnt_cruise_data

The NOC OCP seagoing workstations have mount points for Cook and Discovery in /etc/fstab and preserved under ~/mounts/, e.g. ***

7) Edit shell scripts in mexec_exec/: ctd_syncscript, lad_syncscript (for LADCP), and uhdas_01_linkmerge (if reading UHDAS/CODAS data).

These scripts will be used to sync data from the remote directories to the processing directory structure, and will need to be edited to reflect file naming conventions in the remote directories (e.g. ~/mounts/CTD/Data/CTD_DY113_001.cnv vs. ~/mounts/CTD/Data/Processed/ctd_dy113_001_01.cnv, etc.).

You may also want to use mexec_exec/cruise_backup to sync the processing directory and scripts to an external hard drive; in this case, edit the directories set near the top of cruise_backup, and (recommended) add it to your crontab file to run regularly.

8) Set up or link underway data processing

a) TECHSAS or SCS: edit ocp_hydro_matlab/file_tools/mexec/mtechsas/mtnames.m (for TECHSAS) or ocp_hydro_matlab/file_tools/mscs/msnames.m (for SCS): add or comment/uncomment lines as necessary to reflect the stream names available on your cruise. If adding a new type of stream you can decide on the Mexec abbreviation.

b) RVDAS: get address login credentials for rvdas database machine. Set up ~/.pgpass file containing:

server:port:database:username:password

Run mrjson_get_list.m. Consider adding things to skip under mrvdas_ingest case in setdef_cropt_uway.m. look at mrtables_from_json.m and see if there's anything to add to mrnames.m (new streams) which is a lookup between shortnames (output file prefixes) and the streamnames (instrument_talkIDmsgID). If so you may also need to add it to muwaydirs.m, which has a list of shortnames (first column) and directories to put them in. Then, edit opt_*cruise*.m to delete the line about skipunderway, edit m_setudir.m and rerun m_setup.m. That should run m_setudir.m and create some subdirectories, e.g. nav/pmv, nav/sea, met/sonic, etc. It will also output a list of streamnames that it doesn't know where to put, and of directories it's expecting but doesn't find any table for.

Commented [FYL6]: Could these be combined?

On any system: You may need to edit and force rerun m_setudir.m once the cruise starts. If you added new Mexec stream abbreviations above, add them and the directories where you wish those streams to be processed to the list in m_setudir.m. In any case you may need to uncomment/comment out newly relevant/irrelevant lines.

If new underway streams become available during the cruise, remove ocp_hydro_matlab/mexec_processing_scripts/underway/m_udirs.m and regenerate it, and make the new directories, by running m_setudir.m.

9) Edit template files in ocp_hydro_matlab/mexec_processing_scripts/varlists/ (if necessary)

Template files are used to control lists of variables within scripts. They include ctd_renamelist.csv, sam_varlist.csv, dcs_varlist.csv, and cchdo_varlist.csv and cchdo_ctd_varlist.csv which determine lists of variables to be loaded for CTD, bottle sample, and other files, and how (if) they will be renamed. For SCS ships, there is also the set of scs_renamelist_source.csv files.

The list of variable names that you require in each file will vary from cruise to cruise depending on which samples are being collected. The ctd_, sam_, and cchdo_ template files contain many possible variables, so in most cases you will just need to delete lines. The scs_ files may need to be edited but most likely not.

ocp_hydro_matlab/mexec_processing_scripts/varlists/mcvars_list.m: make sure the two lists in this file include all the variables you want to carry through CTD processing and sample comparison, respectively. It is not necessary to comment out variables you don't have.

2.3 Using Mexec

Each time Matlab is started, run m_setup to initialize the environment for Mexec processing by adding paths and generating global variables. If you clear all variables at any point, run m_common or m_global to regenerate them.

3. CTD data and water bottle sample data

3.1 Sea Bird data acquisition and processing

The first step is to select the SBE output variables in the SBE data acquisition software, SeaSave. Record the setup by saving a .XMLCON file. It is essential that the output variables include scan and pressure temperature, and it is highly useful for them to include NMEA latitude and longitude if those streams are available (generally the case on modern research vessels). For some variables (e.g. turbidity), the conversion from voltage to physical units may result in loss of precision, so better results may be obtained by outputting the raw voltage stream.

Here is an example from JC086.

name 0 = timeS: Time, Elapsed [seconds]
name 1 = depSM: Depth [salt water, m]
name 2 = prDM: Pressure, Digiquartz [db]
name 3 = t090C: Temperature [ITS-90, deg C]
name 4 = t190C: Temperature, 2 [ITS-90, deg C]
name 5 = c0mS/cm: Conductivity [mS/cm]
name 6 = c1mS/cm: Conductivity, 2 [mS/cm]
name 7 = sal00: Salinity, Practical [PSU]
name 8 = sal11: Salinity, Practical, 2 [PSU]
name 9 = sbeox0V: Oxygen raw, SBE 43 [V]
name 10 = sbeox0Mm/Kg: Oxygen, SBE 43 [umol/Kg]
name 11 = sbeox0ML/L: Oxygen, SBE 43 [ml/l]
name 12 = xmiss: Beam Transmission, Chelsea/Seatech/WET Labs CStar [%]
name 13 = flC: Fluorescence, Chelsea Aqua 3 Chl Con [ug/l]
name 14 = turbWETbb0: Turbidity, WET Labs ECO BB [m^-1/sr]
name 15 = altM: Altimeter [m]
name 16 = scan: Scan Count
name 17 = ptempC: Pressure Temperature [deg C]
name 18 = pumps: Pump Status
name 19 = latitude: Latitude [deg]
name 20 = longitude: Longitude [deg]
name 21 = flag: 0.000e+00

For combining CTD data with Niskin bottle sample data, the CTD data file names should incorporate a unique integer cast number, referred to here as station number – not to be confused with a number or alphanumeric designating a planned station/site. (For instance, imagine a cruise starts with a test CTD followed by a full CTD both at site A-55, and then a CTD at site A-53: the cast or station numbers for Mexec processing could be (1, 2, 3) or (990, 1, 2) or even (10, 9, 15) – but not 55, 55, 54, nor 1.1, 1.2, 2.)

