SC/68C/ASI/13

Sub-committees/working group name: ASI

Update on modernization of visual survey simulation programs

Dan W Smith; Debra Palka

Update on modernization of visual survey simulation programs

Dan W Smith[1], Debra Palka[2]
[1] Integrated Statistics, Inc. 16 Sumner St., Woods Hole, MA 02543
[2] Northeast Fisheries Science Center, 166 Water St., Woods Hole, MA 02543

Abstract

The Scientific Committee of the International Whaling Commission has used simulated data to evaluate novel analysis methods for line transect abundance surveys. These datasets have been archived with the Secretariat. However, the original executable code used to create those simulated data was generated using now outdated compilers. To make the code more accessible, the Scientific Committee funded a project to document, update, and streamline the code so it is compatible with newer available compilers. This manuscript describes the updating process, now complete, and provides guidelines and examples showing how to use the programs to create new simulated data scenarios and how to re-compile the code after further development. We plan to make the code and guidelines available on GitHub. The existing simulated datasets, together with others that may be generated with the updated programs, could comprise a library of datasets for use by the Scientific Committee in evaluating future novel analysis methods for line transect data.

Introduction

The Scientific Committee (SC) of the International Whaling Commission (IWC) formally reviews and agrees on the status of abundance estimates submitted to the SC. One issue that makes this task difficult is determining how to evaluate the robustness of novel estimation methods. In several cases the SC used simulated data to evaluate such novel methods. The IWC has archived all previous simulated datasets; however, the code used to create those simulated data has not been easily available to create additional scenarios. Thus, the IWC funded D.W. Smith to update and document existing line transect simulation code so as to make it more accessible to SC members.

As background, the suite of programs (called Transim) discussed here was most recently used to create simulated sighting and effort data from surveys conducted in the Antarctic. In these runs, the simulated study area was populated with animals that followed the general features of the distribution, density and behavior of Antarctic minke whales, and the sighting survey procedures as modeled were similar to those from the IWC-IDCR-SOWER surveys. However, the Transim suite is sufficiently general to simulate line transect sightings and effort data resulting from a 1, 2, or 3 team line transect survey, where each team could have a different user-specified detection function that could depend on various factors such as school size, Beaufort state, or some other unrecorded factor. The Transim user can also specify the amount of time surveyed per day, the speed of the ship, and the amount of time spent in Closing and Passing mode. The spatial distribution and abundance of a target species can be pre-specified to allow spatial and temporal gradients, clustering of groups, or any combination of these. Depending on the Transim version, animals within groups may have the ability to dive, swim, and move randomly or react to the ship. Their dive time distribution can be generated either from a theoretical Poisson distribution or from empirically measured dive sequences. In cases where school sizes are greater than one, all animals in a group can surface either simultaneously or in a non-synchronized fashion. See Palka and Smith (in review) for more details.

Results

The code for the Transim suite of survey-simulation programs has been updated so that it compiles under (at least) gcc/g++ 4.8.5 on gnu linux 3.10.0 (the current environment at Woods Hole's Northeast Fisheries Science Center), and successfully replicates results of each of the final suite of 16 scenarios produced during earlier simulation work[1]. The core code for the post-processing segment of the simulations has been revised so that key configuration parameters can now be obtained from a separate config file rather than being hard-coded into the program.

[1] Apart from whale swim directions, which have played no role in any recent analyses, and differences in a portion of day #42 within the first rep of scenario 47, potentially due to slight variation in the program's interaction with the random number generator under different compilers and OS versions.

Multiple versions of both pre- and post-processing components were used in the earlier simulations, as the pre-processing component evolved slightly during generation of successive scenarios to minimize foreseeable problems and the post-processing component in its earlier form required code editing to change parameters between scenarios. When checking for exact replication, these differences mostly had to be maintained, but in the future varied simulations should be able to be run using fixed versions of the pre- and post-processing code resulting from the recent work.

A guide specifying formats and contents of input (configuration) and output files is provided in appendices for users who may wish to undertake additional simulations using this suite of programs. C++ code and cycling scripts for the model can be obtained from the Northeast Fisheries Science Center via GitHub (not completed yet).

Reference

Palka, D.L., and D.W. Smith. In review. Simulations of the IWC IDCR/SOWER Antarctic minke whale abundance surveys.

Appendix I: Running the Transim/Whalesim Model

Main Programs

A "cookbook" for running the model as it now stands follows. It will assume the user intends to run a multiple-replicate simulation. (Single-rep simulations could in principle be run by invoking the C++ executables directly, but will likely prove simpler to run as special [one-rep] cases of the multi-rep approach.)

The following tcsh command iterates both the preprocessing and main simulation programs (*locatereadship_timefile* and *whalesim*) via a perl script:

runwhalesimnew5delrl [start day number] [end day number] [number of hours] [filename-root] [number of reps] [orig | replicate]

example:
runwhalesimnew5delrl 0 5 16 sc48 3 replicate

Arguments to the command should be in the order above, and the user should not assume there are any defaults (i.e., all arguments should be provided).

