Reference Manual for Quality Control of Subsurface Currents Data (QCSCD)

Version 1.1

By

Charles Sun

1. INTRODUCTION

Conventionally, there are two kinds of ocean currents measurements. One is remote sensing measurements and the other is In-situ measurements. Remote measurements of ocean surface currents and their signatures, including HF (high frequency) Doppler radar systems, satellite-borne sensors, give us a broad overview, and tell us how currents vary in time and space, while In-situ measurements, either Eulerian measurements or Lagrangian measurements, provide us detail information to understand what controls the flow of water in the interior of the oceans.

The objectives of quality control subsurface currents (QCSC) are:

a) To ensure the data consistency within a single data set and within a collection of data sets, and
b) To ensure that the quality and errors of the data are apparent to the user who has sufficient information to assess its suitability for a task.

This manual is a summary of NODC currents data quality control (QC) procedures focused on Eulerian measurements of ocean currents. It is a work-in-progress that reflects to the up-to-dated QC testing procedures designed for the subsurface currents data sets archived at the NODC. This manual is written for the experienced operator with sufficient computer programming skills, including but not limited to *FORTRAN*, Perl, and *R*, and also for those who are just entering the field.

Annex A provides an example of using the quality control subsurface current data (QCSCD) script in R, followed by a complete list of the script in Annex B and the standard operating procedure of converting originator's formats to the NODC ocean currents NetCDF format in Annex C.

2. QUALITY CONTROL

2.1. Summary of Procedures

The conceptual design of the QCSC is based upon the IOC/CEC's "Manual of Quality Control Procedures for Validation of Oceanographic Data" *(IOC/CEC Manual, 1993)*, which described that quality control procedures for meteorological and oceanographic data should comprise two distinct aspects, i.e.,

1) Automatic quality control and
2) Oceanographic and meteorological assessment.

The NODC QCSC scheme adapted the IOC's approach and the procedures are:

a) Select a time series data set from the NODC Archive Management System.
b) Inspect Metadata Required for Quality Control.

c) Convert data into the NODC GOCD NetCDF Format.
d) Input converted data set to the QCSC Scheme.
e) Generate scatter and time series plots.
f) Perform quality tests and set flags.
g) Generate test results report.
h) Generate quality Controlled data sets.
i) Conduct lower level oceanographic assessment, and
j) Conduct higher level oceanographic assessment, if necessary.

Figure 1 is a flow diagram of the above steps of quality control subsurface currents. The above Steps (a) to (h) are the so-called automatic QC, while Steps (i) and (j) are the two levels of oceanographic data assessments. Step (i), is a lower (first) level oceanographic assessment and is, basically, a visual inspection of graphical products (plots) generated in Step (e). Screening plots essentially allows the quality control of data that we receive with checks being made to ensure that the data are free from instrument-generated spikes, gaps, spurious data at the start and end of the record and other irregularities, for example long strings of constant values. These problems are not immediately obvious when just looking at large columns of data. When suspicious values are seen, flags are applied to the data points in question, as a warning to end users. Screening is after all, a procedure based very much on instinct and perception, and opinions will inevitably differ from person to person.

The higher (second) level of oceanographic assessment generally involves the application of further analytical methods, e.g. harmonic analysis, rotary spectrum analysis, and principal component analysis, and detailed data-point by data-point comparisons with other available data. It also involves the validation of anomalous data for which the causes are not readily identifiable, and this may include the investigation of particular process-response mechanisms in the data, e.g. extreme meteorological forcing, inertia oscillations or internal tides in current meter data.

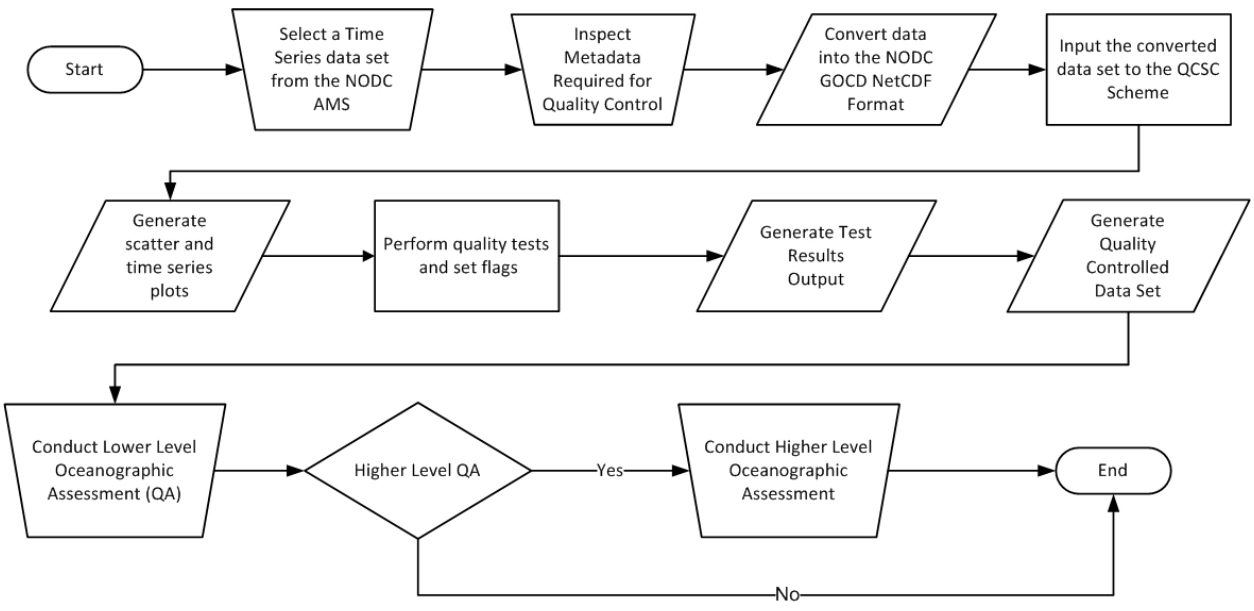


Figure 1 Flow Diagram of Quality Control Procedures for Subsurface Currents Data.

It is envisioned in the context of the minimum requirements for data validation, that any oceanographic assessment should include at least the lower level checks. Some higher level checks should also be undertaken if the data require them and are sufficient for them to be undertaken.

2.2. Quality Flagging

Every current data set is evaluated using a series of QC tests and the results of those tests are recorded by assigning flags to each test. Flags are set in the data record to identify (1) the tests that were performed, and (2) results of those tests. Quality control test designs were based on The QC flags indicating the levels of data quality are shown in Table 1.

Table 1 Flags for subsurface currents data

Flag	Description
1	Data appear to be correct.
2	Data have failed one (or more) QC test but the errors are correctable and useable to users.
3	Data have failed one (or more) QC test and are considered to be either doubtful or of high interest to users for future high level QC tests.
4	Data have failed one (or more) critical QC test and are considered to be erroneous.
9	The element is missing: used as a placeholder.

Because quality assessment can be performed by various users, it is possible that data flagged as doubtful by one user will be considered acceptable by other users or vice versa. Flags can be changed by any processing center as long as a record is kept of what changes are.

2.3. Pre-processing

The checklist of metadata required for QC and example information below shows the information used by NODC to ensure that the data are adequately described:

2.3.1. Data Originator Details
- 2.3.1.1. Name of country responsible for data
- 2.3.1.2. Name of organization responsible for data
- 2.3.1.3. Project Name (if applicable)

2.3.2. Mooring/Instrument Details
- 2.3.2.1. Instrument category e.g. Paddle wheel current meter
- 2.3.2.2. Mooring/Rig Number e.g. 1234
- 2.3.2.3. Instrument model and manufacturer e.g. Aanderaa RCM7 Current Meter
- 2.3.2.4. Principle of measurement, e.g. Vector averaged current
- 2.3.2.5. Additional notes on mooring structure and notes on performance of mooring
- 2.3.2.6. Latitude and Longitude of mooring (degrees) e.g. 53.85°, -3.26°
- 2.3.2.7. Time zone e.g. GMT/UTC
- 2.3.2.8. Site Area and Name of Site
- 2.3.2.9. Method of position fix e.g. GPS
- 2.3.2.10. Water column (Sea Floor) Depth (m) e.g. 352m
- 2.3.2.11. Depth of meter or shallowest sensor (e.g. ADCP bin) e.g. 300m

2.3.3. Timing Details
- 2.3.3.1. Date and Time of Deployment (UTC)
- 2.3.3.2. Date and Time of start of usable data (UTC)
- 2.3.3.3. Date and Time of Recovery (UTC)
- 2.3.3.4. Date and Time of end of usable data (UTC)
- 2.3.3.5. Nominal time interval between successive data cycles in series (second)
- 2.3.3.6. Type of sampling (e.g. instantaneous, averaged)

2.4. QC Test Descriptions

The subsurface currents data quality control tests (SCDQCS) are used to provide objective information about data quality by recording results of well-defined tests. These tests are described in more detail in following sections.