3.1.1 SBE Data Processing

On the CTD logging computer, the SBE Data Processing software should be used for initial processing when the cast is finished. First run:

1) Data Conversion to convert the raw frequency and voltage data to engineering units as appropriate by applying the manufacturer's calibrations stored in the CON file and saving both downcast and upcast to an ASCII format (.cnv) file.
- This step may include oxygen hysteresis correction using SBE default parameters, but we recommend not applying the correction here but instead applying it in mexec processing (this makes it easier to change the parameters if necessary). If you decide to correct for oxygen hysteresis at this stage you will need to change the dooxyhyst flag in opt_*cruise*.m).

  The output file names should contain the three-digit sequential station/cast number, and should not have spaces; otherwise their form is not important. Add the filename (and location) to the ctd_proc, cnvfilename case in opt_*cruise*.m.

The next two steps can be run either in SBE Data Processing or in mexec:

1) Align CTD to align the oxygen sensor in time relative to pressure. Recommended: set the output name to _align so that, for input CTD_CRUISE_nnn.cnv, this step will produce CTD_CRUISE_nnn_align.cnv.
2) Cell Thermal Mass to correct the pressure and conductivity. Recommended: set the output name to _ctm so that, for input CTD__CRUISE_nnn_align.cnv, this step will produce CTD_CRUISE_nnn_align_ctm.cnv.

The first and last .cnv files (that is, original and _align_ctm), as well as the .bl and .ros files, should be copied to /local/users/pstar/cruise/data/ctd/ASCII_FILES, while .hex, .hdr, and .XMLCON files should be copied to RAW_CTD_FILES. On unix/linux systems you can use (mexec_exec) ctd_syncscript to do this after editing cruise name and location of original files.

Commented [YF7]: update: not necessary, can instead change where scripts look for these files. They should definitely be copied to the processing machine, though.

3.2 Mexec CTD data processing

3.2.1 Output file types

ctd_*cruise_nnn*_*.nc contain CTD time series or profiles, with different stages of processing, editing, and averaging (see below).

dcs_*cruise_nnn*.nc contains information about scans (start, bottom, end of cast) and positions

fir_*cruise_nnn*.nc contains information about bottle firing times and corresponding CTD and winch data

sam_*cruise*_all.nc is a combined (all-station) file containing the data from the fir_*cruise_nnn*.nc files along with analysed bottle sample data.

win_*cruise*_nnn.nc contains winch information (wireout, tension).

3.2.2 Processing steps to do immediately following a cast

The basic steps for CTD processing following a cast (see Appendix B) are:
> ctd_syncscript
Copies .hex, .cnv, _align_ctm.cnv, and .bl files from acquisition computer to processing workstation.

>> stn = *n*; ctd_all_part1 % *n* is the integer station number

ctd_all_part1.m calls the following:

mctd_01.m loads and renames variables (as set in varlists/ctd_renamelist.csv and cruise options files) and saves in ctd_*cruise_nnn*_raw.nc.

mctd_02.m does conversions, edits, and corrections as set in cruise options files, calling ctd_apply_autoedits.m, ctd_apply_oxyhyst.m, select_calibrations.m, and apply_calibrations.m to produce ctd_*cruise_nnn*_raw_cleaned.nc and ctd_*cruise_nnn*_24hz.nc files. The only default action is to apply (the manufacturer default) correction for oxygen hysteresis, but examples are available for code to remove out of range values, certain scan ranges, spikes, and/or times when the pumps were off; it is generally not recommended to apply these before first examining data (as problems may be masked). Once sample data are available, the mctd_02 case of the cruise options file is also where calibration functions can be specified (with a setting to apply to all stations or only a subset).

mctd_03.m selects primary sensors (as set in cruise options files), computes derived variables (e.g. salinity) and averages to 1 Hz, using grid_profile.m, producing ctd_*cruise_nnn*_psal.nc.

mdcs_01.m guesses start and bottom of cast and saves in dcs_*cruise_nnn*.nc.

>> stn = *n*; mdcs_03g

mdcs_03g.m brings up a GUI for selection or confirmation of cast start, bottom, and end based on P, T, C, and pumps flag; any modifications are added to dcs_*cruise_nnn*.nc.

For the start of the downcast, select the lowest pressure after the CTD has soaked and been brought to the surface before descending (unless it was brought too close to the surface, causing erroneous conductivity values, in which case, select the start of the good data). For the end of the upcast, select the last scan for which there was good in-water oxygen, temperature, conductivity and salinity data (note that oxygen data becomes out-of-water before the other variables because of the different sensor response times).

Edit opt_cruise.m if necessary: if upon sampling some Niskins were leaky or questionable, edit their flags; if the cast was not full-depth add a nominal water depth (see Appendix A for how to find out where).

>> stn = *n*; ctd_all_part2

ctd_all_part2.m calls the following:

mctd_04.m separates down and up casts, optionally applies m_loopedit.m to the downcast data, and averages to 2 dbar, producing ctd_*cruise_nnn*_2db.nc and ctd_*cruise_nnn*_2up.nc.

mfir_01.m gets times and scans of Niskin bottle firing from .bl file, along with Niskin bottle numbers (default: 1:24) and flags (default: 2 for all fired bottles, 9 otherwise) from cruise options files, and puts in fir_*cruise_nnn*.nc.

mfir_03.m gets CTD data from these scans from the 1-Hz _psal.nc file and adds to fir_*cruise_nnn*.nc.

mwin_01.m gets winch information from the underway stream and saves in win_*cruise_nnn*.nc.

mwin_to_fir.m adds the winch information to fir_*cruise_nnn*.nc.

Commented [FYL8]: Where to explain difference between manufacturer cal and "our" sample-comparison cal?

Commented [FYL9]: Add information on default choices for filling gaps

Commented [FYL10]: Explain option to account for oxy_align time offset?

