MULTIFAN-CL User's Guide

Pierre Kleiber, David A. Fournier, John Hampton, Nick Davies, Fabrice Bouye, and Simon Hoyle

September 1, 2018

Contents

List of Figures

List of Tables

Recent developments

A brief chronological overview of developments made to MULTIFAN-CL over the past 1, or more, years is provided to assist regular users in applying the latest features.

23 February 2011 – version 1.1.1.0

- The number of estimable parameters has been increased from 7000 to 8000. This was found to be necessary since some complex models were exceeding the limit.
- The time period over which the Beverton-Holt stock-recruitment relationship (SRR) is calculated has been made flexible and user-defined. Earlier versions assume the entire model calculation period in deriving the SRR as used for equilibrium yield estimation. This is now the default option and flags may be set to define a part of the model period to be used for the SRR. This feature enables users to explore the effects of selecting a period considered to be more reliable or representative.

5 August 2011- version 1.1.2.0

- The facility to undertake model simulation runs that include projections into the future with stochastic recruitments.
- Population deterministic projection scenarios that explore a range of assumptions regarding alternative recruitment regimes: either, in respect of the stock-recruitment relationship (either the entire modelling period or a part thereof), or, an average of absolute recruitments over a user-specified period.
- A streamlined procedure for controlling the inputs of the parameters associated with estimating recapture rates of tagged fish relative to both release group and fishery group.
- Detailed output of tag recaptures that permits detailed diagnosis of model fit to observations, for example, a consideration of the observed and expected rates of tag attrition.

25 October 2011 - version 1.1.3.1

- New input .ini file format entailing a version number that ensures backward compatibility
- Output file controls have been defined for particular report files. Three output options exist: Full; Streamlined; and Silent. Streamlined and silent options achieve faster performance for single model evaluations.

2 February 2012 - version 1.1.4.2

- New input .ini file format now includes the value of the steepness parameter for the Beverton-Holt stock recruitment relationship where it is initialised or fixed when commencing a model fit.

- The maximum fishing mortality value has been raised to facilitate very high potential "observed" catches being achieved in model projections.
- A new simulation output file containing age-specific fishing mortalities for each time period is produced.
- Simulations can now be operated for projections under zero fishing mortality. This is sometimes required for risk analysis in respect of biological limit reference points in respect of unfished biomass.
- An option has been added for applying a bias correction to the recruitment predictions from the BH-SRR so as to approximate the mean of the estimated recruitments. These corrected predictions may then be used for equilibrium yield calculations and deterministic projections.

15 November 2012 – version 1.1.4.3

- Refinements have been made to the Hessian options 7 and 8, and improvements to the operations for stochastic simulation projections under zero fishing. These allow simulations under no fishing mortality to be run simultaneously with those under specified levels. Input and output filenames have been revised. Should simulations need to be run incrementally, e.g. to undertake Harvest Control Rule evaluations, this is now possible.
- A small fix has been made to the month doubling in respect of tagging data
- An additional output has been added to the report file *plot.rep* to include the tag_fish_group_flags so as to facilitate labeling the unique tagging reporting rates in the MULTIFAN-CL viewer.

25 March 2013 – version 1.1.5.4

- New age_flags[178] for applying the constraint on the sum-product of regional recruitment proportions and the regional recruitment deviates to equal 1 in each model time period. Set age_flags[178] = 1 to activate the constraint.
- New parest_flags[239] that activates stochastic recruitments in projections as deviates about the Beverton-Holt stock-recruitment relationship (BH-SRR) predictions where the deviates and the relationship are taken over a subset of the model time period.
- Revision to the application of age_flags[145] such that negative values will enable low penalty weights upon the fitting of the BH-SRR.
- A small improvement to a routine that returns the BH-SRR parameters.
- Improvements to several routines to maintain stability during the control phases.

5 November 2013 – version 1.1.5.5

- Several fixes were made to the predictions of recruitments under the scenario of zero fishing mortality as made from the Beverton-Holt stock-recruitment relationship and from the average of the absolute recruitment estimates.

14 February 2014 – version 1.1.5.6

- A minor modification was made to this re-build version that enables non-integer tag release length frequency data to be input in the .tag file.

5 September 2014 – version 1.1.5.7

- A correction was made to the tag reporting rate used for calculating the predicted tags caught during the mixing period (fishery-specific or both fishery- and release group-specific), and these rates are subsequently reported with the predicted tags by period at liberty in the plot.rep output file. The redundant output files "fish-rep-rate.dat" and "fish-rep-rate2.dat" are now no longer generated.
- The parest_flags vector was increased from 300 to 400 elements so as to accommodate new development features which has increased the .par file version number to 1046.
- Temporal recruitment deviates may be excluded from the parameters being estimated for the number of terminal time periods specified by the value assigned to parest_flags(400).
- The number of independent variables able to be estimated has been increase to 10,000.
- The interrupt facility using "ctrl-c" and "q" was adjusted to operate properly for the Windows executable.

12 September 2014 – version 1.1.5.8

- A correction was made to the fishing mortality calculations for a potential error which could occur in certain instances relating to the fishery subscript for a temporary storage variable.

12 February 2015 – version 1.1.5.9

- A fix was made to a bug relating model calculations under the assumption of zero fishing effort when applied to stochastic projections that affected:
  - Projection recruitments as Stochastic deviations about BH-SRR predictions in projection years
  - BH-SRR multipliers on estimated recruitments during estimation model years
  - Stochastic recruitments selected from a subset period of the estimation model years

29 February 2016 – version 2.0.0.1

- Substantial revisions and additions have been made that include:
  - Dimensions added for multi-species/sexes
  - Multi-sex equilibrium yield calculations
  - Time-variant selectivities
  - Including age-length data in the model fit
  - Tagging negative binomial likelihood - revised formulation and allow relative weighting
  - Likelihood components report
  - Tagging diagnostics report
  - Fitting the BH-SRR to calendar year (annual) recruitments
  - Fixed terminal recruitment deviates
  - Non-integer tag release frequencies allowed
  - Size composition normal likelihood with a student-t probability density function
  - Zero selectivity-at-age specified for particular young age classes

  - Tail compression of size composition data
  - Minimum sample size threshold applying to size composition data
  - Fitting during the control phases with fixed growth
  - Cobb-Douglas biomass-related effect on catchability - substantial revisions

20 April 2016 – version 2.0.0.2

- Minor revision (build) to:
  - Report initial tag release frequencies specific to age class to the output file "temporary_tag_report". Also minor fixes were made to the calculations for: pooled tag group negative binomial likelihood, and robust normal size composition likelihood

28 April 2016 – version 2.0.0.3

- Minor revision (build) to:
  - Output report of variance of the BH-SRR to the plot.rep

21 June 2016 – version 2.0.1.1

- Revisions (revision) to:
  - The independent variable "recr" was restructured to be recr(2,nyears).
  - The feature implemented by parest_flags(400) was extended into the multi-species dimension.
  - The functionality of age_flags(95) as supplying a multiplier to M or Z for the initial equilibrium population conditions was transferred to age_flags(128). The existing functionality of age_flags(95) to specify the period over with the initial population is averaged is retained.

24 February 2017 – version 2.0.2.1

- Substantial revisions and additions have been made that include:
  - An improved algorithm for applying stochastic recruitment deviates to BH-SRR predictions in simulation projections
  - Predictions of the annualised BH-SRR are allocated by the average seasonal recruitment in the projection periods
  - Tail compression was extended for weight composition data with a revised penalty formulation
  - Dirichlet-Multinomial likelihood for size composition data
  - Improved minimization stability
  - The self-scaling multinomial M-estimator (SSMULT) implementation has been completed, and the phi function formulation was revised
  - Simulation mode for generating pseudo-observed data
  - Autocorrelation in recruitment estimates can be derived
  - Improved the variance calculation to use the OpenBLAS library routine for the singular value decomposition
  - Functional forms for natural mortality at age

- st.dev report includes natural mortality age-specific deviates
- Fixed a bug in the st.dev report
- Fixed a bug for fitting an estimation model that includes fisheries data for a projection period
- Fixed a bug in the input of size frequency that excluded the observation in the first class interval
- Added a "catch" to detect when a vector of zeroes in size frequency data is input in the .frq file. MULTIFAN-CL will stop with an error message.
- New species parameters have been added to the .par file
- MULTIFAN-CL version number has been added to the plot.rep file
- Version of par file is 1052 and MULTIFAN-CL version is now 2.0.2.1

6 March 2017 – version 2.0.2.2

- A revision has been made:
  - An improved formulation to the logistic selectivity at age calculation has been made that is more robust to wide-ranging parameter estimates. This results in more stable minimisation and reduces the need for the generic selectivity penalty activated by parest_flags(74).

16 January 2018 – version 2.0.3.1

- Revisions and additions have been made that include:
  - Enabled likelihood profiling in respect of the model derived quantity for average biomass over a defined period or a specified level of absolute biomass depletion, with each being specified either as total or adult biomass. A bash script file is appended to the User's Guide to assist with running the profile likelihood calculations.
  - An improvement was made to the method for fitting the BH-SRR with estimated autocorrelation in the deviates, such that the relative weight of the AR(1) penalty term is controlled using the standard age_flags(145) setting for activating the BH-SRR fit. The autocorrelation moment estimate is calculated by default (even if the feature is not activated), and is reported to the plot.rep report file.
  - Visual Studio 2017 C++ compilation of a Windows executable. This is a change from previous versions that used a MinGW compilation.
  - A number of improvements relating to multi-sex and multi-species applications:
    - length-specific selectivity may be shared among sexes that may be dimorphic (different growth rates)
    - option for including sex ratio in the predicted size compositions that are aggregated among sexes
    - kludge for the fish_flags grouped_flag_sanity_check() routine to allow for the special case of multi-sexes with shared length-specific selectivity
    - including identifiers for sex in output reports
  - Improvements and refinements to the self-scaling Multinomial and Dirichlet- Multinomial methods for fitting size-composition data.
  - Integrated the routines that apply tail compression to size composition data.
- The following corrections were made relating to multi-sex and multi-species applications:

  - correction to test_plot_output report components (bh_steep_contribution; and tot_catch)
  - correction to initial equilibrium numbers in the aggregate age class
  - correction to the multi-sex calculation of equilibrium spawning biomass of

16 July 2018 – version 2.0.4.0

- specific options available for selection of function minimisers
- generation of simulation pseudo-observations for tagging data
- estimating recruitments using an orthogonal-polynomial parameterisaton
- improvements were made to the self-scaling Multinomial method without random effects estimation for fitting size-composition data to extend this for weight frequency data
- various fixes relating to exploitable biomass report in the *plot.rep* file, and to the AR(1) process penalty term used for estimating autocorrelated recruitments

31 August 2018 – version 2.0.5.1

- self-scaling multinomial M-estimator for composition data is improved by the development of an auto-regressive auto-regressive method for estimating an autocorrelation in the size composition residuals. This is still in development.
- a diagnostic check of the input and output of the *.par* file is added, controlled by a parest_flags setting
- a model fit diagnostic of the Hessian using the singular value decomposition with an option for producing a report
- optimising simulation and operating model calculations, with options to exclude certain calculations and report generation to improve performance
- sharing effort_dev_coffs parameters for the multi-sex model so as to improve robustness when only aggregated data is available
- extensions to the simulation mode for generating pseudo-observations for the estimation model time periods, including tagging data having observation error in the reporting rate probabilities
- implementation for a maturity-at-length feature that enables the input of a maturity at length ogive with the conversion to maturity at age done internally using growth estimates
- correction to the *xinit.rpt* report file to reinstate the recr parameters
- correction for simulation recruitments where age_flags(182)=0
- additional stochastic simulation reports for spawning biomass and Fmult

Descriptions on how to apply these new and revised features are provided in the updated "MULTIFAN-CL User's Guide" dated January, 2018.

List of Tables

List of Figures

Chapter 1

Introduction

MULTIFAN-CL is a computer program that implements a statistical, size-based, age-structured, and spatial-structured model for use in fisheries stock assessment. The model is a convergence of two previous approaches. The original MULTIFAN model1[1] provided a method of analysing time series of length-frequency data using statistical theory to provide estimates of von Bertalanffy growth parameters and the proportions-at-age in the length-frequency data. The model and associated software were developed as an analytical tool for fisheries in which large-scale age sampling of catches was either not feasible or not cost effective, but where length-frequency sampling data were available. MULTIFAN provided a statistically-based, robust method of length-frequency analysis that was an alternative to several ad hoc methods being promoted in the 1980s. However, MULTIFAN fell short of being a stock assessment method as the end point of the analysis was usually estimates of catch-at-age (although later versions included the estimation of total mortality and yield per recruit).

The second model (actually the first, in terms of chronology) was that introduced by Fournier and Archibald[2]. The FA model was a statistical, age-structured model in which estimates of recruitment, population-at-age, fishing mortality, natural mortality and other estimates useful for stock assessment could be obtained from total catch and effort data and catch-at-age samples. In principle, the estimates of catch-at-age obtained from the MULTIFAN model could be used as input data to the FA model and a complete stock assessment analysis conducted.

Such a sequential approach to length-based stock assessment modeling had several serious limitations. First, it was extremely unwieldy. Second, it was difficult to represent and preserve the error structure of the actual observed data in such a sequential analysis. This made estimation of confidence intervals for the parameters of interest and choice of an appropriate model structure for the analysis problematic. It was clear that an integrated approach was required, one that modeled the age-structured dynamics of the stock, but which recognized explicitly that the information on catch-at-age originated with length-frequency samples.

The early versions of MULTIFAN-CL, which were developed for an analysis of South Pacific albacore[3], provided the first attempt at developing a statistical, length-based, age structured model for use in stock assessment. Subsequent versions of the software have added new features, the most important of which have been the inclusion of spatial structure, fish movement and tagging data in the model[4].

[1] Fournier, D.A., Sibert, J.R., Majkowski, J., and Hampton, J. 1990. MULTIFAN: a likelihood-based method for estimating growth parameters and age composition from multiple length frequency data sets illustrated using data for southern bluefin tuna (*Thunnus maccoyii*). *Can. J. Fish. Aquat. Sci.*, 47:301-317.

[2] Fournier, D., and Archibald, C.P. 1982. A general theory for analyzing catch at age data. *Can. J. Fish. Aquat. Sci.*, 39:1195-1207.

[3] Fournier, D.A., Hampton, J., and Sibert, J.R. 1998. MULTIFAN-CL: a length-based, age-structured model for fisheries stock assessment, with application to South Pacific albacore, *Thunnus alalunga*. *Can. J. Fish. Aquat. Sci.*, 55:2105-2116.

[4] Hampton, J., and D.A. Fournier. 2001. A spatially disaggregated, length-based, age-structured population model of yellowfin

MULTIFAN-CL is now used routinely for tuna stock assessments by the Oceanic Fisheries Programme (OFP) of the Secretariat of the Pacific Community (SPC) in the western and central Pacific Ocean (WCPO). Beginning in 2001, the software gained additional users, with stock assessment applications to North Pacific blue shark, Pacific blue marlin, Pacific bluefin tuna, North Pacific swordfish and Northwest Hawaiian lobster underway or planned. The more widespread use of the software prompted the development of this user's guide and software availability via the world wide web. Further development of the software, and of the user's guide, will be an ongoing, collaborative effort.

tuna (*Thunnus albacares*) in the western and central Pacific Ocean. *Mar. Fresh. Res.*, 52:937-963.

Chapter 2

Overview of MULTIFAN-CL

2.1 What is MULTIFAN-CL?

MULTIFAN-CL is a size-based, age-structured model that is used in fisheries stock assessment. In its most basic form, the model is fit to time series of catch and size composition data from either one or many fishing fleets. Size composition data may be in the form of either length or weight-frequency data, or both. The model may also be fit simultaneously to tagging data, if available. Other information is provided to the model in the form of fishing effort data and "prior" information on estimates of various biological and fisheries parameters and their variability. Potentially, other sources of information could also be incorporated into the model fitting process, such as information on growth from tagging data or otolith ring counts, information regarding the impact of environmental variability on various processes, and so on. The ability of MULTIFAN-CL to "integrate" information from a variety of sources is one of its great strengths.

MULTIFAN-CL has the ability to estimate a potentially large number of parameters. Many of the parameters that are estimated describe the process variability in fishing mortality and are of interest for diagnostic purposes (in a similar fashion to residuals) rather than being germaine to understanding the status of the stock. Other parameters, such as recruitment, population biomass, natural mortality, selectivity and catchability are of direct stock assessment importance. Several methods of summarizing stock status in terms of commonly used reference points, such as maximum sustainable yield (MSY) and related quantities, are incorporated into the model. MULTIFAN-CL can also project fish abundance and catch into the future beyond the end of the input data using parameters estimated by the data and assumptions concerning future recruitment and fishing effort in the various fleets.

Two features of MULTIFAN-CL that set it apart from most other stock assessment models are temporal variability in efficiency of fishing gear and spatial variability in the density and size composition of the fish. MULTIFAN-CL does not make the usual assumption that catchability for any particular fishing gear remains constant in time, but instead estimates a time series of catchability for each fishery (or group of fisheries) identified in the model. Thus the common phenomenon of increasing catchability due to developing technology and know-how can be accommodated as well as the possibility of up or down changes in catchability for any other reason.

Most stock assessment models treat the fish population as a homogeneous "unit stock" with the whole population vulnerable to all of the fisheries operating in the model. However MULTIFAN-CL divides the model universe into regions with fish in a particular region at a particular time vulnerable only to fishing gear operating in that region and that time. The fish become vulnerable to other gear at other times only by moving to other regions according to movement coefficients estimated from the data. A classical preoccupation in stock assessment is determining whether a population should be treated as a "unit stock" or as a collection

of regional sub-stocks, each treated independently as a separate unit stock. MULTIFAN-CL does not impose such a rigid choice of stock structure taken from either of these two extremes, but instead allows some degree of intermediate regional autonomy. Regions are connected or isolated by the degree of fish movement between them. Stocks in each region experience region-specific recruitment and time series of abundance trends as well as region-specific fishery characteristics and levels of fishing effort.

For assessments of highly migratory species such as tunas and billfish, the implication of the above is that the analysis is best undertaken if possible on a basin scale, with the extent of population mixing embodied in the spatial configuration and movement parameterization of the model. However, for reasons of data availability, management jurisdictions and the like, assessments are often undertaken on a sub-basin scale. For example, assessments of skipjack and yellowfin tuna in the Pacific Ocean are undertaken for the western and eastern components of the ocean (divided at 150_W longitude) separately. While the these species are distributed throughout the tropical and sub-tropical Pacific, mixing between the western and eastern Pacific is believed to be limited enough (in part because of their relatively short life spans) to have little impact on the dynamics of the respective sub-populations. So we probably do not lose much information by conducting assessments on a sub-basin scale, even though the distributions of the populations are basin scale. Of course, we would not lose anything in a basin-scale analysis, even if the sub-populations were completely separated, and we may in fact gain information in subtle ways (through "cross-fertilization" of information from one region to another).

But what about undertaking MULTIFAN-CL analyses over more restricted spatial horizons? Would, for example, an analysis of skipjack tuna data pertaining to the region approximated by the EEZ of Papua New Guinea provide useful information on local population dynamics and fishery impacts? Given the magnitude of the catch in PNG and the short life span of skipjack tuna, the answer is probably "yes". However, a more satisfactory approach in this case would be to define the PNG EEZ as a "region" (see below) of the western Pacific and model the local dynamics in the context of the entire western Pacific stock. MULTIFANCL would allow such an approach to be taken. We would not, however, advocate the use of this software to conduct isolated analyses of small stock components where the fisheries represent a relatively small proportion of the total catch.

2.2 Some concepts

• Region — The geographic area considered by the model is divided into a mosaic of sub-areas here dubbed regions. The fish population is modeled at the region level, and inter-region mixing is controlled by movement coefficients. Fisheries (see below) are region specific, but biological processes such as growth and natural mortality are assumed to be common among regions. There are various considerations involved in choosing the regions. Ideally, regions should reflect some degree of spatial homogeneity in the population, so definition according to oceanographic or ecological regimes might be appropriate (e.g. the Longhurst bio-ecological provinces in the Pacific[1]). The spatial and seasonal distribution of the various fishing fleets is also a consideration – regions should, as far as possible, group similar types of fishing units. The configuration of regions will also be influenced by the information available on the dynamics of the stocks – there is little point in defining a region for which there is little data to provide information on the dynamics of that sub-population. Finally, socio-political factors might also be important in defining regions. For example, some consideration of management jurisdictions, such as the regions of competence of fisheries management organizations or EEZs in the case of national management, may be important.

• Population — The fish population in the model is specified at any time step as a matrix of fish abundance by age class and by region. Age classes span a range of ages from age of recruitment to a final age class which includes fish of that age and older. The abundances specified by the population matrix are the basic

[1]Longhurst, A., Sathyendranath, S., Platt, T., and Caverhill, C. 1995. An estimate of global primary production in the ocean from satellite radiometer data. *J. Plankton Res.,* 17:1245–1271.

state variables in the population dynamics of the model which simulates changes in the abundances starting from an initial state over a specified number of time steps according to processes of recruitment, growth, movement, and mortality.

• Fishery — Fishing activity in the model is grouped into a number of fisheries each with a characteristic catchability and selectivity pattern. Fisheries specified in the model operate only in one region. Different fisheries can be designated as sharing common catchability and selectivity patterns, and such groupings can span more than one region. Thus for example longline fleets operating in different regions are considered separate fisheries, but they can be designated as having the same catchability and/or selectivity characteristics. Basic input data for the model are catch, effort, and size sample counts for each fishery and time period during which that fishery operates.

• Fishing incident — A fishing incident is the occurrence of a particular fishery in a fishing period. A historical fishery consists of a time series of fishing incidents which correspond to entries in the input file known as the "*.frq*" file (see section 4.1.1).

• Fishing period — A fishing period is a time (month and week of month for length frequency purposes) during which at least one fishing incident occurs.

• Fleet — Synonymous with "fishery". It leads to a less confusing mnemonic for data file headings ("flt") than does "fishery".

• Time period—A time period is the time between recruitment events. It determines the age difference between successive age cohorts and the time units ascribed to parameters such as natural mortality and fishing mortality.

• Estimation — An estimation is a run of MULTIFAN-CL which, if successful, produces estimates of parameters that optimize the fit between predictions of the underlying population dynamic model and the observed data.

• Simulation — A simulation is a run of the underlying population dynamic model over a number of time periods with a given set of parameter values to predict a time series of catch and abundance among other things. Many simulations may occur in the course of an estimation.

2.3 What Information is used?

• Total catches for each fishing incident may be specified in weight or numbers of fish in the *.frq* file. Currently, a choice must be made for each fishery regarding the use of numbers or weight – the two units cannot be mixed within a fishery. The specification of numbers or weight for each fishery is done using the first row of fishery data flags in the *.frq* file (see "Fishery data flags" - page 26). Some amount of missing catch information is tolerated as long as the effort for such fishing incidents is known. Of course, it is not possible to have an entire fishery where catch is unknown.

• Effort data are specified in the *.frq* file in arbitrary units that are consistent within each fishery. Internally, MULTIFAN-CL re-scales the effort for each fishery such that its average is 1. The exception to this is where fisheries in different regions are assumed to have the same catchability, in which case effort is re-scaled over all such grouped fisheries so that their relative effort is preserved (see 3.4.1). Missing effort data for individual fishing incidents may be specified as long as the catch is known. It occasionally happens that there are no effort data at all for some fisheries. In such cases, experience has shown that it is better to use a proxy for fishing effort (such as the number of vessels, or even the catch) rather than simply declare all of the effort data to be missing. The user then has the flexibility to account for the credibility of the effort measure through the specification of the prior for the probability distribution of the effort deviations (see 3.4.3).

• Length and/or weight frequency sample data are provided in the *.frq* file in a specified structure (see "Structure of ..." - page 27). As with the catch and effort data, a convenient method is available for specifying

missing data. It is important to note that the size frequency data should be, or be scaled to, the effective sample size of the fish measured or weighed for each fishing incident. This is because sample size information is used in the computation of the likelihood contributions of the length and weight frequency data (see "Size sample data" - page 94).

• Tagging data if available, may be incorporated into the analysis and are provided in the *.tag* file (see section 4.1.4). Tagging data must be stratified by region, time and size at release, and by time and fishery of recapture. For tagging data to usefully contribute to the analysis, it is necessary to make various assumptions about such processes as the mixing of tagged fish in the untagged population and the reporting of tag recaptures (see 3.7 for details). The tagging data provide information to the mode because we assume that the tagged and untagged populations share the same parameters (mortality, growth, movement, etc).

• Age-length data may be incorporated and are provided in the *.age_length* file (see section 4.1.9). These data must be stratified in respect of the fishery realization from which the sample was collected, i.e. fishery, year, and month, and the dimensions of the age-length matrix for each sample must correspond to those specified for the population in the *.ini* and *.frq* files.

• Specific information on the values of particular population parameters is provided to the model by way of bounds or priors on the estimates. Bounds simply constrain the model to search the parameter space within defined ranges for specific parameters. Bounds may be set by the user for some parameters, e.g. for growth parameters specified in the *.ini* file. Other parameters are bounded internally (i.e., hard-wired), although such bounds are generally wide enough to encompass reasonable parameter estimates[2]. Priors are used to constrain other parameter estimates where there is information (not otherwise available to the model) on what the "best values" of the parameters might be. Priors are specified in terms of a mean and a penalty on the likelihood for deviation from the mean[3]. The penalty is effectively the inverse of twice the variance of the prior distribution, which is assumed to be normal (although many parameters are specified in the model as log transforms).

• Other information could potentially be included in the model. Data such as age samples from otolith ring counts, length-increment data from tag returns, and indices of relative abundance from research surveys could all be included in the model with relatively minor modification of the source code. Likewise, it is easy to add prior information on parameters of interest by specifying additional prior distributions. Users are encouraged to seek advice from the authors regarding such model development.