2.4.1. Platform Identification test

This test is the very first to be done. It is recognized that knowledge of the instrumentation used to make an It is recognized that knowledge of the instrumentation used to make an observation can be useful in the assessment of the quality of the data. Likewise, knowledge of the platform from which the data were collected can also be used. The present version of this manual suggests tests that make use of instrumentation knowledge if available. It is expected that subsequent versions of the Manual will improve on this.

2.4.2. Impossible Date/Time test

This tests if the date and time of the observation is sensible.

2.4.3. Impossible location test

This tests if the location of the observation is sensible. It does so by simply checking that the latitude and longitude have possible values. This test begins by checking if the latitude lies between 90 degrees south and 90 degrees north inclusive and the longitude must lie between 180 degrees west and 180 degrees east. If it does, processing passes immediately to next test. If not, the quality flag is set to be "2".

2.4.4. Position on land test

This tests if the location of the observation is on land or water. It does so by comparing the location with a file of known bathymetric values. QCSC uses the "Bedrock" (base of the ice sheets) version of the so-called ETOPO1 global relief model of Earth's surface, developed by the National Geographic Data Center. ETOPO1 is a one arc-minute global relief model of Earth's surface that integrates land topography and ocean bathymetry. It was built from numerous global and regional data sets and can be downloaded from http://www.ngdc.noaa.gov/mgg/global/global.html .

If the sounding of the station is present, then it is compared with the ETOPO1 bathymetric value at the location of the station. If they agree to within 10%, then processing passes to the next test. If the sounding of the station is not available, the ETOPO1 bathymetric value at the location of the station is positive, the quality flag is set to be "3". Otherwise, the quality flag is set to be “2”.

2.4.5. Global Impossible Parameter Values

This tests the global impossible values of current speeds should not exceed the maximum speed of five (5) and the minimum current speed should be 0.01 m/s. All current directions should lie between 000^{o}and 360^{o}.

This test uses a quality control parameter, known as percent good test (U.S. Integrated Ocean Observing System, 2013), to determine whether the data are sufficient to provide the required data quality.

The following table shows how to set flags, based upon the conditions of the percentages of the values of horizontal velocity fall within the global maxim speed limit, *VELmax*, of *5 m/s* (*PHVmax*) and minimum speed limit, *VELmin*, of *0.01 m/s* (*PHVmin*).

Flag	Description	Instruction
1	Data appear to be correct. The values of horizontal velocity components fall within the global speed limits.	*PHVmax*(VEL >= 5 m/s) < 5%
2	Data have 5% of values fall outside the expected global speed limits.	*PHVmax*(VEL >= 5 m/s) >= 5%
3	Data have 50% or more of values exceed global maxim speed limit.	*PHVmax*(VEL >= 5 m/s) >= 50%
4	Data have 75% or more of values exceed global maxim speed limit or 95% or more of values below global minimum speed limit.	*PHVmax*(VEL >= 5 m/s) >= 75% or *PHVmin*(VEL <= 0.01m/s) >= 95%

2.4.6. Regional Impossible Parameter Values

This test is currently not yet implemented.

2.4.7. Spikes

Spikes are usually considered as singular points which are out-of-range when compared to the surrounding values. This tests the level of noisiness of the data to determine the data quality, which is the percentage of observed values lie outside an envelope of permitted values (PSK). The test begins with a pre-defined, allowable level of noisiness, which is five (5) times of the standard deviation of the observed values (VEL_{sdv}).

Flag	Description	Instruction
1	Data appear to be correct. The values of horizontal velocity components fall within the envelope of speed limits.	Horizontal current speeds have less than five (5) percentages of values fall within the envelope of speed limits, i.e., PSK < 5% Where PSK is the percentages of horizontal velocity >= $5VEL_{sdv}$
2	Data have 5% or more but than 50% of values fall outside the expected the envelope of speed limits.	5% <= PSK <=50% Where PSK is the percentages of horizontal velocity >= $5VEL_{sdv}$.
3	Data have 50% or more but less than	50% <= PSK <=75%

	75% of values exceed the envelope of speed limits.	
4	Data have 75% or more of values exceed the envelope of speed limits.	PSK >= 75%

2.4.8. Constant Current Speed

This checks the occurrence of constant values on the current speed being measured. It depends on the sampling interval used and the resolution of the sensor and recording equipment. The last factor has not been specifically included in this test, and therefore should be considered in the assessment of any data failing the tests. Constant current speeds are uncommon although theoretically two consecutive values may be the same. A flag should be set against each current speed data point which is equal in value to the two previous values, regardless of the sampling interval.

Flag	Description	Instruction
1	Data have five (5) percentages or less of current speeds are equal in value to the two previous values (PSV).	PSV < 5%
2	Data have 5% or more but less than 50% of are equal in value to the two previous values.	5% <= PSV < 50%
3	Data have 50% or more but less than 75% of value are equal in value to the two previous values.	50% <= PSV<75%
4	Data have 75% or more of rate of value are equal in value to the two previous values.	PSV >= 75%

2.4.9. Constant Current Direction

Almost constant current directions may be generated by topographic effects.

The following numbers of consecutive equal values are allowed depending on sampling interval:

Δt (min)	Number of consecutive equal values
5	12
10	6
15	4
20	3
30	2
60	2

A flag should be set against each current direction data point which is equal in value to the previous 12, 6, 4, 3, or 2 previous values, (as applicable).

Flag	Description	Instruction
1	Data have five (5) percentages or	PSV < 5%

	less of current speeds are equal in value to the two previous values (PSV).	
2	Data have 5% or more but less than 50% of are equal in value to the two previous values.	5% <= PSV < 50%
3	Data have 50% or more but less than 75% of value are equal in value to the two previous values.	50% <= PSV<75%
4	Data have 75% or more of rate of value are equal in value to the two previous values.	PSV >= 75%

2.4.10. Rate of Change in Time

This test checks the rate of change in time for current speed and is applicable for subsurface currents can be considered to be sinusoidal functions with definable expected differences between sampling points.

The theoretical differences between two consecutive current speed samples u_1 and u_2 for various sampling intervals (Δt) assuming a smooth sinusoidal semi-diurnal tidal current with a period of 12.42 hours are given below:

Δt(min)	theoretical $\lvert u_1 - u_2 \rvert$	factor	allowable $\lvert u_1 - u_2 \rvert$
5	0.0422 u	2.0	0.08 m/s
10	0.0843 u	1.8	0.15 m/s
15	0.1264 u	1.6	0.20 m/s
20	0.1685 u	1.5	0.25 m/s
30	0.2523 u	1.4	0.35 m/s
60	0.5001 u	1.2	0.60 m/s

where u is the orthogonal tidal current amplitude.

Flag	Description	Instruction
1	Data have five (5) percentages or less of current speeds change rate (PSR) in time exceeds the allowable speed change, given sampling intervals.	PSR < 5%
2	Data have 5% or more but less than 50% of values fall outside the expected the allowable speed rate of change in time.	5% <= PSR <50%
3	Data have 50% or more but less than 75% of rate of change in time exceed the allowable speed difference.	50% <= PSR <75%
4	Data have 75% or more of rate of	PSR >= 75%

	change in time exceed the allowable speed difference.	

2.4.11. Visual Inspection

This is the final step in the procedures of quality control subsurface currents data. It is a screening review of the QC tests to assess if the quality flag of each data set is properly set. This test involves the visual inspection of time series plots and bivariate scatter diagrams to assess the test flags are set properly in according with the patterns or trends outliers or anomalous gaps in the data. The general features of the data are also compared with those for the same area from any other available sources. The following figure shows an example of the outcome of the QCSCD.

3. SUMMARY

The design of quality control subsurface currents in this manual has been compiled from various sources, including UNESCO/IOC and US/IOOS programs. The nine (9) tests identified apply to current observations from Eulerian measurements excluding from acoustic Doppler current profiler observations (ADCP). The individual tests are described and the procedures have been developed and tested at the U.S. NODC.

4. REFERENCES

U.S. Integrated Ocean Observing System, 2013. Manual for Real-Time Quality Control of In-Situ Current Observations: A Guide to Quality Control and Quality Assurance of Acoustic Doppler Current Profiler Observations. 45pp.

UNESCO, 1993. IOC Manual and Guides 26, Manual of Quality Control Procedures for Validation of Oceanographic Data, Section 2.2, Appendix B: Current Meter Data. Prepared by CEC: DG-XII, MAST and IOC:IODE. 436pp. Available online at: http://unesdoc.unesco.org/images/0013/001388/138825co.pdf

UNESCO, 2010. IOC Manual and Guides No.22, Revised Edition, GTSPP Real-Time Quality Control Manual. First Revised Edition. 145pp.