Commented [FYL11]: Info about default and other options

Commented [FYL12]: More info on default and other options

mfir_to_sam.m puts the data from fir_*cruise_nnn*.nc into appended sam_*cruise*_all.nc.
station_summary.m adds position, start and end time, depth obtained by calling best_station_depths.m, and information from sam_*cruise*_all.nc for this station to station_summary_*cruise*.nc and table station_summary_*cruise*.txt. Rather than storing station depths in a text file, they are calculated by best_station_depths.m. Immediately following a cast, it will try to calculate them from CTD+altimeter; later you can specify to include information from the LADCP (if available), but in any case, if you have casts that are not full-depth, you should use the underway bathymetry data to fill in a list of bottom depths under the best_station_depths case in opt_*cruise*.m.
mdep_01.m adds the depths to the various ctd_*cruise_nnn*_*.nc files.
get_sensor_groups.m adds serial numbers to the sample file.

>> stn = *n*; mctd_checkplots

mctd_checkplots.m produces a set of plots to check for sensor drift or other problems. It will make plots comparing the two sensors, plots comparing up- and downcast data. It will also make plots comparing station *n* with other stations, first querying for either some number of preceding stations (including 0) or a list of specific station numbers to use. If you notice loops/static instabilities here, you can activate the cruise option to apply loop editing in mctd_04.m.

>> stn = *n*; mctd_rawshow

mctd_rawshow.m allows inspection of 24 Hz data. If this reveals editing needed (e.g. spikes that are large enough to affect averaged data), there are normally two options:

1) specify automatic edits (based on scan ranges, data ranges, behaviour if pumps go off, and despiking) in opt_*cruise*.m under mctd_02.m case; and/or
2) >> stn = *n*; mctd_rawedit
mctd_rawedit.m brings up a GUI for selecting and deleting spikes in the data (starting from ctd_*cruise_nnn*_raw_cleaned.nc if available, ctd_*cruise_nnn*_raw.nc otherwise), and saves the selections to ctd_*cruise_nnn* _raw_cleaned.nc, as well as recording them in (text file) mplxyed_*yyyymmdd_HHMMSS*_ctd_*cruise_nnn*

If there are significant instrument or cable problems, the inspection may show spikes in pressure or temperature that are large enough to affect the other data streams, requiring reprocessing from before the cellTM stage*** details in ***.
If automatic edits were added or mctd_rawedit was used,

>> stn = *n*; ctd_all_postedit

ctd_all_postedit.m reruns mctd_02, mctd_03, mctd_04, mfir_03, and mfir_to_sam to apply edits and propagate changes to the ctd_*cruise_nnn*_raw_cleaned.nc file through to other files.

If necessary, iterate the rawedit and postedit steps, and/or mctd_checkplots.m, ctd_all_part2.m to loopedit.

Note: If you need to restart processing from the mctd_01 stage after doing manual edits in mctd_rawedit, to reapply them run ctd_apply_autoedits.m.

3.3 Water Bottle Sample Data

3.3.1 Loading data

The result of the sample data processing is to create a master sample data file, sam_*cruise*_all.nc, populated with CTD firing data, sensor data, and subsequently the water sample data as they become available. The CTD winch, firing and sensor data are pasted into this file during running of ctd_all_part2.m, as described above. This section describes how the water sample data are included in the process.

To link analysed water sample data with the corresponding CTD data (and with each other) we use CTD cast number (referred to here as station number, or field statnum in the Mstar files) and Niskin position (i.e. the carousel position, or field position in the Mstar files). The two parameters can be combined to make a unique "sample number" variable, e.g.:

```
>> sampnum =  d.statnum*100 + d.position;
```

Scripts including msal_01.m, moxy_01.m, mnut_01.m, mco2_01.m, mcfc_01.m, miso_01.m perform the following steps, depending on parameters and code specified in opt_*cruise*.m:

1) Read in sample data from excel (single or multiple sheet) or ascii csv files (with or without header lines preceding one or more column header rows and zero or more units rows) or from netcdf or Matlab files. Currently, for excel/csv files, every data row is required to have a value for every column (e.g. if station number is a column, even if the file contains data from only a single station, the number must be filled in on each line). For samples collected but not yet analysed, the "data" is a record of where (cast and Niskin) they were collected, and goes into a *parameter*_flag field for reference.
2) Map variable (or column) names to standardized names used in the Mexec processing and Mstar files, and check units (if supplied) or add them (if not).
3) Calculate additional quantities and apply corrections, e.g. calculate oxygen concentration from titre, standard, blank, and draw temperature values; apply salinity standards offsets and average salinity readings; modify flags based on opt_*cruise*.m
4) Save data to *type_cruise_nnn*.nc file(s), where type is e.g. sbe35, sal, oxy, etc., and to sam_*cruise*_all.nc. The individual data type files contain more information on each than the concatenated sam file. For instance, for oxygen the replicate values and their individual flags will be recorded (as botoxya, botoxyb, etc. and botoxya_flag, botoxyb_flag, etc.) in oxy_*cruise_nnn*.nc, but only the average (of good values) and correspondingly updated flags will be copied to sam_*cruise*_all.nc. Before saving to sam_*cruise*_all.nc, flags will additionally be updated for consistency with Niskin bottle flags (i.e., if a Niskin has been flagged as 4 or 3 for having closed at the wrong depth or leaked, respectively, all samples from that bottle will be flagged 4 "bad"). For salinity, only CTD samples are copied to sam_*cruise*_all.nc, while TSG samples are written to tsg_*cruise*_all.nc.

The key information to determine early in the process is:

a) The format and column headers/variable names of the input sample files

b) whether information from file headers is required (e.g. salinometer bath temperature and standard seawater K15 value) and can be read in or should be coded into opt_*cruise*.m.
c) whether you will want to recalculate variables or whether concentrations and salinities will be provided as-is by analysts. For salinity, it is recommended that you read in conductivity ratio values for both samples and standards, and inspect them (using msal_01.m) before coding and applying standards offsets and then calculating conductivity and salinity.

Thinking through these questions early on can help to not only make the files easier to parse (e.g. by making sure they do include columns for CTD station and Niskin, or that standards and samples are clearly denoted) but also to make sure that all required information is collected (e.g. lab temperature) and that everyone is using the same convention for any flags supplied (e.g. WOCE flags: readthedocs.io/exchange_format).