[2]Most of the hard-wired bounds are set in the source code files newmau5a.cpp and newmaux5.cpp using the set_value function.

[3]Most priors are specified in the source code files callpen.cpp and alldevpn.cpp. User control over priors is available through setting various flags in the *.par* file.

Chapter 3

Description of Model Processes

This chapter associates the biological and fishery related structure of the model with the formal parameters that can be either actively estimated or fixed during a phase of a MULTIFAN-CL fit as well as other structural parameters that are always fixed. The way various flags specify the model structure is discussed here. For more mathematical treatment of the model, see the Technical Annex (Appendix A).

3.1 Initial Population

MULTIFAN-CL simulates population trends forward in time starting from an initial state specified by the abundances of fish by region and age class. Initial abundance of age class 1 is actually recruitment and is dealt with separately (see 3.2). This initial state is determined in one of three ways depending on the setting of age flag 94 (see 4.5.11).

1. age flag 94 = 0 — initial abundances for age classes > 1 are estimated parameters. This makes for an (A - 1) $\times$ R matrix of parameters, where A is the number of age classes and R is the number of regions. The age class 1 abundances are also estimated but these are recruitment parameters and are dealt with differently (see 3.2).

2. age flag 94 = 1 — initial abundances of age classes > 1 by region are exponentially distributed from age class 1 abundances (recruitment by region) with age varying rate parameter, -χMa, where Ma is the natural mortality by age and χ is a multiplier set by age flag 95 (see 4.5.11).

This eliminates estimation of initial abundance parameters because they are calculated based on recruitment and natural mortality parameters. However, the assumption is made that there is an equilibrium age distribution at the start under a prior regime of no fishing.

3. age flag 94 = 2 — initial abundances of age classes > 1 by region are exponentially distributed from age class 1 abundances (recruitment by region) with age varying rate parameter, -$barZa,r$ = -(Ma + barFa,r), the negative of average total mortality by age and region, where the average is over a number of early time periods given by age flag 95. The fishing mortality values going into the average are those from the previous function iteration in the fitting procedure, and in the initial iteration these values are assumed to be zero.

This eliminates estimation of initial abundance parameters because they are calculated based on recruitment and total mortality parameters. However, the assumption is made that there is an equilibrium age distribution at the start under a prior regime of nearly constant fishing mortality.

3.2 Recruitment

An important structural setting is the number of recruitment events per year which is set in the header of the *.frq* file. It is recorded in age flag 57, and is not normally re-set during a fit. There are two options for estimation of recruitment.

3.2.1 Average recruitment and deviations

In this mode, recruitment is estimated firstly by a parameter for total recruitment over all regions and times. Time-series variation in total (i.e., summed over regions) recruitment is estimated when age flag 30 = 1. This is set automatically by MULTIFAN-CL during the initial fit phase. In essence, deviations from the average recruitment (recruitment devs) are estimated for each recruitment event. The variability of the recruitment devs is controlled by setting parest flag 149. A common setting is parest flag 149 = 14, which sets the SD for the lognormal prior distribution to about 0.6 (the mean is of course 0).

It is common for the estimated recruitment deviations in the terminal time periods to be poorly estimated due to the low number of observations for the most recent cohorts. Excluding the estimation of the terminal temporal recruitment deviates in the most recent model time periods is therefore advantageous in reducing error in the current or latest model quantities of interest to managers, e.g. absolute biomass estimates. By setting parest flag 400 to the number of model time periods from the last period that are excluded from the estimation of temporal recruitment deviates causes the deviates for these terminal time periods to be: excluded from the calculation of mean recruitment; excluded from the vector of estimated parameters; and, set to zero. Having fixed deviates of zero (default setting for parest flag 398 = 0) results in the absolute recruitments for the terminal time periods to be fixed at the average of the log(recruitments). An alternative option (parest flag 398 = 1) results in the absolute recruitments for the terminal time periods being fixed at the arithmetic mean of the estimated recruitments.

By setting region flag 1 = 1 for any region, or for all regions, the proportion of total recruitment occurring in the region or regions can be estimated [B.4]. If region flag 1 is left unset for all regions, then the recruitment distribution as given in the *.ini* file is maintained.

Estimating the distribution of recruitment among regions as described above provides the model with greater flexibility to fit the data. However, the time-series variation in recruitment is still the same in all regions. The recruitment distribution by region can be made to vary in time by setting age flags 70 and 71 to 1. This essentially allows recruitment by time period in each region to be estimated somewhat independently. The amount of variation in the recruitment distribution is controlled by a penalty which may be set via age flag 110. A constraint is activated by setting age flag 178 = 1 which ensures the sum-product of the regional recruitment proportions and the regional recruitment deviates is equal 1 in each model time period. This must be applied if the deviates are to be used for generating stochastic recruitments in projections by applying the deviates to predictions of the stock-recruitment relationship (see 4.4).

3.2.2 Recruitment by orthogonal polynomials

Temporal-spatial recruitments

Concern is warranted for the tendency for some complex fish models to estimate local minima, which is sometimes evident in alternative minimization methods following different paths, and hence finding different maximum likelihood estimates (MLE). Ideally, the MLE should be approximately the same irrespective of the minimizer used. Local minima may be an outcome of a model that is "over-parameterised" or overly complex, such that the minimisation becomes unstable. For models having complex temporal-spatial stratification it is not unusual for a large number of independent recruitments parameters to be estimated. A method

is available in MULTIFAN-CL that may make complex models more robust by reducing the number of recruitment parameters by using orthogonal polynomials for estimating the recruitments in each region and time period. It uses a linear hierarchical model for the year, season, region, and season-region interaction levels for calculating the Gram-Schmidt orthogonalisation basis matrix. Polynomials having a user-specified number of degrees are estimated for each of the possible four levels. This capability has been extended for also the multi-species/sexes cases. In the multi-sex case the orthogonal polynomial parameters are shared among the sexes, which invokes a 50:50% sex ratio in the recruitments implicitly.

Note that this method is an alternative to that of average recruitments with deviations [3.2.1] and shares none of the flag settings with that approach.

The method is invoked by setting parest_flags(155) to > 0, which also specifies the number of polynomial degrees for the year level. The degrees for each level can be specified individually using parest_flags(216-218, 221), however, it is recommended that the same degree is applied to each level, and is taken from the value of parest_flags(155) by default (i.e. when parest_flags(216-218, 221) are all zero). Turning off the levels of: season, region, and season-region interaction is done by setting parest_flags(216-218) to -1. This may be useful for examining the effects of successively adding these levels. The parest_flags numbers applying to this method are the same for the multi-species/sexes cases and are used when setting the multi-species parest_flags. For the relevant flags see [4.5.12].

Flexibility exists for increasing the polynomial degrees from that of an existing solution, which may be a useful approach to sequentially increase the polynomial complexity in successive phases of a doitall fit. This is achieved by increasing the value in parest_flags(155) with each phase.

The time periods for which the polynomials are estimated may be defined for each level being estimated, with the default being the entire model estimation period. Also, the upper and lower bounds for the degrees may be specified for each of the levels.

Besides the parest_flags, the other essential flags that specify this method are the season_region_flags which are input from the *.frq file*. These user-defined flags specify the seasons and regions within which recruitment is assumed to occur, thus allowing control over the temporal and spatial distribution of recruitment. See the description of these flags in [4.1.1].

Recruitment by environmental driver

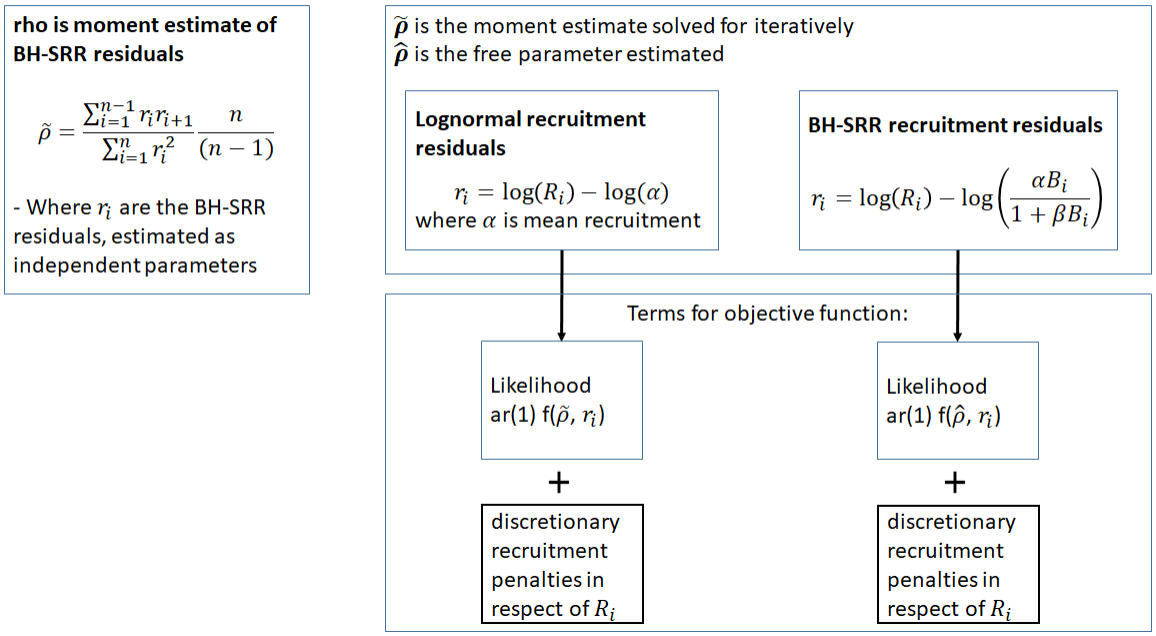
Recruitment can also be made sensitive to an environmental driving variable by setting ageflags 101 and 102, and setting age flag 72 to a value that regulates the degree of correlation between recruitment and the driving variable. An *.env* file must be present to supply the environmental data (see section 4.1.7).

See section 4.5.11 for details on use of flags mentioned above.
This feature is currently in development.

3.2.3 Autocorrelation in recruitment

An autocorrelation coefficient of the recruitment parameters may be estimated based upon either the deviates of the BH-SRR or the independent log-normal recruitments. By default, the analytical moment estimate is calculated for the autocorrelation of the recruitment deviates from the estimated Beverton-Holt stock recruitment relationship (BH-SRR). This is reported along with the results for the BH-SRR fit in the file *plot.rep*. If the autocorrelation is to be estimated, either of two types of recruitment residuals can be used: BH-SRR deviates, or independent log-normal recruitments, as shown in the diagram below.

Schema of approaches for calculating and estimating autocorrelation in recruitments



1. Autocorrelated deviates of the BH-SRR.

   The residuals used for calculating the moment estimate of the autocorrelation are the deviates of the estimated absolute recruitments from the BH-SRR predictions, r_i. These residuals are also used for calculating the AR(1) penalty term when estimating an autocorrelation as a free parameter ρ where the AR(1) penalty term replaces the log-normal error assumption of the BH-SRR fit. The weight of this penalty is specified using the age_flags(145) that both activates the fitting of the BH-SRR ([4.5.13]) and defines its relative weight in the integrated model fit. Activating the feature for the estimated autocorrelation is achieved using age_flags(135) where the lognormal error assumption is replaced with the matrix solution for the autocorrelation penalty function. Turning on the estimation of the parameter is achieved using age_flags(136). The estimated parameter ρ is reported to the *plot.rep* file in the yield analysis section along with the moment estimate, e.g.

   # BH autocorrelation moment estimator 1.527e-01
   # BH autocorrelation parameter estimate 2.545e-02

   The estimated parameter ρ is stored in the second row of the species_pars parameters in the *.par* file e.g

   # species parameters
   0
   2.54533130772897e-02
   0
   0
   ...

   The advantages of this method relates to the utility of the estimated autocorrelation for predicting recruitments in simulation projections. Future recruitments are often assumed to be a function

of projected spawning biomass, therefore this quantity should be considered when estimating a recruitment autocorrelation.

2. Lognormal random recruitment deviates with autocorrelation.

   The residuals used for calculating the moment estimate are the deviates of the estimated absolute recruitments from the average recruitment, in other words, they are the independent temporal recruitment deviates. The formulation is implemented by age_flags(129). An iterative method is used for estimating the autocorrelation using the moment estimate. The analytical solution for the autocorrelation, being the moment estimate, is used to calculate a penalty within each evaluation that is added to the total objective function. With successive evaluations a solution for the coefficient obtained iteratively. The algorithm is as follows:

   (a) perform first model evaluation and calculate the residuals

   (b) calculate the moment estimate of rho

   (c) input rho into the autocorrelation penalty function

   (d) add the penalty to the total objective function of the population model fit

   (e) repeat (a) to (d) until convergence

Whether either of methods 1 and 2 are activated or not, the analytical solution for the autocorrelation moment estimate for the BH-SRR residuals is calculated by default. This is reported to the *plot.rep* output file whenever the BH-SRR is estimated for the equilibrium yield estimation. The analyst may then decide whether it be used in generating projection recruitments or if it should be estimated as a free parameter using either methods 1 or 2.

3.3 Age and Growth

MULTIFAN-CL assumes a modified von Bertalanffy growth pattern set firstly by three parameters: the average lengths of the youngest and oldest age classes plus the rate constant K. Estimation of these is activated by setting parest flags 12, 13, and 14 respectively. Size at age up to some age class, a^*, can be modified from von Bertalanffy by activating independent parameters for the average lengths of these younger age classes. The age class a^* is set by parest flag 173, making for an additional a^*-1 parameters. Estimation of these parameters is turned on by setting parest flag 184 to 1. The values of these parameters are stored in the *.par* file in the 3rd row of age pars. The penalty (for difference from 0) on these parameters is defined by one tenth of the value in parest flag 182, and defaults to 0.01 when this flag is set to 0 (see section 4.1.5).

Some amount of variance in length at age is assumed. Estimation of the standard deviation of length at age is activated by parest flag 15. This standard deviation can be made to vary exponentially with length by setting parest flag 16. That's two more parameters.

The Richards curve is also available via a fourth growth curve parameter. This parameter is activated by setting parest flag 226 to 1, and estimated by setting parest flag 227 to 1.

3.3.1 Maturity at Age

For the computation of spawning biomass, a maturity schedule is used that is a vector in respect of age class being the proportions mature. If only females are to be included in the spawning biomass, age-specific sex ratios can also be included in the maturity schedule. In this case, each element of the maturity schedule

would be the proportion of the female population in an age class that is mature multiplied by the proportion of the total population in that age class that is female. A more complete way to model reproductive output is to include spawning fraction at age and fecundity per kg at age, as well as maturity and sex ratio. The maturity schedule then represents reproductive potential at age. This calculation is usually performed in terms of the "spawning potential" at length using the length-specific vectors for:

- Maturity ogive from ovary histology
- Fecundity
- Spawning fraction
- Proportion females

The maturity at length ogive is the normalised product from these vectors that is calculated external to MULTIFAN-CL. This ogive is then converted to a vector in respect of ages for application in the age-structured model calculations, which is done using a growth function.

The conversion from the maturity at length ogive to maturity at age can be done in two ways:

- external to MULTIFAN-CL using an assumed growth curve and then maturity at age is assumed at the "fixed" values input to the model fit in the *.ini* file [4.1.3]
- internally within MULTIFAN-CL using the model's growth function (either estimated or fixed) for the mean length-at-age and standard deviation

The second approach avoids potential for bias in the calculation of spawning biomass when the growth function is estimated within the model fit, and the resulting growth curve differs from that used when applying the first approach. This is preferable in that it integrates the growth estimation and maturity at age calculation into each model evaluation. Noting that, with estimated growth varying with each model evaluation during the minimisation procedure towards a solution, the maturity at age will vary with each evaluation. Whereas the first approach maintains a constant maturity at age throughout the minimisation procedure, irrespective of varying growth estimates.

The maturity at length ogive is input from the *.ini file [4.1.3] and implementation of the second approach for the conversion to maturity at age internally is activated by age_flags(188) with two possible approaches using cubic spline functions:

- Simple spline - simply evaluates the cubic spline function at the mean length at age as calculated from the von Bertalanffy curve
- Weighted spline - evaluates the cubic spline function at the mean length at age as calculated from the von Bertalanffy curve, and assumes that the length at age is normally distributed around the mean length; a rough integration over the length distribution is performed with steps equal to 0.5*standard devation

The final maturity at age ogive obtained using the fitted growth parameters is reported in the *.par* file [4.1.5].

3.4 Catch

Catch by age, time period, and fishery is determined by fishing mortality at age, time period, and fishery applied to estimated abundance by age and region at the start of each time period. Fishing mortality is in

turn a product of 1) fishery and time specific effort, 2) a fishery specific catchability that can be allowed to vary with time, and 3) a fishery and age specific selectivity that does not vary with time[1].

3.4.1 Catchability

For each fishery, catchability is parameterized by an average over time which is enabled by fish flag 1. The average catchability can be modified by two types of variation in time. If estimation of long term trends in catchability is desired, fish flag 10 enables a series of cumulative time deviations to be added (in log form) to the average catchability[2]. The time between application of one deviation to the next is determined by fish flag 23, in which the number of months between the addition of successive catchability deviations is specified. This flag thereby determines the number of extra parameters that must be estimated. The log- transformed catchability deviations have normally distributed priors of mean 0 and SD's determined by fish flag 15. The default setting for fish flag 15 is 50, which implies a SD of 0.1.

Seasonal variation in catchability can also be accommodated either by sinusoidal or free form variation within each year. Sinusoidal variation is enabled by fish flag 27 and adds two parameters per fishery, which are stored in the 1st and 2nd rows of fishery parameters in the .par file. Freeform variation adds up to 12 independent catchability deviations (i.e., up to 12 more parameters) at even intervals within each year, and is enabled by fish flag 47, which also indicates the number of such deviations per year. If enabled, these parameters are stored in the seasonal catchability pars section of the .par file.

There are two additional ways in which variability in catchability can be introduced into the model. In the first, catchability is allowed to be a function of the level of fishing effort (with an exponent term on the fishing effort as an additional parameter). This might be appropriate in cases where it is believed that fishing vessel operators collaborate and share information on the location of fish, thus enhancing the overall performance of the fleet (parameter >1) or where crowding effects reduce fleet performance (parameter <1). This effect is activated by setting age flag 156 = 1 (to use the effort-dependent catchability hypothesis with parameter(s) stored in the 7th row of fishery parameters in the *.par* file) and by setting fish flag 51 = 1 to activate the estimation of the parameter(s).

The second hypothesis is that catchability is related to the level of relative abundance of the stock, parameterized by the addition of a multiplicative term to eqn. A.5: $B\beta$, where B is the biomass index for the region in which the fishery occurs (normalized to average 1) and β is a parameter. Such an effect is suspected (though not demonstrated for tuna as far as we know) to occur for schooling species, where the abundance of schools does not decline in proportion to the population ($\beta > 0$). This biomass-dependent effect on catchability may therefore be used to depict hyper-stability in catch rates, and is sometimes called the "Cobb-Douglas" effect. This effect is activated by setting age flag 125 = 1 (to use the biomass-dependent catchability hypothesis) with the parameter(s) stored in the 8th row of fishery parameters for each fishery (columns) in the .par file, and by setting fish flag 53 = 1 to activate the estimation of the parameter(s). Pre-defined or assumed parameter values for specific fisheries may be input (i.e. inserted directly into fish pars 8 of the .par file) and fish flags 53 set to 0, and with additional model evaluations, a new solution .par file is obtained, including the estimated catchabilities given the assumed scenario for the biomass effect. This may be useful for exploring alternative assumptions for this effect. To obtain stability in the initial population conditions, it is also necessary to estimate region-related parameters (starting at row 2 of the region parameters in the .par file, with the number of subsequent rows being equivalent to the number of years over which average F is calculated) by setting age flag 126 = 1, and setting age flag 127 to a defined penalty value, (default = 1000). These region-related parameters are used to estimate the Cobb-Douglas multipliers, in the sense that after all is calculated they are fitted to the resulting biomass effect via the penalty.

[1] In fact, there is structure in the model and in the *.par* file to allow for time variable selectivity. However, preliminary simulations suggested that such selectivity deviations could not be estimated in practice. This issue needs to be revisited in due course, but in the meantime it is not recommended that estimation of selectivity deviations be attempted.

[2] An implementation of time variable catchability using a Kalman filter is also available and is invoked using fish flag 39. This feature has not yet been fully tested, so users are advised to proceed with caution and skepticism.

It may be reasonable (and parsimonious of parameters) to assume that certain fisheries share some aspects of catchability. For such cases grouping flags are provided to indicate how the fisheries are to be grouped. Separate grouping flags are provided for overall catchability (fish flag 29), seasonality (fish flag 28), effect of effort on catchability (fish flag 52), and effect of abundance on catchability (fish flag 54). Refer to 4.5.5 for details.

A special case concerns the grouping of catchability for fisheries in different regions. For example, a common assumption in tuna assessments is that catchability is the same for longline fisheries operating in different regions. This allows the catch per unit effort for these fisheries to provide information on the relative abundance among regions. Normally, effort data are normalized (each effort observation is divided by the average effort for that fishery) during the data pre-processing phase of the analysis. However, this process effectively removes any information on relative CPUE of different fisheries. When catchability is assumed to be common among fisheries, we want to preserve such relativities, so the effort normalization is performed across all such fisheries.

3.4.2 Selectivity

Fishery-specific selectivity coefficients (ranging from 0 to 1) describe the age-specific component of fishing mortality. Estimation is activated automatically during the initial fit phase, but if desired, estimation can be deactivated by setting fish flag 48 = 0.

Although fishing mortality is always age-specific, selectivity may be defined as either age-specific or length-specific. The main difference between the approaches is in the way the model predicts the expected catch at length for comparison with the observed size frequency data.

- With age-specific selectivity, the model multiplies numbers at age by selectivity at age to predict the age distribution of the catch, and converts this into the size distribution of the catch based on the growth curve.
- With length-specific selectivity, the model converts numbers at age into numbers at size based on the growth curve, and then multiplies numbers at size by selectivity at size to predict the size distribution of the catch.

Several methods are available to parameterize age-specific selectivity. In the first method, one parameter is estimated per age class, although the values for some number of terminal age classes may be constrained to be equal, or set to zero (see below). This is the default method, which is used if fish flag 57 = 0.

The second method imposes a specific functional form on the relationship between selectivity and age class. Two such functional forms are currently available: logistic (fish flag 57 = 1) and double normal (fish flag 57 = 2). Logistic selectivity forces the relationship between selectivity and age class to follow a logistic curve, with the two parameters describing the shape of the curve stored in fish pars 9 and fish pars 10. This type of selectivity form would be appropriate where selectivity is believed to increase continuously with age class, e.g. as might be the case for longline fisheries. Double normal selectivity allows selectivity to first increase with age class up to a certain age and then to decrease, i.e., a dome-shaped selectivity. The increasing and decreasing parts of the curve are described by normal density functions that allow different shapes for each part. Double normal selectivity is described by three parameters, fish pars 9, fish pars 10 and fish pars 11.

The third and final method uses a cubic spline to describe selectivity by age class. This method is invoked by setting fish flags 57 = 3. The number of parameters used to generate the cubic splines is specified by the user by setting fish flag 61 to the desired number of parameters. This number should be greater than 1 but smaller than the number of age classes for which selectivity is allowed to differ. It defines the number of equally-spaced spline coefficients over the range 0 to 1. The estimated parameters of the cubic spline are stored in the "fishery selectivity" section of the *.par* file. The number of cubic spline parameters may be

increased by setting fish flag 62 equal to the number of additional parameters in the input *.par* file for a MULTIFAN-CL run. For example, if the current number of parameters as defined by fish flag 61 is 5, setting fish flag 62 = 2 would increase the number of parameters to 7 for that fishery. Note that these runs are not fully nested because the positions of the original 5 cubic spline coefficients on the 0–1 scale will be shifted slightly when the number of coefficients is increased to 7. This means that the new fit with 7 coefficients will not have the same initial likelihood function value as the previous fit with 5 coefficients. However, the difference will not be great and re-convergence will be rapid if the number of coefficients being added is small relative to the number used in the previous fit.

The above describes how purely age-based selectivity coefficients are estimated in MULTIFAN-CL. However, these age-based selectivity coefficients may be modified in such a way that recognizes that selectivity is fundamentally a size-based process. In particular, we would like to recognize that age classes that have large overlap in their size distributions should have similar selectivity coefficients. A method implementing such an approach (still age-based but recognizing the influence of size) is invoked by fish flag 26. Purely age-based selectivity coefficients are retained if fish flag 26 = 0. An initial parameterization penalizes the selectivity coefficients for age classes with similar mean lengths to be themselves similar, and is activated by fish flag 26 = 1. A subsequent parameterization uses the standard deviations of length at age, and this option is activated by fish flag 26 = 2. Generally, we would recommend the use of this latter parameterization.

The number of selectivity coefficients to be estimated may be up to the number of fisheries times the number of age classes. However, there are several ways in which it may be desirable to constrain either the number of selectivity parameters, or the way that they vary with respect to age class:

- Use of function forms or cubic spline parameterizations — As described above.

- Grouping over fisheries — As for catchability, selectivity coefficients may be grouped over fisheries if it is reasonable to assume that the grouped fisheries sample the age structure of the population in the same way. The grouping flag is fish flag 24. Note that fisheries so grouped should have identical specifications for fish flags 3, 16, 26, 57, 61, 62 and 75.

- Grouping terminal age classes — It might normally be expected that a certain number of terminal age classes would have the same selectivity, particularly if there is little difference in mean lengths and/or large overlap in the length distributions of those age classes. A number of terminal age classes can be forced to be the same by setting fish flag 3 to the last age class for which selectivity is allowed to vary. Such grouping of terminal age classes is compatible with all other options for parameterizing selectivity. Users must ensure that fisheries grouped for selectivity have the same specifications for fish flag 3.