- "**Start day number**" is the day to start _after_ (thus 0 if you want to start on day 1);
- "**End day number**" is the last day for which you want to generate data. Day numbers default to starting with 1 Jan 1995; thus, for example, day number 35 will appear in output as 4 Feb 1995.
- "**Number of hours**" is the numbers of hours of surveying per day; 16 is the safest choice as it has been most thoroughly tested, but smaller numbers might be OK too.
- "**Filename-root**" is the root for config and output files, e.g. "sc48", where config files would include *sc48.scf* and initial output files for the first rep would include *sc48.e1* and *sc48.l1* (note that a lowercase "L" precedes the numeral "1" in the latter filename).
- "**Number of reps**" should be self-explanatory. (Doing more than 100 reps might require some modification of later-stage processing scripts.)
- "**orig|replicate**": here supply one of the words "orig" or "replicate", thus specifying either a completely new run or a run to replicate one from the past. In the replication case, files containing random-number seeds from that past run should already exist and have names of the form *[filename-root].w[repnum]*.

For runs expected to be long (many model days, many reps), the script can be run in the background, e.g.

runwhalesimnew5delrl 0 5 16 sc48 3 replicate &

but it may be helpful to first try a shorter subset of the run to estimate how much file space will be needed for the longer run's output.

The tcsh script *runwhalesimnew5delrl* runs the perl script called *runwhale_rs_tf_lc* (original version of the latter by Dave Hiltz, NEFSC; modified by DWS), which in turn runs the following compiled C++ programs:

a) *locatereadship_timefile*: to distribute whales etc.;
b) *mkroster*: to create the observer roster; and
c) *whalesim*: to run the core of the simulation.

The C++ executables require several configuration files; existing config files from past runs can be used if suitable, or new files can be created according to formats prescribed in Appendix II. Necessary configuration files include:

a) *[filename-root].lcf*: contains location information,
b) *[filename-root].scf* : contains ship information,
c) *[filename-root].rcf*: contains roster information, and

d) *[filename-root].rea*: contains reaction information.

Again, see Appendix II for further details on these files.

Pre-Processing Step

The pre-processing step creates a whale configuration file based on location and reaction configuration files and a roster file based on a roster configuration file; this happens as part of the *runwhale_rs_tf_lc* invocation mentioned above. These files then serve as input to the main ("whalesim") program.

Post-Processing Step

The post-processing step implements closing-mode behavior by the ship, errors in observation parameters, confusion of whale group ID's, truncation of times, and elimination of all but first sightings, if any of these are desired.

To iterate the post-processing programs, the first command is:

run_clmode_brstop_as.csh [filename-root] [number of reps] [orig | replicate]

example:
run_clmode_brstop_as.csh sc48 3 replicate

As above, choosing one of the words "*orig/replicate*" indicates whether this post-processing is to be original or a replication. For the replication case, random-seed files from a previous run should exist, and should have names of the form *[filename-root].clmode.sd[repnum]*-- e.g., *sc48.clmode.sd2* for the seed file for the second rep of sc48.

The post-processing executable *clmode_brstop* requires one config file, *[filename-root].ccf*, whose format is described in Appendix II.

The result of the pass above will be effort files named *[filename-root].pd.e[repnum]* and sightings files that still contain all sightings. The latter files will be named *[filename-root].as.l[repnum]*, where "*as*" indicates "*all sightings*." If you don't want all sightings in the final output, you should next execute:

run_makefirstsight.csh [filename-root] [number of reps]

example:
run_makefirstsight.csh sc48 3

which will run the executable "*makefirstsight*" to reduce the all-sightings files to ones containing only first sightings (the resulting new sightings output files will be named *[filename-root].pd.l[repnum]*).

Both of the latter scripts should run sufficiently quickly that placing runs in the background will not be necessary. The scripts will generate some output to the screen while running, but this can usually be ignored unless needed for debugging.

Files

A directory on the GitHub site will contain source code for the model's components. If executable code is not already available, executing "*make*" in this directory with a suitable compiler should create the executable "*whalesim*" for the core of the model. The command "*make locatereadship_timefile*" will create the executable for the preprocessing component that distributes whales in space; "*make clmode_brstop*" will create the executable for the post-processing step that implements closing-mode behavior; and "*make makefirstsight*" will create the final executable that reduces output to first sightings. These executables can then be iterated by the scripts mentioned above.

Versions

The present C code compiles under (at least) gcc/g++ version 4.8.5 with Gnu make 3.82.
The tcsh and csh scripts run under (at least) tcsh version 6.18.01.
The perl script runs under (at least) perl version 5.16.3.
The above all run under (at least) Gnu linux 3.10.0.