ANNEX A: Example of the Procedures of Quality Control Subsurface Ocean Currents Data

The procedures used in the present quality control subsurface ocean currents data are as follows:

1) Create a working directory, */nodc/users/csun/Projects/gocd*, in the NODC user's directory.
2) Select a time series data set to be quality controlled from the NODC Archive Management System , for example, "9700246",
3) Create a directory with a directory name beginning with A and the accession number in the working directory created in Step, for example ,"A9700246".
4) Create three subdirectories, *"plots"*, *"qcd"*, and"*raw"*, in the directory *"A9700246"* .
5) Convert the originator's data into the NODC GOCD NetCDF format and place the converted NetCDF format data in the *"raw"* directory.
6) Invoke the "QCSCD.R" script at the Linux command prompt by typing:

   Rscript qcscd.R –accession A9700246 –dt 60 [-verbose | -log filename]

   Where the command line options, –accession and –dt, are mandatory, which specifies the name (of the input file location) and the sampling intervals in minutes of the data set. The other command line option, -verbose, is optional. Once it is set, QCSCD will output quality control results to the terminal. Otherwise, the quality control results will be saved to a file located in the directory created in Step 3. The default filename is "*gocd_A9700246.log*". In addition, the quality control results can be written to a file specified in another command line option, *-log filename*.

   The success run of QCSCD will save the QCed files with QC flags to the "*qcd*" directory.

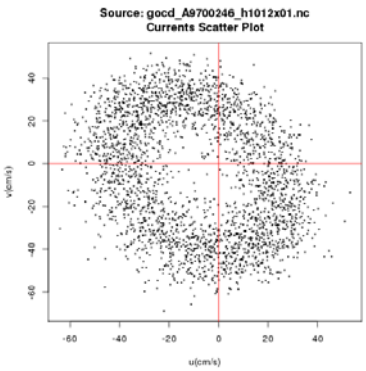
7) Beside the default log file, "*gocd_A9700246.log*", there are two files will be generated, "*gocd_A9700246_qcf.csv*" and "*A9700246_plots.html*".

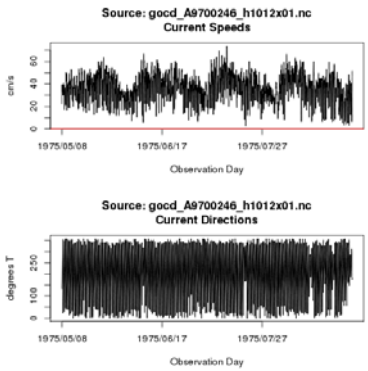
   The "*A9700246_plots.html*" is a HTML file containing metadata, the QC flags of quality control tests, scatter, and time series plots of the input file. It is primarily used at the last step, visual inspection, of quality control procedures.

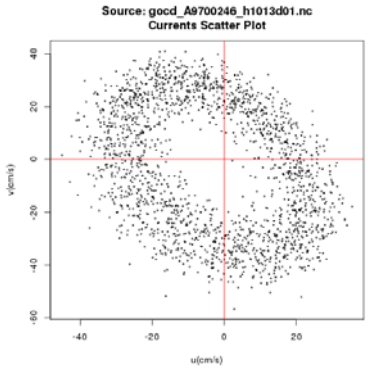
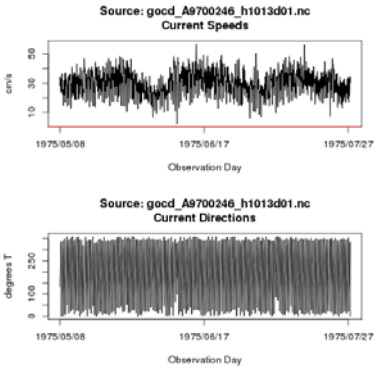
Table 1 Partial list of A9700246_plots.html located at /nodc/users/csun/Projects/gocd/A9700246 directory.

Plot No. 1
Loc.: (40.868N 67.404W)
Instrument Type: V-0208
Inst. Depth: 45.0 m
Seafloor Depth: 85.0 m
ETOPO1 bathymetry: 81 m

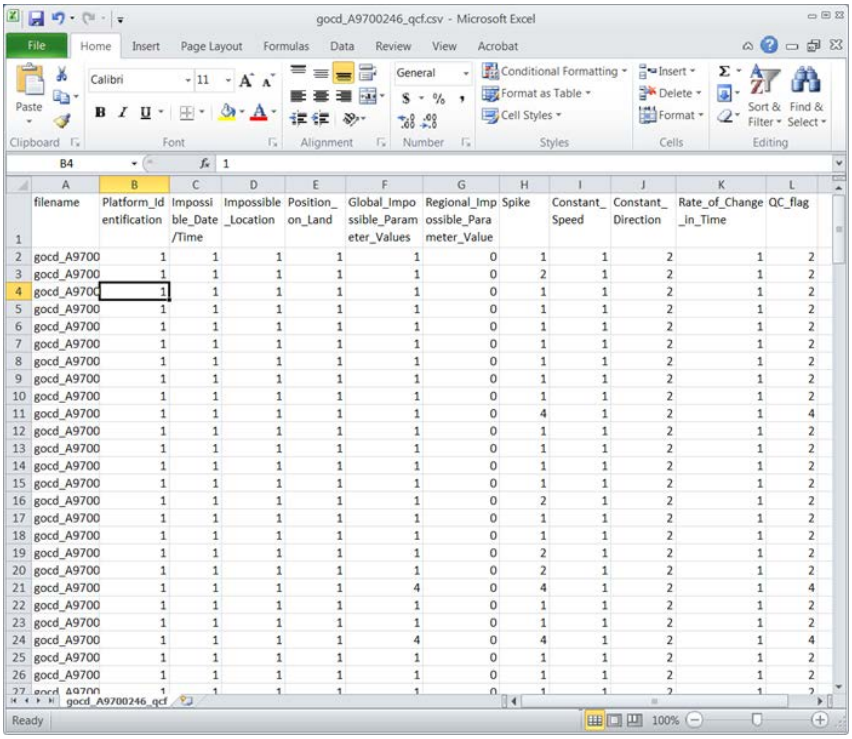
Test Name	Test Flag
Impossible_Date/Time	1
Impossible_Location	1
Position_on_Land	1
Global_Impossible_Parameter_Values	1
Regional_Impossible_Parameter_Value	0
Spike	1





Constant_Speed 1 Constant_Direction 2 Rate_of_Change_in_Time 1 Overall QC flag 2		
Plot No. 2 Loc.: (40.868N 67.404W) Instrument Type: V-0166 Inst. Depth: 75.0 m Seafloor Depth: 85.0 m ETOPO1 bathymetry: 81 m **Test Name / Test Flag** Impossible_Date/Time 1 Impossible_Location 1 Position_on_Land 1 Global_Impossible_Parameter_Values 1 Regional_Impossible_Parameter_Value 0 Spike 2 Constant_Speed 1 Constant_Direction 2 Rate_of_Change_in_Time 1 Overall QC flag 2		

The "*A9700246_acf.csv*" file contains the QC flags of each tests and an overall QC flag indicating the level of quality of the data. Figure 1 is a screenshot of the first several lines of the file.



	filename	Platform_Identification	Impossible_Date/Time	Impossible_Location	Position_on_Land	Global_Impossible_Parameter_Values	Regional_Impossible_Parameter_Value	Spike	Constant_Speed	Constant_Direction	Rate_of_Change_in_Time	QC_flag
2	gocd_A9700	1	1	1	1	1	0	1	1	2	1	2
3	gocd_A9700	1	1	1	1	1	0	2	1	2	1	2
4	gocd_A9700	1	1	1	1	1	0	1	1	2	1	2
5	gocd_A9700	1	1	1	1	1	0	1	1	2	1	2
6	gocd_A9700	1	1	1	1	1	0	1	1	2	1	2
7	gocd_A9700	1	1	1	1	1	0	1	1	2	1	2
8	gocd_A9700	1	1	1	1	1	0	1	1	2	1	2
9	gocd_A9700	1	1	1	1	1	0	1	1	2	1	2
10	gocd_A9700	1	1	1	1	1	0	1	1	2	1	2
11	gocd_A9700	1	1	1	1	1	0	4	1	2	1	4
12	gocd_A9700	1	1	1	1	1	0	1	1	2	1	2
13	gocd_A9700	1	1	1	1	1	0	1	1	2	1	2
14	gocd_A9700	1	1	1	1	1	0	1	1	2	1	2
15	gocd_A9700	1	1	1	1	1	0	1	1	2	1	2
16	gocd_A9700	1	1	1	1	1	0	2	1	2	1	2
17	gocd_A9700	1	1	1	1	1	0	1	1	2	1	2
18	gocd_A9700	1	1	1	1	1	0	1	1	2	1	2
19	gocd_A9700	1	1	1	1	1	0	2	1	2	1	2
20	gocd_A9700	1	1	1	1	1	0	2	1	2	1	2
21	gocd_A9700	1	1	1	1	4	0	4	1	2	1	4
22	gocd_A9700	1	1	1	1	1	0	1	1	2	1	2
23	gocd_A9700	1	1	1	1	1	0	1	1	2	1	2
24	gocd_A9700	1	1	1	1	4	0	4	1	2	1	4
25	gocd_A9700	1	1	1	1	1	0	1	1	2	1	2
26	gocd_A9700	1	1	1	1	1	0	1	1	2	1	2