- Non-decreasing selectivity with age — This assumption constrains selectivity coefficients to increase, or remain constant, with increasing age and is activated with fish flag 16 = 1. This is an assumption commonly employed for longline fisheries. This option should not be used if specific functional forms for selectivity are being employed (i.e., fish flag 57 = 1 or 2).

- Dome-shaped selectivity — For fisheries targeting small fish, it may be reasonable to assume that a certain number of terminal age classes have zero selectivity. This constraint is activated by setting fish flag 16 = 2, with fish flag 3 determining the number of non-zero age classes. This option is incompatible with logistic selectivity (fish flag 57 = 1).

- Selectivity smoothing — Age-based selectivity coefficients attract a small smoothing penalty based on the second and third differences of the selectivity coefficient vector. The penalty weight may be controlled by setting fish flag 41 (for the second difference penalty) and fish flag 42 (for the third difference penalty). Setting these flags to 0 or 1 results in the default penalties. Setting the flags to higher values results in higher penalties and smoother selectivity coefficients. Smoothing penalties are disabled for functional form and cubic spline selectivity.

- Zero selectivity for young ages — This assumption constrains selectivity coefficients to be zero for a specified number of the youngest age classes. This helps to avoid selectivity being non-zero owing to the

functional form for selectivity rather than data (often limited for the youngest age classes). The fish flag 75 activates this constraint having the value of the number of age classes starting from 1 for which selectivity is assumed to be zero. In other words the minimum age class at which selectivity is non-zero = fish flag 75 + 1. Users must ensure that fisheries grouped for selectivity have the same specifications for fish flag 75.

Time-variant selectivity may be estimated for specified fisheries where the intention is to allow fishery-specific selectivity to vary through time in a manner such as "time-blocks" where abrupt shifts in the selectivity-at-age patterns occur during the model calculation period, such as may be due to changes in fishing practice, for example a change in the target species. Each time-block is delineated by the analyst a priori based upon expert opinion as to when such changes may have occurred. Another form of time-variant selectivity entails unique forms in each season. The manner is to estimate time-variant selectivity functions for the single fishery as applied to each time-block. In addition to long-term variation in selectivity, it is also intended to allow for seasonal patterns in selectivity. Therefore this feature allows for long-term and seasonal variation in selectivity, which includes an interaction such that each time-block has unique seasonal selectivities.

Although the assumed functional form for the fishery's selectivity is constant over the model calculation period, the parameters alter among the time-blocks and seasons that change the shape of the form in each. The goal of this approach is to allow for seasonal coefficients that would be constant among years comprising a specified time-block; where this block may be all or part of the fishery's history. This would allow for three possible assumptions for time-variant selectivity in a fishery:

a) Multiple time-blocks,

b) Single time-block with seasonal coefficients, and

c) Multiple time-blocks with unique seasonal coefficients in each time block (time-block + season interaction).

The number of time-blocks in each fisheries is specified by fish flag 71 being the total number of breaks over all fisheries, and the years in which the breaks occur is provided in the input file *selblocks.dat* (see section 4.1.8). Therefore, a single time-block is specified by fish flag 71 = 0, being the default value.

A check is made in respect of the grouping of fisheries for the selectivities as specified by the analyst in fish flag 24 to ensure that all those fisheries with shared selectivities for which time-variant selectivities are required are represented in the inputs.

The implementation of seasonality in a fishery's selectivity is controlled by fish flag 74. By default, if this flag has a zero value it is set to 1 immediately following initialisation or following assignment from the *.par*. Alternatively a number greater than one specifies the number of seasonal selectivity patterns within a year, and by default this is four.

When using length-specific selectivity, MFCL calculates the expected proportions observed at length by assuming that selectivity is a length-specific process, rather than age-specific. It may be implemented for a fishery in MFCL by setting fish flag 26 to 3 and fish flag 57 to 3.

The following functional forms are available for length-specific selectivity. For logistic selectivity set fish flag 61 = -1. The parameters are held in fish pars 9 and 10. For uni-modal selectivity, set fish flag 61 to -2. The parameters are held in fish pars 9, 10, and 11.For cubic spline parameterization, set fish flag 61 to the required degree of the cubic spline.

Multi-sex length-based selectivity

For the case of multiple sexes, the same "length-based" selectivity may be assigned for both males and females. For instances of sexual dimorphism, assuming the same age-specific selectivity or estimating different

length-specific selectivity may not be desirable. Shared selectivity at length is achieved by using one set of splines or logistic functions for both sexes, but to evaluate them at the different relative mean lengths at age, thus producing different age based selectivities among sexes. This method maps the mean lengths for both sexes into the same interval [0,1], and selects the node values in [0,1] specific to each sex for the cubic splines or the logistic functions. The approach is not a strictly "length-based" selectivity, but rather uses the same spline or logistic selectivity function as shared among the sexes, but because it is mapped on the sex-specific mean lengths-at-age for both sexes, the selectivity-at-age is different.

The feature is conditional upon the data being multi-sex, and setting age_flags(193) equal to 1 activate it, with fish_flags(i,26)=2 to enable the estimation of age-specific selectivities with the standard deviation of length-at-age accounted for. It is applicable to either the spline or logistic functional forms (fish_flags(i,57) = 1 or 3). It can accommodate the conditions for asymptotic selectivity for old ages, specified by fish_flags(i,3), and also the fixing of selectivity to a zero value for defined young ages, specified by fish_flags(i,75); and since these settings are age-specific, they can be defined differently among the sexes. It may be preferable to apply the generic selectivity penalty specific to the fisheries that employ this feature using fish_flags(i,72), particularly to stabilise the parameters during the early phases of the minimisation.

3.4.3 Fishing effort

Effort "data" are not data in the sense of the catch and size data, which are predicted by the model and fit to observations using likelihood functions. Effort data are used in the model as independent, or forcing variables. However, the observed effort is not necessarily assumed to be the true or effective effort — each effort observation is associated with an estimated effort deviation parameter (or effort dev) which is the difference between the observed effort and the model's estimate of the effective effort[3]. Unlike the catchability deviations, effort devs are transient (or non-cumulative) in nature and may be thought of as noise in the relationship between fishing effort and fishing mortality (see eqn. A.5).

Effort devs are assumed to be normally distributed with mean zero and variance specified by the user in the form of a penalty on the objective function. There are two way of specifying these penalties. The standard method specifies a penalty for each fishery by way of fish flag 13. In this standard method, the default value of the penalty is 10, which is equivalent to SD of approximately 0.22. Lower penalties allow the effort devs to be estimated with more freedom and imply more noise in the fishing effort – fishing mortality relationship. For example, setting fish flag 13 = 1 is equivalent to SD of approximately 0.7 and would imply that there is minimal information in the effort data concerning fishing mortality. Conversely, setting higher penalties implies less noise and more information in the effort data concerning fishing mortality. There are several considerations that influence the choice of effort dev penalties, including:

- The nature of the fishing gear and the way in which it samples the population —
e.g., a gear that targets fish schools, such as purse seine, might be expected to have a noisier relationship between effort and fishing mortality and should therefore attract smaller effort dev penalties than a more widespread, diffuse gear such as longline;

- The geographical distribution of fishing effort in relation to the region in which the fishery occurs — a widely distributed fishery would be expected to sample the population in that region with less noise than a very geographically restricted fishery, in which variation in the sub- regional distribution of the fish might create additional noise in the effort - fishing mortality relationship.

A useful variation on the effort dev penalties is to scale the penalties according to the square root of the observed effort. The idea behind this feature is that we want to have relatively small penalties for very low

[3]There is a version of the model, invoked by setting age flag 92 = 1, in which the effort devs are solved directly in the catch equations using the Newton-Raphson technique and assuming that the observation errors in the total catches are zero. This has the advantage of eliminating a large number of parameters from the estimation. This formulation of the model has not yet been extensively tested, and users are advised to proceed with appropriate caution and skepticism.

observed effort and relatively high penalties for high observed effort. This is achieved by setting fish flag 13 to the negative value of the desired penalty. For example, setting fish flag 13 = -10 would invoke a penalty of 10 (SD = 0.22) at the average level of observed effort for that fishery, lower penalties (higher SD) for below average effort and higher penalties (lower SD) for above average effort. See 4.5.7 for more options regarding effort dev estimation.

An alternative method, first implemented in version 6 of the frq file, allows specification of time-specific and non-integer penalties. The frq file is expanded by inserting an additional time-dependent penalty column (column 7) after the effort column (moving the first length frequency column to column 8). If fish flag 66 is set to 1, values in this time-dependent penalty column are used to scale the precision of the CPUE estimates. Time dependent CV estimates are obtained by multiplying the time-dependent penalty column by fish flag 13, and then transforming in the usual way (CV = 1/sqrt(2*penalty)). Thus a higher value in column 7 will result in a lower CV.

When time dependent penalties are used for a fishery, all values in column 7 for that fishery must be greater than 0. Time dependent penalties can be turned off for a fishery by setting all values for that fishery to -1.

Time-dependent penalties over-ride penalty scaling by the square root of the observed effort.

An enhancement is possible in respect of a sex-disaggregated model. For instances where no sex-disaggregated data are available for fitting a multi-sex model, the model is not informed in respect of the region-specific sex-ratios. In this case, it may be desirable to constrain the sex-specific effort deviates to be similar or shared among the sexes. This may avoid implausible model estimates by preventing the model from compensating for the shared average catchability via having strongly positive or negative effort deviate coefficients (effort_dev_coffs). A special case was made to allow for effort_dev_coffs grouping flags such that the respective fisheries corresponding to the two sexes could be grouped so as to share the same effort_dev_coffs parameters. The fishery grouping is specified using fish_flags(79).

It may be necessary to control the maximum fishing mortality in the catch equation (rmax) to either facilitate very high observed catches to be achieved in certain model projection scenarios, or to limit mortality to plausible levels for small or endangered populations. The control is achieved by setting the value of age_flag(116), where maximum mortality =age_flag(116)/100. Note however this constraint will also apply in the model estimation period that may cause computational problems during model fitting. It is recommended to use a maximum mortality of 0.7 or 1.1 for model estimation, but can be set to 0 for deterministic projections (in which case the default maximum mortality of 5.0 will apply).

3.5 Natural Mortality

The instantaneous rate of natural mortality (M) consists of an average over age classes and age-specific deviations from the average. Estimation of average M is activated by setting age flag 33 = 1. If average M is not estimated, the value specified in the *.ini* file will be maintained in the model. Many data sets are not very informative regarding M, therefore it is often desirable to constrain the estimation of average M according to some prior knowledge of its true value. This is accomplished by setting age flag 82 to the prior mean ($\times$ 100) and age flag 84 to the penalty weight for deviation from the mean. Age flag 84 determines $1/(2\sigma 2)$ for the log-normally distributed prior on average M (see 5.4.1). The minimum and maximum ages to be included in the average may be specified by setting age flag 83 and age flag 84, respectively — if unspecified, the average is calculated over all age classes.

Another way of constraining the estimation of average M takes advantage of a well-known life history invariant, the ratio of natural mortality to growth rate. By theory this ratio is 1.5, and for a wide variety of fishes has been shown empirically to be close to 1.6 with a standard error of .58[4]. A log-normal prior can be set on

[4]Jensen, A.L. 1996. Beverton and Holt life history invariants result from optimal trade-off of reproduction and survival. *Can. J. Fish. Aquat. Sci.,* 53:820–822.

M/K with age flag 130 giving the target ratio ($\times$ 100) and the penalty given by age flag 133 determining $1/(2\sigma 2)$ (see 5.4.1). The range of ages included in the average M defaults to the full range of age classes, and a narrower age can be specified by age flags 131 and 132.

Estimation of the age-specific deviations from average M is activated by setting age flag 73 = 1. It is not advisable to attempt to estimate the deviations in a completely unconstrained fashion; therefore, a range of options for constraining the estimation through various penalties is provided. See 4.5.16 for details. The deviations are estimated in log form and are stored in the *.par* file in the 2nd row of age_pars (see section 4.1.5). This is the default option activated when age_flags(109)=0.

An alternative method for estimating age-specific natural mortality is to the age-specific mortality indices directly as free parameters. This is activated when age_flags(109)=1. The age-specific parameters are stored in the *.par* file in the 5$^{\text{th}}$ row of age_pars (see section 4.1.5).

Two methods are possible for estimating functional forms for age-specific natural mortality: splines and Lorenzen[12], activated by age_flags(109) set to 2 and 3, respectively. For the splines option, parest_flags(121) specifies the number of nodes, while for the Lorenzen[5] option parest_flags(121) is set to 2 for estimating the two parameters. When estimating functional forms, it is recommended to disable the estimation of the other options by setting: age_flags(73) = 0, age_flags(77, 78, 79, 80) = 0, and also age_flags(33) = 0 to disable the estimation of average natural mortality, which no longer needed when estimating the age-specific formulation that is independent of any mean value. The estimated parameters for the functions are stored in age_pars(5).

3.6 Movement

In any analysis in which the number of regions is more than 1, movement coefficients will need to be specified and/or estimated. Initially, two coefficients for each non-zero element of the movement matrix (see "Movement matrix" - page 26) must be specified in the *.ini* file for each movement period (see "Movement" - page 31). Estimation of movement coefficients is activated by setting age flag 68 = 1. The number of periods for which the coefficients are estimated is controlled via the movement map specified in the *.ini* file and subsequently written into the .par file.

Movement parameters may be made age specific in a simple linear fashion by first setting age flag 89 = 1, which creates the additional parameters (one per coefficient) and a location in the *.par* file for their values to be stored. Setting age flag 88 = 1 then activates the estimation of these parameters. A logistic-type nonlinearity in the relationship between movement and age class may be similarly added by setting age flag 91 = 1 (to create the parameters) and age flag 90 = 1 (to estimate them).

3.7 Tag-Specific Processes

In the following sections, issues specific to the modeling of tagging data are discussed.

3.7.1 Tag fishery grouping

Ideally, the fisheries that are defined in the model should be identifiable also as the fisheries recapturing tags. However, in some instances the fisheries identified in the tag data may consist of groups of the fisheries defined in the *.frq* file. This is the case for tagging data used in tuna assessments in the western and central

[5]Lorenzen, K. 1996. The relationship between body weight and natural mortality in juvenile and adult fish: a comparison of natural ecosystems and aquaculture. *J.Fish.Biol.49*:627-647

Pacific. Here, purse seine fisheries are defined by setting method, typically log sets, fish aggregation device (FAD) sets and unassociated school sets. However, the type of set was rarely reported for tag recaptures by purse seine vessels. We therefore aggregate the model predictions of tag recaptures across these fisheries so that predictions and observations are comparable in the tag likelihood function. The grouping of fisheries for this purpose is defined by fish flag 32.

3.7.2 Tag pooling

MULTIFAN-CL models the tagging data by projecting the numbers of tagged fish in each tag release group over time from the time of release to the end of the time period used in the analysis. If the number of tag release groups is large, the computational load will increase greatly and the speed of execution may become unacceptably slow. A reasonable compromise is to maintain the tagged population dynamics separately for each tag group over some period of time, after which the tagged fish enter a "pooled" population and therefore lose their tag group identity. By default, such pooling occurs as tagged fish enter the last age class (plus group), but earlier pooling may be specified by setting age flag 96 to the number of time periods after release that pooling is to occur. If you attempt to pool tagged fish after their entry into the last age class, an admonishment from MULTIFAN-CL will result.

3.7.3 Tag mixing

In general, the population dynamics of the tagged and untagged populations are governed by the same model structures and parameters. An obvious exception to this is recruitment, which for the tagged population is simply the release of tagged fish. Implicitly, we assume that the probability of recapturing a given tagged fish is the same as the probability of catching any given untagged fish in the same region. For this assumption to be valid, either the distribution of fishing effort must be random with respect to tagged and untagged fish and/or the tagged fish must be randomly mixed with the untagged fish. The former condition is unlikely to be met because fishing effort is almost never randomly distributed in space. The second condition is also unlikely to be met soon after release because of insufficient time for mixing to take place. Depending on the disposition of fishing effort in relation to tag release sites, the probability of capture of tagged fish soon after release may be different to that for the untagged fish. It is therefore desirable to designate one or more time periods after release as "pre-mixed" and compute fishing mortality for the tagged fish based on the actual recaptures, corrected for tag reporting (see below), rather than use fishing mortalities based on the general population parameters. This in effect desensitizes the likelihood function to tag recaptures in the pre-mixed periods while correctly discounting the tagged population for the recaptures that occurred. The number of fishing periods designated as pre-mixed may be specified for each tag release group via the first tag flag in the *.par* file.

3.7.4 Tag reporting

For most tagging programs, not all tag recaptures are observed because some are not reported to the tagging authorities (the curse of all tagging programs). Therefore, MULTIFAN-CL estimates fishery-specific tag reporting rates. Estimation is activated by fish flag 33 = 1, with the parameters stored in the 3rd row of fishery parameters in the *.par* file. The parameters may take values in the range 0–1; however a global upper bound less than 1 may be specified (multiplied by 100) as parest flag 33. Tag reporting parameters may be grouped over fisheries using fish flag 34, which should provide coarser grouping than that for the tag recapture data (fish flag 32) if that grouping is employed.

If there is independent information on tag reporting rates, for example from tag seeding experiments, reporting rate priors may be specified. The prior mean is specified (multiplied by 100) as fish flag 36, with the penalty

weight being fish flag 35. The usual relationship between the penalty weight and prior variance is in effect ($var = 1/2p$).

Random walk changes in tag reporting rates over time may be activated by fish flag 37 = 1, with the frequency of changes set by fish flag 45 (in a similar fashion to catchability deviations). This feature is currently experimental and should be treated with caution.

Reporting rates by release group and fishery

Individual tag releases are generally grouped into tag release groups, which represent the tags released by quarter x region x tagging program stratum. Tag recapture reporting rates are typically assumed to be associated with the fishery group recapturing the tags, but independent of the tag release group. However, this assumption is not necessary in MULTIFAN-CL. Reporting rate can vary among tag release groups. This feature, added in 2010, is designed to accommodate reporting rate variation through time, or between releases by different tagging programs.

The probability of a recapture being reported may be specific to each tag release group and the factors surrounding it, such as the physical characteristics of the tag employed (colour, printed information), the perceived value of the incentive (reward), and the extent of publicity associated with the tag release group. It is, therefore, advantageous to the analysis of tag-recapture data to estimate the probability of a recaptured tagged fish being reported, specific to each tag release group. This parameter, termed the tag release group-specific reporting rate, can be estimated for each release group and for each fishery group, i.e., all possible combinations of release groups and fishery groups. Since these combinations rapidly expand the number of parameters (e.g. 5 release groups x 20 fisheries produces 100 tag release group-specific reporting rates), the facility exists to share these parameters among similar release groups and fisheries.

The MULTIFAN-CL formulation for calculating expected recaptures remains the same except that recapture reporting rates now include the added subscripted dimension of tag release group, i.e. a matrix dimensioned by fishery and release groups. The *.par file now includes the following additional parameters:

tag flags

tag fish rep - initial values of reporting rates by fishery and release group

tag fish rep group flags - grouping of reporting rates by fishery and release group

tag_fish_rep active flags - switch for turning on estimation, all matrix is activated=1

tag_fish_rep target - prior mean of reporting rate, usually set to be the same as init value

tag_fish_rep penalty - penalty on deviations from the prior mean

The tag_fish_rep target and tag_fish_rep penalty parameters specify the mean and variance of the priors associated with the tag release group-specific reporting rates being estimated. Note that the # tag flags parameters have nrows being the number of tag release groups (nrel); whereas the five reporting rate parameter blocks have nrows being nrel+1, and the extra row is to account for an aggregate tag release group. This aggregate tag release group is to accommodate an accumulation of the surviving tagged populations after n time periods over the recapture phase. In other words, individual tag release groups are not retained indefinitely in the model but instead after a set number of time periods over which most recaptures will have occurred or tagged fish will die, the release group survivors are removed from its original tagged population and are pooled into an aggregate tag release group. It is for this additional aggregate release group that an extra row of parameters is required in each parameter block.

Estimating tag release group-specific reporting rates may be done using two approaches.

The first applies when using the old format of the *.ini* file (version no. 1000) in which the tagging reporting rate parameters are absent. This entails three stages to the model fitting procedure. In the first stage, the program is run with the -makepar setting in the normal way to generate the .par file. In the second stage, the activation matrix, the grouping matrix and the two prior distribution matrices (target and penalty) are edited

in the par file. In the third stage, the age flag(198) is set to 1 in order to activate estimation of group-specific reporting rates, and the model evaluations are run to a desired solution.

The second applies when using the current format of the *.ini* file (version no. 1001) where the tagging reporting rate parameters are entered in sections 3 to 7 as matrices (see section 4.1.3). Age flag(198) is set to 1 in order to activate estimation of group-specific reporting rates, and the model evaluations are run to a desired solution.

Output of the fitted tag release group-specific reporting rates is reported in the output *.par* file in the section titled # tag fish rep. The *plot.rep* file output is in the usual format and the group-specific results are output in the section titled

Reporting rates by fishery by tag group (no time series variation).

3.8 Biological Reference Points

Biological Reference Points (BRPs) are commonly used in fisheries stock assessments as a means of comparing the current estimated condition of the stock with population states or levels of exploitation that are considered to be either limits or targets. In MULTIFAN-CL three different types of BRP analysis, described in the sections below, are available. In each case, the analyses are integrated into the overall model, thus allowing confidence intervals for the BRPs to be computed using the Delta method.

3.8.1 Impact analysis

A very simple BRP that we have found to be useful in the context of age structured models is the impact of fishing on the population. Impact is defined as the difference between an estimated population state at a point in time and the equivalent population state that would have occurred in the absence of one or more fisheries. A key assumption of this analysis is that the time series of recruitment that is estimated with fishing would also have occurred without fishing; in other words, fishing is assumed not to have had a significant effect on recruitment. The analysis is implemented after a converged fit is completed simply by executing an additional population simulation with the catchability coefficients of selected fisheries set to zero. This is invoked by setting fish flag 55 to 1 for the fleets one wishes to disable, and most likely setting age flag 193 to 1 as well (see 4.5.15).

The ratio of the fished population biomass to the unfished biomass is the measure of fishery impact. Impacts can be examined at the region level, for groups of regions, or for the whole model domain, and may be compared to reference levels agreed to by the management agency. For example, an impact ratio of 0.5 would be approximately equivalent to MSY under the assumptions of the Schaefer production model. The results of the impact analysis are output to a mirror of the *plot.rep* file called *plotq0.rep*.

Figure 3.1 shows examples of biomass estimates where either all fisheries are active, or just longline fisheries, or both longline and surface fisheries inactive. The surface fisheries appear to have a greater impact than the longline fisheries. The red line came from the absolute biomass section of the *plot.rep* file, and the the other lines came from the absolute biomass section in two instances of the *plotN0.rep* file formed by setting fish flag 55 = 1 for all fisheries or just for the surface fisheries.

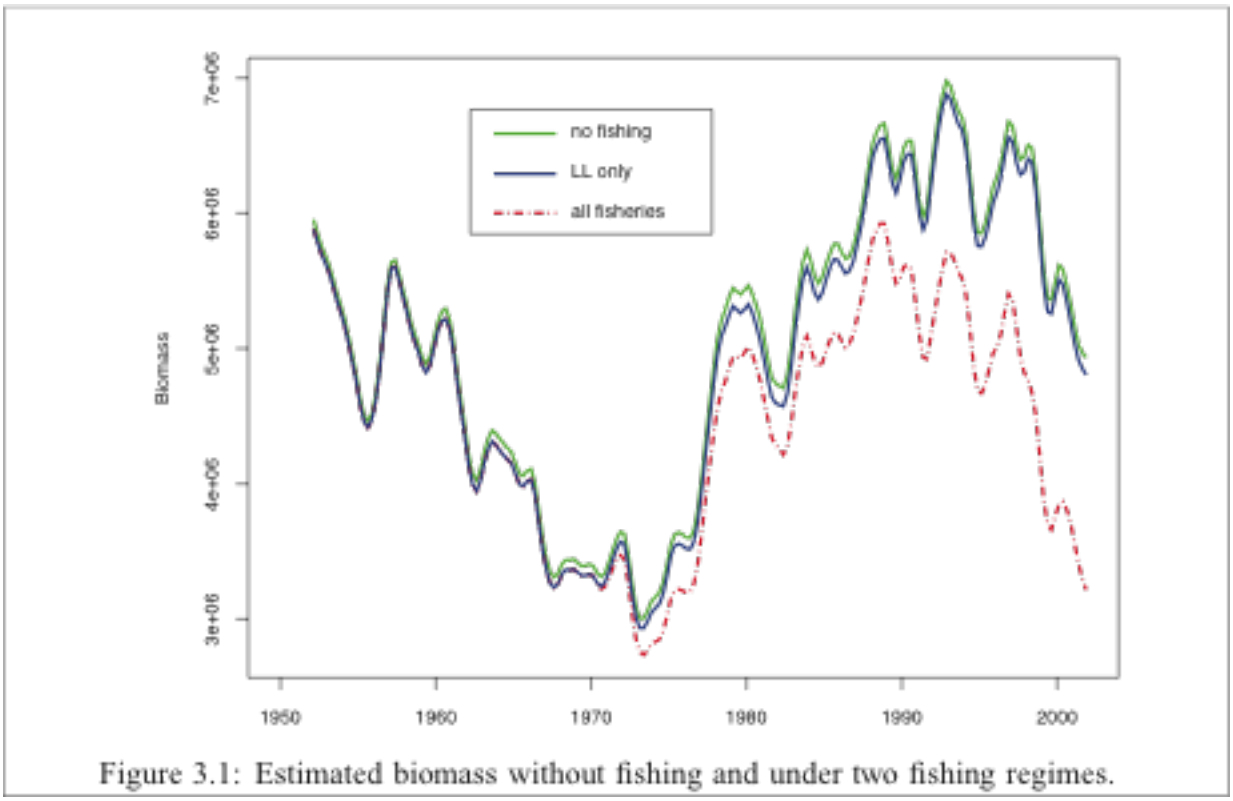


Figure 3.1: Estimated biomass without fishing and under two fishing regimes.