Suggested Trial

To confirm that things are set up satisfactorily before attempting a completely novel run, it may be useful first to replicate a past simulation. The GitHub directory will contain config and seed files for past scenario #48; once the executables are compiled, running the "*run**" scripts in the order above should generate output files *sc48.pd.e[n]* and *sc48.pd.l[n]* that can be compared with the past results supplied in a subdirectory. The script *removeswimdir_l.csh* can then be used to remove swim directions from both sets of sightings files *(*.l[n]*), resulting in files named *sc48.pd.nsd.l[n]* that should match exactly.

If you wish to try an additional trial in a directory that already contains output from an earlier run, the script *rmoutput.csh* can be run beforehand to remove most output from the previous run.

Caveats

Not all combinations of specified model conditions have been tried, and not all would be expected to work. Whale movement and reaction in particular have been absent from all recent runs and thus should be activated, if at all, only with great caution and in simple scenarios where intended behavior can be confirmed.

Appendix II: Configuration File Formats

Original version presumably by Randall Gray, CSIRO/Tasmania, ca. 1997; revised and updated by D.W. Smith, Woods Hole, 2021 [IWC contractor; NOAA affiliate (via Integrated Statistics Inc) for Protected Species Branch, Northeast Fisheries Science Center].

The model takes several input files, including the following:

- a location configuration file (named *.lcf in iterative runs),
- a ship configuration file (*.scf),
- a roster file (*.rcf),
- a reaction configuration file (*.rea),
- a whale configuration file (*.wcf), and
- under some circumstances a dive time file (which may be automatically generated).

In all recent runs the whale configuration file has been generated automatically from the location and reaction configuration files by the perl script *runwhale_rs_tf_lc* (or its predecessors with similar names), while the reaction configuration file has been designed to generate essentially no reaction. Thus the discussion to follow will focus on the location, ship, roster, and post-processing configuration files.

The model uses MKS units (meters, kilograms, and seconds), and angles are all measured in radians within the code (though they appear as degrees in some output). Pre-processing or post-processing must be applied to convert from or to ROTF (rods, ounces [troy], and fortnights) or whatever the chosen system of units may be.

All indexing (with the exception of the B1 ... B22 parameters in the parameter sets) is done starting from zero; thus the first whale is 'whale 0', and the first weather condition is 'weather 0'.

Configuration file templates follow, but it may often prove helpful also to use a past set of configuration files as a model. As can be seen below, configuration parameters are quite numerous, meaning that many possible combinations have never been tried; thus, checking the behavior of the model after any run is advised. Care should also be taken to assure that lines in different configuration files do not conflict-- e.g. the values for "nplatforms" or "n platforms" in the ship and roster configuration files.

Note further that there should be exactly one space between parameter names and their values in config files, as e.g. in "speed 5.9201", and between words in multi-word parameter names, as e.g. in “min dive time”.

In all configuration templates below,

- things in brackets-- [] --are optional (unlike in the previous section), while
- things in braces-- {} --indicate that such blocks may be repeated.
- A vertical bar or pipe, "|", means "or" and separates two or more alternatives.

The brackets, braces, and bars are NOT part of the syntax of the configuration file.

- %f denotes a floating point number;
- %d denotes an integer; and
- %p denotes a floating point number in [0, 1].
- %t denotes a time in one of the following formats:

```
DD,HH:MM:SS
DD,HH:MM
DD,HH
DD
HH:MM:SS
HH:MM
```

When the model reads a configuration file, lines with a # as the first character are treated as comments and blank lines are skipped.

Location Configuration Files

```
=================================================================
#
#      Locate configuration file template
#

[expected number of whales %f | density of whales %f]
[expected number of clusters %f]

baseline density of clusters %f
baseline stddev within clusters %f
# set these to 0 for no custering

[firstday density of whales %f]
[lastday density of whales  %f]
#   NOTE regarding the last two lines above: should have both if one is
#   present, and must write out decimal values for the %f's--
#   can't use scientific notation because "bc" is used for calculations
#   within script "runwhalesimnew5delrl", can't handle it
today density of whales XX
#   the "XX" here is a placeholder than gets replaced automatically
#   via the "bc" calculations mentioned above at run time

[density gradient parameter %f]
# ^ a value of 1.0 gives uniform distribution

[non-monotonic density gradient]
speed %f
variance 0
# variance must be set to 0 for current version of code

min dive time %f
max dive time %f
dive mean %f
# note that dive mean given here will presumably have no effect
# if making a nonsynchronous dive time file (actual mean will depend
# on what is hard-coded in locatereadship_timefile, which is currently
# 153.0)

[make nonsynchronous dive time file]

[surface mean %f]
[cue body proportion %f]
# use cue b.p. = 0.3 if no cue effect

region %f %f [%f] | region compute
```

```
[{sclass prob and size %d %f %d}]

# the first %d is the ordinal number of the group-size-class,
# starting from 0; the %f is a *cumulative* probability of
# all group-size-classes <= the present one; and the second
# %d is the group size for this class (thus usually 1 more
# than the first %d)

[{sclass probfar and size %d %f %d}]
# like the above, except that it is the group-size-class
# distribution for the "far from ice" end of the search
# area-- i.e. the area last reached by the ship in Woods
# Hole simulations-- for cases where there is a gradient
# in group-size distribution: in such cases the "sclass prob"
# numbers above will be those for the "near end", otherwise
# those "sclass prob" numbers will apply to the whole area
# populated with whales

end
```

==

If clustering is present, baseline density of clusters is the density of whale-group clusters before any transformations to create density gradients. (The members of clusters are whale groups, which take up no space in the model though they may be composed of multiple whales.) Baseline stddev of clusters is the standard deviation of whale groups' distances from their clusters' (theoretical) centers; thus it is a *within*-cluster measure.