3.3.2 Checking data

As sam_*cruise*_all.nc gathers all the sample and corresponding CTD data, you can inspect and compare them however you like. Several functions are available to facilitate checking various data quantities against each other, neighbouring profiles, and/or the CTD. It is generally helpful to run them iteratively as the cruise progresses.

checkbottles_01 plots a single sample parameter for multiple stations, as well as its anomaly (either from the mean, or, if a gridded section file, ctd/grid_cruise_*.nc, has been produced, from the gridded profile), as a function of pressure and as a function of temperature; a graphical user interface enables selection and printing of outliers to be flagged in opt_*cruise*.m or investigated further.

checkbottles_02 plots multiple parameters for a single station, overlaying sample values on CTD or gridded (see above) profiles and labelling Niskin numbers. This script is particularly useful for detecting leaking or closed-at-the-wrong-depth Niskins, which will exhibit bad values for multiple parameters (including, potentially, obvious outliers in oxygen draw temperature).

mctd_evaluate_sensors compares data from a given CTD sensor (e.g. cond1, oxygen2, temp2, or can specify by serial number) to calibration (bottle sample, or for temperature, SBE35) data and to data from the other CTD, as functions of station number (a proxy for time), pressure, temperature, and the parameter itself, to enable the user to get a sense of how the sensors are behaving, look for drifts, pressure dependencies, scale factors, etc., and propose a calibration function to correct the CTD data. Calibration functions are coded in opt_*cruise*.m, and a switch allows mctd_evaluate_sensors to apply them to test out. The script also has the option to plot large-residual sample and CTD values against individual station profiles, to check whether they represent bad sample analyses, or simply larger variance in properties at a given bottle stop (e.g. in a high gradient region).

Sample flags set in opt_cruise.m under msal_01 etc. cases, and/or Niskin flags set under the mfir_01 case, can be updated to reflect problems identified here. Flags are then applied to the sam_*cruise*_all.nc file by rerunning the script under whose case they were set (e.g. msal_01, mfir_01).

Flag meanings for Niskin bottles (mfir_01): initial flags should be 2 (bottle good, sampled), 3 (leaking [and did not sample]), 4 (did not trip correctly [wire not released or bottle was seen to snap closed on landing]), 7 (unknown problem [maybe leaking but not so obviously we didn't sample]), 9 (did not sample, no further information). Then after examining data update initial flags of 7 to 2 (good), 3 (several sample values suspicious so probably leaking), or 4 (clearly sample is from a different depth, i.e. did not trip correctly), and add additional 3 or 4 flags as indicated by the data.

3.4. Sensor Calibration

Calibration functions are entered in opt_*cruise*.m under the mctd_02, ctd_cals case. They are entered as strings which describe setting dcal.*sensor* (e.g. dcal.cond1, dcal.oxygen2, or can be specified by serial number) as a function of d0.*sensor*, as well as (optionally) d0.statnum, d0.press, d0.temp[1/2] etc. Each string is the value of a field whose name is the cruise name (to prevent applying copy-pasted calibration functions from a previous cruise). Additionally, flags under this case must be set to 1 in order for the calibrations to be applied when mctd_02 is run. More details are given in set_mexec_defaults.m (under mctd_02, ctd_cals).

It is beneficial to calibrate temperature (if comparison data are available) before choosing a calibration for conductivity. Mctd_evaluate_sensors (see previous section) can help with testing the calibrations entered into opt_*cruise*.m.

3.5 Outputting data in other formats

3.5.1 1hz files for LADCP processing

3.5.2 LADCP processing for bottom depth

To run basic processing of LADCP data from cast nnn (after mout_1hzasc has been run):

> lad_linkscript_ix # to copy data from network machine

>> cd ladcp/ix

>> cfgstr.orient = 'DL'; process_cast_cfgstr(nnn, cfgstr);

This will generate plots as well as matlab files in ladcp/ix/DL_GPS/processed/nnn/

And if you have dual instruments, you can process the uplooker on its own:

>> cfgstr.orient = 'UL'; process_cast_cfgstr(nnn, cfgstr);

And both together:

>> cfgstr.orient = 'DLUL'; process_cast_cfgstr(nnn, cfgstr);

If you have the CTD 1 Hz file, you can include bottom tracking as a constraint:

>> cfgstr.orient = 'DL'; cfgstr.constraints = {'BT'}; process_cast_cfgstr(nnn, cfgstr);

SADCP data can also be used (see below).

Commented [FYL13]: check

3.5.3 WOCE exchange format CTD and bottle data

mout_exch_sam.m and mout_exch_ctd.m, respectively, write bottle sample and corresponding CTD data from sam_cruise_all.nc, and 2-dbar downcast CTD profiles from ctd_cruise_nnn_2db.nc, to WOCE exchange format (ascii) files. mout_exch_ctd.m writes one station/file at a time. File headers are customized in opt_cruise.m; the header information should include a note on which quantities are calibrated (and how, including salinity standard batch number) and which are not.

3.5.3 Summary tables (***to be updated)

station_summary.m produces a table of CTD casts, with columns including start, bottom, and end times; depth; number of bottles fired; number of salinity samples; and numbers of other samples, customized in opt_cruise.m

tsg_summary prints out/makes plots of some info from merged files (not sure this one works, some of the input files may not be current)

sam_listing is a function that just prints the CTD data from bottle firing times for a particular station

mout_sam_csv

mctd_makelists

4. Underway Data

4.1 TECHSAS/SCS/RVDAS

4.1.1 Data access

Mexec uses 'short names' to access the TECHSAS and SCS streams or RVDAS tables through lookup tables set in mtnames.m, msnames.m, or mrnames.m, respectively. Additional lines can be added to these files, and irrelevant lines commented out, as necessary.