Setting fish flag 55 = 1 for selected fisheries also provides a means of investigating the interaction among fisheries — if fishery x is disabled, then the increase in catch of fishery y is a measure of the impact of x on y. Figure 3.2 shows the ratio of predicted longline catch with surface fisheries active to longline catch without the surface fisheries as well as the corresponding ratio of surface catch with and without the longline fisheries. The surface fisheries appear to be having a greater effect on longline fisheries than the reciprocal effect of longline on surface fisheries.

3.8.2 Yield analysis

A classical Beverton-and-Holt-type equilibrium yield analysis is carried out if a stock-recruitment relationship is estimated (age flag 145 > 0). Equilibrium yield is computed in the following fashion:

- A Beverton and Holt stock-recruitment relationship (SRR) is estimated by setting age flag 145 = x, where x is the penalty weight used to penalize recruitment deviations from the SRR. A value of x between 1 and 5 is normally sufficient to estimate the SRR without overly reducing recruitment variability, and negative values assign low penalty weight = $10^{\text{age\_flags}(145)}$. Note that parest flag 149, which specifies the penalty on deviations of recruitment from the average, should be set to zero when the SRR is being estimated. The other flags that need to be set are: age flag 146 = 1 (activate estimation of SRR parameters); age flag 147 = n, where n is the number of recruitment periods between spawning and recruitment (default = 1); age flag 149 = 0 or 1, where 0 results in "stock" and the subsequent yield computation being in terms of fish weight while 1 signifies the use of fish numbers for both SRR estimation and yield computation.

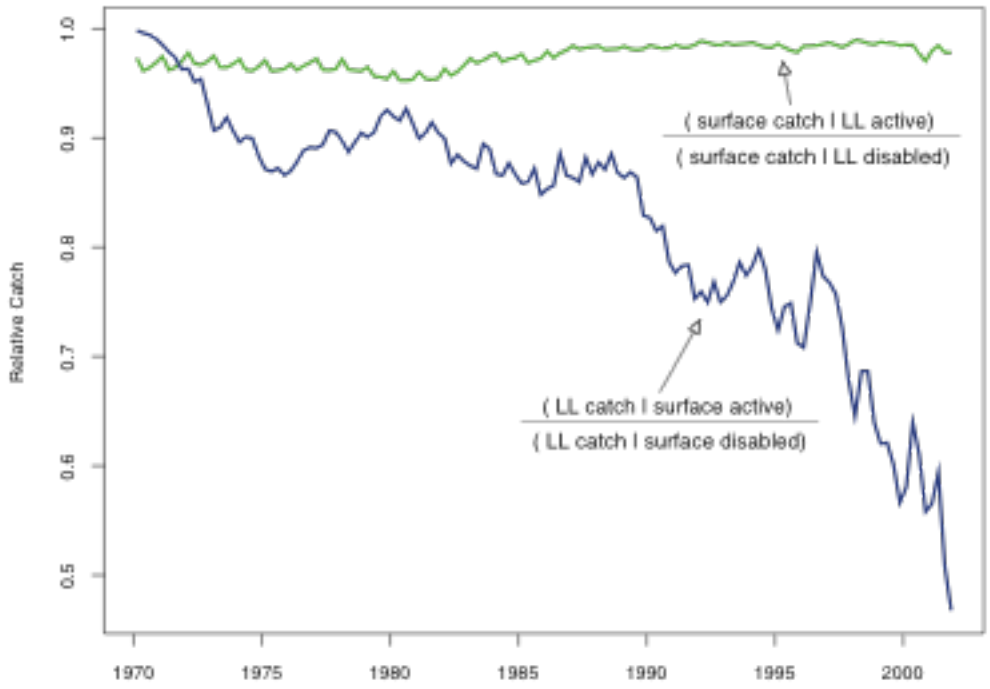


Figure 3.2: Catch in selected fisheries with other fisheries active relative to catch with other fisheries inactive.

• Fisheries data are usually very uninformative regarding the SRR. For this reason it is recommended that the SRR be constrained in a biologically meaningful way. We do this by specifying a beta-distributed prior on the SRR "steepness", which is the ratio of recruitment at 20% of unfished equilibrium biomass to recruitment at unfished equilibrium biomass. The prior is specified by the parameters A and B of the beta distribution, which are controlled by age flags 153 and 154 (see 5.4.2).

• For the sex-disaggregated model, the recruitments estimated for the female population only, are shared among both sexes. Similarly, the Beverton-Holt stock-recruitment relationship (SRR) estimated for the female population is shared among the sexes, i.e. the predicted recruitment is distributed uniformly among sexes. A yield must therefore be calculated for each population given the same equilibrium recruitments and then summated over both sexes. Which of the two sexes is the female sex is specified by the species_flags input from the *.ini* file (section 4.1.3).

• The SRR may be fitted to the total recruitments for a calendar year (annual) rather than the component recruitments (quarterly or seasonal). Many models assume a quarterly temporal stratification which frequently results in high variability in the estimated recruitments, most likely due to the seasonality in the underlying recruitment patterns. This high variability may unduly affect the fit of the SRR and it may be preferable to fit the relationship to the annual recruitments. An alternative option is to assign the average annual biomass to the x-variable and total "annualised" recruitments to the y-variable. Consequently, whereas multiple recruitments are estimated during a calendar year, the SRR parameters are estimated in respect of the average annual values. This option is conditional upon age flag 182 =1. When a subset of the model analysis period is used for fitting the SRR (age flag 199 and 200), the start and end years used are the nearest complete calendar years. Using this option means the SRR predictions used for calculating equilibrium yield, of course relate to a calendar year, as does the equilibrium yield.

• Yield estimates based upon the Beverton and Holt stock-recruitment relationship (SRR) can be calculated according to two options. The default option (age flag 199 = 0) assumes average recruitment is determined

from the SRR calculated over the full model analysis period. An alternative option (age flag 199 > 0) calculates the SRR using recruitments and biomass over a specified historical period, defined by age flags 199 and 200 (see 4.5.12). This may be a necessary option if the recruitment deviates in the terminal time periods have been fixed (parest flag 400, see 3.2.1), and consequently recruitments for these period should be excluded from the SRR fit as they are not true estimates.

• Estimated annual recruitment (deviates) are highly variable and differ considerably from the SRR predictions in each year. The SRR is fitted to these absolute recruitments according to a log-normal error distribution. Predictions from the relationship in normal space will therefore approximate the median of the estimated recruitment distribution. A bias correction may be applied to the recruitment predictions from the BH-SRR to approximate the mean of the estimated recruitments. Using the estimated error distribution of the SRR, corrected predicted recruitments ($\hat{Z}$) may be calculated:

$$\hat{Z} = exp\left[\left(\hat{Y}\right) + \frac{s^2_{log(R)}}{2}\right]$$

where $s^2_{log(R)}$ is the variance of the SRR predicted recruitments (in log-space). To activate this bias correction age_flag(161) is set to 1 (see 4.5.12).

• An age-specific vector of fishing mortality (F) is computed for the whole model domain using the method described in A.1.8. The period over which the average is computed is defined by age flags 148 and 155. Age flag 148 specifies the number of recruitment periods from the end of the analysis over which the average is to be computed, e.g., if recruitment is quarterly and we want to compute the average for the last 5 years, then set age flag 148 = 20. Age flag 155 specifies the number of recruitment periods from the end of the analysis to omit from the average. For example, in the previous example of quarterly recruitment, if we wanted to compute the average for the past 5 years but excluding the most recent year, we would set age flag 155 = 4. Omitting the most recent periods from the average may be desirable because estimates of fishing mortality at the end of the analysis are usually very uncertain. By default the effort deviations are ignored in calculating the average F vector, but they can be included by setting age flag 194 to non-zero. This can make a difference if for some fisheries there is a trend away from zero in the effort deviations.

• Equilibrium yield is computed (see A.1.11) for a range of F vectors that are obtained by multiplying the average F vector by a series of multipliers (*Fmult*) ranging from 0.1 to 50 (unless yield becomes negative at *Fmult* < 50 in which case the computation is terminated.

• The maximum yield over the range of *Fmult* is determined and is designated as the *MSY*. The equilibrium biomass (or population number) that results in the *MSY* is designated B_{MSY}. F_{MSY} is defined as MSY/B_{MSY}.

• Ratios of B_{t} $/B_{\mathrm{MSY}}$ (example in Figure 3.3) and F_{t} $/F_{\mathrm{MSY}}$ are computed for the entire time series of the analysis, as a means of comparing historical exploitation and population states with the current MSY conditions. This of course does not imply anything about what the historical MSY conditions might have been, as the age-specific pattern of fishing mortality may have varied substantially[6].

[6]In which case one could argue that such ratios for the period prior to that over which the average F vector is computed are not very meaningful! A proper historical treatment should probably compare biomass and fishing mortality with the MSY levels determined using the age-specific selectivity in operation at those times.

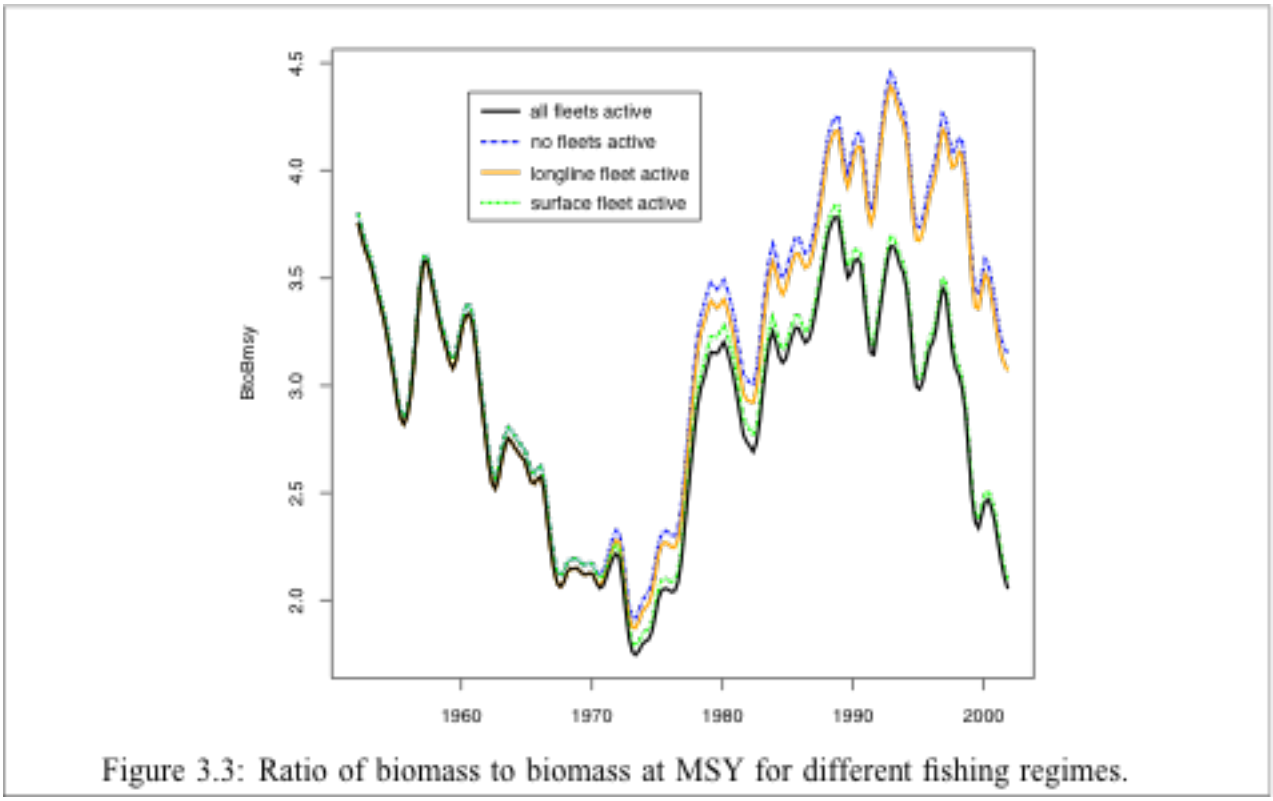


Figure 3.3: Ratio of biomass to biomass at MSY for different fishing regimes.

A simplification of the above procedure is used to also compute yield per recruit (YPR) for the same series of *Fmult* applied to the same spatially- aggregated F vector. In this case, the yield resulting from one recruit is calculated and the SRR does not come into play. A region-specific YPR analysis is also undertaken. Here, *Fmult* is applied to a 2-dimensional (age and region) F matrix and the YPR computed for each region individually.

The results of the SRR estimation, equilibrium yield analysis and spatially-aggregated YPR analysis are written into the *plot.rep* file. The results of the region-specific YPR analysis are written into a separate file called *catchequ*, but this should ultimately be put in *plot.rep* too.

3.8.3 Pella-Tomlinson biomass dynamics

An alternative to estimating a SRR and undertaking a Beverton-Holt-type equilibrium yield analysis is to introduce a likelihood component for deviations of the estimated spatially-aggregated biomass from biomass predicted by the Pella-Tomlinson production model. This allows estimation of the parameters of the model — the instantaneous rate of population growth r and the population carrying capacity K. In principle, the shape parameter m may also be estimated, but in practice most fisheries data sets are uninformative with respect to this parameter. The default value (2) forces a symmetrical production curve (Schaefer model).

The biomass dynamics penalty is invoked by setting age flag $150 > 0$, with the value determining the degree to which the model is penalized for deviation from the Pella-Tomlinson model. Estimation of r and K is activated by setting age flag 151 and age flag $152 = 1$.

We do not have much experience in using this feature of the model and therefore cannot comment on its utility. One problem that occurs with some tuna assessments is that the biomass is mainly recruitment driven and often has long term decadal scale changes that are thought to be driven by climatic regime shifts. In such cases, we would not expect the Pella-Tomlinson biomass dynamics model to work very well.

3.8.4 Target reference points and Likelihood Profiles

Target values can be set for certain primary or derived parameters by imposing priors with strong penalty weights to force MULTIFAN-CL to converge to, or close to, a given value perhaps away from the value MULTIFAN-CL would naturally converge to. A number of BRP parameters are targetable in this way as well as other parameters and derived quantities. Targeting is accomplished by the use of various flags, see 4.5.12, 4.5.13, 4.5.14 for examples.

Targeted convergence can be useful in a couple of ways. If several MULTIFAN-CL runs are forced in this way to a series of targets for a particular parameter, then a plot of the final likelihood values (with residual penalty for the target subtracted off) against the target values is an approximation of the likelihood profile for that parameter. Targets can also help check the robustness of a convergence. A variety of targets can be set to pull MULTIFAN-CL in various directions away from a convergence point. The targets can then be relaxed to see if MULTIFAN-CL re-converges at, or near, the original convergence point.

3.8.5 Likelihood profile for F to F_{MSY} ratio

An indication that a population is being over fished is a finding that F is greater than F_{MSY}. Therefore it is of interest to estimate F/F_{MSY} and to investigate the uncertainty in the estimate. A likelihood profile can be generated as stated above by setting targets for the fishing mortality multiplier at MSY, x_{MSY}, which is the reciprocal of F/F_{MSY}. Age flag 165 sets a target of the flag setting divided by 100. Age flag 166 sets the corresponding penalty. As part of this process MULTIFAN-CL calculates a weighted average of a fixed set of x values, and age flags 198 and 169 influence the accuracy of that computation. Age flag 199 defaults to 105 and must be greater that the current estimate of MSY. It is a divisor in calculating weighting factors for the weighted average (see A.1.12). Values less than MSY can generate NaNs which can lead to a crash. Values more than 10×MSY lead to some inaccuracy in determination of x_{MSY}. The current estimate of MSY is printed to standard output whenever this target process is active to allow appropriate adjustment of age flag 169. Age flag 168 defaults to a value of 50, which should be adequate for most occasions.

3.8.6 Likelihood profile for average biomass or biomass depletion

The profile likelihood may be calculated in respect of fixed values for either: an average absolute biomass, or a level of absolute biomass depletion, with each being specified either as total or adult biomass. In the case of the quantity being a level of depletion, the initial biomass is calculated as the average over the first 5 time periods, and the terminal biomass is an average over a user-defined period. Similarly, in the case of the quantity being an average biomass, the calculation period is user-defined. The periods are specified using age flag 173 and 174; with the default (0) setting being: 1 to n years in the case of average biomass; and, n years - 6 to n years in the case of absolute biomass depletion. The quantity can be either total or adult biomass as specified using age flag 172. This quantity is "fixed" by means of a penalty being added to the likelihood for which a weight may be assigned, effectively constraining the model fit to attain a "target" value for the quantity. This target is a scaled value of the quantity obtained from the maximum likelihood estimate, MLE, model value. Normally, a range of scalars is applied, e.g. 0.5 to 1.5, and the total objective function is then recorded from the converged fit at which each of the target values is achieved. The method requires a bash script to perform the algorithm external of MULTIFAN-CL as follows:

- run the MLE model using the solution .par input file and set parest flag 347 ==0 so as to produce the file: "relative_depletion" or "avg_bio" depending upon the setting for parest flag 346 - value == 1 uses biomass depletion; value == 2 uses average biomass
- read from the file the MLE value for the fixed quantity, relative depletion or average biomass

- loop over a range of scalars for the fixed quantity
- for a given value of the scalar a series of 7 separate sets of model evaluations are done, with the penalty weight specified by parest flag 348 and number of evaluations increasing gradually with each subsequent set
- after the final set of evaluations the total objective function value for that scalar value is obtained, and the "test plot output" file is renamed to "test_ plot_ output_***" where *** is the scalar value

This bash script that performs the algorithm, "scriptloop2", is provided in Appendix [C].

Chapter 4

Running MULTIFAN-CL

A MULTIFAN-CL analysis consists of three major components. The first sets the structure of the model in terms of the number and types of fisheries, some biological attributes of the fish, and the spatial organization of regions within the model universe. The second component performs a "fit" in which model parameters are estimated by adjusting them to optimize the fit of model predictions to observed data. The third component examines and interprets the results, which are essentially the estimated values of the parameters as well as other values derived there from. This chapter is concerned with the first two of these components. The third will be dealt with primarily in chapter 6, *Interpreting Results*.

To structure a MULTIFAN-CL model and fit it to data, it is necessary to construct 1) an "*.ini*" file for entry of some structural information and starting values and bounds for some parameters; 2) a "*.frq*" file for entry of more structural information plus the observed catch, effort and sample data; and 3) a "*.tag*" file if tagging data are to be included. The fitting procedure runs through a number of phases, usually with an increasing number of parameters being estimated or with relaxation of constraints on parameters on proceeding from phase to phase. MULTIFAN-CL constructs a number of output files. Among them is a "*.par*" file, which provides continuity between phases of the analysis. The *.par* file provides a complete record of all parameter estimates obtained from a particular model run. The analyst orchestrates the fitting procedure by manipulating a variety of flags either by directly editing their values in the *.par* file between phases, or by setting their values by way of command line arguments to MULTIFAN-CL. This often leads to complex and lengthy command lines which are most conveniently incorporated into batch scripts, or "doitall files" which serve the dual purpose of running a complex analysis and of documenting the details of how the analysis was conducted.

Many of the files pertinent to a given analysis are linked together by virtue of a base file name, usually indicative of the particular analysis, for example the species name. Thus we could have input files *bigeye.ini*, *bigeye.frq* and *bigeye.tag*. Some of the output file names also include the base file name, but some output files have fixed names and will overwrite corresponding files from earlier analyses launched from the same file directory. It is therefore good practice to conduct separate analyses in separate file directories.

4.1 Constructing Input Files

MULTIFAN-CL reads input files record by record and reads data within each record from left to right with data items separated by white space. When a hash (#) is encountered, the remainder of the record is considered to be a comment and is skipped. Data items are expected to be read in a strict order but without regard to how they might be grouped on records. Therefore the files can be organized into records

and sprinkled with comments so as to facilitate reading and editing by humans. Listings of example files are shown in the following sections.

4.1.1 The *.frq* file

The main body of the *.frq* file supplies catch, effort, and size sample data. In addition, it has a header section that defines much of the structure of the data and the model. Below are sections of example *.frq* files for yellowfin tuna and a multi-sex case interspersed with explanatory text. A copy of this or a similar example file can be obtained from www.multifan-cl.org/.

Initial structural declarations

This is a listing through the first ten values read by MULTIFAN-CL.

```
# WCPO YELLOWFIN MULTIFAN-CL ANALYSIS 1962-2001
#
# Number of Number of Movement Number of Year1 a b c d e
# Regions Fisheries flag tag groups
5 15 1 27 1962 0 0 4 0 4
```

The meanings of the data in the order in which they appear are:

- number of regions (N_{R})
- number of fisheries (N_{F})
- movement flag, which should always be set to 1 as it sets the generic movement submodel as documented in Appendix A.
- number of tag groups (N_{G}), which is the number of tag "cohorts" (all tag releases in a model region during a particular time interval) being modeled (see *.tag* file). If tagging data are not being included in the analysis, this flag is set to 0.
- first year for which catch and effort data are included in the catch/effort section of the file
- number of sexes or species (denoted "a" in example)
- flag to indicate whether the frequency data are length (0) or age(1) data (denoted "b" in example)
- number of recruitments per year assumed in the model (denoted "c" in example)
- the month in which recruitment is assumed to occur. The default (0) is that recruitment occurs in the first month of a recruitment period, i.e., in January for a single recruitment per year or January, April, July, and October for four recruitments per year. Recruitment is always assumed to occur in the first week of any month. (denoted "d" in example)
- *.frq* file version number – indicates for which version of MULTIFAN-CL the data files were constructed (intended for backward compatibility, but in practice causes MULTIFAN-CL to announce "Die yuppie scum" and quit when it encounters a *.frq* file that is incompatible with the MULTIFAN-CL code version) (denoted "e" in example)

For the case where multi-sex or multi-species data are provided, the structural declaration must specifiy the *.frq* file version number >= 8, and an index is inserted for the number of sexes/species (num species) in the flag denoted "a" in the above example, as in the following.

WCPO YELLOWFIN MULTIFAN-CL ANALYSIS 1962-2001 – multiple sexes

#

Number of Number of Movement Number of Year1 num b c d e

Regions Fisheries flag tag groups species

5 15 1 27 1962 2 0 4 0 8

Species/sex by tag groups and regions

For the case where multi-sex or multi-species data are provided, the tag groups and regions must defined in respect of which species/sex they relate. These sections must be added with the *.frq* file version number being specified >= 8. Firstly, the number of tag groups in which each species/sex occurs must be specified, as in the following example.

number of tag groups for

each species

1 2

19 19

Secondly, a vector having a length equal to the number of regions in the model must be added for each species/sex which defines in which regions each occurs. The value 1 indicates it occurs in the region, and 0 indicates it does not occur. The following example illustrates all regions are inhabited by both species/sexes.

regions in which species 1 occurs

1 2 3 4 5

1 1 1 1 1

regions in which species 2 occurs

1 2 3 4 5

1 1 1 1 1

Region sizes

The *.frq* file continues with information on relative size of regions. N_{R} numbers are read by MULTIFAN-CL, one for each region. It is here that the order in which regions are indexed is defined. The comment line is to help humans remember the indexing but is not essential as far as MULTIFAN-CL is concerned. The numbers entered here may be any index of relative region size, as they are normalized within the model. Relative region size is important when the catchability of fisheries in different regions is grouped, i.e., assumed to be the same. Taking region size into account allows the model to obtain information on relative population size in the regions from the catch and effort data of fisheries so grouped.

1-north 2-mid.west 3-mid.east 4-south.west 5-south.east

Relative region size

1.35 1.00 1.20 0.25 1.20

Fishery locations

Next comes a N_{F} long vector of region indices indicating in which region each fishery operates. It is here that the order in which fisheries are indexed is defined. Again, the comment lines are not essential but useful for humans in making sure that the indexing is correct.

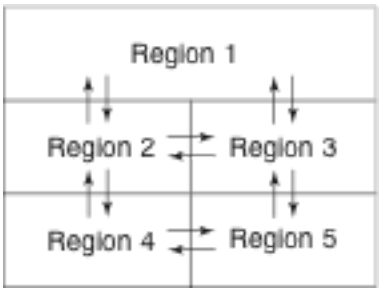
```
# Region in which each fishery is located:
# 1 -- LL 1
# | 2 -- LL 2
# | | 3 -- LL 3
# | | | 4 -- LL 4
# | | | | 5 -- LL 4A
# | | | | | 6 -- LL 5
# | | | | | | 7 -- PS/LOG 2
# | | | | | | | 8 -- PS/FAD 2
# | | | | | | | | 9 -- PS/LOG 2
# | | | | | | | | | 10 -- PS/SCH 3
# | | | | | | | | | | 11 -- PS/FAD 3
# | | | | | | | | | | | 12 -- PS/SCH 3
# | | | | | | | | | | | | 13 -- PH/R 3
# | | | | | | | | | | | | | 14 -- PH/H 3
# | | | | | | | | | | | | | | 15 -- ID 3
# | | | | | | | | | | | | | | |
1 2 3 4 5 2 2 2 3 3 3 2 2 2 4
```

Movement matrix

Next comes the movement matrix, a $N_{\mathrm{R}} \times N_{\mathrm{R}}$ matrix of 1-s and 0-s giving information on movement of fish between regions. 1-s indicate direct transfer of fish from one region to another and 0-s indicate lack of direct transfer. The row labels indicate the origin of movements and column labels indicate the destination. The movement matrix is symmetrical to allow for movement in either direction, and the diagonal is all 0-s.