Speed is the speed of whales.

The first argument of the region line in the case where the region size is precisely specified is the size of the region in the x axis (sx); the second is the length in the y axis (sy); and the optional third is the "target" maximum sighting distance (radial). The locations generated will lie in [-sx/2, sx/2] x [0, sy].

If, instead, "region compute" is specified, dimensions of the region to be populated with whales will be computed based on whale speed, ship speed, and maximum sighting distance.

By default whales are uniformly distributed around the y axis and have a speed of zero, a variance of zero, and a random orientation.

If dive times are simply generated according to the specified min, max, and mean rather than being generated according to the rules for making a nonsynchronous dive time file, they will be drawn from a Poisson distribution slightly modified to accommodate the specified min and max.

A whale may only be detected when it surfaces (i.e. the probability of detection is independent of its time at the surface); thus surface mean should generally be set to zero, which is also the default.

Density gradients in whale distribution are generated as described in the section titled "Whale distribution component" in Palka and Smith (in review).

Ship Configuration Files

```
=================================================================

#
#     Ship configuration file template
#

# start, finish and velocity/speed  need to be specified carefully:
# ships can easily leave the region containing whales if individual
# whales' locations have been pre-specified

start %t
finish %t

# Set the number of ships in the run.

nships %d

[widest search angle in degrees %f]

# default for widest search angle is 90 degrees, but if errors are to be
# added in post-processing, sightings beyond abeam (i.e. at > 90 deg.)
# should be included initially because errors may bring them back within
# 90 degrees

# Individual ships may now optionally be selected and modified.
# the default starting location is the origin
# the default speed is 0, but all recent runs have used 5.9201 m/s
# (~11.5 kt)
# and the default heading is true north


[ship %d
  [location %f %f]
  [velocity %f %f]
  [speed %f]
  [direction %f %f]

# Set the number of platforms on the ship.
  nplatforms %d

# Select a platform to configure.  All platforms must be configured.
  {platform %d

# Set visibility radius in metres (default = 1853.25m ie 1 nm)
    radius %f

# Set probability of losing a whale
    [p(loss) %p]

# Set probability of missing a whale (default 0.0)
    [p(miss) %p]

# Select a function to specify parameter sets for
# 0 == [like #10; used for debugging]
# 1 == r log(x) model
# 2 == generalised product model
# 3 == generalised product model with multiple whale types
# 4 == generalised product model with tracking
# 5 == generalised product model with tracking and multiple whale
#      types
# 6 == Norway invlogit
# 7 == Norway invlogit with multiple whale types
# 8 == Norway invlogit, incl. whale's orientation to observer
# 9 == Norway invlogit plus B6 param = min radial distance for wide scanning
#      test
#10 == Norway invlogit plus params for school size
#11 == see everything (at least within activation radius ?-- used for
#      debugging)
#12 == Norway invlogit plus params for school size,
#   *plus* params for cue effect added by DWS for D. Palka (Jan 2004)

    function %d

# Set the number of parameter sets
    nparamsets %d

# Specify up to 22 parameters within the set (B1 ... B22); all
# parameter sets and parameters which are not specified have zeros in
# all parameters.

    {parameter set %d %f {[%f]}}
  }
]

# There must be nships such blocks, and the file must be
# terminated by an end statement

end
```

==

Velocity and speed/direction are mutually exclusive; only one or the other should be specified. (Velocity and direction have not been specified in Woods Hole runs [thus the direction has taken the default value]. Probabilities of losing or missing a whale [p(loss) and p(miss)], independent of specified detection functions, have also not been specified.)

Detection function #12 has been used in all recent runs in Woods Hole; it is described in the section of Palka and Smith (in review) titled "Component for detection of a whale group." Specifications for other functions may be found in the code of source file prob.cxx.

While there is provision for having more than one ship in a run, it is not recommended (and has not been attempted for any model runs done in Woods Hole). The simulator will run more efficiently with only one ship, and better results are likely from running the simulation twice with the same whale configuration and different starting points for the ship.

Roster Configuration Files

The roster configuration file creates roster configurations for single ships and single functions. That is, each function on each ship must have its own roster configuration file (if it is to be used), though a single roster file suffices if the same function is used with different parameter sets. If there are multiple roster files, these must then be concatenated to produce the working roster.