Figure 1 Screenshot of Quality Control Summary Report

ANNEX B: List of the Script “QCSCD.R”

```
#
scriptName <- 'qcscd.R'
Version <- '1.0'
Version_Date <- '2014-08-20'
#

currentTime <- format(Sys.time(),'%Y-%m-%dT%H:%M:%S')

library('ncdf4')

gocd_proj_folder  <- '~/Projects/gocd'
gocd_data_folder  <- paste(gocd_proj_folder,'/data',sep='')
accession         <- 'A9700246';
seafloor_depth    <- 9999.9;
log_folder        <- paste(gocd_proj_folder,'/',accession,sep='')
logFile           <- paste(log_folder,'/gocd_A9700246.log',sep='')
dt                <- 60; #default sampling interval = 60 minutes
verbose           <- FALSE

sampling_intervals <- c(5,10,25,20,30,60); # in minutes

no_consecutive_qual_values <-c(12,6,4,3,2,2); # in minutes
no_consecutive_equal_values_allowed <-  2; # for dt = 60 minutes

consecutive_diff_values <-c(8,15,20,25,35,60); # in cm/seconds
consecutive_diff_values_allowed  <- 60; # for dt = 60 minutes

gocd_nc_data_folder           <- paste(gocd_proj_folder,'/',accession,sep='')
 gocd_nc_raw_data_folder       <- paste(gocd_nc_data_folder,'/raw',sep='')
#gocd_nc_raw_data_folder       <- paste(gocd_nc_data_folder,'/test_raw',sep='')
gocd_qced_data_folder         <- paste(gocd_nc_data_folder,'/qcd',sep='')
gocd_nc_raw_data_plots_folder <- paste(gocd_nc_data_folder,'/plots',
  sep='')
plot_html_file                <- paste(gocd_nc_data_folder,'/',accession,
  '_plots.html',sep='')

args <- commandArgs(trailingOnly = TRUE)

if (length(args) != 0) {
  for( i in 1:length(args)) {
    if (args[i] == '-verbose') {
       verbose <- TRUE
    }
    if (args[i] == '-log') {
      if (!(is.na(args[i+1]))) {
        logFile <- args[i+1]
      }
    }
    if (args[i] == '-dt') {
      if (!(is.na(args[i+1]))) {
        dt <- args[i+1]
      } else {
        cat ('Invalid sampling inverval.',args[i+1],'\n Program stop.\n');
        quit()
      }
    }
    if (args[i] == "-accession") {
      if (!(is.na(args[i+1]))) {
        accession <- args[i+1]
      } else {
        cat ('Missing (or invalid) accession number.',args[i+1],'\n Program stopped.\n');
        quit()
      }
    }
  }
}

if (!verbose) { sink(logFile) }
```

```
for( i in 1:length(sampling_intervals)) {
  if (dt == sampling_intervals[i]) {
    no_consecutive_equal_values_allowed <- no_consecutive_qual_values[i]
    consecutive_diff_values_allowed  <- consecutive_diff_values[i]
  }
}

nc_files <- list.files(gocd_nc_raw_data_folder,full.names = FALSE,
  recursive = FALSE)

fillValue <- 9999.9
global_max_spd <- 500.0 # 500 cm/s, 5m/s
global_min_spd <- 1.0 # 1 cm/s
envelope <- 5
test_passed <- TRUE
test_failed <- TRUE
test_performed <- TRUE
qc_test_names <- c(
  'Platform_Identification',
  'Impossible_Date/Time',
  'Impossible_Location',
  'Position_on_Land',
  'Global_Impossible_Parameter_Values',
  'Regional_Impossible_Parameter_Value',
  'Spike',
  'Constant_Speed',
  'Constant_Direction',
  'Rate_of_Change_in_Time')

tests_performed <- c(1,1,1,1,1,0,1,1,1)

data.arr <- array("",dim=c(1,1))
data.arr[1] <- paste('filename',',',
  'Platform_Identification',',',
  'Impossible_Date/Time',',',
  'Impossible_Location',',',
  'Position_on_Land',',',
  'Global_Impossible_Parameter_Values',',',
  'Regional_Impossible_Parameter_Value',',',
  'Spike',',',
  'Constant_Speed',',',
  'Constant_Direction',',',
  'Rate_of_Change_in_Time',',QC_flag',sep='')

source("~/rLib/getSeafloorDepth.R")

cat(paste("Current Date Time:",currentTime),'\n')
#
# Open an existing ETOPO1 file for reading
#
filename <- paste(gocd_data_folder,'/','ETOPO1_Bed_g_gmt4.grd',sep='')
cat(paste('Reading ETOPO1 file: ',filename,sep=' '),'\n')
triplet <- getSeafloorDepth(filename)
cat('Reading ETOPO1 file completed.','\n')
x_grid <- triplet$x
y_grid <- triplet$y
z_grid <- triplet$z
no_tests <- length(qc_test_names)
fileConn<-file(plot_html_file,open='w')
writeLines("<!DOCTYPE html>",fileConn)
writeLines(" <html> ",fileConn)
writeLines(" <body> ",fileConn)
writeLines(" <table border='1'> ",fileConn)
no_plot = 1
no_plots_per_row = 1
for( np in 1:length(nc_files)) {
  u <- vector(); v <- vector()
  cspd <- vector(); cdir <- vector()
  nc_file <- nc_files[np]
  record <- paste(nc_file,',',sep='')
  filename <- paste(gocd_nc_raw_data_folder,'/',nc_file,sep='')
  qcedfilename <- paste(gocd_qced_data_folder,'/',nc_file,sep='')
```

```
  file.copy(filename,qcedfilename,overwrite = TRUE)
  cat('\n','No.',np,'Assessing ',nc_file,'\n')
  qc_flags <- c()
  qc_codes <- c()
  qc_index <- c()
  qc_index <- 0
  for( n in 1:no_tests) {
    qc_flags[n] <- 0
    qc_codes[n] <- 0
    qc_index[n] <- 2**(n-1)
  }

  nc <- nc_open( qcedfilename, write=TRUE )

  for( i in 1:nc$ndims ) {
    d <- nc$dim[[i]]
    if (d$name == 'time') {
      time <- as.double(d$vals)
    }
  }

  for( i in 1:nc$nvars ) {
    var     <- nc$var[[i]]
    varname <- var$name
    if (varname == 'gocd_station_id') { station_id <- ncvar_get( nc, var) }
    if (varname == 'latitude') { lat <- ncvar_get( nc, var) }
    if (varname == 'longitude') { lon <- ncvar_get( nc, var) }
    if (varname == 'position_quality_flag') { pos_qc <- ncvar_get( nc, var) }
    if (varname == 'depth') { depth <- ncvar_get( nc, var) }
    if (varname == 'seafloor_depth') { seafloor_depth <- ncvar_get( nc, var) }
    if (varname == 'time_quality_flag') { time_qc <- ncvar_get( nc, var) }
    if (varname == 'current_speed') { cspd <- ncvar_get( nc, var) }
    if (varname == 'current_speed_quality_flag') {
      cspd_flag <- ncvar_get( nc, var) }
    if (varname == 'current_direction') { cdir <- ncvar_get( nc, var) }
    if (varname == 'current_direction_quality_flag') {
      cdir_flag <- ncvar_get( nc, var) }
    if (varname == 'u') { u <- ncvar_get( nc, var) }
    if (varname == 'u_flag') { u_flag <- ncvar_get( nc, var) }
    if (varname == 'v') { v <- ncvar_get( nc, var) }
    if (varname == 'v_flag') { v_flag <- ncvar_get( nc, var) }
    if (varname == 'pi_name') { pi_name <- ncvar_get( nc, var) }
    if (varname == 'instrument_type') { instr_type <- ncvar_get( nc, var) }
    if (varname == 'instrument_serial_number') {
      ser_num <- ncvar_get( nc, var) }
    if (varname == 'sampling_interval') { delta_t <- ncvar_get( nc, var) }
    if (varname == 'project_name') { project_name <- ncvar_get( nc, var) }
    if (varname == 'agency_name') { agency_name <- ncvar_get( nc, var) }
    if (varname == 'experiment_name') {
      experiment_name <- ncvar_get( nc, var) }