Normally, proximal exchange of fish is assumed to occur only across common borders. This is a convention used to minimize the number of movement parameters to be estimated rather than a requirement of the model. Note that exchange of fish is not limited to adjacent regions in one time step. The implicit movement scheme used (see 3.6, A.1.3) allows exchange among regions that are either directly linked or indirectly linked through other regions.

A schematic map of the regions in this example is:



which shows the common borders between regions from which the movement matrix in the following *.frq* fragment is derived:[1]

Movement matrix (1=contiguous regions, 0=non-contiguous)

Matrix in the form

R2 R3 R4 R5

1 1 0 0 #R1

1 1 0 #R2

0 1 #R3

1 #R4

Note that only the upper triangle (without the diagonal) of the movement matrix is entered in the *.frq* file. MULTIFAN-CL automatically fills in the lower triangle.

For the case where multi-sex or multi-species data are provided, a movement matrix must be specified for each species or sex.

Fishery data flags

A $5 \times N_{\mathrm{F}}$ matrix comes next giving values for fishery data flags.

Data flags

0 0 0 0 0 1 1 1 1 1 1 1 1 0

2001 2001 2001 2001 2001 2001 2001 2001 2001 2001 2001 2001 2001 2001 2001

8 8 8 8 8 8 8 8 8 8 8 8 8 8

0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0

The first row indicates the type of catch data for each fishery: 0 for catch in number and 1 for catch in weight. The second and third rows indicate the year and month respectively for the start of the projection period for each fishery (N.B. at present the start of the projection period should be set the same for all fisheries). Rows 4-5 are currently unused.

Season-region flags

Next comes a matrix of season-region flags being $s \times nregs$, where the number of rows s are the number of "seasons" or recruitments per year, and the number of columns are the number of regions. These flags specify

[1] Note that when there is only one region ($N_R = 1$), no vestige of a movement matrix should appear in the *.frq* file. However, see footnotes under "Periodic Movement" below (page 47) and in structuring movement and recruitment in the *.ini* file (page 54).

the seasons and regions within which recruitment is assumed to occur. The value 1 = recruitments occurs (presence), and the value 0 = recruitments does not occur (absence). For example if recruitment is assumed to be uniformly present the section is:

Season-region flags

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

Whereas, the next example has recruitment occurring only in the summer seasons (first two quarters) and in only two regions.

Season-region flags

0 0 1 0 1

0 0 1 0 1

0 0 0 0 0

0 0 0 0 0

For the case where multi-sex or multi-species data are provided, a season-region flags matrix must be specified for each species or sex.

Periodic movement

Aspects of movement behavior are defined next starting with the number of movement periods per year, that is, the number of times per year in which fish may move from one region to another. This number is followed by a vector giving for each movement period the week within the year at which the movement occurs.[2]

num movement periods

4

move weeks

5 17 29 41

MULTIFAN-CL makes the simplifying and approximate assumption that there are exactly four weeks in every month (and therefore 48 weeks in a year). Thus week within year is given by $4(m - 1) + w$ for month m and week within month w, i.e., the first weeks of months 2, 5, 8, and 11 lead to the movement weeks in the example above. It is a good idea for movement weeks to correspond to weeks in the catch/effort part (described below) of the *.frq* file. Movement behavior is further structured by entries in the *.ini* file (see "movement map" – page 31).

Catch at age fit flags

An option for fitting to catch-at-age composition data is currently in development and is not yet fully operational and remains untested. However, it is essential for the two related flags (*age_nage* and *age_age1*) to be included in the *.frq* file for all versions >= 5. These are entered in the *.frq* immediately preceding the catch/effort/sample data. The flag *age_nage* is the switch for activating this option and must be set to 0 so as not to implement the feature. The following two lines must be entered into the *.frq* file.

[2]Note that when there is only one region ($N_R = 1$), the number of movement periods should be 1, not zero even though there actually is no movement in this case. Also, there should be a single dummy integer entered for the move weeks.

age_nage age_age1

0 -1

Structure of catch/effort/sample data

The last part of the header section of the *.frq* file gives information on the structure of the catch, effort, and sample data to follow.

Number of fishing incidents

1854

No. intervals 1st width grouping factor

100 10 2 1 # length frequency data

100 1.1561 1.1561 1 # weight frequency data

This part begins with the number of fishing incidents (N_{C}), i.e. the number of sets of catch/effort observations (with or without length or weight frequency data). This is followed by eight numbers indicating the structure of sample data as follows:

- number of length frequency intervals (N_{L})
- lower length of the first interval
- width of length intervals
- length frequency grouping factor – this allows the length intervals specified in the *.frq* file to be grouped into any multiple of the original interval in the model. For example, if the length interval width is 2 cm and a grouping factor of 2 is specified, the length frequency data are grouped into 4 cm intervals for the analysis.
- number of weight frequency intervals (N_{W}). If weight frequency data are not being included in the analysis, this flag is set to 0.
- lower weight of first interval (if $N_{\mathrm{W}} = 0$, set this flag to 0)
- width of weight intervals (if $N_{\mathrm{W}} = 0$, set this flag to 0)
- weight frequency grouping factor – as for length frequency data (if $N_{\mathrm{W}} = 0$, set this flag to 0)

Note that the specification of length and weight intervals (first interval and interval width) do not need to be integers, although this will often be the case. One instance where it is necessary to use floating point numbers is to define weight intervals in terms of live weight where the actual observations are in terms of processed weight. If the conversion of processed to live weight is a simple proportional increase, e.g., live weight is 10% higher than processed weight, then this proportion may be applied as a scaling factor to the first weight interval and the interval width. For example, consider a case where "gilled and gutted" processed weights are collected and the data are aggregated into 1 kg weight intervals with the lower weight of the first interval being 5 kg. If live weight is 10% higher than processed weight, the lower weight of the first weight interval is set to 5.5 and the interval width to 1.1. If either length or weight samples are not used, dummy numbers (0 is a useful convention) must still be included so that there are eight numbers in this section of the file.

4.1.2 The catch/effort/sample data

The final section of the *.frq* file gives catch, effort and sample data with one item for each of the N_{C} fishing incidents:

```
#year month week fishery catch effort length_sample weight_sample
#
1962 2 1 1 67523 54317 0 0...0 # 100 length frequencies
-1 # no weight frequencies
1962 5 1 1 30360 19354 0 0...0 # 100 length frequencies
-1 # no weight frequencies
1962 8 1 1 6802 3755 0 0...0 # 100 length frequencies
-1 # no weight frequencies
1962 11 1 1 31622 26369 0 0...0 # 100 length frequencies
-1 # no weight frequencies
.
.
.
2001 2 1 15 589 22912 -1 # no length frequencies
0 1...1 # 100 weight frequencies
2001 5 1 15 456 26769 -1 # no length frequencies
0 2...0 # 100 weight frequencies
2001 8 1 15 -1 23354 -1 # \
-1 # |
2001 11 1 15 -1 21554 -1 # |
-1 # |
2002 2 1 15 -1 22912 -1 # |
-1 # | projection period
2002 5 1 15 -1 26769 -1 # |
-1 # |
2002 8 1 15 -1 23354 -1 # |
-1 # |
2002 11 1 15 -1 21554 -1 # |
-1 # /
```

The data in each item appear in the following order:

- year
- month
- week
- fishery index
- catch (amount in numbers or weight (see "Fishery data flags" above), or -1 if catch is missing)
- effort (or -1 if effort is missing)
- time-dependent CPUE penalty - only in *.frq* file versions >= 6.

• N_L numbers giving length sample count in each length interval (or -1 if length sample is missing, but note that if length frequency data are not used in the analysis at all ($N_L = 0$), these numbers are omitted)

• N_W numbers giving weight sample count in each weight interval (or -1 if weight sample is missing, but note that if weight frequency data are not used in the analysis at all ($N_W = 0$), these numbers are omitted)

The format of the catch/effort/sample data for the *.frq* file version 8 (see structural declaration above) that provides input data for multiple sexes or species requires the input for each sex/species to be on separate rows, and includes additional columns: 5 to (4 + 2*n, where n is the number of sexes/species) that precede the columns holding the catch and effort data. For the case of multiple sexes, 4 additional columns are inserted (columns 5 to 8), that identify which of the sexes/species are included in the row of fisheries data. These index columns also facilitate the input of data aggregated among sexes/species. An example for multiple sex data follows with an explanation of the format.

#year month week fishery sex1 sex2 sex1 sex2 catch effort LF/WF_samples

index index agg agg

#

1962 2 1 1 1 0 1 0 420833 54317 0 0...0 -1

1962 2 1 1 0 1 0 1 25440 54317 0 0...0 -1

1962 5 1 1 1 0 1 0 18049 19354 0 0...0 -1

1962 5 1 1 0 1 0 1 12311 19354 0 0...0 -1

. . .

1962 8 1 1 1 0 1 1 6802 3755 0 0...0 -1

1962 8 1 1 0 1 1 1 6802 3755 0 0...0 -1

1962 11 1 1 1 0 1 1 31622 26369 0 0...0 -1

1962 11 1 1 0 1 1 1 31622 26369 0 0...0 -1

For each single fishery realization, a row must be entered for each sex/species, even if all data are aggregated among sex/species. For a two-sex model, this means that duplicate rows are input for each fishery realization. The purpose of the rows is to: create the realization; enter the data; create the "region" the additional sex/species; and, create the extra fishery for the second, or more, sex/species. The format of the additional indicator columns are as follows, in respect of the example above for two sexes,:

- a "1" in the 5th and 6th columns denote to which sex the data in the row relates (columns "sex1 index" and "sex2 index").
- a "1" in the 7th and 8th columns denote whether the data in the row are aggregated for both sexes (columns "sex1 agg" and "sex2 agg"), i.e. if both columns have the entry "1", this indicates catches and size data are aggregated.

Note that the effort in the duplicated rows is identical since this is the same fishing incident, but the catch entered will be different if it is dis-aggregated with respect to sex/species. In the above example the fishing incidents in 1962 2 1 1 and 1962 5 1 1 the catch and size composition data are not aggregated among the sexes (columns 7 and 8 have entries: 1 0 and 0 1). For the fishing incidents in 1962 8 1 1 and 1962 11 1 1, these data are aggregated among the sexes (columns 7 and 8 have entries: 1 1 and 1 1). Another example follows for the sexes/species indicator columns, where three sexes/species occur, entailing 6 additional columns to the catch data. In respect of columns 5 to 10, for example a single fishing realization is:

. . . 1 0 0 0 1 1. . .

. . . 0 1 0 0 1 1. . .

. . . 0 0 1 0 1 1. . .

This indicates there are three sexes/species, with only two sexes/species present in the catch and size data which are aggregated, and these sexes/species are indexed 2 and 3; while sex/species 1 is completely absent from the catch and size data.

Fishing incident items do not need to correspond to file records in the *.frq* file, i.e., data for fishing incidents can be split over several lines. The above first example shows each item split into two file records. However, there is a certain advantage to keeping fishing incident items on single file records, even though it makes for lengthy records, as in the above second example. It facilitates use of record counters in file editors or unix utilities such as *awk* and *wc* to assure that the number of fishing records (N_{C}) is accurate (MULTIFAN-CL will not function as intended if this is not the case). It also facilitates reading the *.frq* file into a spreadsheet if that is desired.

Projection period

MULTIFAN-CL will project estimates of abundance and catch beyond the end of the time series of known catch and effort. This is effected simply by adding more catch/effort records to the *.frq* file denoting a future regime governed either by effort or catch. If prescribed effort data are used, the "-1" should be entered for catch and vice versa. The example data above show a projection period of 1.5 years for one of the fleets. Projections can be included for any of the fleets and for as long a period as desired.

4.1.3 The *.ini* file

The *.ini* file supplies information mainly on biological characteristics of the fish. This file is used prior to the first phase of an estimation procedure, and the information therein is incorporated into the initial *.par* file (see section 4.1.5). The following example of an *.ini* file is interspersed with explanatory text. A copy of this example or a similar file can be obtained from <http://www.multifan-cl.org/>.

File format – version number

This fragment of the *.ini* file continues through comment lines to the first value read by MULTIFAN-CL which is the file version number that denotes the particular format used.

ini version number

1001

Where a value of 1000 denotes an old format and a value of 1001 denotes the current format for a single sex/species model. If an incorrect value is entered (<1000), the old format will be assumed and the value will be taken to be that of the next input variable, being the number of age classes.

For a multiple sex/species model, the version number of the *.ini* is specified as 1002 which informs MULTIFAN-CL that this is a multi-sex/-species model with the region_flags and species_flags supplied (see below).

Age classes

The next fragment of the *.ini* file continues through comment lines to the second value read by MULTIFAN-CL which is the number of age classes N_{A}

WCPO YELLOWFIN MULTIFAN-CL ANALYSIS 1962-2001

#

number of age classes

20

starting with the age of recruitment up to the oldest class, which is a "plus" group. The time resolution of age classes or recruitment frequency (e.g., annually or quarterly) is defined in the *.frq* file.

Tag reporting rate parameters

For the current format, the next fragment of the *.ini* file consists of five sections for the tagging reporting rate parameters.

> # tag fish rep - initial values of reporting rates by fishery and release group
>
> # tag fish rep group flags - grouping of reporting rates by fishery and release group
>
> # tag_fish_rep active flags - switch for turning on estimation, all matrix is activated=1
>
> # tag_fish_rep target - prior mean of reporting rate, usually set to be the same as init value
>
> # tag_fish_rep penalty - penalty on deviations from the prior mean

Each section is a matrix by fishery (across) and release group (down), the number of columns being the number of fisheries, and the number of rows being the number of tag release groups nrel+1. The additional row provides the parameters for an aggregate tag release group (see section 3.7.4).

Multiple sexes/species

For a multi-sex/-species model the region_flags and species_flags must be flags supplied in an *.ini* having version number 1002. The region_flags specify the "kludged" regions for sex/species 1, 2, to *n*. The number of regions (i.e. columns) are effectively replicates of that number specified for sex/species 1. The first row activates the estimation of average distribution of recruitment among regions (is set = 1 by default when estimation of total recruitment is activated, see 4.5.11). The third row indicates the sharing of these parameters among the sexes/species. For a two-sex model which shares the recruitment distribution for each sex among the regions the region_flags will be as follows, using a 6-region model for example:

region control flags

1 1 1 1 1 1 1 1 1 1 1 1

0 0 0 0 0 0 0 0 0 0 0 0

1 2 3 4 5 6 1 2 3 4 5 6

0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0

For a multi-species model having no sharing among the species, all entries in the third row will be zero.

The species_flags have rows for n number of sexes/species and 10 columns. The first column defines the activation of multiple sexes/species that share temporal variation in recruitment. For a multi-species

application with no sharing, the species_flags would be as follows (no activation) for a two-species, six-region example:

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

For a multi-sex model the first column would be set to 1 for each row because it is reasonable to assume temporal variation in recruitment would be shared among the sexes.

The second column applies only to the multi-sex case, and indentifies which of the sexes 1 to n is the female sex. For example, a two-sex case with sex 2 being denoted as the female sex, the species_flags would be as follows (activation for species 2):

1 0 0 0 0 0 0 0 0 0

1 1 0 0 0 0 0 0 0 0

The second column, therefore identifies for which sex the spawning biomass will be used in deriving the single Beverton-Holt stock-recruitment relationship (BH-SRR) that will predict average recruitment for both sexes.

Maturity

The maturity schedule comes next in the *.ini* file, contained in a N_{A} vector of the proportions mature by age class

MATURITY AT AGE

0 0 0 0 0 0 0.25 0.50 0.75 1 1 1 1 1 1 1 1 1 1 1

This vector is used in the computation of spawning biomass. If only females are to be included in the spawning biomass, age-specific sex ratios can also be included in the maturity schedule. In this case, each element of the maturity schedule would be the proportion of the female population in an age class that is mature multiplied by the proportion of the total population in that age class that is female. A more complete way to model reproductive output is to include spawning fraction at age and fecundity / kg at age, as well as maturity and sex ratio. The 'maturity' schedule then represents reproductive potential at age.

For the multi-sex/-species case (an *.ini* having version number 1002), the maturity schedules for additional sexes/species must be input. For example, a two-sex model, with identical schedules:

MATURITY AT AGE

0 0 0 0 0 0 0 0 0 0 0.05 0.1 0.2 0.4 0.6 0.7 0.8 0.85 0.9 0.95 1 1 1 1 1 1 1 1

The multi-species maturity

0 0 0 0 0 0 0 0 0 0 0.05 0.1 0.2 0.4 0.6 0.7 0.8 0.85 0.9 0.95 1 1 1 1 1 1 1 1

In the case where maturity at age is calculated internally based upon an input maturity at length schedule and estimated growth [3.3.1], the values entered for the maturity at age schedule can be proxy or dummy values.

Maturity at length

For an *.ini* having version number > 1002 the maturity at length schedule comes next, contained in a N_{L} vector of the proportions mature by length class. The length classes must correspond to that specified in the *.frq* file.

For the multi-sex/-species case, the maturity at length schedules for additional sexes/species must be input (as for maturity at age above).

In the case where maturity at age is calculated externally based upon an input maturity at length schedule and an assumed growth curve [3.3.1], the values entered for the maturity at length schedule can be proxy or dummy values.

Natural mortality

4.1.3 Next in the file is the starting value for natural mortality rate averaged over age classes.

natural mortality

0.25

A vector of natural mortality by age can be estimated by MULTIFAN-CL. The parameters actually estimated are age-specific deviations from the log of the mean natural mortality. The starting values of the deviations are normally zero, but if age-specific natural mortalities are to be fixed at specific values or the deviations are to be estimated starting from non-zero values, then the deviations may be set in the 2nd row of the age parameter matrix (see below).

For the multi-sex/-species case (an *.ini* having version number 1002), the natural mortality for additional sexes/species must be input. For example, a two-species model, with different mortality for each:

natural mortality

0.250298600000

The multi-species natural mortality

0.112277282

Movement

Movement information is then given starting with a "movement map" which groups the movement periods into seasons.

movement map

1 2 2 1

The number of entries in the movement map must correspond to the number of movement periods entered in the *.frq* file (see page 47). Based on the movement weeks defined in the example *.frq* file above, the movement map shown here would define two seasons with one movement behavior in weeks 41 and 5 of the year and another behavior in weeks 17 and 29. If it were desired to have different behavior in each of the movement periods, the movement map would be the sequence "1 2 3 4".

movement coefficients (12 for each move map entry)

0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1

0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1

0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1

0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1

Starting values of movement coefficients come next, with a set of coefficients for each entry in the movement map[3]. Within each set there is one coefficient for each possible direct transfer between regions, i.e., one for each "1" in the full movement matrix, the upper triangle of which is given in the *.frq* file (see "Movement matrix" - page 46). The coefficients are entered in row by row order of 1's in the full movement matrix. Thus coefficients in each row of the example above would correspond to movements between regions indicated as follows in the order given:

1→2, 1→3, 2→1, 2→3, 2→4, 3→1, 3→2, 3→5, 4→2, 4→5, 5→3, 5→4,

The time units of the coefficients are the inverse of the recruitment frequency, e.g., if recruitment is quarterly, the units are quarter-1.

For the case of multiple sexes/species a matrix of movement coefficients must be entered for each sex/species. The matrix dimensions will be the same for each sex/species given that the number of regions and movement maps will be common.

Age parameters

Next comes a matrix composed of 10 sets of age-specific parameters (termed *"age pars"*), one for each age class.

age_pars

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.00012

multi-species age_pars

11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0.57759 0.39304 0.174720432 -0.10498 -0.11553 -0.11528 -0.11484 -0.11405 -0.11272 -0.11059 -0.10730 -0.10247 -0.09571 -0.08669 -0.07523 -0.06194 -0.04634 -0.02971 -0.013233 0.00187 0.01463 0.02453 0.03156 0.03600 0.03833 0.03901 0.03843 0.03692363

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.00012

Recruitment

The initial proportion of total recruitment occurring in each region[4] follows in a vector, the order and length of which must be consistent with the region indexing defined in the *.frq* file (see "Region sizes" - page 45).

1-north 2-mid.west 3-mid.east 4-south.west 5-south.east

recruitment distribution among regions (5)

0.15 0.40 0.30 0.05 0.10

For the multi-sex/-species case (an *.ini* having version number 1002), the initial proportion of total recruitment occurring in each region for additional sexes/species must be input. For example, a two-sex model, with identical proportions for each:

recruitment distribution by region

0.05 0.06 0.4 0.35 0.05 0.09

multi-species recruitment distribution by region

0.05 0.06 0.4 0.35 0.05 0.09

[4] In case of only one region: *ibid.*

Growth and Steepness

Information about growth, both in size and in weight, comes next.

The von Bertalanffy parameters (mean length 1, mean length nage, K)

Initial value Lower bound Upper bound

28.0 20.0 40.0

145.0 140.0 200.0

0.1 0.0 0.3

Weight-length parameters

2.784e-05 2.8992

Steepness - sv(29)

0.9

Variance parameters (avg SD, SD dependency on length)

Initial value Lower bound Upper bound

6.0 3.0 8.0

0.40 -0.69 0.69

The values are entered in the file in the following order:

- The starting value for the mean length of the youngest age class followed by lower and upper bounds.
- The starting value for the mean length of the oldest age class followed by lower and upper bounds.
- The starting value for von Bertalanffy growth rate parameter (k) followed by lower and upper bounds. The units of k are the inverse of the recruitment frequency.
- The two parameters, a and b of the length-weight relationship: weight $= a \times \text{length}^b$. Weight and length would normally be in kg and cm, respectively. These two parameters are fixed and are not currently estimated by MULTIFAN-CL.
- The starting value for the steepness parameter of the Beverton-Holt stock-recruitment relationship (see section 3.8.2).
- The starting value for the average standard deviation of length at age followed by lower and upper bounds.
- The starting value for age dependency of standard deviation of size at age followed by lower and upper bounds. This parameter has been log-transformed, so that a value of 0 would infer no age-dependency, while a value of 0.69 would indicate that the standard deviation of the oldest age class is approximately twice that of the youngest age class ($\exp(0.69) = 2$).

For the multi-sex/-species case (an *.ini* having version number 1002), the parameters for additional sexes/species must be input. For example, a two-species model, with different parameters for each:

The von Bertalanffy parameters (mean length 1, mean length nage, K)

Initial value Lower bound Upper bound

25.0 20.0 40.0

150.0 140.0 200.0

0.15 0.0 0.3

#

The multi-species von Bertalanffy parameters (mean length 1, mean length nage, K)

28.0 20.0 40.0

180.0 140.0 200.0

0.075 0.0 0.3

Weight-length parameters

FAR Seas values

2.512e-05 2.9396

sv(29)

0.9

Multi-species length-weight parameters

1.9729e-05 3.0247

Multi-species sv(29)

0.75

Variance parameters (Average SD by age class, SD dependency on mean length)

Initial value Lower bound Upper bound

6.0 3.0 15.0

0.40 -1.00 1.00

multi-species Variance parameters (Average SD by age class, SD dependency on mean length)

Initial value Lower bound Upper bound

6.0 3.0 12.0

0.1 -1.5 1.5

Mean size constraints

The option to specify mean size constraints is a feature of MULTIFAN-CL that allows the analyst to provide some information on the structure of individual length-frequency samples. This may be useful, particularly in the initial phases, in situations where biologically unreasonable growth estimates are being obtained. Mean length constraints specified for samples where there are obvious modes that should lie on or close to the growth curve give MULTIFAN-CL "a helping hand" in its interpretation of the length frequency data. Such constraints may be removed during the later phases of the estimation if desired.

This final fragment of the *.ini* file begins with the number of constraints followed by information on each.

The number of mean constraints

3

Fishery Incident Age Lower Upper Lower Upper

class length length proportion proportion

1 94 1 23.0 30.0 0.1 0.9

2 94 1 23.0 30.0 0.1 0.9

2 94 2 39.0 50.0 0.1 0.9

Each mean size constraint consists of a specification of the fishery, the number of the fishing incident for that fishery (in time sequence), the age class number for which the constraint is being imposed, the lower and upper bound for the mean length constraint, and the lower and upper bound for the proportion at age of that age class in the length frequency sample.

4.1.4 The *.tag* file

Tag release and recovery information is organized into groups consisting of the tags released within a particular model region and a particular year and month. The releases are further stratified by length intervals, which would normally be the same as those defined for the fishery length frequency data. The history of tag recoveries for each group is then summarized by release length interval, fishery, year and month of recapture. Care must be taken to ensure that the year/month specification of tag recapture periods corresponds to that used in the *.frq* file for each fishery.

The data in the *.tag* file begins with header information as in the following example listing.

WCPO YELLOWFIN MULTIFAN-CL ANALYSIS 1962-2001

#

RELEASE GROUPS STARTING LENGTH NUMBER INTERVALS INTERVAL LENGTH

27 10 100 2

TAG RECOVERIES

11 11 120 104 83 131 109 47 28 0 88 49 0 2 0 73 36 31 2

46 50 8 0 1 8 0 4

The header data read by MULTIFAN-CL are ordered as follows:

- number of tag groups (N_{G}) – This must be consistent with N_{G} in the header information in the *.frq* file.
- lower length of first length interval for size measurement of tagged fish
- number of intervals N_{L}
- width of intervals
- a vector $\{N_{\mathrm{r\,g}};\ \mathrm{g} = 1\ .\ .\ .N_{\mathrm{G}}\}$ giving the number of recapture categories, or strata, in each tag group

Following the header there must be N_{G} data sections, one for each of the tag groups. Here is a listing of the first group in the example *.tag* file.