The following Mexec Matlab commands can be used for a quick look at RVDAS data (substitute mt or ms in place of mr for TECHSAS or SCS respectively).

help mrvdas	lists the 'mr' commands (alternates: mtechsas or mscs)
mrlookd	tells you filename, start, end
mrnames	lists mexec 'shortnames', full filenames in cell array.
mrdfinfo winch	provides info about that datastream (eg winch)
mrgaps gyro_s 10s	lists gaps in datastream of more than 10s
mrposinfo([yyyy mm dd hhmm])	gives you position for that time
mrlistit	list segments of data
mrload	load data from specified table/stream and time range

4.1.2 Preparation at the start of the cruise

RVDAS

1. Install postgresql (on Mac: find out where command-line psql is on your path (possibly under /Library/) and make /usr/local/bin/psql a symbolic link to it (alternately, edit the line in m_setup.m that sets MEXEC_G.RVDAS.psql_path).
2. If necessary, edit the lines in m_setup.m setting MEXEC_G.mexec_data_root, MEXEC_G.mexec_source_root, and MEXEC_G.other_programs_root to where: you want the processed data; ocp_hydro_matlab; and the directory containing seawater and gsw toolboxes. Make sure that in the data processing directory (e.g. ~/cruise/data/) you have a subdirectory rvdas containing directories json_files/ and rvdas_csv_tmp/ (the rest will be generated automatically).
3. Determine the address (name or ip) and login credentials of the RVDAS server, as well as the location of its .json files and the database name for your cruise. Add the address, database, and username to m_setup.m, and to file ~/.pgpass, a text file containing one or more lines specifying machine:port:database:username:password. Add the directory containing the .json files (on the remote/server machine) to m_setup.m under MEXEC_G.RVDAS.jsondir. (If for some reason the files are not on the same machine as the database, see alternate B below). Also, in opt_da001.m, change skipunderway to 0.
4. Run mrjson_get_list.m, editing the list of files to exclude instruments you're not interested in. [Alternate B: If necessary, copy over the .json files and make a list file (containing full paths) manually and just do the last couple of lines of mrjson_get_list.m where it calls mrjson_load_all.m, with outfile being mrtables_from_json.m (with full path).] At this point you should be able to run mrlistit, mrlast, and mrload. You can also run mrdefine, which will output the list of tables and variables in the matlab workspace.

5. Inspect mrtables from_json.m and if necessary add instruments/tables/messages/patterns to exclude to setdef_cropt_uway.m or (probably better) opt_*cruise*.m (under case mrvdas_ingest, oopt *your ship* [e.g. sda]), and/or add more code for parsing variables/units to mrrename_tables.m, and/or add new streams and give them shortnames in mrnames.m (and if you have added new shortnames, also add directories for them in muwaydirs.m).
6. If you've added to setdef_cropt_uway.m/opt_*cruise*.m, rerun mrjson_load_all.m to regenerate mrtables_from_json.m (with more variables commented out etc.).
7. Run mrdefine, inspecting lists of original tables and renamed tables, and iterate if necessary.
8. Find and remove m_udir.m and then run m_setup.m to regenerate it.
9. Set MEXEC_G.default_navstream and MEXEC_G.default_hedstream in m_setup.m to point to the shortnames of the best position and heading streams. Possibly edit munderway_varname to add new names for lon, wind speed, etc. (alternately, you can make sure they match what was used last time by editing mrrename_tables.m, but if you want to keep different names, perhaps for comparison of different sources of wind etc., add the alternate possibilities in munderway_varname; various scripts call this and select the first match they find, so for instance any of lon, long, and longitude will be treated the same, etc.).
10. (Optional): compare table described in the json files with tables actually present in rvdas:
    js = mrshow_json_all('~/cruise/data/rvdas/original_json_files/list_json.txt');
    jslist = fieldnames(js);
    whos jslist
    t = mrgettables;
    tlist = fieldnames(t)
11. If any data streams are reported in voltage and require factory calibration coefficients to convert to physical units, enter these in opt_*cruise*.m under mday_01_clean case.

4.1.3 Automatic daily processing

Standard underway data (including navigation, surface air and water, and bathymetry) can be processed on a day-by-day basis by running

```
>> days = [ddd]; uway_daily_proc
```

where days is a vector of the yeardays you want to process, not exceeding yesterday (the last complete day). If days is not found it defaults to yesterday.

uway_daily_proc.m:

- goes through the list of underway data streams found in mrnames (or mtnames/msnames) finds which ones are present in the rvdas table list or the scs or techsas link directory, and calls mday_01.m to load them, producing a series of daily files from each data stream, located in their individual directories (e.g. bathy/singleb/singleb_jc238_d197_raw.nc). It is also possible to exclude some streams/set a limited list of streams for uway_daily_proc.m, but by default it processes everything found in both mrnames (etc.) and

m_udir.m. If no data are available on a day from a given stream, that stream is skipped.
- performs additional processing and cleaning steps on some streams by calling mday_01_clean_av.m, which has cases for different streams and calls various scripts to do automatic processing steps including renaming variables to standard names (e.g. head_gyr, depth), correcting (especially in TECHSAS) or standardising units strings, applying factory calibration coefficients (i.e. to convert from voltage to physical units) where necessary, searching for and flagging backwards time steps or duplicate times in nav streams, NaNing out-of-range values, correcting echosounder depth for speed of sound variations based on the Carter tables, and averaging bathymetry to 30-s; output files are stream_*cruise*_d*ddd*_edt.nc.
- For bathymetry, when both the singlebeam (EA600/EA640) and multibeam (EM120/EM122) centre beam streams are present, mbathy_av_merge.m is called to paste in the depths from the other instrument for subsequent comparison.
- After all days have been loaded and edited, calls m_uway_append.m to add the days' files to the appended file for each data stream (e.g. sim_cruise_01.nc). The list of daily files appended into the master files is given in the header information of that file.
- Finally, calls mnav_best.m, mwind_true.m, mtsg_medav_clean_cal.m, and (for scs) upate_allmat.m, which calculate new variables from the appended files.

The following sections contain further details of the individual data streams, manual quality control/editing steps, and the final steps operating on the appended files.

4.1.4 Navigation: additional processing

The navigation streams in whose position and heading we have most confidence are set in m_setup.m. Script mnav_best.m loads these appended files and averages and merges to a common, 30-s time base, producing a file bst_*cruise*_01.nc containing position, heading (using proper vector averaging), course and speed over ground, and distance run. This is the 'definitive' cruise navigation file.

Navigation streams, including different sources of GPS position, do not always agree to within their stated accuracies (see e.g. DY146 cruise report), so it is worth comparing the different streams carefully especially if they are being used for e.g. ADCP processing or wind calculations.