  }
  lon <- as.double(lon)
  lat <- as.double(lat)
  x_ind <- which(x_grid > lon - 1./60. & x_grid <= lon)
  y_ind <- which(y_grid > lat - 1./60. & y_grid <= lat)
#
# Change the sign of seafloor depth for postive downward.
#
  ETOPO1_value <- z_grid[x_ind,y_ind]*(-1.0)
  no_obs <- length(time)
  timestamp <- as.Date(time,origin='1900-01-01T00:00:00',
    format='%Y-%m-%dT%H:%M:%S')
  if (length(u) == 0) {
    qc_comp <- FALSE
    u <- cspd*sin(cdir*pi/180.0)
    v <- cspd*cos(cdir*pi/180.0)
  } else {
    qc_comp <- TRUE
  }
  if (length(cspd) == 0) {
    qc_resu <- FALSE
    cspd <- sqrt(u*u + v*v)*0.5
```

```
    cdir <- 90.0 - atan(v/u)*180.0/acos(-1)
  } else {
    qc_resu <- TRUE
  }

    if (no_plot == 1) {
      writeLines("  <tr> ",fileConn)
    }
    graphics.off()  # Turn off all graphics devices to begin.
    sc_plot_filename <- gsub(".nc","_scatter_plot.png",nc_file,fixed=TRUE)
    plot <- paste(gocd_nc_raw_data_plots_folder,'/',sc_plot_filename,sep='')
    png(plot)
    title_text <- paste("Source: ",nc_file,"\nCurrents Scatter Plot",sep="")
    plot(u,v,type='p',cex=0.25,xlab='u(cm/s)',ylab='v(cm/s)')
    title(main=title_text, line=1)
    abline(h = 0, v = 0, col = "red")
    cat('Scatter Plot completed:',sc_plot_filename,'\n')
    dev.off()
#
#  Time Series Plot
#
    dev.next()
    ts_plot_filename <- sub('.nc',"_ts_plot.png",nc_file,fixed=TRUE)
    plot <- paste(gocd_nc_raw_data_plots_folder,'/',ts_plot_filename,sep='')
    png(plot)
    par(mfrow=c(2,1))
    title_text <- paste("Source: ",nc_file,"\nCurrent Speeds",
      sep="")
    plot(timestamp,cspd,type='l',ylab='cm/s', xlab='Observation Day',xaxt='n')
    title(main=title_text, line=1)
    axis(1,at=timestamp[seq(1,length(cspd),960)],
      labels=format(timestamp[seq(1,length(cspd),960)],"%Y/%m/%d"))
    abline(h = 0, v = global_min_spd , col = "red")

    title_text <- paste("Source: ",nc_file,"\nCurrent Directions",
      sep="")
    plot(timestamp,cdir,type='l',ylab='degrees T', xlab='Observation Day',xaxt='n')
    title(main=title_text, line=1)
    axis(1,at=timestamp[seq(1,length(cdir),960)],
      labels=format(timestamp[seq(1,length(cdir),960)],"%Y/%m/%d"))
    cat('Time Series Plot completed: ',ts_plot_filename,'\n')
    dev.next()

    uv_ts_plot_filename <- sub('.nc',"_uv_ts_plot.png",nc_file,fixed=TRUE)
    plot <- paste(gocd_nc_raw_data_plots_folder,'/',uv_ts_plot_filename,sep='')
    png(plot)
    par(mfrow=c(2,1))
    timestamp <- as.Date(time,origin='1900-01-01T00:00:00',
      format='%Y-%m-%dT%H:%M:%S')
    title_text <- paste("Source: ",nc_file,"\nEast-West Component Speeds",
      sep="")
    plot(timestamp,u,type='l',ylab='u(cm/s)', xlab='Observation Day',xaxt='n')
    title(main=title_text, line=1)
    axis(1,at=timestamp[seq(1,length(u),960)],
      labels=format(timestamp[seq(1,length(u),960)],"%Y/%m/%d"))
    abline(h = 0, u = 0, col = "red")
    title_text <- paste("Source: ",nc_file,"\nNorth-South Component Speeds",
      sep="")
    plot(timestamp,v,type='l',ylab='v(cm/s)', xlab='Observation Day',xaxt='n')
    title(main=title_text, line=1)
    axis(1,at=timestamp[seq(1,length(v),960)],
      labels=format(timestamp[seq(1,length(v),960)],"%Y/%m/%d"))
    abline(h = 0, v = 0, col = "red")
    cat('Time Series Plot completed: ',uv_ts_plot_filename,'\n')

    dev.off( )

    if (lat < 0) {
      clat <- paste(sprintf("%.3f",abs(lat)),'S',sep='')
    } else {
      clat <- paste(sprintf("%.3f",lat),'N',sep='')
    }
```

```
    if (lon < 0) {
      clon <- paste(sprintf("%.3f",abs(lon)),'W',sep='')
    } else {
      clon <- paste(sprintf("%.3f",lon),'E',sep='')
    }
    writeLines("    <td valign='top'> ",fileConn)
    line = paste('Plot No. ',np,'<br />Loc.: (', clat,' ',clon,')<br />',sep='')
    writeLines(line,fileConn)
    line = paste('Instrument Type: ',gsub("^\\s+|\\s+$", "", instr_type),
      '<br />',sep='')
    writeLines(line,fileConn)
    cdepth <- paste(sprintf("%.1f",depth),' m',sep='')
    csdepth <- paste(sprintf("%.1f",seafloor_depth),' m',sep='')
    line = paste('Inst. Depth: ',cdepth,'<br />','Seafloor Depth: ',
      csdepth,'<br />','ETOPO1 bathymetry: ',ETOPO1_value,' m',sep='')
    writeLines(line,fileConn)

#
# QCT1-1: Platform Identification test
#
  test_no <- 1
  if (nchar(instr_type,type="chars",allowNA=FALSE)) {
    cat(paste(test_no,' ','QCT1-1:',qc_test_names[test_no],': Passed',
      'Instrument Type:',instr_type),'\n')
    qc_flags[test_no] <- 1
    qc_codes[test_no] <- 1
  } else {
    cat(paste(test_no,' ','QCT1-1:',qc_test_names[test_no],': Failed'),'\n')
    qc_flags[test_no] <- 9
    cat(paste(test_no,' ','Instrument Type: unknown'),'\n')
    qc_codes[test_no] <- 1
  }
  record <- paste(record,qc_flags[test_no],',',sep='')
  line = "<table border='1'><tr>";
  writeLines(line,fileConn)
  line = paste("<td><strong>Test Name</strong></td>",
    "<td><strong>Test Flag</strong></td></tr>",sep='')
  writeLines(line,fileConn)
  line = paste("<tr><td>",qc_test_names[test_no],'</td><td>',qc_flags[test_no],
    '</td></tr>',sep='')
#
# QCT1-2: Impossible Date/Time test
#
  test_no <- 2
  source("~/rLib/ymd2jd.R")
  ref_date_time <- as.double(ymd2jd(1900,1,1))
  if (length(which(time <= 0)) > 0
    || length(which(timestamp > currentTime)) > 0)   {
    cat(paste(test_no,' ','QCT1-2:',qc_test_names[test_no],': Failed'),'\n')
    qc_flags[test_no] <- 4
    qc_codes[test_no] <- 1
  } else {
    cat(paste(test_no,' ','QCT1-2:',qc_test_names[test_no],': Passed'),'\n')
    qc_flags[test_no] <- 1
    qc_codes[test_no] <- 1
  }
  time_qc <- time_qc + qc_flags[test_no]

  time_qc[which(is.na(time))] <- 9
  ncvar_put(nc,'time_quality_flag',time_qc)
  record <- paste(record,qc_flags[test_no],',',sep='')
  line = paste("<tr><td>",qc_test_names[test_no],'</td><td>',qc_flags[test_no],
    '</td></tr>',sep='')
  writeLines(line,fileConn)
#
# QCT1-3: Impossible location test
#
  test_no <- 3
  if (abs(lat) <= 90.0 || abs(lon) <= 180.0) {
    cat(paste(test_no,' ','QCT1-3:',qc_test_names[test_no],': Passed'),'\n')
    qc_flags[test_no] <- 1
    qc_codes[test_no] <- 1
```

```
  } else {
    cat('QCT1-3 Impossible Location Test: Failed')
    cat(paste(test_no,' ','QCT1-3:',qc_test_names[test_no],': Failed'),'\n')
    cat(paste(test_no,' ','QCT1-3:',lat, lon),'\n')
    qc_flags[test_no] <- 2
    qc_codes[test_no] <- 1
    cat(paste(test_no,' ','Latitude:',lat,'Longitude:',lon),'\n')