Roster configuration file template:

```
=================================================================
#
#     Roster configuration file template
#

# Set ship and function this file is for

ship %d
function %d

# Set starting and finishing time for this roster
start %t
finish %t

# Set the number of platforms this is for
n platforms %d

# Set the number of watches
n watches %d

# Set the number of weather conditions
n weathers %d

# Set the length of a watch
watch %t

# Activate a new weather condition
{[weather %d %t]}

end

=====
```

OR, if weather is to change during each day in an iterated run, the format for a weather line changes from the above to
{[weather %d XX,%t]}
where XX is a placeholder for a day number that will get replaced during iteration, and the %t (time) contains no units greater than hours, e.g.:

```
weather 0 XX,0:0
weather 1 XX,2:40
weather 2 XX,10:40


=================================================================
```

This determines the weather/watch combination for each platform at a given time. A parameter set for a given function is then selected based on the current watch and weather condition in the following way:

PSn = Wc + Wn * NWc

where
PSn is the parameter set number,
Wc is the weather condition,
Wn is the watch number and
NWc is the number of weather conditions possible.

The primary change made in the roster configuration file between different scenarios run in Woods Hole has been to vary weather conditions.

The roster file itself:

```
======================================================================
#
#     Roster file for ship's platforms
#
#  This file is usually generated by mkroster using a roster
#  configuration file.


{%lu %d %d %d %d}

end

======================================================================
```

The first field in a line is a long unsigned integer which represents a number of seconds past the beginning of the simulation (usually taken to begin at 00:00:00, Jan. 1, 1995). The next arguments select respectively the ship number, platform number, function number and parameter set to use.

Once the internal time reaches the target time in the roster file, the current function and parameter set are updated for the selected ship and platform pair.

Note on Reaction Configuration Files

The reaction configuration file for all recent runs has been designed to result in essentially no reaction (see sample), so the user need not be concerned with most details of that file unless this changes.

Note, though, that MaxDive and MaxSpeed in that file should not be set any lower than "max dive time" and "speed" in the location configuration file.

Post-processing (Closing Mode) Configuration Files

Post-processing configuration file template:

```
======================================================================
#
error_model %d
truncate_times %d
confuse_wids %d
#
======================================================================
```

error_model can take values (-1, 0, 1, 2, 3, 4): -1 adds no errors, while 0 through 4 add errors according to five different models. In the most recent suite of runs done in Woods Hole, error model 3 was used for all scenarios incorporating errors. This model generates errors as described for scenarios 37-54 in the section of Palka and Smith (in review) titled "Radial distance and sighting angle measurement errors." Specifications for other error models may be found in the code and comments of source file clmode_brstop.cxx.

truncate_times can take values 0 or 1, determining whether times are to be truncated to the minute (1) or not (0).

confuse_wids can also take values 0 or 1, determining whether whale ID's are to be confused according to a process described in the section of Palka and Smith (in review) titled "Duplicate sighting mis-identification"
(1), or not (0).

Example Configuration Files

Sc48.scf – ship configuration file

```
start 0
finish 0,16:0
nships 1
ship 0
location 0 18532.5
speed 5.9201
# 11.5 knots, in m/sec
#
nplatforms 3
#
#  like scenario 34 for 2006 runs;
#  also scenario #17 (formerly #1) for 2005 runs;
#  and, e.g., scenarios #29, #31
#
#         -- 1-9 Apr 2005; 5 Jan 2006; 18 Nov 2007 / DWS

#  IO platform
platform 0
radius 9000
function  0
nparamsets 1
parameter set 0  0.00 00  0 0 0
function 12
nparamsets 6
parameter set 0  0.0018 1700 4.0 0.45 -1.0  450 0.15 0 0
parameter set 1  0.0022 1100 5.0 0.35 -1.25 350 0.10 0 0
parameter set 2  0.0025  700 6.0 0.20 -1.50 250 0.05 0 0
parameter set 3  0.0018 1700 4.0 0.45 -1.0  450 0.15 0 0
parameter set 4  0.0022 1100 5.0 0.35 -1.25 350 0.10 0 0
parameter set 5  0.0025  700 6.0 0.20 -1.50 250 0.05 0 0


#  TOP platform
platform 1
radius 9000
function  0
nparamsets 1
parameter set 0  0 0 0 0 0
function 12
nparamsets 6
parameter set 0  0.0010 1900 3.5 0.50 -1.00 450 0.15 0 0
parameter set 1  0.0015 1500 4.5 0.40 -1.25 350 0.10 0 0
parameter set 2  0.0020 1100 5.5 0.25 -1.50 250 0.05 0 0
parameter set 3  0.0010 1900 3.5 0.50 -1.00 450 0.15 0 0
parameter set 4  0.0015 1500 4.5 0.40 -1.25 350 0.10 0 0
parameter set 5  0.0020 1100 5.5 0.25 -1.50 250 0.05 0 0


#  bridge
platform 2
radius 9000
function  0
```

```
nparamsets 1
parameter set 0  0 0 0 0 0
function 12
nparamsets 6
parameter set 0  0.0022 700 5.5 0.40 -1.0 450 0.15 0 0
parameter set 1  0.0024 500 6.0 0.35 -1.25 350 0.10 0 0
parameter set 2  0.0026 300 6.5 0.30 -1.50 250 0.05 0 0
parameter set 3  0.0022 700 5.5 0.40 -1.0 450 0.15 0 0
parameter set 4  0.0024 500 6.0 0.35 -1.25 350 0.10 0 0
parameter set 5  0.0026 300 6.5 0.30 -1.50 250 0.05 0 0


end
##********end of ship config file *************
```