4.1.5 Meteorology: additional processing

Ship speed, position and heading from the bst navigation file are merged onto the wind data in the surfmet file. The absolute wind speed is calculated and vector averaged in one multi-step script mwind_true.m. The output files from this processing are
surfmet_*cruise*_true.nc
surfmet_*cruise*_trueav.nc
The latter file is reduced to 1-minute averages, with correct vector averaging when required. In order to avoid ambiguity, variable units are explicit in whether wind directions are 'towards' or 'from' the direction in question. The result is a bit cumbersome, but should be unambiguous if the units are read carefully.

Commented [FYL14]: still here? or later?

Commented [FYL15]: check they still are, and are correct!

Note: TECHSAS stores wind speed in m/s, but says the variable unit is knots. This is corrected in mday_01_clean_av.

Wind over the stern: The standard test of whether the relative wind processing has been done correctly would be to observe no change in the calculated absolute wind when the ship changes direction or speed. This can be misleading, since the anemometer sited on the foremast under-reads speed by a significant margin when the wind is over the stern. Therefore if either the 'before' or 'after' wind direction is over the stern, there can be a significant change in the apparent true wind speed during such manoeuvres.

Wind relative direction near 0/360: The age old problem of wind direction near the 0/360 boundary still remains. Since the anemometer is set up with 0/360 at the bow, the relative wind is very often around this heading. Even though the anemometer data are recorded at the data rate generated by the sensor (nominal 1 Hz), there is a problem with the raw data. In particular, when the wind is near 0/360, the TECHSAS files will sometimes contain headings in between, e.g. in the range 150 to 210, reminiscent of when simple numerical averaging of heading was occurring. When these bad headings are used in correct calculation of true wind, bad data are the result.

Irradiance and surface pressure

Downwelling PAR and TIR data are read into both TECHSAS and RVDAS in V rather than W/m^2, so factory calibration coefficients should be entered into opt_*cruise*.m under the mday_01_apply_fcal case.

4.1.6 Ocean surface variables: additional processing

Ocean surface variables may come in in multiple streams, e.g. tsg/ocl, sbe38dropkeel, and some are in surfmet along with wind etc.

By default, temperature and salinity are edited for bad times (e.g. when pumps were off), which can be found using mtsg_findbad.m and added to opt_*cruise*.m under the mtsg_medav_clean_cal case, and averaged to 30 s using mtsg_medav_clean_cal.m. They may then be calibrated by comparison with bottle samples (salinity) or CTD 5-m temperature. TSG salinity samples are read in at the same time as CTD samples using msal_01.m (see Section 3). mtsg_bottle_compare.m helps compare salinity and choose a time-varying calibration which can be added to opt_cruise.m under tsgsal_apply_cal*** case. mtsg_medav_clean_cal.m is then rerun with flag usecal set to 1 to incorporate the calibrated data. Finally, mtsg_surfmet_merge combines the tsg data with the other surfmet variables.

Temperature variables: On TECHSAS, sea surface temperature (that is, temperature at the seawater intake) is called temp_r or temp_m, while the housing temperature (temperature inside the inline CTD, applicable to the conductivity measurements) is called temp_h. On SCS on the JCR, there are two sea surface temperature sensors, sstemp and sstemp2; the conductivity measurement temperature, tstemp, and the fluorometer temperature, sampletemp. On the Cook, temp_housing is the temperature applicable to conductivity, temp_remote that at the seawater intake

(actually a little way up the pipe), and sbe35dropkeel_temp the temperature on the hull.

You can re-run mtsg_findbad.m, mtsg_bottle_compare.m, and m_tsg_medav_clean_cal.m as many times as required to get a clean record.

If you want to check a calibration already applied, edit the switch at the beginning of mtsg_bottle_compare.m from 'uncal' to 'cal' and rerun.

4.1.7 Bathymetry: additional processing

Following ingesting and combining daily data in m_daily_proc.m, bathymetry data can be cleaned by interactive script mbathy_edit_av, which allows the user to outline and remove bad data points from the EA600 and the EM120/EM122 centre beam for each day.

4.2 VMADCP

4.2.1 Acquiring and processing/editing data

4.2.1.1 VMDAS plus CODAS

Vessel mounted ADCP data processed using the old University of Hawaii CODAS software (Matlab/Python hybrid version, see /local/users/pstar/cruise/sw/uh_adcp/programs/index.html for documentation) can be loaded into daily and appended Mstar files using mcod_01.m, mcod_02.m, and mcod_mapend.m. The latter sorts by time, so sequences can be added in any order or any number of times.

The full round of processing from VMDAS to Mstar, including the calls to CODAS quick_adcp.py and gautoedit.m, can be run for a sequence or set of sequences using wrapper scripts vmadcp_proc.m and vmadcp_edit.m, as follows:

>> doall = 1; vmadcp_proc % runs vmadcp_linkscript to sync and link files; writes q_py.cnt using initial angle and amplitude set in opt_cruise.m, and calls quick_adcp.py to load data into database; calls mcod_01, mcod_02, mcod_mapend

> This script will prompt for the instrument to use (75 or 150, probably), sequence number or vector of sequence numbers, and possibly the mode (narrowband or broadband), if not set in opt_cruise.m
>
> If this fails on "cannot find asetup", it may indicate there's not enough data in the sequence

Optional additional processing (better done on multiple sequences, and can be done back at home):

> At some point, examine cal/botmtrk/btcaluv.out and cal/watertrk/adcpcal.out (in whichever sequence directories they are found; they will not be generated for every sequence), to refine the angle and amplitude calibrations. Add these to the vmadcp_proc aa75 and/or aa150 cases in opt_cruise.m. Once you have done this, you can rerun vmadcp_proc.m, setting doall = 2:

>> doall = 2; vmadcp_proc % writes q_pyrot.cnt using angle and amplitude set in opt_cruise.m and calls quick_adcp.py to apply angle and/or amplitude corrections; calls mcod_01, mcod_02, mcod_mapend

>> vmadcp_edit % writes q_pyedit.cnt, calls gautoedit.m to enable interactive data editing, calls quick_adcp.py to apply edits; calls mcod_01.m, mcod_02.m, mcod_mapend.m

4.2.1.2 UHDAS plus CODAS

In pycodas terminal (prompt should start with (pycodas) bash-4.2$)