  }
  ncvar_put(nc,'position_quality_flag',qc_flags[test_no])
  record <- paste(record,qc_flags[test_no],',',sep='')
  line = paste("<tr><td>",qc_test_names[test_no],'</td><td>',qc_flags[test_no],
    '</td></tr>',sep='')
  writeLines(line,fileConn)
#
# QCT1-4: Position on land test
#
  test_no <- 4
  if (seafloor_depth == 9999.9) {
    cat(paste(test_no,' ','QCT1-4:',qc_test_names[test_no],': Not performed'),'\n')
    qc_flags[test_no] <- 9
    qc_codes[test_no] <- 0
    cat('seafloor_depth: unknown\n')
  } else if (ETOPO1_value < 0 ) {
    cat(paste(test_no,' ','QCT1-4:',qc_test_names[test_no],': Failed'),'\n')
    qc_flags[test_no] <- 3
    qc_codes[test_no] <- 1
    cat(paste(test_no,' ','ETOPO1:',ETOPO1_value,'m'),'\n')
    cat(paste(test_no,' ','seafloor_depth:',seafloor_depth,'m'),'\n')
  } else if ((ETOPO1_value-seafloor_depth)/ETOPO1_value > 0.10 ) {
    cat(paste(test_no,' ','QCT1-4:',qc_test_names[test_no],': Failed'),'\n')
    qc_flags[test_no] <- 2
    qc_codes[test_no] <- 1
    cat(paste(test_no,' ','ETOPO1:',ETOPO1_value,'m'),'\n')
    cat(paste(test_no,' ','seafloor_depth:',seafloor_depth,'m'),'\n')
  } else {
    cat(paste(test_no,' ','QCT1-4:',qc_test_names[test_no],': Passed'),'\n')
    qc_flags[test_no] <- 1
    qc_codes[test_no] <- 1
  }
  record <- paste(record,qc_flags[test_no],',',sep='')
  line = paste("<tr><td>",qc_test_names[test_no],'</td><td>',qc_flags[test_no],
    '</td></tr>',sep='')
  writeLines(line,fileConn)
#
# QCT2-1: Global Impossible Parameter Values
#
  test_no <- 5
  p_max_cspd <- length(which(abs(cspd) > global_max_spd))/length(cspd)
  p_min_cspd <- length(which(abs(cspd) < global_min_spd))/length(cspd)
  if ( p_max_cspd >= 0.75 || p_min_cspd >= 0.95 )  {
    cat(paste(test_no,' ','QCT2-1:',qc_test_names[test_no],': Failed'),'\n')
    qc_flags[test_no] <- 4
    cat(paste(test_no,' ','QCT2-1: QC Flag: ',qc_flags[test_no]),'\n')
    cat(paste(test_no,' ','QCT2-1: PHVmax: ',p_max_cspd),'\n')
    cat(paste(test_no,' ','QCT2-1: PHVmin: ',p_min_cspd),'\n')
    qc_codes[test_no] <- 1
  } else if ( p_max_cspd >= 0.50 )  {
    cat(paste(test_no,' ','QCT2-1:',qc_test_names[test_no],': Failed'),'\n')
    qc_flags[test_no] <- 3
    cat(paste(test_no,' ','QCT2-1: QC Flag: ',qc_flags[test_no]),'\n')
    cat(paste(test_no,' ','QCT2-1: PHVmax: ',p_max_cspd),'\n')
    cat(paste(test_no,' ','QCT2-1: PHVmin: ',p_min_cspd),'\n')
    qc_codes[test_no] <- 1
  } else if ( p_max_cspd >= 0.05 )  {
    cat(paste(test_no,' ','QCT2-1:',qc_test_names[test_no],': Failed'),'\n')
    qc_flags[test_no] <- 2
    cat(paste(test_no,' ','QCT2-1: QC Flag: ',qc_flags[test_no]),'\n')
    cat(paste(test_no,' ','QCT2-1: PHVmax: ',p_max_cspd),'\n')
    cat(paste(test_no,' ','QCT2-1: PHVmin: ',p_min_cspd),'\n')
    qc_codes[test_no] <- 1
  } else {
    cat(paste(test_no,' ','QCT2-1:',qc_test_names[test_no],': Passed'),'\n')
```

```
    qc_flags[test_no] <- 1
  }
  record <- paste(record,qc_flags[test_no],',',sep='')
  line = paste("<tr><td>",qc_test_names[test_no],'</td><td>',qc_flags[test_no],
    '</td></tr>',sep='')
  writeLines(line,fileConn)
#
# QCT2-2: Regional Impossible Parameter Values
#
  test_no <- 6
  cat(paste(test_no,' ','QCT2-2:',qc_test_names[test_no],': Not tested'),'\n')
  qc_flags[test_no] <- 0
  qc_codes[test_no] <- 0
  record <- paste(record,qc_flags[test_no],',',sep='')
  line = paste("<tr><td>",qc_test_names[test_no],'</td><td>',qc_flags[test_no],
    '</td></tr>',sep='')
  writeLines(line,fileConn)
#
# QCT2-3: Spike
#
  test_no <- 7
  cspd_sd_raw = sd(cspd)
  cspd_lim <- sd(cspd[which(abs(cspd) < 10*cspd_sd_raw)])*envelope
  no_cspd_spike <- length(which(abs(cspd) > cspd_lim))/no_obs
  if ( no_cspd_spike >= 0.75 ) {
    cat(paste(test_no,' ','QCT2-3:',qc_test_names[test_no],': Failed'),'\n')
    qc_flags[test_no] <- 4
    cat(paste(test_no,' ','QCT2-3: QC Flag: ',qc_flags[test_no]),'\n')
    cat(paste(test_no,' ','QCT2-3: PSK: ',no_cspd_spike),'\n')
    qc_codes[test_no] <- 1
  } else if( no_cspd_spike >= 0.50 ) {
    cat(paste(test_no,' ','QCT2-3:',qc_test_names[test_no],': Failed'),'\n')
    qc_flags[test_no] <- 3
    cat(paste(test_no,' ','QCT2-3: QC Flag: ',qc_flags[test_no]),'\n')
    cat(paste(test_no,' ','QCT2-3: PSK: ',no_cspd_spike),'\n')
    qc_codes[test_no] <- 1
  } else if( no_cspd_spike >= 0.05 ) {
    cat(paste(test_no,' ','QCT2-3:',qc_test_names[test_no],': Failed'),'\n')
    qc_flags[test_no] <- 2
    cat(paste(test_no,' ','QCT2-3: QC Flag: ',qc_flags[test_no]),'\n')
    cat(paste(test_no,' ','QCT2-3: PSK: ',no_cspd_spike),'\n')
    qc_codes[test_no] <- 1
  } else {
    cat(paste(test_no,' ','QCT2-3:',qc_test_names[test_no],': Passed'),'\n')
    qc_flags[test_no] <- 1
    cat(paste(test_no,' ','QCT2-3: QC Flag: ',qc_flags[test_no]),'\n')
    cat(paste(test_no,' ','QCT2-3: PSK: ',no_cspd_spike),'\n')
    qc_codes[test_no] <- 1
  }
  record <- paste(record,qc_flags[test_no],',',sep='')
  line = paste("<tr><td>",qc_test_names[test_no],'</td><td>',qc_flags[test_no],
    '</td></tr>',sep='')
  writeLines(line,fileConn)
#
# QCT2-4: Constant Speed (Stationarity Checks)
#
  test_no <- 8
  no_stuck <- 0
  for( n in 3:no_obs ) {
    if (!(is.na(cspd[n-1]) && is.na(cspd[n]))) {
      if (cspd[n] == cspd[n-1] && cspd[n] == cspd[n-2]) {
        no_stuck <- no_stuck + 1
      }
    }
  }
  p_no_stuck = no_stuck / no_obs
  if ( p_no_stuck >= 0.75 ) {
    cat(paste(test_no,' ','QCT2-4:',qc_test_names[test_no],': Failed'),'\n')
    qc_flags[test_no] <- 4
    cat(paste(test_no,' ','QCT2-4: QC Flag: ',qc_flags[test_no]),'\n')
    cat(paste(test_no,' ','QCT2-4: PSV: ',p_no_stuck),'\n')
    qc_codes[test_no] <- 1
```

```
  } else if ( p_no_stuck >= 0.50 ) {
    cat(paste(test_no,' ','QCT2-4:',qc_test_names[test_no],': Failed'),'\n')
    qc_flags[test_no] <- 3
    cat(paste(test_no,' ','QCT2-4: QC Flag: ',qc_flags[test_no]),'\n')
    cat(paste(test_no,' ','QCT2-4: PSV: ',p_no_stuck),'\n')
    qc_codes[test_no] <- 1
  } else if ( p_no_stuck >= 0.05 ) {
    cat(paste(test_no,' ','QCT2-4:',qc_test_names[test_no],': Failed'),'\n')
    qc_flags[test_no] <- 2
    cat(paste(test_no,' ','QCT2-4: QC Flag: ',qc_flags[test_no]),'\n')
    cat(paste(test_no,' ','QCT2-4: PSV: ',p_no_stuck),'\n')
    qc_codes[test_no] <- 1
  } else {
    cat(paste(test_no,' ','QCT2-4:',qc_test_names[test_no],': Passed'),'\n')
    qc_flags[test_no] <- 1
    cat(paste(test_no,' ','QCT2-4: QC Flag: ',qc_flags[test_no]),'\n')