sc48.rcf – roster configuration file

```
ship 0
function 12
start 0
finish 0,16:0
watch 0,2:0
n platforms 3
n weathers 3
n watches 2
#  note that weather will not necessarily change at these exact times
#  after post-processing to incorporate closing mode        --Feb 2005, DWS)
weather 0 XX,0:0
weather 1 XX,2:40
weather 2 XX,10:40
#  (runwhalesimnew* puts in successive day numbers for these XX as it cycles,
#   as I understand it--                                      Apr 2003, DWS)
#  (but it edits the *.r and *.s files it creates once it is done
#   [has always done so], so it is hard to see what has happened--
#                                                             Feb 2005, DWS)
end
##******end of roster config file *********
```

sc48.lcf – location configuration file

```
speed 0
variance 0
region compute
# ^ -- try new syntax-- DWS
baseline density of clusters 0
baseline stddev within clusters 0
firstday density of whales 0.000000043674
lastday density of whales  0.000000014558
#   note, can't use scientific notation ^ here if using
#   "bc" for calculations within script
today density of whales XX
##  trying 0.15 whales per nm^2 for *starting* day's density,
##  0.05 for last day's density (hence 0.10 for avg) --
##  "today's" density of whales should get computed, based on current
##  day number, by "bc", invoked within runwhalesimnew* ;
##  result should get inserted in place of the "X"'s above by
runwhale_rs_tf_lc
density gradient parameter 0.6
non-monotonic density gradient
cue body proportion 0.3
#  use cue b.p. = 0.3 if no cue effect
make nonsynchronous dive time file
min dive time 5
max dive time 480
dive mean 75
# dive _mean_ given here will presumably have no effect if
# doing nonsynchronous dive times (actual mean will depend
# on what is currently hard-coded in locatereadship_timefile)
#                    (9 Apr 05 DWS)
#
# dive params based on Antarctic data from Debi;
# max was 240 when based on min(?) avg blows per minute,
# but now increased (20 Feb 03 DWS)
#
# group size distrib from later batch of SH minke data--
# for "near-to-ice" area
sclass prob and size 0  0.4215   1
sclass prob and size 1  0.6634   2
sclass prob and size 2  0.8048   3
sclass prob and size 3  0.8925   4
sclass prob and size 4  0.9335   5
sclass prob and size 5  0.9575   6
sclass prob and size 6  0.9702   7
sclass prob and size 7  0.9787   8
sclass prob and size 8  0.9815   9
sclass prob and size 9  0.9872   10
sclass prob and size 10 0.9886   11
sclass prob and size 11 0.9914   12
sclass prob and size 12 0.9928   13
sclass prob and size 13 0.9956   14
sclass prob and size 14 0.9970   15
sclass prob and size 15 0.9970   16
sclass prob and size 16 0.9984   17
sclass prob and size 17 1.0000   18
```

```
#
# distrib for "far-from-ice" end
#
sclass probfar and size 0 0.6703   1
sclass probfar and size 1 0.9385   2
sclass probfar and size 2 0.9921   3
sclass probfar and size 3 0.9992   4
sclass probfar and size 4 0.99994  5
sclass probfar and size 5 1.0000   6
#
end
##*******end of locate config file *******
```

sc48.rea – reaction configuration file

```
# these values should generate essentially no reaction
ReactionRadius 926.625
AlphaDive 10000.0
AlphaSpeed 10000.0
AlphaDirection 10000.0
BetaDive 10000.0
BetaSpeed 10000.0
BetaDirection 10000.0
MaxDive 480
MaxSpeed 5.9201
StdDevSpeed 1.0
StdDevDirection 0.1
StdDevDive 1.0

end
```

sc48.ccf – post-processing configuration file

```
error_model -1
truncate_times 0
confuse_wids 0
end
```

Appendix III: Output Data Formats

(Much of the following section is a slightly adapted excerpt from Annexes 2 and 3 of Palka and Smith [in review].)

Effort Output Data Format

The effort output files are initially named effort.log and are concatenated into files named *[filename-root].e[repnum]*.

Each line in an effort file provides the time at which a condition started; for example, the time when a new watch team starts, or the weather changes, or IO mode starts after being in Closing mode, or surveying ends for the day.