0) If you previously applied calibrations (angle, amplitude, xducer_dx or _dy) to data in postprocessing:
rm -rf
/local/users/pstar/cruise/data/vmadcp/postprocessing/DY113/proc_editing/os*

1) uhdas_01 #syncs data from acquisition computer to koaeula

2) uhdas_02 #syncs to postprocessing/proc_editing (and makes links if necessary)

3) uhdas_03 #copies previously made edits (archived in proc_archive) to proc_editing so they will be applied to newly expanded dataset; adds new data to dataset

4) cd ~/cruise/data/vmadcp/postprocessing/DY113/proc_editing

5) cd os150nb
6) dataviewer.py -e & #edit using selectors and/or thresholds; remember to apply edits to every segment of time series
7) quick_adcp.py --steps2rerun apply_edit:navsteps:calib --auto
8.1) cat cal/watertrk/adcpcal.out
8.2) cat cal/watertrk/guess_xducerxy.out
8.2) cp -Rp . ../os150nb_a
9) quick_adcp.py --steps2rerun apply_edit:rotate:navsteps:calib --rotate_amplitude amp --rotate_angle ang --xducer_dx dx --xducer_dy dy --auto #if adcpcal.out shows amplitude or angle calibration needed
10.1) cd ..
10.2) dataviewer.py -c os150nb_a os150nb & #to see the effect of the edits and calibration. If you are not happy with them, repeat 6)-9). When you are happy:
10.4) rm -rf os150nb_a #only keep the edited, calibrated version here now

11) cd ../os75nb #and repeat 6)-10)

12) cd ..
13) dataviewer.py -c os75nb os150nb & #to compare two instruments and check for additional needed edits; if any noticed, follow steps above

14) uhdas_04 #generates .nc files in ~/cruise/data/vmadcp/DY113/postprocessing/proc_edit/os75nb/contour/ and os150nb/contour/

15) uhdas_05 #syncs proc_edit to proc_archive

4.2.2 Output VMADCP station data for LADCP processing

In matlab

16 – 19) vmadcp_stations_to_ladcp(start_station, end_station)

16) os = 75; nbb = 1; mvad_01
os = 150; nbb = 1; mvad_01
%makes mstar format files containing vmadcp data in
/local/users/pstar/cruise/data/vmadcp/mproc/

17) stn = nnn; cast = 'ctd'; os = 75; nbb=1; mvad_03
stn = nnn; cast = 'ctd'; os = 150; nbb=1; mvad_03
%vmadcp upper ocean velocity profiles corresponding to ctd station nnn; run for each station for which new or edited vmadcp data are available

18) mvad_for_ladcp('ctd',nnn,75,1)
mvad_for_ladcp('ctd',nnn,150,1)
%makes file of station nnn data to be used as constraint in LADCP processing

19) cfgstr.orient = 'DLUL'; cfgstr.constraints = {'GPS';'BT';'SADCP'};
process_cast_cfgstr(nnn, cfgstr);

Appendices

A. Quick guide to using Mexec functions and interacting with Mstar files

You must have ocp_hydro_matlab on your Matlab path. Each time you start Matlab:

>> m_setup % to add paths and set global variables

If you clear the workspace variables:

>> m_global % to set global variables

To load an Mstar file:

>> [d, h] = mload(filename, '/'); % loads all data and header information from filename

Both d and h are structures, d containing "data" and h containing "header" information. The names of the variables in the structure d are also given in h.fldnam, while their units are in h.fldunt. Times in Mstar files (generally in seconds) are relative to the time specified by the vector h.data_time_origin [yyyy mm dd HH MM SS] or [yyyy mm dd]. The .nc suffix of filename is not required but the path is.

To save variables from the workspace to an Mstar file:

>> mfsave(filename, d, h) % where d and h are like Mstar data and header structures; see help mfsave for other options

Help mode: on Linux/Unix systems, get_cropt can be called on the Matlab command line to find out about options:
>> help_cropt = 1; scriptname = ''; oopt = ''; get_cropt
Displays a list of scriptnames and oopts in any of the setdef_cropt_*.m.
>> help_cropt = 1; scriptname = scriptname; oopt = ''; get_cropt
Displays a list of all calls to get_cropt in file scriptname.m, if this file exists, and if not, of all files which use scriptname = scriptname to call get_cropt.
>> help_cropt = 1; scriptname = scriptname; oopt = oopt; get_cropt
Displays which of the setdef_cropt_*.m contains this (scriptname, oopt) case (and therefore a help string explaining it), along with a list of opt_*.m files which contain it (and therefore examples of modifying the defaults).

Commented [FYL16]: move this to appendix/tips section below

Commented [YF17R16]: Still works?

* If a script crashes while writing a file, that file may be left with an "open to write" flag and subsequent calls or scripts will fail. Reset using "mreset". Scripts crashing may also result in a proliferation of temporary wk*.nc files (which would normally be removed at the end of the script); these contain the date of generation so they can be cleaned up periodically.

* Once or twice in Mstar scripts we got a mysterious error message along the lines of "not a binary mat file" or "a preference with that name or group already exists". Just re-run the program and next time it will likely be fine; if not, try >> clear all; m_setup; if that doesn't work, exit and restart Matlab.

* Occasionally there may be an error with a library, possibly related to writing the NetCDF files. Restarting Matlab usually clears this up.

Because of the interactive prompting options, sometimes a prompt will appear even if you have passed input arguments in MEXEC_A.MARGS_IN. If Matlab says "Busy|", you don't need to do anything even if it looks like it is asking for input.

You do not need to be in a particular directory to run the mexec programs, as long as you specify full file paths when using e.g. mload (all the processing scripts are set up to specify full file paths).

* In matlab/Mstar to open a Mstar nc file: mload

To save data to struct array d, header info to array h:
>> [d h]=mload('filename','/') % '/' means all vars, or you can list the ones you want.