#---------------------------------

1 - RELEASE REGION YEAR MONTH

2 1989 8

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 2 4 9 26 25 27 18 23 14 8 5 7 3 2 2 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

#

#

LENGTH RELEASE FISHERY RECAP YEAR RECAP MONTH NUMBER

48 6 1989 8 4

48 6 1989 11 1

50 9 1992 2 1

52 6 1989 11 3

56 6 1989 8 2

56 6 1989 11 2

56 6 1990 5 1

58 6 1989 8 1

58 6 1990 2 1

60 6 1989 8 2

62 6 1989 11 1

Each group is first identified by the region index, year, and month of release in that order followed by a vector of N_{L} size frequencies (100 of them in this case) for all the fish released in the group. Normally these frequencies will be integer numbers, but real (decimal) numbers can be input if necessary, as may be the case if independent analyses of tag release data have been undertaken prior to its input to MULTIFAN-CL. If this were group g, data must follow for $N_{\mathrm{r\ g}}$ recapture categories (11 of them in this case), identifying each category and enumerating the recoveries as follows:

- fish length at release
- index of fishery in which fish was recaptured
- year of recapture
- month of recapture
- count of recovered fish in category defined by the four items above

For the case of multiple sexes/species, the format for the header of each of N_{G} data sections, one for each of the tag groups, must include species flags. These species flags are read in conditional upon the input data (in the *.frq* file) being for more than one sex/ species. For an example, the following header denotes release group one consisting of releases and recaptures for only sex/species 1:

1 - RELEASE REGION YEAR MONTH SPECIES 1 FLAG SPECIES 2 FLAG

2 1998 2 1 0

Whereas, the following header denotes release group two consisting of species 1 and 2 combined releases:

1 - RELEASE REGION YEAR MONTH SPECIES 1 FLAG SPECIES 2 FLAG

2 1998 5 1 1

Tag release groups consisting of data combined over both sexes were treated to have associated recaptures that were also combined over both sexes; i.e. for these release groups, the sex is unknown for both the releases and recaptures. This is the current state of this feature in MULTIFAN-CL, and data for species- or sex-specific recaptures for tag release groups having unknown species or sex, are not supported.

The fishing index, year of recapture, and month of recapture must have a corresponding record in the *.frq* file, i.e., a fishing incident must have occurred in order for one or more tag recaptures to have occurred. In this example 26 other data groups must follow to complete the *.tag* file. A copy of this or a similar example file can be obtained from <http://www.multifancl.org/>.

4.1.5 The *.par* file

The *.par* file is both an input and an output file. Normally the *.par* file which is output from one phase of a fit is then input to the succeeding phase. This file plus the *.frq* file contain all the information needed by MULTIFAN-CL to restart an analysis from where it left off. The *.par* file contains the latest values of all parameters plus other structural information about the current state of the analysis. Most of the structural information is embodied in a multitude of flags located at the top of the file. Information on the various types and meanings of the normally used (and some not so normally used) flags is given below in [4.5].

Note that for the case of multi-sex/species model certain flags and parameters are replicated over the number of sexes/species dimensioned in the model.

For *.par* versions > 1054, the facility exists for debugging the input and output of the file. This intersperses the file content with *check_numbers* at various points. These are unique integer values which are inspected by MULTIFAN-CL to check the sequence of the parameters read from the file. This is largely a code development tool. In MULTIFAN-CL versions > 2.0.4.0 this facility is activated using parest_flags(195) [4.5.1]. Note that the position of the *check_numbers* in the file are not indicated in the following list as they will vary depending upon the code developments, and hence the subsequent MULTIFAN-CL versions.

All segments of the file are labeled with comment records (i.e., with "#" in the first column). Here is a list of all segment labels.

> # The parest_flags – 400 of them in one lengthy record – vector parest_flags(1, 400); Version number is parest_flags(200)
>
> # The number of age classes – a single number – int nage;
>
> # age flags – 200 of them in one lengthy record – vector age_flags(1, 200);
>
> # fish flags – one record per fishery with 100 flags each – matrix fish_flags(1, nfsh, 1, 100);
>
> # tag flags (if tagging data included) – one record per tag group with 10 flags each – imatrix tag_flag(1,num_tag_releases,1,10);
>
> # tag fish rep (only for par file versions > 1041) - initial values of reporting rates by fishery and release group; – matrix tag_fish_rep(1, ntg+1,1,nfsh)
>
> # tag fish rep group flags (only for par file versions > 1041) - grouping of reporting rates by fishery and release group; tag_fish_rep_group_flags(1, ntg +1,1,nfsh)
>
> # tag_fish_rep active flags (only for par file versions > 1041) - switch for turning on estimation, all matrix is activated=1; tag_fish_rep_active_flags(1, ntg+1, 1, nfsh)
>
> # tag_fish_rep target (only for par file versions > 1041) - prior mean of reporting rate, usually set to be the same as init value; tag_fish_rep_target(1, ntg+1, 1, nfsh)
>
> # tag_fish_rep penalty (only for par file versions > 1041) - penalty on deviations from the prior mean; tag_fish_rep_penalty(1, ntg+1,1,nfsh)
>
> # region control flags – 10 for each region – matrix region_flags(1, 10, 1, nregions)
>
> # species flags – matrix(2,10) defining the multi-sex instance and identifying the female sex
>
> # percent maturity – the maturity schedule as given in the .ini file or as calculated internally based upon maturity at length input from the .ini file and the estimated growth curve. – vector pmature(1, nage);
>
> # total populations scaling parameter – tot population dvariable totpop;
>
> # rec init pop level difference –
>
> # recruitment times –12 numbers... for each month? 1 ⇒ recruiment 0 ⇒ no recruitment ??
>
> # relative recruitment – time variation in recruitment, one per fishing period

Reporting rate dev coffs – (1,num fisheries,2,num fish times) a ragged array

availability coffs – no. age classes

annual coffs for relative recruitment Only present conditional upon the historical parest_flags(155) being > 0. The vector length is the maximum number of polynomial degrees, (equal to the total number of calendar years).

orthogonal poly coffs for relative recruitment The orthogonal polynomial parameters from the *.par* are only present conditional upon the historical parest_flags(155) being > 0, i.e. that this flag was set in the run that generated the estimated orthogonal recruitment parameters. The parameters are dimensioned according to the temporal and spatial stratification of the model, and therefore have no fixed format. The number of rows is the sum total of the number of vectors at each of the four polynomial levels, and the number of columns is the maximum number of polynomial degrees, (equal to the total number of calendar years).

relative initial population – matrix (1, nreg) × (2,nage)

fishery selectivity – (1,nfsh,1,nage)

natural mortality coefficient – est of natural mortality averaged over age classes

average catchability coefficients – average catchability for each fishery

initial trend in catchability coefficients – (1,nfsh)

diffusion coefficients – no. move coeffs (2×no. 1-s in move matrix)

age dependent diffusion coefficients – no. move coeffs (2×no. 1-s in move matrix)

nonlinear diffusion coefficients – no. move coeffs (2×no. 1-s in move matrix)

regional recruitment variation – matrix (nyrs × nregions). Specifies time-series deviations from the average in regional recruitment distribution.

effort deviation coefficients – (1,nfsh,1,nft) a ragged array... nft=vector of no. incidents per fishery

correlation in selectivity deviation – (1,nfsh)

extra fishery parameters – (1,50,1,nfsh)

1. amplitude for sinusoidal seasonal catchability
2. phase for sinusoidal seasonal catchability
3. tag reporting rates
4. σ^2 for negative binomial likelihood for tags (parest flag 111, see section 4.5.8)
5. parameter for added zeros in negative binomial (see section 4.5.8)
6. another parameter for added zeros
7. effort effect on catchability
8. Biomass effect on catchability
9. Logistic or double-normal selectivity (version < 1048)
10. Logistic or double-normal selectivity (version < 1048)
11. Double-normal selectivity (version < 1048)
12. Dirichlet-Multinomial likelihood
13. Catch-at-age likelihood (not operational)

14. SSM-RE log-length N variance
15. SSM-RE log-weight N variance
16. SSM-RE length rho correlation
17. SSM-RE weight rho correlation
18. SSM-RE log-length variance
19. SSM-RE log-weight variance
20. SSM-RE length sample size covariate
21. SSM-RE weight sample size covariate
22. Dirichlet-Multinomial no random effects length lambda exponent
23. Dirichlet-Multinomial no random effects length sample size covariate
24. Dirichlet-Multinomial no random effects weight lambda exponent
25. Dirichlet-Multinomial no random effects weight sample size covariate
26. to 50. Unused

> # species parameters (species_pars)– (1,20,1,nsp) where nsp is the number of sexes/species (see section 4.1.1)

1. Unused
2. Estimated ρ parameter for auto-correlated recruitments

Rows 3 to 20 are unused.

> # seasonal_catchability_pars
>
> # age-class related parameters (age_pars)– (1,10,1,12) (see 3.4.1). – a 10 × nage matrix containing by row:

1. Unused
2. age-specific deviations from log of average natural mortality
3. deviations from von Bertalanffy curve
4. growth curve deviations
5. log of natural mortalilty at age if age flag 109 set, and functional form parameters if age flag 109 = 2 or 3..

Rows 6–10 are unused. These parameters can be pre-set in the *.ini* file ("Age parameters" – page 31).

> # region parameters – (region pars) – a 100 x nregions matrix. The first row contains the average over time of the regional recruitment distribution. If age flag 125 and 126 = 1, row 2 to n contain the parameter scalars for the biomass-dependent effect on catchability, where n is the number of years over which average F is calculated. Other rows have not yet been used.
>
> # catchability deviation coefficients – catch dev coffs(1,nfsh, 2,nft) a ragged array..
>
> # selectivity deviation coefficients – (1,fsh.num fisheries, 2,fsh.num fish times, 1,fsh.nage)
>
> # sel_dev_coffs – sel dev coffs(1,nfsh,1,nft,1,ng)
>
> # year_flags

season_flags

The logistic normal parameters

log_length variance length rho log_length_dof length_psi length_exp

log_weight variance weight rho log_weight_dof weight_psi weight exp

length_tot_exp weight_tot_exp

maturity at length – for *.par* versions > 1055 the maturity (spawning potential) schedule by length class as input from the *.ini*

The von Bertalanffy parameters – as in *.ini*

First Length bias parameters – vb bias(1,num fisheries)

Common first Length bias flags – common vb bias(0,num fisheries)

Common first Length bias coffs – common vb bias coffs(1,num fisheries)

Seasonal growth parameters – sv(1,30) nnewlan.cpp 47: int NUMSV=30; sv(27) and sv(28) are a and b of length-weight relationship.

Cohort specific growth deviations – growth dev.allocate(1,fsh.nyears);

Variance parameters – as in *.ini*

The number of mean constraints – 1 number (from *.ini*) if not zero, followed by constraint data as in *.ini* file.

The diffusion coefficients – newmult.cpp 55: D(1,nregions,1,nregions)

The grouped_catch_dev_coffs flag – 11 ⇒group stuff to follow; 01 ⇒not

The grouped_catch_dev_coffs – grpcatch.cpp 149: grouped catch dev coffs.allocate (1, ngroups, 2, num grouped fish times)

Historical_flags – flags taken from the initial 00.par or the input .par file.

The parest_flags

fish flags

End_historical_flags – end of flags taken from the initial 00.par or the input .par file.

Objective function value – 1 number

The number of parameters – 1 number

Likelihood component for tags – 1 number (hidden behind #)

Penalty for mean length constraints – 1 number ... 0 if age_flag(52) set

Maximum magnitude gradient value – 1 number ... 0 if age_flag(52) set

Average fish mort per fishing incident – 1 number (hidden behind #)

Average fish mort per year – 1 number (hidden behind #)

Average fish mort per year by age class – nage numbers (hidden behind #)

4.1.6 The *mfcl.cfg* file

This is an optional file which, if it exists in the directory from which MULTIFAN-CL is run, will tune the memory allocation. It supplies three numbers:

1. GRADSTACK BUFFER SIZE (default, 1000000)
2. CMPDIF BUFFER SIZE (default, 56000000)
3. ad array mbl size (default, 25000000)

4.1.7 The *.env* file

This is an optional file which needs to be supplied only if spatially aggregated recruitment is to be estimated in relation to some environmental driving variable (see 4.5.11 and 3.2). This file supplies the environmental data as a sequence of numbers corresponding to recruitment times in the model.

4.1.8 The *selblocks.dat* file

An input file called *selblocks.dat* is supplied by the analyst that specifies the periods associated with particular fisheries having long-term variation in selectivity. The format places the years for the first year of each time-block, subsequent to the initial time-block, are listed as a vector on each row for the relevant fisheries. An example follows:

1997 2002

1997 2002

2000

2000

The first two fisheries have three time-blocks: the initial time-block starting in the first year of the fisheries' time period, and the second and third time-blocks starting in 1997 and 2002, respectively. The third and fourth fisheries have two time-blocks: the initial time-block starting in the first year of the fishery's time period, and the second time-block starting in 2000, respectively. To which fisheries these time-blocks relate, is controlled by fish flag 71. By convention, each row in fish_flags points to fisheries 1 to n fisheries in the model, and a zero in column 71 represents time-invariant selectivity for the fishery in the relevant row indicated. A number greater than zero both activates the estimation of time-variant selectivity for the fishery in the relevant row, and informs the number of breaks in the fishery's time period and therefore the number of time-blocks to be estimated (n breaks + 1). In this example, the column of fish flag 71 has the following elements:

2
2
0
0
1
0
0
1
0
...

Which indicates that fisheries 1 and 2 have 2 breaks (i.e. three time-blocks), and fisheries 5 and 8 have 1 break (i.e. two time-blocks). The rows in selblocks.dat relate, in order, to the fisheries identified by fish flag 71. In this example, the rows relate to fisheries 1 and 2 (2 breaks for each), and fisheries 5 and 8 (1 break for each).

4.1.9 The *.age_length* file

When direct observations of ages at length are available this is input in a file called **.age_length* where * is the root name for the arguments supplied to the program, for example *alb.age_length*. The context of age-length data being fitted in the model is activated by parest flag 240.

The format of the input file consists of three sections:

- a header indicating how many fishery-specific samples of age-length are to be input (how many matrices of age-length),
- a vector of sample-specific scalars for the effective sample size i.e. the number of elements equals the above number, and
- for each sample, a header providing the details of the sample and the age-length matrix itself.

An example follows for a model having 14 fisheries with 12 age-length samples having been collected from various fishing incidents.

```
# num age length records
12
# effective sample size
1 1 1 1 1 1 1 1 1 1 1 1
# Year Month Fishery Species
1996 5 4 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
...
```

Only one sample *i.e.record* is illustrated for this example, and therefore only one sample matrix is shown, however in the input file there are 12 matrices being input. The scalar for the effective sample size among all the 12 samples is “1”, and therefore the effective sample size is the same as the observed sample size. The header of the sample record supplies details of the fishery and fishing incident from which it was sampled, and in the case of the multi-species/sex disaggregated model, which species/sex the record relates to. The matrix of frequencies of fish ages observed in each length class follows, with the rows being the length classes and the columns being the age classes. Note that the matrix dimensions correspond to those in the *.ini* for the number of age classes, e.g.

number of age classes
28

and in the *.frq* for the number of length classes, e.g.

#Datasets LFIntervals LFFirst LFWidth LFFactor WFIntervals WFFirst WFWidth WFFactor

3376 95 10 2 1 200 1 1 1

Therefore, in this example the matrix has the dimensions: 95, 28

If there are more than one record in the set of observations, a matrix is entered for each record subsequently, and each is pre-ceded by the header specifying to which fishery realisation it was taken from, e.g.

Year Month Fishery Species

1999 11 9 1

NOTE: if the fishing incident from which the age-length sample was collected does not have an associated catch in the fisheries data of the .frq file, this will cause MULTIFAN-CL to crash

4.1.10 The *.tag_sim* file

When generating simulation tagging data from projection time periods, it is necessary to supply two sources of information: details of the simulation tagging event(s); and, the event- and fishery-specific tag reporting rates assumed ([4.4.2]). This information is input in a file called *.tag_sim*.

The format consists of three sections:

- a header indicating how many simulation tagging events are to be input,
- a series of vectors, each holding the details of a particular event, and
- a matrix of the assumed tag reporting rates in respect of each event and the defined fisheries.

An example follows for a projection having 14 fisheries with 5 simulation tagging events.

```
# RELEASE GROUPS
5
#
#————————————————————
# 1 - RELEASE REGION YEAR MONTH Fishery Predicted numbers
2 2019 5 2 50
...
#————————————————————
# 5 - RELEASE REGION YEAR MONTH Fishery Predicted numbers
6 2019 5 10 1000
#
# Reporting rates for each event: rows = fisheries; cols = tag events
0.087733432 0.087733432 0.087733432 0.087733432 0.087733432
0.891764148 0.891764148 0.891764148 0.891764148 0.891764148
0.131841385 0.131841385 0.131841385 0.131841385 0.131841385
0.024424731 0.024424731 0.024424731 0.024424731 0.024424731
0.663929513 0.663929513 0.663929513 0.663929513 0.663929513
```

0.144619528 0.144619528 0.144619528 0.144619528 0.144619528
0.899525685 0.899525685 0.899525685 0.899525685 0.899525685
0.227107007 0.227107007 0.227107007 0.227107007 0.227107007
0.788129658 0.788129658 0.788129658 0.788129658 0.788129658
0.045505415 0.045505415 0.045505415 0.045505415 0.045505415
0.025934941 0.025934941 0.025934941 0.025934941 0.025934941
0.035171551 0.035171551 0.035171551 0.035171551 0.035171551
0.007716504 0.007716504 0.007716504 0.007716504 0.007716504
0.009808246 0.009808246 0.009808246 0.009808246 0.009808246

For this example the reporting rates were assumed to be constant among the simulation tagging events, however, the flexibility exists for assuming rates that are specific to fisheries and tagging events.

4.2 Temporal structure of model

To conduct a simulation, MULTIFAN-CL needs to have set up a schedule of calculation events such as times of recruitment, times to calculate abundance and catch, and times to move fish among regions. Each of these events determine a time "node", and much of the schedule of nodes is determined by the *.frq* file.

A sequence of regularly spaced nodes is determined by the number of recruitments per year and the month of first recruitment in a year given by the 8th and 9th numbers in the *.frq* file (see page 24). For example, if annual recruitment is assumed, new recruits would appear in the population in January (default) or in the month number (1-12) specified. If quarterly recruitment is assumed, the default behavior would be for recruitment to occur at the first week of January, April, July and October. The time of a fishing incident is indicated by the year, month and week in the *.frq* file (see "catch/effort/sample" – page 28), and nodes are established at the given times. In principle, it would be possible to have a data resolution of 48 fishing periods per year (12 months by 4 weeks per month). However, for most assessments data will be more coarsely aggregated. If data were quarterly, for example, fishing incidents could be associated with months 2, 5, 8 and 11 and week 1 in each case as in the example *.frq* file above.

Movement events also establish nodes at times indicated by the movement weeks entered in the *.frq* file (see "Periodic movement" – page 27).

To optimize calculation speed, it is desirable to minimize the number of nodes. For example with four recruitments per year, there must be at least four nodes per year. If the catch/effort data are also quarterly, but entered at different weeks than the recruitments, there will be eight nodes per year. And if movement events occur at still different weeks, the number of nodes per year will be multiplied further. However, if care is taken to keep the various events in phase, the number of nodes will be minimized. Thus the movement weeks in the example *.frq* file above are adjusted to coincide with the fishing incidents, but the recruitment month in the example is 0 which defaults to first week of January and is therefore out of phase. It would improve speed to set it to 2,5,8,or 11.

Another aspect of the node schedule is that for a particular catch/effort entry in the *.frq* file, the effort given is applied only during the time period starting at the immediately preceding node, regardless of how that node was determined. Thus if catch/effort data were entered annually at say month 6, week 2 (i.e., 26th week of the year), and recruitment month was set at 6 (i.e., 25th week of the year), then the population dynamics would be that of an intense fishery during one week of the year and zero effort at other times. Another example would be two fisheries, one with catch/effort entered annually and the other entered quarterly. Assuming the data were phased so that the annual fishery occurs at one of the weeks of the quarterly fishery,

the annual fishery would operate in the model only during one quarter of the year.[5] If the two fisheries were not in phase, the temporal situation would be further complicated for the annual fishery and for the quarterly fishery as well. Obviously it is desirable to minimize nodes so as to minimize confusion as well as to optimize calculation speed.

4.3 Conducting a Fit

The executable program file for MULTIFAN-CL is mfcl. A general form of the command line to run MULTIFAN-CL is as follows:

mfcl example.frq in.par out.par [-makepar|-switch args|-file fname]

with choice of one or none of the options within the square brackets. In the normal course

of events, a sequence of mfcl commands are issued, one for each phase of the fit:

mfcl example.frq 00.par 01.par ...

mfcl example.frq 01.par 02.par ...

mfcl example.frq 02.par 03.par ...

.

.

.

The *.par* file names could be anything, but it is common practice to add the extension .par and to name them in a numerical (or alphabetical) sequence so that they can be identified with the phase of the fit in which they are produced.

4.3.1 Creating an initial parameter file (`-makepar`)

MULTIFAN-CL normally needs an initial input *.par* file containing starting values for all the information noted above. Such a file would be tedious and error prone to assemble by hand. Therefore MULTIFAN-CL provides a way to use the *.ini* and *.frq* files to produce an initial *.par* file with the -makepar option:

mfcl example.frq example.ini 00.par -makepar

Note that a *.ini* file has taken the place of the input *.par* file, and the output *.par* file in this example will be named 00.par. Also note the point made in section 4.1.3 regarding the version number denoting the particular *.ini* file format. If the new format is employed, the version number *must* be specified at top of the *.ini* file.

4.3.2 Command line manipulation of flags (`-switch`)

On progressing from one fitting phase to another, it is usual practice to instruct MULTIFANCL to change something about the fitting process, for example the status of a parameter from being held constant to being estimated or relaxing the constraints on one or more parameters. This can be accomplished by editing the flag values in the *.par* file which will be input in the next phase. Information is given below in Section 4.5 on the various types and meanings of the flags that are manipulated in normal practice. Given the multitude of

[5] If the exploitation rate is not too severe, this approximation is probably not too bad. After all, some classical fishery models assume fishing occurs in instantaneous spikes.

flags, editing their values directly in *.par* files is difficult and error prone. Therefore the -switch option is available to edit flag values. It indicates that a number of switch statements are entered on the command line:

mfcl ex.frq 01.par 02.par -switch n switch$_1$ · · · switch$_n$

where n is the number of switch statements and where each switch statement is a vector of three integers {I_1, I_2, I_3}, and where new flag values are set according the following rules:

Table 4.1: Rules followed to assign flag values

I1		Action
1	—	parest flag $I_2 = I_3$
2	—	age flag $I_2 = I_3$
-n; ($n < 999$)	—	fish flag I_2 for fishery $n = I_3$
-999	—	fish flag I_2 for all the fisheries $= I_3$
-9999	—	tag flag I_2 for all tag release groups $= I_3$
-n; ($10000 \leq n < 99999$)	—	tag flag I_2 for tag release group n - 9999 $= I_3$
-n; ($100000 \leq n < 100009$)	—	region flag n - 99999 for region $I_2 = I_3$

Setting flags for fisheries, tag groups, or regions that have not been defined in the *.frq* file will generate an error message. Note that there is a limit of 10 types of region flag. However, the only region flag used so far is type 1. Therefore values of I_1 other than -100000 for region flags for now would be pointless. Unfortunately, as yet there is no short cut for identifying all the regions.

Here are some example switch settings:

mfcl ex.frq 01.par 02.par -switch 2 1 14 1 2 95 5

would change the 14th parest flag to a value of 1 and the 95th age flag to a value of 5.

mfcl ex.frq 01.par 02.par -switch 3 -3 16 1 -4 16 1 -5 16 1

would change the 16th fish flag for fisheries 3, 4, and 5 to a value of 1.

mfcl ex.frq 01.par 02.par -switch 1 -999 36 50

would change the 36th fish flag for all fisheries to a value of 50 and the 95th age flag to a value of 5. And

mfcl ex.frq 01.par 02.par -switch 1 -100000 3 1 -100000 4 1

would change the 1st region flag for regions 3 and 4 to a value of 1.

4.3.3 Flag switches in a file (`-file`)

It is evident that editing more than a few flags with the -switch option would lead to a long command line which can be difficult to enter without error. Therefore there is a provision for preparing switch statements in a text file which MULTIFAN-CL can access like this:

mfcl ex.frq 01.par 02.par -file 02.switch

As with other input files, comments are indicated by "#" and can be sprinkled throughout the file. This is an important feature for documenting the way a fit was run. The file name, 02.switch in the example above, could be anything, but it would behoove the analyst to associate the name with the phase to which it applies.

It is common practice that the command lines to MULTIFAN-CL are entered into scripts, or batch files in which case a special form of the -file option can be used. This is described in the next section.

4.3.4 Batch execution (*doitall* file)

Early stages of any stock assessment can be exploratory in nature with MULTIFAN-CL being run from the command line with -switch statements entered from the keyboard at each phase. However, the fitting procedure is typically rerun many times in search of a reliable and convincing convergence, and it quickly becomes advisable to enter a series of command lines into a batch file which can automatically run all or a portion of the phases of the fitting procedure. These MULTIFAN-CL batch files have commonly been named "doitall" or some derivative thereof. There is an obvious advantage in having a *doitall* file automatically orchestrate a lengthy fit that might take several days to run, but another big advantage is that the sequence of flag settings and edits that structure or "program" the fit is automatically documented in the *doitall* file and can be reproduced, whereas such a sequence entered from the keyboard could be difficult to recall. Furthermore, the *doitall* file can be copied and edited to test the effects of subtle, or major, differences in the way a fit is conducted. Advantageous use of the -file option for lengthy flag editing (see 4.3.3) can be incorporated into a *doitall* file so that the entire orchestration of a fit is documented in one file. Finally, several *doitall* scripts can themselves be called from a higher level script that might, for example, replicate fits with several different starting values for one or more parameters.