    cat(paste(test_no,' ','QCT2-4: PSV: ',p_no_stuck),'\n')
    qc_codes[test_no] <- 1
  }
  record <- paste(record,qc_flags[test_no],',',sep='')
  line = paste("<tr><td>",qc_test_names[test_no],'</td><td>',qc_flags[test_no],
    '</td></tr>',sep='')
  writeLines(line,fileConn)
#
# QCT2-5: Constant Direction (Stationarity Checks)
#
  test_no <- 9
  no_exceed <- 0
  for( n in 2:no_obs ) {
    if (!(is.na(cdir[n-1]) && is.na(cdir[n]))) {
      if (abs(cdir[n] - cdir[n-1]) > consecutive_diff_values_allowed) {
        no_exceed <- no_exceed + 1 }
    }
  }
  p_no_exceed = no_exceed / no_obs
  if ( p_no_exceed >= 0.75 ) {
    cat(paste(test_no,' ','QCT2-5:',qc_test_names[test_no],': Failed'),'\n')
    qc_flags[test_no] <- 4
    cat(paste(test_no,' ','QCT2-5: QC Flag: ',qc_flags[test_no]),'\n')
    cat(paste(test_no,' ','QCT2-5: PSV: ',p_no_stuck),'\n')
    qc_codes[test_no] <- 1
  } else if ( p_no_exceed >= 0.50 ) {
    cat(paste(test_no,' ','QCT2-5:',qc_test_names[test_no],': Failed'),'\n')
    qc_flags[test_no] <- 3
    cat(paste(test_no,' ','QCT2-5: QC Flag: ',qc_flags[test_no]),'\n')
    cat(paste(test_no,' ','QCT2-5: PSV: ',p_no_stuck),'\n')
    qc_codes[test_no] <- 1
  } else if ( p_no_exceed >= 0.05 ) {
    cat(paste(test_no,' ','QCT2-5:',qc_test_names[test_no],': Failed'),'\n')
    qc_flags[test_no] <- 2
    cat(paste(test_no,' ','QCT2-5: QC Flag: ',qc_flags[test_no]),'\n')
    cat(paste(test_no,' ','QCT2-5: PSV: ',p_no_stuck),'\n')
    qc_codes[test_no] <- 1
  } else {
    cat(paste(test_no,' ','QCT2-5:',qc_test_names[test_no],': Passed'),'\n')
    qc_flags[test_no] <- 1
    cat(paste(test_no,' ','QCT2-5: QC Flag: ',qc_flags[test_no]),'\n')
    cat(paste(test_no,' ','QCT2-5: PSV: ',p_no_stuck),'\n')
    qc_codes[test_no] <- 1
  }
  record <- paste(record,qc_flags[test_no],',',sep='')
  line = paste("<tr><td>",qc_test_names[test_no],'</td><td>',qc_flags[test_no],
    '</td></tr>',sep='')
  writeLines(line,fileConn)
#
# QCT2-6: Rate of Change in Time
#
  test_no <- 10
  no_spd_exceed <- 0
  for( n in 2:no_obs ) {
    if (!(is.na(cspd[n-1]) && is.na(cspd[n]))) {
      if (abs(cspd[n] - cspd[n-1]) > consecutive_diff_values_allowed) {
```

```
        no_spd_exceed <- no_spd_exceed + 1 }
    }
  }
  p_no_exceed = no_spd_exceed / no_obs
  if ( p_no_exceed >= 0.75 ) {
    cat(paste(test_no,' ','QCT2-6:',qc_test_names[test_no],': Failed'),'\n')
    qc_flags[test_no] <- 4
    cat(paste(test_no,' ','QCT2-6: QC Flag: ',qc_flags[test_no]),'\n')
    cat(paste(test_no,' ','QCT2-6: PSR: ',p_no_exceed),'\n')
    qc_codes[test_no] <- 1
  } else if ( p_no_exceed >= 0.50 ) {
    cat(paste(test_no,' ','QCT2-6:',qc_test_names[test_no],': Failed'),'\n')
    qc_flags[test_no] <- 3
    cat(paste(test_no,' ','QCT2-6: QC Flag: ',qc_flags[test_no]),'\n')
    cat(paste(test_no,' ','QCT2-6: PSR: ',p_no_exceed),'\n')
    qc_codes[test_no] <- 1
  } else if ( p_no_exceed >= 0.05 ) {
    cat(paste(test_no,' ','QCT2-6:',qc_test_names[test_no],': Failed'),'\n')
    qc_flags[test_no] <- 2
    cat(paste(test_no,' ','QCT2-6: QC Flag: ',qc_flags[test_no]),'\n')
    cat(paste(test_no,' ','QCT2-6: PSR: ',p_no_exceed),'\n')
    qc_codes[test_no] <- 1
  } else {
    cat(paste(test_no,' ','QCT2-6:',qc_test_names[test_no],': Passed'),'\n')
    qc_flags[test_no] <- 1
    cat(paste(test_no,' ','QCT2-6: QC Flag: ',qc_flags[test_no]),'\n')
    cat(paste(test_no,' ','QCT2-6: PSR: ',p_no_exceed),'\n')
    qc_codes[test_no] <- 1
  }
  record <- paste(record,qc_flags[test_no],',',sep='')
  line = paste("<tr><td>",qc_test_names[test_no],'</td><td>',qc_flags[test_no],
    '</td></tr>',sep='')
  writeLines(line,fileConn)
  line = paste('<tr><td>Overall QC flag</td><td>',max(qc_flags),
    '</td></tr></table>',sep='')
  writeLines(line,fileConn)
  writeLines("    </td> ",fileConn)
  if (qc_comp) {
    u_flag <- u_flag + max(qc_flags[5:10])
    v_flag <- v_flag + max(qc_flags[5:10])
    u_flag[which(is.na(u))] <- 9
    v_flag[which(is.na(v))] <- 9
    ncvar_put(nc,'u_flag',u_flag)
    ncvar_put(nc,'v_flag',v_flag)
  }
  if (qc_resu) {
    cspd_flag <- cspd_flag + max(qc_flags[5:10])
    cdir_flag <- cdir_flag + max(qc_flags[5:10])
    cspd_flag[which(is.na(cspd))] <- 9
    cdir_flag[which(is.na(cdir))] <- 9
    ncvar_put(nc,'current_direction_quality_flag',cdir_flag)
    ncvar_put(nc,'current_speed_quality_flag',cspd_flag)
  }

  data.arr[np+1] <- paste(record,max(qc_flags),sep='')
#
# Calcuate the sum of qc_index
#
  qc_sum <- sum(qc_index*qc_codes)
#
# Convert qc_sum from decimal to binary code
#
#qc_string <- paste(rev(as.integer(intToBits(qc_sum))),collapse="")
  date_modified <- format(as.POSIXlt(Sys.time(),"GMT"),'%Y-%m-%dT%H:%M:%SZ')
  ncatt_put( nc, 0,'date_modified',date_modified)
  ncatt_put( nc, 0,'date_issued',date_modified)
  ncatt_put( nc, 0,'QC_Manual','Reference Manual for Quality Control Subsurface Currents Data
(QCSCD) Version 1.0')
  ncatt_put( nc, 0,'QC_Version','1.0')
  qc_test_string <- c(qc_test_names)
  qc_test_string <- paste(qc_test_string,collapse=", ")
  ncatt_put( nc, 0,'QC_test_names',qc_test_string)
```

```
  qc_code_string <- toString(qc_codes)
  ncatt_put( nc, 0,'QC_test_codes',qc_code_string)
  qc_flag_string <- toString(qc_flags)
  ncatt_put( nc, 0,'QC_test_results',qc_flag_string)
  qc_version <- paste(scriptName,Version,Version_Date,sep=", ")
  ncatt_put( nc, 0,'QC_Software',qc_version)
  nc_close(nc)
  writeLines("    <td> ",fileConn)
  line = paste('<img src=','"plots/',sc_plot_filename,'"',
    ' alt="scatter plot">', sep='')
  writeLines(line,fileConn)
  writeLines("    </td> ",fileConn)
  writeLines("    <td> ",fileConn)
  line = paste('<img src=','"plots/',ts_plot_filename,'"',
    ' alt="Speed and Direction time series plot">', sep='')
  writeLines(line,fileConn)
  writeLines("    </td> ",fileConn)
  writeLines("    <td> ",fileConn)
  line = paste('<img src=','"plots/',uv_ts_plot_filename,'"',
    ' alt="U, V time series plot">', sep='')
  writeLines(line,fileConn)
  writeLines("    </td> ",fileConn)
  if (no_plot == no_plots_per_row) {
    writeLines("  </tr> ",fileConn)
    no_plot <- 1
  } else {
    no_plot <- no_plot + 1
  }
}
writeLines("       </tr> ",fileConn)
writeLines("     </table> ",fileConn)
writeLines("   </body> ",fileConn)
writeLines(" </html> ",fileConn)
close(fileConn)
qc_test_result_file <- paste(gocd_nc_data_folder,'/gocd_',accession,'_qcf.csv',
  sep='')
write(data.arr,qc_test_result_file)
```