* >> m_read_header(file) allows you to read the header info only

* >> mhistory: returns to you the text you need to copy into a script to recreate the program steps you just ran (if you have been using MEXEC_A.MARGS_IN and not mfsave.m)

B. Processing checklists

	Cruise: DY146	CTD number *nnn*				
	Step	Notes and output				
A	DatCnv.psa AlignCTD.psa CellTM.psa	DY146_CTD_*nnn*.cnv DY146_CTD_*nnn*_align.cnv DY146_CTD_*nnn*_align_ctm.cnv				
T	ctd_syncscript	RAW_CTD_FILES, ASCII_FILES				
M	stn = *nnn*; ctd_all_part1 mctd_01 mctd_02 mctd_03 mdcs_01 mout_1hz_asc	ctd_dy146_*nnn*_raw.nc ctd_dy146_*nnn*_raw_cleaned.nc ctd_dy146_*nnn*_24hz.nc ctd_dy146_*nnn*_psal.nc dcs_dy146_*nnn*.nc ctd.*nnn*.02.asc				
M	stn = *nnn*; mdcs_03g	Select cast (start and) end dcs_dy146_*nnn*.nc				
M	Edits to opt_jc238.m:	a) If Niskins leaking or misfired, add flags under mfir_01 case b) If cast not full-depth, add water dep in best_station_depths case				
M	stn = *nnn*; ctd_all_part2 mctd_04 mfir_01 mfir_03 mwin_01 mwin_to_fir mfir_to_sam station_summary mdep_01	ctd_dy146_*nnn*_2db.nc ctd_dy146_*nnn*_2up.nc fir_dy146_*nnn*.nc win_dy146_*nnn*.nc bot_dy146_*nnn*.nc sam_dy146_all.nc				
M	stn = *nnn*; mctd_checkplots					
M	stn = *nnn*; mctd_rawshow	Look for spikes, bad ranges				
M	stn = *nnn*; mctd_rawedit and/or edits to opt_jc238.m under mctd_02, rawedit_auto	If edits required ctd_dy146_*nnn*_raw_cleaned.nc				
M	stn = *nnn*; ctd_all_postedit mctd_02 mctd_03 mctd_04 mfir_03 mfir_to_sam mout_1hz_asc	If edits done ctd_dy146_*nnn*_raw_cleaned.nc ctd_dy146_*nnn*_24hz.nc ctd_dy146_*nnn*_psal.nc ctd_dy146_*nnn*_2db.nc ctd_dy146_*nnn*_2up.nc fir_dy146_*nnn*.nc sam_dy146_all.nc ctd.*nnn*.02.asc				
T	lad_linkscript_ix	*nnn*DL000.00, *nnn*UL000.00				
M	run_proc_ladcp (first 3 rounds)	Plots and .mat files				

A = on acquisition computer, T = in shell (terminal), M = In Matlab

D. Known bugs and future changes

D.1. Bugs

Errors involving MEXEC_A.MARGS_OT if a file is interrupted: clear all, run m_setup, and try again.

Matlab netcdf errors about "a preference with that name or group already exists" occur, apparently when multiple Matlab sessions are accessing the low-level netcdf functions at the same time. The only response is to start again at the point of interruption and hope there won't be more coincidences, although in extreme cases it may be necessary to restart one or multiple Matlab sessions. The error seems particularly likely to be triggered by running m_setup, so one adaptation is to keep a couple of Matlab sessions running and set up, rather than starting a new one while Mexec scripts are running in another window . . . however this only reduces rather than eliminates the occurrence of these errors.

```
Error using netcdflib
Library failure in call to open. eacces:permissionDenied. Error message from the
NetCDF
library: "Permission denied".

Error in netcdf.open (line 50)
            [varargout{:}] = netcdflib ( 'open', filename, mode );

Error in nc_attput>nc_attput_tmw (line 41)
ncid  =netcdf.open(ncfile, nc_write_mode );

Error in nc_attput (line 28)
   nc_attput_tmw ( ncfile, varname, attribute_name, attval )

Error in m_openio (line 20)
nc_attput(ncfile.name,nc_global,'openflag','W'); % set to W if file is open to write.
Usual state is R.

Error in mheadr (line 22)
ncfile = m_openio(ncfile);

Error in mctd_02a (line 98)
mheadr

Error in ctd_all_part1 (line 16)
stn = stnlocal; mctd_02a; %rename variables following templates/ctd_renamelist.csv
```

This means you should remove ctd/ctd_cruise_nnn_raw.nc (or even ctd/ctd_cruise_nnn*.nc) and start again. Make a note on the processing logsheet.
If the beginning part of this error occurs when running smallscript_botnav, in terminal:
chmod u+w ctd/ctd_dy113_nnn_raw*.nc

Exit with error because file
/local/users/pstar/dy113/mcruise/data/ctd/ctd_dy113_002_2db.nc is already open for write
It may be the case that this program has crashed or been interrupted before, leaving the write flag set in the file
If required you can reset the write flag using

mreset(filename) or mreset(ncfile)

where filename or ncfile.name is a char string containing the name of the mstar file

Error in m_openot (line 27)
  ncfile = m_exitifopen(ncfile); % exit if write flag set

Error in mcalc (line 57)
ncfile_ot = m_openot(ncfile_ot);

Error in mctd_04 (line 176)
mcalc

>> mreset('ctd/ctd_dy113_002_2db.nc')

About to reset mstar openflag on file ctd/ctd_dy113_002_2db.nc.
Do you really want to do this ?
  Reply y/yes. Default is no. y

File ctd/ctd_dy113_002_2db.nc has been modified

Attempt to reference field of non-structure array.

Error in mtposinfo (line 51)
  tin = pdata.time+MEXEC_G.uway_torg;

Error in mctd_02a (line 111)
  [botlat botlon] = mtposinfo(tbotmat);

Error in ctd_all_part1 (line 16)
stn = stnlocal; mctd_02a; %rename variables following templates/ctd_renamelist.csv

run techsas_linkscript in terminal (have to do this each new day UTC)

some error about f.ladcpdo

run lad_linkscript_ix and make sure LADCP files were copied/links were made

Notes: If you run m_setup.m with skipunderway set to 0 (set in opt_da001.m) before mrtables_from_json.m is available, it will get stuck. If one of the matlab scripts seems to freeze and not respond to ctrl-C, try hitting enter; if this unsticks it (but it fails), it may be getting stuck on the system command where it tries to run the psql string; check ~/.pgpass, and if that doesn't fix it, try putting a debug point in mr_try_psql.m to check the psql string and maybe try running it from a terminal directly.