The following partial listing of an example *doitall* file is interspersed with commentary. A copy of this or a similar example file can be obtained from <http://www.multifan-cl.org/>.

```
#!/bin/sh
# WCPO YELLOWFIN MULTIFAN-CL ANALYSIS 1962-2001
# ------------------------
# PHASE 0 - create initial par file
# ------------------------
#
if [ ! -f 00.par ]; then
mfcl yft.frq yft.ini 00.par -makepar
fi
```

The first thing to note is that this *doitall* file is a "script" file designed for execution in a bash shell of a LINUX/UNIX type environment,[6] and the purpose of the first line above is to invoke a bash shell. A consequence is that aside from the first line, all text on a line following a hatch (#) is comment material as it is in all text to be input to MULTIFAN-CL. Furthermore, the *doitall* file contains bash shell programming syntax, such as the lines immediately above and below the mfcl command line above. The construct shown here is used throughout the *doitall* file. It tests for presence of a file in the local directory named "00.par" in this case, and in the absence of such a file, it executes the mfcl command. Otherwise it skips to material following the "fi" line. This construct is set up around all phases in the *doitall* file, so that if a fit is interrupted in any way (e.g., computer crash, power failure, or purposeful interrupt), then when the *doitall* file it re-invoked, it will start at about the point where it left off, skipping calls to mfcl for all the *.par* files accumulated to that point.

The creation of the initial *.par* file has traditionally be termed "phase 00" for the purpose of naming the sequence of *.par* files, but the initial *.par* file could be named anything, e.g., "aa.par" if an alphabetical sequence were desired.

```
# ------------------------
```

[6] For running MULTIFAN-CL in a MS-Windows environment it is strongly suggested to use the cygwin command interface, available under GNU General Public License from http://cygwin.com. All example scripts shown here have been found to work with cygwin.

PHASE 1 - initial par

#

if [! -f 01.par]; then

mfcl yft.frq 00.par 01.par -file - <<PHASE1

The above mfcl command illustrates the beginning of a "here file" construct used with the -file option which points in this case to a file name of "-". That is a code for the standard input which normally would be the keyboard, but " << PHASE1" usurps that and declares that standard input shall henceforth come from material in following lines of the *doitall* file itself until such time as an instance of "PHASE1" occurs by itself on a single line (shown two file fragments below). Thus the text between << PHASE1 and PHASE1 is not bash shell material but is material to be read by MULTIFAN-CL. It is a mixture of switch statements (see 4.3.2) and comments. The text PHASE1 in this example could be anything at all as long as it's identical at top and bottom of the switch material. The text can be re-used in subsequent "here" file constructs, but it might be helpful for human readers of the *doitall* file to choose text that is identified with the phase number, as in this example.

As is often the case in the first phase after the makepar phase, the flag editing material in this example is quite long. Discussion here concentrates on the structure of the file, but the commentary in the listing itself deals with the use of particular flags. For more complete documentation on the use and manipulation of flags see 4.5.

2 113 1 # estimate initpop/totpop scaling parameter

-1 49 5 # divide LL LF sample sizes by 5 (default=10)

-2 49 5 # "

-3 49 5 # "

-4 49 5 # "

-5 49 5 # "

-2 50 5 # divide LL WF sample sizes by 5 (default=10)

-3 50 5 # "

-5 50 5 # "

-15 50 1 # divide LL WF sample sizes by 1 (sampling coverage is 100%)

1 32 3 # sets initial estimation with exploitation rate constraint

2 107 1000 # penalty for exploitation rate

2 108 10 # exploitation rate target is 0.1

1 111 4 # sets likelihood function for tags to negative binomial

1 141 3 # sets likelihood function for LF data to normal

2 57 4 # sets no. of recruitments per year to 4

2 69 1 # sets generic movement option (now default)

2 94 2 2 95 20 # initial age structure based on av. Z over 1st 20 periods

-999 26 2 # sets length-dependent selectivity option

-9999 1 1 # sets no. mixing periods for all tag release groups to 1

-999 14 10 # sets maximum incident fmort to 1.0

sets non-decreasing selectivity for longline fisheries

-1 16 1 -2 16 1 -3 16 1 -4 16 1 -5 16 1 -15 16 1

grouping of fisheries

selectivity. catchability tag return reporting rate

-1 24 1 -1 29 1 -1 32 1 -1 34 1

-2 24 1 -2 29 1 -2 32 2 -2 34 2

-3 24 1 -3 29 1 -3 32 3 -3 34 3

-4 24 1 -4 29 1 -4 32 4 -4 34 4

-5 24 1 -5 29 1 -5 32 5 -5 34 5

-6 24 2 -6 29 2 -6 32 6 -6 34 6

-7 24 3 -7 29 3 -7 32 6 -7 34 6

-8 24 4 -8 29 4 -8 32 6 -8 34 6

-9 24 5 -9 29 5 -9 32 7 -9 34 6

-10 24 6 -10 29 6 -10 32 7 -10 34 6

-11 24 7 -11 29 7 -11 32 7 -11 34 6

-12 24 8 -12 29 8 -12 32 8 -12 34 7

-13 24 9 -13 29 9 -13 32 8 -13 34 7

-14 24 10 -14 29 10 -14 32 9 -14 34 8

-15 24 11 -15 29 11 -15 32 10 -15 34 9

Note that one or more switch statements can be placed in a line. They can also appear in any order and can be grouped in logical units as in the group of grouping flag settings above.

#tag-rep. priors tag-rep means effort penalties

-1 35 1 -1 36 50 -1 13 -50

-2 35 1 -2 36 50 -2 13 -50

-3 35 1 -3 36 50 -3 13 -50

-4 35 1 -4 36 50 -4 13 -50

-5 35 1 -5 36 50 -5 13 -50

-6 35 234 -6 36 42 -6 13 -10

-7 35 234 -7 36 42 -7 13 -10

-8 35 234 -8 36 42 -8 13 -10

-9 35 234 -9 36 42 -9 13 -10

-10 35 234 -10 36 42 -10 13 -10

-11 35 234 -11 36 42 -11 13 -10

-12 35 200 -12 36 80 -12 13 1

-13 35 200 -13 36 80 -13 13 1

-14 35 200 -14 36 80 -14 13 1

-15 35 1 -15 36 80 -15 13 -10

sets penalties for catchability deviations

-12 15 1 # No effort for Philippines and Indonesian fisheries so

-13 15 1 # small penalty gives additional ability for catch prediction

-14 15 1 # with minimal impact on population parameters

-15 15 20

-999 33 1 # estimate tag-reporting rates

1 33 90 # maximum tag reporting rate for all fisheries is 0.9

PHASE1

fi

PHASE1 marks the end of switch material for phase 1. Phase 2 illustrates a somewhat different "here" file and switch material construct.

tempfile=‘mktemp mfclXXXXXX‘ # get temp file

trap "rm -f $tempfile" 0 # and clean it when done

PHASE 2

----

if [! -f 02.par]; then

cat - <<PHASE2 >$tempfile

1 149 100 # set penalty on recruitment devs to 100/10

-999 3 17 # all selectivities equal for age class 17 and older

-999 4 4 # possibly not needed

-999 21 4 # possibly not needed

1 189 1 # write *.fit files (obs. and pred. size data)

1 190 1 # write plot.rep

1 1 2500 # set max. number of function evaluations per phase to 2500

1 50 -1 # set convergence criterion to 1E+01

PHASE2

mfcl yft.frq 01.par 02.par -file $tempfile

if [$? -ne 0]; then exit; fi

fi

The first line creates a temporary file name that is guaranteed to be unused. The next two lines then to set up a statement that will remove the temporary file and that will be executed whenever the *doitall* file stops,

for whatever reason. Flag switch material is then copied from a "here" file to the temporary file and fed to MULTIFAN-CL through the normal -file option. The reason for this alternate construct is that the operator interruption of MULTIFAN-CL (see 6.1.2) does not function under LINUX when MULTIFAN-CL is running with standard input directed away from the keyboard as in the construct above, but works fine with this construct. The statement after the mfcl command line prevents the *doitall* file from continuing to subsequent phases if an interrupt has occurred.

A flag manipulation trick worth noting is in the last line of the flag switch material above. The convergence criterion is set so that MULTIFAN-CL will not be too fussy about getting very close to an optimum point of the likelihood surface in this phase of the fit. Such will be the case in subsequent phases as well until the convergence criterion is reset in the final phase (see below) to something small (10^{-4}). This means that MULTIFAN-CL will proceed rapidly through intermediate phases getting approximate solutions and then work hard in the final phase to get a more exact solution.

Most of the remaining phases in this *doitall* file will now be skipped with a few instructive examples retained.

```
# ---------

# PHASE 8

# ---------

if [ ! -f 08.par ]; then

mfcl yft.frq 07.par 08.par -file - <<PHASE8

-999 27 1 # estimate seasonal catchability for all fisheries

-12 27 0 # except those where

-13 27 0 # only annual catches

-14 27 0 # are available

1 14 0 # de-activate K for the time being

PHASE8

fi
```

Though flag switch statements can occur in any order, if a flag is set more than once within the same "here" file, the last setting is the one that prevails. In the example above, fish_flag 27 is set to 1 for all fisheries, but is then set back to 0 for three of the fisheries. Also note that a previously active parameter can be de-activated (von Bertalanffy K in this case). It will keep whatever value it had at the end of the previous phase and not participate in the parameter fitting until such time as it is re-activated as in the next fragment shown of the *doitall* file.

```
# ---------

# PHASE 13

# ---------

if [ ! -f 13.par ]; then

mfcl yft.frq 12.par 13.par -switch 1 1 14 1 # estimate von Bertalanffy K

fi
```

Note in the above example that simple -switch options can be used in the *doitall* file when only one or a few flags are to be set.

```
# ---------
```

```
# PHASE 17
# ---------
if [ ! -f 17.par ]; then
mfcl yft.frq 16.par 17.par -file - <<PHASE17
1 50 4 # set convergence criterion to 1E-04
2 107 0 # remove overall exploitation rate penalty
1 1 1000
PHASE17
fi
mfcl yft.frq 16.par 16.par -switch 1 1 145 3 # calc. hessian matrix
mfcl yft.frq 16.par 16.par -switch 1 1 145 4 # produce SD report
mfcl yft.frq 16.par 16.par -switch 1 1 145 5 # produce correlation report
```

In this final example the convergence criterion is set small so as to achieve a more precise convergence on the optimum of the likelihood surface. Three further mfcl command lines serve to calculate hessian matrix and to use it to generate approximate standard error estimates and a correlation matrix for the parameters.

Depending upon the processor employed and model complexity, the computation of the Hessian and derivatives for dependent variables (parest flag 145 = 3) may be extremely intensive. For example, for a complex tropical tuna model this calculation requires up to 24 hours. A technique has been developed that uses R script to split this calculation into several parts. A related R script takes the generated binary outputs and integrates them into a single Hessian file (see Appendix A.4 – Hessian calculation and derivatives of dependent variables). This utility will approximately halve the processing time for this calculation.

4.3.5 Minimising method

A selection of methods for minimizing the statistical objective function are employed in MULTIFAN-CL, comprising variations on the quasi-Newton approach, with the differences being in the implementation. At each ith minimization step, the calculation has the step vector s_i and the gradients g_{i+1} and g_i. The difference g_{i+1} - g_i indicates information about the Hessian in the s_i direction. The Hessian approximation is updated using this information. For the limited memory Newton approach, the more recent g_i and s_i are saved, and used to update the search direction, instead of calculating a new Hessian approximation at each step. This may improve minimization efficiency, depending upon the specific case of the integrated model fit, such as when the number of independent parameters being estimated is large, because less memory is stored and quicker minimisation steps may be taken. For this approach, the memory held between minimisation steps is specified by defining the number of steps saved, which affects the "precision" of the minimization. Also, a variation of the limited memory Newton was created that employs quad-double precision (16 byte floating point arithmetic) for the update routine, thus saving more steps without running into numerical instability, and also most likely with increased efficiency.

The method is selected using a setting for parest_flags(351), and the number of steps for the limited memory Newton minimiser is specified by parest_flags(192). Generally a large number of steps performs well (around 400). The angle bound adjusts the degree of change to the minimising direction and the default setting of parest_flags(352)=0 is recommended. For most applications having a moderate number of parameters, the quasi-Newton has been found to perform best.

4.4 Projecting the Future

Projections usually involve conducting a simulation over a certain period of time in which the initial population state and other population parameters are obtained from the stock assessment model. The fishing regime for the projection is specified in terms of either catch or effort. Uncertainty in the initial population state, future recruitment and other model parameters may be incorporated into the projection by sampling the parameters from a multi-variate probability distribution (e.g. the posterior distribution in a Bayesian analysis or the variance-covariance matrix in a likelihood-based analysis), and conducting a sufficient number of repeats to characterize the overall variability. The variables that are usually monitored in the projection include population biomass, recruitment, fishery catch (if effort is specified) and fishing mortality (if catch is specified).

We take a similar approach to that described above, but we integrate the projection into the main stock assessment analysis by building the projection period into the *.frq* file. This involves (1) specifying the year and month that define the beginning of the projection period (via the fishery data flags 2 and 3 in the *.frq* file) and (2) specifying the catch or effort for the projection period in the appropriate records for those time periods. If catch is specified in the projection period, then effort is defined as missing using -1. If effort is specified, then catch is defined as missing, again using the -1 convention. Currently, size composition data cannot be specified in the projection – for all fisheries, selectivity is assumed constant over time. (If it were desired to change the selectivity of a fishery in the projection, this might be possible by defining a new fishery.) While it is possible to define different starting points for the projection for different fisheries, we recommend that a common start point be defined for all fisheries. This makes it easier to "insulate" the likelihood function from the projection period.

There are a number of reasons why we have preferred this integrated approach:

• The complete analysis (stock assessment plus projection) is fully specified by the set of standard data and .par files and can be undertaken in a standard model run.

• The variance of variables of interest, such as population biomass, is computed for the projection period automatically when the variance report for the analysis is generated.

• Recruitment and effort deviations during the projection period are legitimate sources of process variation that should be incorporated into estimates of confidence intervals for projection variables. While the point estimates of these variables for the projection period will tend to zero (the mean of their prior distributions), their variance will be recognized and will contribute to the variance of projected biomass and other dependent variables.

The way in which the projections have been structured in the model code ensures that the results of the projections do not affect the overall objective function. Thus, once a converged fit has been obtained, a range of projection specifications can be tested effectively with only one function evaluation for each specification.

Annual recruitments assumed for **deterministic projections** can be selected according to the following three options.

1. Average recruitment determined from the Beverton-Holt stock-recruitment relationship (SRR). This is the default option, with age flags 190, 191, and 195 = 0. The SRR is calculated over either the full model estimation period, or for the period specified by age flags 199 and 200. Projections having future recruitments predicted from the "annualised" SRR (age flag 182, see 3.8.2) must be adjusted such that the annual recruitment is correctly assigned to each time period (quarter or season) within each future calendar year. In this case, the average seasonal recruitment distribution is calculated, being the average estimated recruitment among seasons and regions, based upon the estimated recruitments. This is used to allocate the SRR predictions in the projection years; thus maintaining the seasonal recruitment pattern in the future. This option is activated by age flag 183 = 1.

2. Use the average of the absolute annual recruitments over a specified historical period. Age flags 190 and 191 denote the start and end points of this period prior to the final year of the model analysis period - not including the projection time periods. For example the model analysis period may be 1952.125 to 2009.875 = 232 time periods, and a partial period for deriving average recruitment may be specified from 1981 to 2009, and so age_flag(190) = 116, and age_flag(191) = 0. The age flag 195 must be set to 0.

3. Average recruitment determined from the SRR but the predicted recruitment is apportioned among regions according to the regional recruitment distribution over a specified historical period. This is activated by age flags 195 = 1, with age flags 190 and 191 being set as described for option 2 above, i.e. to define the period used for the average proportions among regions. The SRR is either calculated over the full model estimation period, or for the period specified by age flags 199 and 200.

Table 4.2: Age flags used for projections

190	**191**	**195**	**183**	**Recruitment for projections**
0	0	0	0	SRR predictions based upon the fit to estimated recruitments for all, or part, of the model estimation period. (For part of the period being specified by age flags 199 and 200.
0	0	0	1	Annualised SRR predictions with allocation in each season based upon the average estimated seasonal recruitment proportions.
>0	>=0	0	0	Use average recruitment from the absolute estimates for the period defined by age flags 190 and 191
>0	>=0	1	0	Use SRR predictions (all or part of the estimation period) but with a regional distribution according to the average distribution of the absolute recruitments for the period defined by age flags 190 and 191.

For projection recruitment options 1, 2 and 3 above (that employ the SRR predictions), a bias correction may be applied to the predictions of the SRR to approximate the mean of the estimated recruitment distribution in normal space (see section 3.8.2). This correction is activated by setting age_flag(161) to 1 (see 4.5.17).

4.4.1 Stochastic projections

The above description is for a single model evaluation in a deterministic projection, with parameter values held constant during the projection period. MULTIFAN-CL can be run in simulations with three sources of stochasticity: annual recruitments, population numbers at age in the first projection year, and fishery effort deviations. This feature has utility for risk analyses and evaluating relative performance of assumed future fishery management strategies. It entails five steps:

1. Hessian calculation for the fitted model parameters over the model estimation period (e.g. 1952 – 2008).

2. Construction of "*.frq" and "proj.par" files for input to deterministic projections.

3. Hessian option 7: calculation of derivatives for projection dependent variables

4. Hessian option 8: generate time series of stochastic recruitments

5. Simulation run.

The first step has been described previously (see 4.5.1) and the deterministic projection in the second step has been described above. It is recommended to test the newly constructed projection input par file operation with a single model evaluation to ensure that it works, e.g.:

```
./mfclo32 bet.frq proj.par ttt -switch 6 1 1 1  2 190 0 2 191 0 2 148 4 2 155 0 -999 55 1
```

The third and fourth steps entail Hessian calculations to generate the stochastic inputs for the parameters. For the third step, age flag(20) must be set to the number of simulations, and parest flag(145) is set to 7 and 8 for steps 3 and 4 respectively. An example follows for 200 simulations:

```
./mfclo32 bet.frq proj.par ttt -switch 2 1 145 7 2 20 200
./mfclo32 bet.frq proj.par ttt -switch 5 -999 55 0 1 145 8 1 234 1 1 235 20 1 237 0
```

Note that the input *.par* file "*proj.par*", must specify age flag(20) to the number of simulations, (in this example 200). Also note that if the flag setting for stochasticity in the numbers-at-age in the first projection year (parest flag 237) is not set, therefore this source of stochasticity will *not* occur in simulation projections in the fifth step, even if the flag settings are enabled. To add this source of stochasticity, it is necessary to enable it in the fourth step.

In the fourth and fifth steps it is necessary to supply the appropriate flags to specify the sources of stochasticity.

1. Recruitments

   Future recruitments are randomly resampled (multinomial) from historical estimates for each simulation year. The period of the historical time series from which future stochastic recruitments are generated is specified according to parest flags(232 and 233). The default setting for both equal to 0 defines the entire period of the time series. Specified values for: parest flag(232) is the first time period; and parest flag(233) is the last time period. Note that parest flag(238) must be set to 0 to allow placement of stochastic recruitments. Age flag(20) must be > 0 to initiate the simulations and set equal to the number of simulations desired.

   Future stochastic recruitments may also be applied as deviates to the predictions of the stock-recruitment relationship in each projection run. This is activated by parest_flags(239). If the stock-recruitment relationship has been estimated over a subset of the model time period using age_flags(199, 200), then these flag values must correspond to those of parest flags(232 and 233).

   When the stochastic recruitments are derived from the "annualised" SRR predictions (age flag 182, see 3.8.2) plus the random deviate, by default this is allocated to each time period (quarter or season) within each future calendar year according to the average seasonal recruitment distribution. Thus maintaining the seasonal recruitment pattern in the future.

2. Numbers at age in 1st year of projections **STILL IN DEVELOPMENT**

   Random selection of the population state numbers-at-age in the first year of the projection period is taken from a parametric distribution (Choleski decomposition) and is activated by setting parest flag(237) to 1. If parest flag(231) is set to 0 a pre-defined seed for the random number generator is used, else the flag value supplied will be used.

3. Fishery effort deviations **STILL IN DEVELOPMENT**

   Random selection of fishery effort deviations for each year of the projection period is undertaken in veff_dev.cpp (ln 23) where deviates are sampled from a parametric distribution with a user-supplied standard deviation. Parest_flag 234 turns on the random effort devs, flag 235 specifies the standard deviation, and flag 236 provides an iseed for the random deviate (if desired).

In addition to undertaking projections with specified future fishing effort or catch, it may also be intended to undertake simulation projections that extend the model under no fishing effort or catch. This might be necessary for examining unfished historical and future biomass for the purpose of evaluating stock status relative to an "unfished" reference point. In this case it is necessary to undertake the third and fourth steps with fish_flag(55) = 1 for all fisheries, so that the inputs are generated for running both positive and zero fishing simulation projections. Although this can be applied in the -switch settings, it is easier to apply this flag setting in the input *proj.par* file. With this setting, the following files are generated:

- *simulated_numbers_at_age*
- *simulated_numbers_at_age_noeff*
- *simyears*

The first two files contain the numbers-at-age in the first projection year and the stochastic recruitments for each simulation are generated, while the third file contains the random indices of the recruitment deviates for the fitted Beverton-Holt stock-recruitment relationship. All three files are necessary inputs for the fifth step.

The fifth step is to undertake the simulation projections. For the simulations parest flag(1) is set = 1 since the simulations relate to a single model set of parameters. An example follows for the fifth step that runs 200 simulations and activates all three sources of stochasticity, and undertakes a zero-fishing model evaluation in each simulation:

```
./mfclo64 bet.frq proj.par ttt -switch 7 1 1 1 -999 55 1 2 20 200 1 145 0 1 234 1 1 235 20 1 237 1
```

The simulation run output is contained in the files:

- "projected_numbers_at_age";
- "projected_numbers_at_age_noeff";
- "projected_randomized_catch_at_age";
- "projected_randomized_catches";
- "projected_fmort_at_age_year";
- "projected_spawning_biomass";
- "projected_spawning_biomass_noeff";
- "Fmults.txt";
- "other_projected_stuff".

If particular performance indicators relating to projection model status in respect of FMSY or the unfished spawning biomass are required, the reports "Fmults.txt" and "projected_spawning_biomass_noeff" are produced. Following the yield calculation for each projection simulation, the fishing mortality multiplier (Fmult) and several other MSY-related quantities are reported to "Fmults.txt".

The simulation file output is controlled by the same flags that control the standard reports [4.5.1], and generally parest_flags(190) and parest_flags(186) should be set to 1.

4.4.2 Generating pseudo-observations

To implement full simulation capability in MULTIFAN-CL, a simulation mode is possible such that an "operating model" (OM) can produce pseudo-observed data with specified observation and process error for both estimation and projection time periods as might be used in management strategy evaluations (MSE).

The general algorithm for producing predictions for the projection period follows. Size frequency observations have been used by example.

- As normal, the analyst supplies catch or effort in the fishery data for the projection period.
- Analyst populates the length and/or weight frequency columns of the fisheries data section in the *.frq* file with non-zero "pseudo" frequencies in the fisheries data during the projection period for the fisheries for which simulation data are to be generated. The total of the frequencies is the desired effective sample size for the pseudo-observations.
- Analyst sets parest_flags(241) = 1 that activates the feature for generating simulation data from projections

Essentially the algorithm is activated by parest_flags(241) with the data to be generated being specified by entries made in the projection section of the fisheries data in the *.frq* file. Should pseudo-observations be required for the estimation model time periods, this is activated by parest_flags(242).

Size composition data

The entries made in the projection section of the fisheries data in the *.frq* file act simply as "switches" for activating the generation of pseudo-observations for that fishery incident. The entries themselves in each size class interval can be any "dummy" value. Observation error is applied to the predictions with the random number seed being supplied via the command line option arguments: "-length_seed" and "-weight_seed", followed by the seed number for each. Alternatively a default seed value will be used. Multinomial error is applied to the predictions with the multinomial sample size being taken from the sum of the "dummy" entries supplied as frequencies in the size composition data columns of the fishery data in the *.frq* file for the projection periods. This is where the analyst needs to be careful in specifying the data to be produced as total of the entries for each fishing incident defines the level of Multinomial observation error.

Relative abundance indices

The general method for fitting to standardized indices of relative abundance derived from catch-per-unit-effort (CPUE) in MULTIFAN-CL is to express the indices as standardized effort, assume constant catchability, and apply high penalty to the total catch likelihood. The generation of pseudo-observations from simulations for the projection period will depend upon the projected "observed" fishery data supplied by the analyst; i.e. projected catch or effort. As such there are two algorithms for generating the pseudo-observations.

1. Projected "observed" catch supplied

In this case the analyst supplies "observed" catches in the fishery data for the projection period, and "-1" for the effort. As no "observed" effort is supplied, a Newton-Raphson solution for the predicted fishing mortality directly provides the predicted catch. The model estimates no predicted effort. To generate a predicted effort, a new formulation was developed using the N-R solution for the fishing mortality, the catchability in the final year of the estimation period, together with the average observed effort input the estimation period (in true units):

$$\tilde{E}_y = (\frac{F_y}{q_{y=term}})\bar{E} \tag{4.1}$$

In MULTIFAN-CL the normalization of effort applies only to effort for the estimation period and is used in calculating the average effort used in $\bar{E}$. This calculation takes correct account for the case of grouped fisheries. This method ensures the generation of the predicted effort in the same units as the input effort for the estimation period.

The random number seed is supplied from the command line option argument "-effort_seed" followed by the number. Alternatively a default seed value will be used. Lognormal error is applied to the predictions with the standard deviation taken from the analysts value specified in age_flags(186). If no value is specified, i.e. age_flags(186)=0, then no observation error is applied to the prediction.