ANNEX C: Standard Operating Procedures for Converting Originator's Data Formats to the Global Ocean Currents (GOC) NetCDF Format

The Standard Operating Procedure (SOP) describes the step-by-step Instructions for converting originator's data formats to the Global Ocean Currents NetCDF format conventions. Use Accession Number 9700246 as an example.

1) Log in to the NODC Archive Management System (AMS) located at https://archive.nodc.noaa.gov/AMS/prod/.
2) Search for Accession Number 9700246 by typing '9700246' in the text area of the row labeled as 'accession_id'
3) Click on the "Search" button located near the end of the AMS Web page to search for the metadata information of the Accession number.
4) The AMS will return a Web page containing basic metadata information of the Accession Number.
5) Click on the link, 'metadata', appeared at the first column of the table to obtain other metadata information about the accession.
6) The AMS will create a Web page containing additional metadata information about the accession.
7) Click on the link, 'archive' from the 'accessions_id' field.
8) A Web page with a header called 'Index of /archive/arc0001/9700246' should be created. The header indicates the physical location of the data of the Accession inside the NODC, which should be the folder of '/nodc/archive/arc0001/9700246'.
9) Log in to the NODC computational server, 'amonite'.
10) Change your working directory to '/nodc/archive/arc0001/9700246'.
11) Review the contents of the accession in the directory and its subdirectories stated in Step 10 and look for the following metadata information required by the GOC format conversion program:

    - agency_name
    - project_name
    - experiment_name
    - pi_name
    - pi_url
    - pi_email
    - latitude of the measurement
    - longitude of the measurement
    - time zone of the measurement
    - sampling_interval
    - instrument depth
    - seafloor depth
    - mooring_name
    - instrument_serial_number
    - speed and direction of ocean currents and/or
    - east-west and north-south components of ocean currents

12) Create a folder under the current working directory of the GOCD project, i.e., /nodc/users/csun/Projects/gocd. The name of the folder should begin with the uppercase letter

of 'A' followed by a seven-digit accession number, for example, A9700246, where 9700246 is the accession number for the ocean currents data set being archived at NODC.

13) Create a folder under the current working directory of the GOCD project, i.e., /nodc/users/csun/Projects/gocd. The name of the folder should begin with the uppercase letter of 'A' followed by a seven-digit accession number, for example, A9700246, where 9700246 is the accession number for the ocean currents data set being archived at NODC.
14) Search if a data format converter has been written for the data submitted by the same PI or institution in the past, which can be reused for converting the Accession program, if not, write a new converter in FORTRAN 90 (preferred) and place the new converter, convUSGS.f90, in the 'pgms' folder of the GOCD working directory.
15) Write a main program in FORTRAN 90 (preferred) and place the main program in the Accession working folder, 'A9700246'. The default naming convention of the main program will begin with 'conv' followed by an uppercase letter, 'A', then the accession number, '9700246', i.e., 'convA9700246.f90'
16) Create a linux 'make' program. The make program essentially is used to update targets (exe or object files) according to the dependency instructions, typically present in a file called 'Makefile'. This however can be overridden by specifying the filename with the -f switch. The makefile should look like this:

```
# macros for files ending in .f95
CF      = f95
FFLAGS   = -O2
#
# # macros for loader
LDFLAGS   = -L/home/csun/lib
INCFLAGS = -I/usr/include -I/home/csun/Projects/gocd/nc/V2-0/pgms
LIBFLAGS = -lnetcdf -lnetcdff -lf90
SOURCE = ~/Projects/gocd/nc/V2-0/pgms
INCH = ~/include/goc.h
#
# #program source code files and object code files
#
OBJ1       = convA9700246.o convUSGS2.o check.o check2.o def_dim.o
OBJ2       = def_time_coverage.o def_crs_var.o def_station_varid.o def_lat_varid.o
OBJ3       = def_lon_varid.o def_position_qc_varid.o def_depth_varid.o
OBJ4       = def_seafloor_depth_varid.o def_time_varid.o def_time_qc_varid.o
OBJ5       = def_spd_varid.o def_dir_varid.o def_spd_qc_varid.o def_dir_qc_varid.o
OBJ6       = def_u_varid.o def_u_qc_varid.o def_v_varid.o def_v_qc_varid.o
OBJ7       = def_meta_varid.o def_global_att.o
OBJS       = $(OBJ1) $(OBJ2) $(OBJ3) $(OBJ4) $(OBJ5) $(OBJ6) $(OBJ7)
PRGM       = convA9700246.x
#
$(PRGM): $(OBJS)
        $(CF) -o $(PRGM) $(OBJS) $(LDFLAGS) $(LIBFLAGS) $(INCFLAGS)
check.o: $(SOURCE)/check.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/check.f90
check2.o: $(SOURCE)/check2.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/check2.f90
def_crs_var.o: $(SOURCE)/def_crs_var.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_crs_var.f90
def_time_coverage.o: $(SOURCE)/def_time_coverage.f90
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_time_coverage.f90
def_dim.o: $(SOURCE)/def_dim.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_dim.f90
def_global_att.o: $(SOURCE)/def_global_att.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_global_att.f90
def_station_varid.o: $(SOURCE)/def_station_varid.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_station_varid.f90
def_lat_varid.o: $(SOURCE)/def_lat_varid.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_lat_varid.f90
```

```
def_lon_varid.o: $(SOURCE)/def_lon_varid.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_lon_varid.f90
def_position_qc_varid.o: $(SOURCE)/def_position_qc_varid.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_position_qc_varid.f90
def_depth_varid.o: $(SOURCE)/def_depth_varid.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_depth_varid.f90
def_seafloor_depth_varid.o: $(SOURCE)/def_seafloor_depth_varid.f90
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_seafloor_depth_varid.f90
def_time_varid.o: $(SOURCE)/def_time_varid.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_time_varid.f90
def_time_qc_varid.o: $(SOURCE)/def_time_qc_varid.f90
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_time_qc_varid.f90
def_spd_varid.o: $(SOURCE)/def_spd_varid.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_spd_varid.f90
def_spd_qc_varid.o: $(SOURCE)/def_spd_qc_varid.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_spd_qc_varid.f90
def_dir_varid.o: $(SOURCE)/def_dir_varid.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_dir_varid.f90
def_dir_qc_varid.o: $(SOURCE)/def_dir_qc_varid.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_dir_qc_varid.f90
def_u_varid.o: $(SOURCE)/def_u_varid.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_u_varid.f90
def_u_qc_varid.o: $(SOURCE)/def_u_qc_varid.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_u_qc_varid.f90
def_v_varid.o: $(SOURCE)/def_v_varid.f90
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_v_varid.f90
def_v_qc_varid.o: $(SOURCE)/def_v_qc_varid.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_v_qc_varid.f90
def_meta_varid.o: $(SOURCE)/def_meta_varid.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/def_meta_varid.f90
convUSGS2.o: $(SOURCE)/convUSGS2.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c $(SOURCE)/convUSGS2.f90
convA9700246.o: convA9700246.f90 $(INCH)
        $(CF) $(FFLAGS) $(LDFLAGS) $(INCFLAGS) -c convA9700246.f90
#
clean:
        rm -f *.o *.mod *.MOD
clobber: clean
        rm -f $(PRGM)
#
```

17) To invoke the make program, type the Linux command, ‘make’, at the system prompt. The make program automatically can link dependent modules and also decide which files are to be recompiled by looking at the object code timestamps. The makefile contains all the dependency information required for compiling. The target is the final output that is required at the end of the make process. It typically is reliant on many other files to have compiled successfully.
18) The target, denoted as ‘convA9700246.x’, should be created in the working directory of the accession, successfully, if no compilation errors were generated.
19) To invoke the format conversion process, enter the following line at the system prompt:
20) ./convA9700246.x goc_A9700246_inv.csv convA9700246.log convA9700246.err
21) Numerous netCDF files should be created along with an indexing (inventory) file, ‘gocd_A9700246_inv.csv’, of the netCDF files. The conversion process will create one log file, convA9700246.log, and one error message file, convA9700246,err. All files can be found in the current working directory.
22) Use the Linux “tar” command, ‘tar -czf goc_A9700246_Vx-y.tgz *.nc’, to create a compressed file containing all netCDF files, where x-z is the current version number of the GOC netCDF format conventions.

23) Move converted data files to the directory created in Step 21. Then, add the inventory file without the header (first) line to the GOCD inventory file, gocd_inv.csv, located at the above folder, '/nodc/web/dsdt.nodc/gocd'.
24) Run "run_readGOCnc.pl" to convert the GOCD NetCDF files to text, csv, and htm format and write out the outputs under the sub-folders, "txt", "csv", "htm", of the GOC Web data folder (/nodc/web/dsdt.nodc/goc) respectively.
25) Create a subdirectory, "raw", in the current working directory.
26) Move converted data files to the directory created in the above step.