The fields in the effort file include:

1. Year
2. Month
3. Day
4. Hour:Minute:Second
5. Platform number
6. Watch code
7. Weather code
8. Mode code
9. X position
10. Y position

The codes used in these fields are:

Platforms:	0 (IO platform)
	1 (Topman)
	2 (Bridge)
	-1 (no change from before)
Watch codes:	0 (first watch)
	1 (second watch)
	-1 (no change from before)
Weather codes:	0 (calm, like Beaufort 0)
	1 (not so calm, like Beaufort 1 and 2)
	2 (bad weather, like Beaufort 3 and above)
	-1 (no change from before)
Mode codes:	I (IO mode)
	C (Closing mode)
	N (going on-effort during Closing mode now)
	F (going off-effort during Closing mode now)
	E (end of effort for the day)
	W (weather condition changed now; or initial weather for day)

The effort file contains one row for each thing that changes. As in the example below, for the first time of the day, the mode is indicated in the first line and then there are three rows to indicate the watch for each platform. There is a code to indicate which watch team on a platform worked. Thus, in the example below, on Month=1, Day=1, Year=1995, surveying started at 0:0:0; each platform started with watch team 0; and the IO mode started. Two hours later, at 2:0:0, the watch teams on each platform switched, so watch 1 was now on duty and watch team 0 was off duty. Since IO mode had not changed, there is no row for such a change at time 2:0:0.

As a side effect of the processing that generates these files, lines specifying which teams were on watch might occasionally appear when the teams are not actually changing. These lines should correctly specify the teams that continue on watch at such times.

An example of the effort data (with column headings added) follows:

```
Yr   mon day hh:mm:ss     platform watch  weather mode     Xposn      Yposn
1995   1   1  0: 0: 0        -1       -1        0    W        0       18532
1995   1   1  0: 0: 0        -1       -1       -1    I        0       18532
1995   1   1  0: 0: 0         0        0        0    I        0       18532
1995   1   1  0: 0: 0         1        0        0    I        0       18532
1995   1   1  0: 0: 0         2        0        0    I        0       18532
1995   1   1  2: 0: 0         0        1        0    I        0       61157
1995   1   1  2: 0: 0         1        1        0    I        0       61157
1995   1   1  2: 0: 0         2        1        0    I        0       61157
```

Sighting Output Data Format

The sightings output files are initially named platform.log and are concatenated into files named *[filename-root].l[repnum]*, where the character before *[repnum]* is a lowercase "L".

The fields in the sighting data files are:

1. Year
2. Month
3. Day
4. Hour:minute:second
5. Platform code number (0=IO, 1=Topman, or 2=bridge)
6. Whale group id number
7. Distance ahead along the trackline - see picture below (in meters)
8. Perpendicular distance from the track line to a whale group (in meters)
9. Radial distance between ship and whale group (in meters)
10. Angle of sighting (in degrees)
11. Swim direction of whale group (in degrees)
12. Group size (if 0, assume 1 whale)
13. Watch code (0 or 1)
14. Weather code (0, 1, or 2)
15. Cue code (0=body, 1=blow)
16. X position (in meters)
17. Y position (in meters)
18. Mode (0=IO mode, 1=Closing mode).
19. Group size confirmation status (0=group size not confirmed, 1=confirmed group size)

The value of the time variable is the time when the sighting was detected. The X, Y position is the ship's position at the time of the sighting.

Sighting angles and swim directions (Figure A3.1) range from [0° – 360°). For sighting angles, 0° is assigned to a group that is directly ahead on the track line, 90° is assigned to a group directly abeam on the right hand side of the ship, 180° to a group on the track line but behind the ship, and 270° to a group directly abeam on the left hand side of the ship. With current parameter settings, sightings outside of a 60-to-300 degree window will usually not be recorded.

Swim directions follow the same circular assignment but indicate the direction in which a group is swimming. For example, a swim direction of 0° indicates that a group is swimming in the same direction as the ship's travel, where the group is located parallel to or on the track line. A swim direction of 270° indicates that a group is swimming in a right-to-left direction no matter if the group is on the right or left hand side of the ship (see example on right side of Figure A3.1).

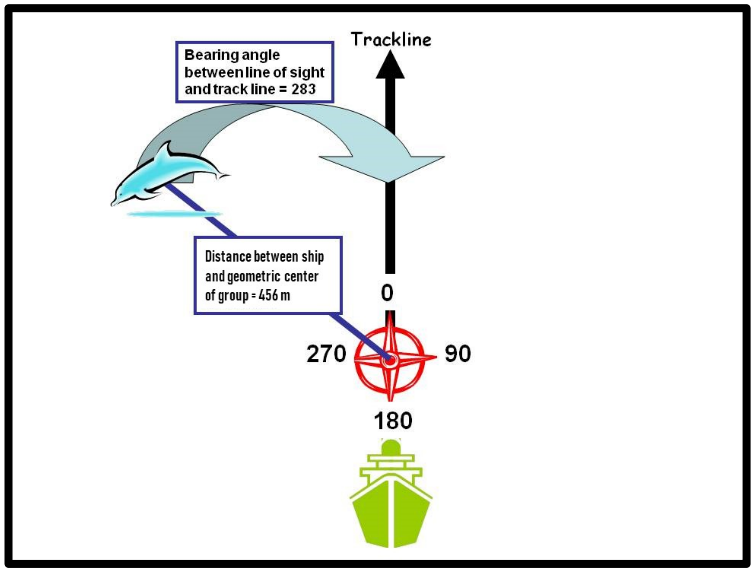
Perpendicular distances are either positive or negative. Negative perpendicular distances indicate a sighting is on the left side of the ship, that is, where the sighting angles were greater than 180°.

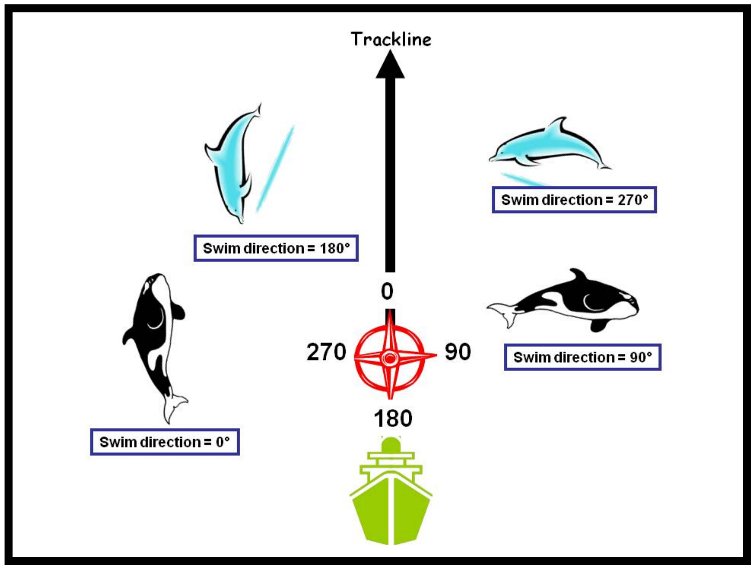
Group sizes recorded in this file are the same for all teams/platforms that detect a given group, except in scenarios where recorded group sizes could have errors. Sizes for a given group can be different for different platforms in cases where initial detections of that group did not trigger a closing and a group size recording error occurs, but a subsequent detection of that group by a different platform does trigger that group to be closed on. After the group is closed on, the recorded group size is the correct group size.

Example of sighting data:

year	mon	day	hh:mm:ss	plat	id	ahead	perp	radial	angle	swim direction	school size	watch	weather	cue	X	Y	mode	confirmed
1995	1	1	1:14: 0	1	94	2503.93	1629.40	2987.41	33.05	212.51	2.00	0	0	0	0	45043	0	0
1995	1	1	1:16: 0	1	184	1073.34	384.11	1140.00	19.69	348.59	2.00	0	0	1	0	45866	0	1
1995	1	1	1:18: 0	0	184	721.11	428.70	838.91	30.73	348.59	2.00	0	0	1	0	46552	0	1

Figure A3.1 Diagram of the definition of the sighting angle (left side) and swim direction (right side).


Trackline
Bearing angle between line of sight and track line = 283
Distance between ship and geometric center of group = 456 m
0
270
90
180



Additional Output Files

The following further output files are also still produced, in some cases for historical reasons, but they have not been used in statistical analyses of data produced in Woods Hole and would generally be useful mainly for debugging. They appear in rough order of likely usefulness. For iterative runs they are concatenated over the days within a replicate if so noted:

- platform2.log -- concatenated into [filename-root].ll[repnum], where "ll" is lowercase "LL"; like platform.log except that the last four columns are replaced by two other final columns: a time in seconds (only) counting from the start of the first day in this replicate, and a number indicating how many whales in the present group surfaced at this time
- seq_overruns.log -- concatenated into [filename-root].sq[repnum]; records cases where a preset dive sequence for a whale group has been overrun (requiring the model to start again at the beginning of that sequence)
- whale.log -- saved as whale.log.[repnum].[daynum] but not concatenated; at present only the files for days 1 and 2 of each rep are saved
- reaction.log -- concatenated into [filename-root].rl[repnum], where “rl” is lowercase “RL”; these files are usually quite large, so they are next gzipped, with ".gz” thus appended to the filename; then, since whale reactions have not been activated in any recent runs, most are automatically deleted, with only the *.rl* files for the first and last reps retained
- [filename-root].d[repnum] -- currently contains only information about numbers and densities of whales; may be correct only for last day of this rep (this file was apparently also used to record some form of output analysis in CSIRO runs)
- [filename-root].o[repnum] -- records some files used when invoking this replicate