2. Projected “observed” effort supplied In this case the analyst supplies "observed" effort in the fishery data for the projection period, and "-1" for the catch. The implementation in MULTIFAN-CL for fishing mortality uses the "observed" effort directly with the terminal catchability (ignoring age-specific effects for now).

$$F_y = q_{y=term} E_y \tag{4.2}$$

And the predicted catch is,

$$C_y = (1 - e^{F_y}) N_y \tag{4.3}$$

Therefore, the relative abundance variability is inherent in the predicted catch given that effort is "fixed" by the analyst. Observation error is therefore applied to the predicted catch. The random number seed is supplied from the command line option argument "-catch_seed" followed by the number. Alternatively a default seed value will be used. Lognormal error is applied to the predictions with the standard deviation taken from the analysts value specified in age_flags(187). If no value is specified, i.e. age_flags(187)=0, then no observation error is applied to the prediction.

Tagging data

Tagging data provide an influential observation type for the integrated model fit in respect of estimating absolute abundance and exploitation levels, and are therefore important in simulation analyses using operating models (OM). The feature for generating simulation data includes producing pseudo-observed tagging data for both the projection and estimation model time periods. Broadly, the algorithm for generating simulation tag recapture pseudo-observations is as follows:

- Assign the release sample to an initial tag release cohort of numbers at age; this is based upon the predicted catch-at-age for the fishing incident associated with the simulation release event.
- Perform catch calculations upon the simulation tagged population. These calculations are essentially similar to the operations done for tag groups within the estimation period. This generates the predicted tag recaptures for the simulation tag release groups.

The simulation tagging data feature entails two components:

- Simulated pseudo-observations for real tagging events that occurred during the estimation model time periods; denoted *sim_realtag*
- Simulated pseudo-observations for virtual tagging events to occur during the projection model time periods; denoted *sim_tag*

As for the other data types described above, the generation of tagging data is activated by parest_flags(241), and in the case of *sim_tag* the random number seed is supplied from the command line option argument "-tag_seed" followed by the number. In the case of *sim_realtag*, the generation of simulation tagging data for real tagging events is activated by parest_flags(242), and the random number seed is supplied from an input file *simseed* that contains a single integer value. With each simulation the value is incremented, and restoring it to the original value enables one to replicate exactly the variability in the *sim_realtag* simulation data.

For the *sim_realtag*, the tag release samples are as occurred during the real tagging events that were used for the estimation model fit; while for the *sim_tag* the tag release samples are virtual samples and are specified by the user. Regarding the estimation time periods, the structure of the historical data was used as it occurred, i.e. the same tag release numbers by size. For the *sim_realtag* case, while the method for calculating the tag releases at age was different, the algorithm for generating predicted recaptures at length was similar to that of the *sim_tag* case.

In MULTIFAN-CL observations from real tagging events are fitted in respect of age, and not in respect of their length at release. However, the observations are input in respect of their lengths. Therefore, the ages and lengths are decoupled and so it is not a simple matter of reporting the expected values for the model tag recaptures, since these are indexed only in respect of age. For projection *sim_tag* data this is managed by a conversion of ages to lengths. While the *sim_realtag* algorithm is somewhat analogous to that used for the projection tagging events, it differs in: retaining a link to the observed release length of each simulated tag recapture; and, accommodates the recaptures made for events released close to the end of the estimation period which are predicted to occur into the projection time periods.

This algorithm has an exception for the case of pooled tags, i.e. where tagged fish may be aggregated into a pooled tag group after a user-specified period at liberty. This has utility when fitting to tagging data in reducing the amount of calculations required. However, for simulating tagging data, the aim is to generate pseudo-observations that resemble real observations. Therefore, simulated data for a pooled tag group would be inconsistent because these do not exist in the real observations (where all observed recaptures relate to a specific release event). A sanity check was therefore added such that in the special case of simulating tagging data, age_flags(96) must be set very high to avoid simulation tagged fish from entering the pooled tag group.

A key aspect of this feature is to include error in respect of the fishery-specific reporting rates. For estimation period tagging events (*sim_realtag*) the estimated reporting rates are already available, while for the projection events (*sim_tag*) the reporting rates must be assumed. In both cases, it is implemented by rolling into the simulation multinomial probability of a tag recapture (so that an extra binomial simulation is not needed). In the *sim_tag* case a matrix of simulation reporting rates specific to fishery and tag-event in the projections is specified at the bottom of the input file **.tag_sim*. This enables total flexibility in the reporting rates to be assumed in the simulation projections.

OM simulations may be computationally intensive, particularly when implemented as part of management strategy evaluations that use a range of OMs, harvest strategies and over a large number of simulations. Opportunities to optimise these processes are possible in respect of the OM simulation procedures and the reports being generated. Preceding the simulation loop, the OM calculation can be optimized to exclude the initial function evaluation and derivative computations, and to simply undertake the dynamic model calculations by setting parest_flags(353). These calculations will otherwise be done by default during an feval=0 as part of the minimisation procedure that precedes the stochastic simulation loop. This is of particular benefit for MSE frameworks where each evaluation entails a single simulation that includes the preceding feval=0 run. Note that within the simulation algorithm these calculations are also not performed (is not called within the minimising_routine and gradient_switch=0).

By default, reports of simulation model quantities are generated (*other_projected_stuff*, *projected_randomized_catches*, *projected_randomized_catch_at_age*), which over a large number of simulations may comprise a large amount of data. When simulations are run on remote hosts, this can be prohibitive for data transfer over the internet or within networks. A parest_flags(191) was assigned to

activate these detailed reports to be produced, otherwise the default is for no reports to be produced. This flag is compatible with the operation of the other report control parest_flags(186-190).

Output files

Pseudo-observations are reported in output files as follows:

- size composition - "test_ lw_sim"
- effort - "effort_sim"
- catch - "catch_sim"
- tagging - "report.simtag_#" and "report.realtag_#"

For the size composition, catch and effort data, each fishing incident indexed by the fishery, true model projection year (e.g. 2019), month and with the random number seed at the beginning of each simulation data block (to enable repeatability among simulations). The simulated tagging data for real tagging events during the estimation period is reported in the *report.realtag_#* file, and for projection tagging events in the *report.simtag_#* file; where _# is the simulation number. The format of both output files reflects that of the input **.tag* file containing observed tagging data ([4.1.4]), which facilitates ready construction of input files within an MSE framework.

4.5 Flag Settings

The lists below give information on the various types and meanings of flags that are manipulated in normal practice. Many flags are not used and are therefore available for future enhancements of MULTIFAN-CL. A complete list of all the flags is given in Appendix B, including those that are unused, obsolete, or suspected to be obsolete. Many examples of their use are given in Section 4.3.4.

Some flags are used to indicate grouping of fisheries for the purpose of defining shared parameters or for grouping observed and predicted data in the likelihood functions. Grouping flags work by assigning an integer to each item (fishery or tag group or ???). The numbers are simply group labels. Thus items to be grouped together are assigned the same number. The assigned numbers should form a continuous set of integers from 1 to a maximum with no gaps. That restriction might be unnecessary, but until this has been investigated it would be better to adhere to it.

4.5.1 Program Control Flags

Table 4.3: Program control flags

Flag	Value	Use
parest_flag(1)	n	quit after n number of function evaluations
parest_flag(50)	n	quit when max. gradient is less than 10^n (n can be negative)
age_flag(92)	0	catch errors version (default)
	1	meaningless
	2	catch calculated by Baranov catch equation
	3	catch calculated by linearized catch equation (as in Stock Synthesis 2)
parest_flag(41)	1	do total weight fit only
parest_flag(32)		Sets initial control

Table 4.3: Program control flags

Flag	Value	Use
	1	old standard
	2	a slightly faster initial control sequence
	3	constrain fit with overall exploitation rate target. Penalty and target given by age flag107 and 108 respectively (page 58)
	4	Yukio control sequence??
	5	Takeuchi-san control sequence??
	6	a control sequence that does not estimate growth curve parameters, apart from the variance and variance trend
	7	A control sequence that does not estimate *any* growth curve parameters, including variance and variance trend
age_flag(32)	0	totpop fixed
	1	totpop estimated (check if this is turned off in phase 1!)
parest_flag(145)	0	normal estimation
	1	compute hessian
	2	compute derivatives for dependent variables
	3	both hessian and derivatives for dependent variables
	4	st. dev. report (*.var*)
	5	correlation report (*.cor*)
	6	write inverse hessian (*.hesinv*)
	7	compute derivatives for dependent vars for projections
	8	covariance report for projections
	9	SVD diagnostic report
parest_flag(351)	0	quasi-Newton minimiser
	1	limited memory Newton minimiser
	2	limited memory Newton minimiser with double precision
parest_flag(192)	n	keep n terms in the limited memory Newton minimiser, default (0)= 7
parest_flag(352)	n	angle bound = $n/100$ for limited memory Newton minimiser
parest_flag(146)	1	scales the gradient for totpop
parest_flag(152)	1	re-scales the gradients to 1
parest_flag(189)	1	write *.fit files
parest_flag(190)	1	Enables flag settings to write to output files controlled by parest_flag(186 to 188). When set = 0 output is reduced.
parest_flag(186)	1	When parest_flag(190)=1, writes to files *fishmort* and *plotq0.rep*
parest_flag(187)	1	When parest_flag(190)=1, writes to *temporary_tag_rep* file.
parest_flag(188)	1	When parest_flag(190)=1, writes to *ests.rep* and *plot.rep*
parest_flag(191)	1	Generate additional simulation data reports.
parest_flag(195)	1	Insert check_numbers in *.par* file
parest_flag(197)	1	create new input par file (obsolete)

Output control

It may not always be efficient to produce the full output from a MULTIFAN-CL model as this requires more processing time and entails producing what are sometimes large files. For example when multiple model projections are undertaken, it is efficient to reduce the output from a single model evaluation. This output can be controlled using parest_flags(186 to 191).

parest_flag (190) = 0 : generates par file, turns off output of *plot.rep*, *plotq0.rep*, *ests.rep*, *temp_tag_report*, *fishmort* files and other report files.

parest_flag (190) = 1 : turns on output of report files subject to options specified by parest_flag(186 to 189) as follows:

- parest_flag (188) = 0 : turns off *ests.rep*, *tag.rep*, **.fit*
  - parest_flag(189) = 0 : turns off output of the **.fit* files
- parest_flag (187) = 0 : turns off *temporary_tag_report*
- parest_flag (186) = 0 : turns off *fishmort* and *plotq0.rep*

This facilitates output according to three options:

- Full output: 1 189 1 1 190 1 1 188 1 1 187 1 1 186 1
- Streamlined option: 1 189 0 1 190 1 1 188 0 1 187 0 1 186 0
- Silent option: 1 189 0 1 190 0 1 188 0 1 187 0 1 186 0

Note that output of the **.fit* files requires parest_flags(190, 188, 189) to be set to 1.

The simulation output is also controlled by these flags. For output of simulations with specified projection fishing mortality parest_flag (190) is set to 1, and for output of simulations with zero projection fishing mortality parest_flag (190) and parest_flag (186) are set to 1. To produce additional simulation report files parest_flag (191) is set to 1 [4.4.2].

4.5.2 Catch Estimation

Table 4.4: Key flag settings used with catch data

Flag	Value	Use
age_flag(115)	n	SS2-type catch estimation
age_flag(116)	n	Maximum fishing mortality is $n/100$. Default mortality is 5.0 when $n = 0$.
age_flag(144)	n	common weight for catch likelihood — automatically set to 10000
fish_flags(i,45)	n	fishery-specific weights instead of af(144). If used, need to set for all fisheries to the desired value.

4.5.3 Length/weight Frequency Data

Table 4.5: Key flag settings used with length/weight frequency data

Flag	Value	Use
parest_flag(141)		LF likelihood function
parest_flag(139)		WF likelihood function

Table 4.5: Key flag settings used with length/weight frequency data

Flag	Value	Use
	0	modified minimum χ^2 with factor of 2 in variance and proper distribution
	1	modified minimum χ^2
	2	modified minimum χ^2 with factor of 2 in variance
	3	full normal distribution including constants
	4	Dirichlet multinomial fit
	5	Dirichlet multinomial mixture fit
	6	Option is not active
	7	Option is not active
	8	Normal with student-t probability distribution
	9	Self-scaling Multinomial with random effects
	10	Self-scaling Multinomial without random effects
	11	Dirichlet Multinomial without random effects
fish_flags(i,49)	0	effective length frequency sample size = actual sample size/10
	n	effective length frequency sample size = actual sample size/n
fish_flags(i,50)	0	effective weight frequency sample size = actual sample size/10
	n	effective weight frequency sample size = actual sample size/n
parest_flag(193)	n	Constant for normal distribution related to number of size class intervals, I: default is $1/I$; else $(n/100)/I$
parest_flag(350)	1	activates the use of the predicted sex-ratios in catches when aggregating predicted size compositions of multi-sex/species catches
parest_flag(310)	1	When parest_flag(141)=8, activates estimation of tot_exp parameter for student-t distribution for length data
parest_flag(300)	1	When parest_flag(139)=8, activates estimation of tot_exp parameter for student-t distribution for weight data
parest_flag(290)	1	When parest_flag(141)=8, activates estimation of log(variance) parameter for student-t distribution for length data
parest_flag(280)	1	When parest_flag(139)=8, activates estimation of log(variance) parameter for student-t distribution for weight data
parest_flag(292)	1	When parest_flag(141)=8, activates estimation of log(degrees of freedom) parameter for student-t distribution for length data
parest_flag(282)	1	When parest_flag(139)=8, activates estimation of log(degrees of freedom) parameter for student-t distribution for weight data
parest_flag(311)	1	Activates tail compression of length frequency data
parest_flag(301)	1	Activates tail compression of weight frequency data
parest_flag(313)	n	When parest_flag(311)=1, specifies the proportion in *each* tail of the length frequency samples = $n/100$
parest_flag(303)	n	When parest_flag(301)=1, specifies the proportion in *each* tail of the weight frequency samples = $n/100$
parest_flag(312)	n	Applies a minimum length frequency sample size = n (only active if parest_flag(311)=1)
parest_flag(302)	n	Applies a minimum weight frequency sample size = n (only active if parest_flag(301)=1)
fish_flags(i,82)	1	Activates estimation of SSM-RE log-length variance
fish_flags(i,84)	1	Activates estimation of SSM-RE log-weight variance
parest_flag(316)	n	Specifies the bounds (=$n/100$) for SSM-RE log-length and log-weight variance
fish_flags(i,78)	1	Activates estimation of SSM-RE length rho correlation

Table 4.5: Key flag settings used with length/weight frequency data

Flag	Value	Use
fish_flags(i,80)	1	Activates estimation of SSM-RE weight rho correlation
parest_flag(315)	n	Specifies the bound (=n/100) for SSM-RE length rho correlation
parest_flag(370)	n	Specifies the bound (=n/100) for SSM-RE weight rho correlation
fish_flags(i,67)	1	Activates estimation of SSM-RE log-length N variance
fish_flags(i,76)	1	Activates estimation of SSM-RE log-weight N variance
fish_flags(i,85)	1	Activates estimation of SSM-RE length sample size covariate exponent
fish_flags(i,86)	1	Activates estimation of SSM-RE weight sample size covariate exponent
parest_flag(317)	n	Specifies the upper bounds (=n/100) for SSM-RE length and weight sample size covariate exponent
parest_flag(334)	n	Specifies the upper bound (=n) for SSM-RE log-length N variance
parest_flag(335)	n	Specifies the upper bound (=n) for SSM-RE log-weight N variance
fish_flags(i,68)		Grouping of fisheries for: SSM-RE - log-length variance, length rho, log-length N variance; and, DM_noRE - length sample exponents
fish_flags(i,77)		Grouping of fisheries for: SSM-RE - log-weight variance, weight rho, log-weight N variance; ; and, DM_noRE - weight sample exponents
parest_flag(320)	n	Activates tail compression for SSM-RE length samples; n is the minimum number of class intervals
parest_flag(330)	n	Activates tail compression for SSM-RE weight samples; n is the minimum number of class intervals
parest_flag(336)	n	Specifies upper bound (=n) on SSM-RE length effective sample size, and assigns penalty weight to tail compressed samples;
parest_flag(337)	n	Specifies upper bound (=n) on SSM-RE weight effective sample size, and assigns penalty weight to tail compressed samples;
parest_flag(338)	n	defines the SSM-RE M-estimator coefficients, n=3 is recommended
fish_flags(i,69)	1	Activates estimation of DM_noRE exponent for the scalar of length frequency sample size multiplier
fish_flags(i,87)	1	Activates estimation of DM_noRE exponent for the scalar of weight frequency sample size multiplier
sample size covariate exponent		
fish_flags(i,89)	1	Activates estimation of DM_noRE length frequency relative sample size covariate exponent
fish_flags(i,88)	1	Activates estimation of DM_noRE weight frequency relative sample size covariate exponent
parest_flag(342)	n	If $n > 0$ specifies upper bound (=n) on DM_noRE assumed maximum for length and weight effective sample sizes, else if $n =$ 0 the default maximum = 1000

4.5.4 Growth Parameter Estimation

Table 4.6: Key flag settings used in growth parameter estimation

Flag	Value	Use
parest_flag(12)	1	estimate mean length of first age class; the default value is set to 1 during the control phases
parest_flag(13)	1	estimate mean length of last age class; the default value is set to 1 during the control phases
parest_flag(14)	1	estimate K
parest_flag(15)	1	estimate "generic" standard deviation of length-at-age
parest_flag(16)	1	estimate length-dependent standard deviation
parest_flag(157)	1	estimate density-dependent growth parameter
parest_flag(168)	1	re-scales age in the growth function
parest_flag(169)	1	makes the re-scaling a function of length
parest_flag(171)	1	puts a power term on age
parest_flag(173)	n	ages 2:n have independent lengths
parest_flag(182)	n	penalty wt. for length estimation is $n/10$
parest_flag(184)	n	activate estimation of independent lengths of age classes specified by parest_flag(173).
parest_flag(226)	1	apply "Richards curve" growth parameter $x > 0$
	$\neq$0,1	apply $x < 0$
parest_flag(227)	1	activate estimation of "Richards curve" growth parameter
parest_flag(21)	n	seasonal growth – under construction
fish_flags(i,11)	1	estimate selectivity bias in first age class; never been tested
fish_flags(i,22)		grouping flag for common selectivity bias; never been tested

4.5.5 Maturity at Age Estimation

Table 4.7: Key flag settings used in maturity at age estimation

Flag	Value	Use
age_flag(188)	1	convert maturity at length to age using simple spline for mean length at age
	2	convert maturity at length to age using weighted spline for mean length at age

4.5.6 Catchability Estimation

Table 4.8: Key flag settings used in the estimation of catchability

Flag	Value	Use
age_flag(57)	n	recruitments per year – also a month doubling kludge factor needed for calculating correct penalties in random walk catchability changes (equivalent to age_flag(93)?)
fish_flags(i,1)	1	estimate average catchability
fish_flags(i,10)	1	estimate time-series changes in catchability
fish_flags(i,15)	n	penalty weight for catchability deviation prior is n (default = 50)

Table 4.8: Key flag settings used in the estimation of catchability

Flag	Value	Use
fish_flags(i,23)	n	only have a catchability deviation if the number of months since the last deviation is $> n$
fish_flags(i,27)	1	estimate seasonal catchability (trig version)
fish_flags(i,47)	n	$n > 0$ estimate seasonal catchability (explicit parameter version), n = no. of parameters (max. of 12)
fish_flags(i,28)		grouping flags for common seasonal catchability
fish_flags(i,29)		grouping flags for common catchability deviations throughout time series
fish_flags(i,60)		grouping flags for common catchability at start of time series
fish_flags(i,39)	n	$n/100$ is variance for Kalman filter catchability deviations (default = 1, sd = 0.1)
fish_flags(i,51)	1	estimate fish pars(7), a parameter for the effect of effort level on catchability
fish_flags(i,52)		grouping flags for common effect of effort on catchability
age_flag(156)	1	use fish pars(7) in the model.
fish_flags(i,66)	1	flags for variable q in implicit method
age_flag(104)	1	enables catchability devs for implicit q
fish_flags(i,53)	1	estimate fish pars(8), a parameter for the effect of abundance on catchability
fish_flags(i,54)		grouping flags for common effect of abundance on catchability
age_flag(125)	1	use fish pars(8) in the model
age_flag(126)	1	Activate the estimation of region_pars(2) to stabilise the initial population calculation when age_flag(125) = 1
age_flag(127)	n	n is the weight of the region_pars(2) estimation penalty when age_flag(125) = 1, default = 1000

4.5.7 Selectivity Estimation

Table 4.9: Key flag settings used in the estimation of selectivity

Flag	Value	Use
fish_flags(i,48)	n	$n > 0$? turns on selectivity estimation (on by default)
fish_flags(i,3)	n	first age class for common terminal selectivity; default: one less than no. age classes
fish_flags(i,16)	n	n=1 selectivity is non-decreasing with age; n=2 selectivity is zero for ages given by fish_flags(i,3) and older
fish_flags(i,19)	n	if $n > 0$, sets num ages for sel dev coffs
fish_flags(i,20)	n	penalty weight = $n/10$ for estimating sel_ dev_ coffs; default value is 1; note: active if fish_flags(i,19) > 0
parest_flags(323)	n	n=0 activates estimation of age-specific selectivity deviate coefficients (sel_dev_coffs), note: fish_flags(i,19) must be > 0
fish_flags(i,21)		obsolete, kaput, unused, inactive, idle
fish_flags(i,24)		grouping fleets for common selectivity (N.B.: selectivity features set by other fish flagslisted here must be identical within groups)
fish_flags(i,26)	1	uses age-specific selectivities that are consistent by length
	2	uses age-specific selectivities as above, with sd of length-at-age accounted for (preferable)

Table 4.9: Key flag settings used in the estimation of selectivity

Flag	Value	Use
	3	uses length-specific selectivity
fish_flags(i,41)		penalty weight for second difference smoothing
fish_flags(i,42)		penalty weight for third difference smoothing
fish_flags(i,56)		penalty weight for making selectivity a non-decreasing function of age; default value is 10-6, active if fish_flags(i,16)=1
fish_flags(i,57)	0	no functional form for selectivity (default)
	1	logistic selectivity curve
	2	double normal selectivity curve
	3	cubic spline selectivity curve, or length-specific selectivity
fish_flags(i,61)	n	n is number of nodes for cubic spline selectivity curve if n > 0
	-1	logistic length-specific selectivity curve
	-2	double normal length-specific selectivity curve
fish_flags(i,62)	n	add n nodes to cubic spline selectivity curve
age_flags(36)	n	Maximum bound for selectivity deviates
fish_flags(i,71)	n	n is number of total number of breaks in the time period of fishery i, a single time-block is specified by $n = 0$, the default.
fish_flags(i,74)	n	n is the number of seasonal selectivity patterns within a year, the default value is 1, and for $n > 1$ the default value is 4.
parest_flags(74)	n	$n > 0$ activates and specifies the weight for a generic selectivity penalty
fish_flags(i,72)	n	$n > 0$ activates and specifies the weight for a generic selectivity penalty specific to fisheries.
fish_flags(i,75)	n	n is number of age classes starting from 1 for which selectivity is assumed to be zero. $n = 0$, is the default.
age_flags(193)	n	n=1 activates estimation of length-specific selectivity shared among sexes

4.5.8 Effort Deviation

Table 4.10: Key flag settings used with effort deviates

Flag	Value	Use
age_flag(34)	1	turns on effort dev. estimation; Note: fish_flags(i,4) must also be set > 1
age_flag(35)	n	n is the maximum bound on effort dev. estimation
fish_flags(i,4)	n	if $n > 0$, applies effort dev. to fishing mortality; if $n > 1$, turns on effort dev. estimation, note: age_flag(34) must also be set = 1
fish_flags(i,13)	n	penalty weight for effort devs prior (default=10). If n is negative, the weights are scaled by the square root of the effort.
fish_flags(i,38)	n	$n/100$ is variance for Kalman filter effort deviations (default = 1, sd = 0.1)
fish_flags(i,65)	1	fat-tailed Cauchy distribution for effort devs
fish_flags(i,66)	1	use the time varying CPUE weights in column 7 of the version 6+ frq file
age_flag(160)	n	penalty weight for fitting to effort in catch conditioned model

fish_flags(i,79)	grouping of fisheries for shared effort deviations in a multi-sex model

4.5.9 Tagging Data

Table 4.11: Key flag settings used with tagging data

Flag	Value	Use
parest_flag(111)	0	least squares likelihood function for tagging data
	1	robust least squares likelihood function for tagging data
	2	Poisson likelihood function for tagging data
	3	negative binomial likelihood function for tagging data. (fish_pars(4,fi) are the variance-determining parameters
	4	negative binomial (recommended) with fish_pars(4,fi) as the variance-determining parameters with relative weighting using parest_flags(306)
parest_flags(305)	1	Activates the estimation of negative binomial variance for the specific option specified by parest flags(111) = 4
parest_flags(306)	n	Specifies the bounds for the negative binomial variance parameter for the option specified by parest flags(111) = 4 (section 5.3)
age_flag(96)	n	pool tags into general tag population starting n periods after release
tag flags(i,1)	n	number of tag mixing periods is n
age_flag(100)	1	use negative binomial with added zeros as tag likelihood function. fish_pars(5,fi) and fish_pars(6,fi) are the parameters
parest_flag(33)	n	global upper bound for tag-reporting rate (*100)
fish_flags(i,32)		grouping flag for tag recaptures
fish_flags(i,33)	1	estimate tag-reporting rate
fish_flags(i,34)		grouping flag for common reporting rate
fish_flags(i,35)	n	penalty weight for tag-reporting rate prior is n
fish_flags(i,36)	n	reporting rate prior mean is $n/100$
fish_flags(i,37)	1	estimate random walk changes in tag-reporting rate (with frequency of devs defined by fish_flags(i,45))
fish_flags(i,43)	0	for parest_flag(111)=3, over-dispersion parameter (a) for neg. bin. = 5 (fish_pars(i,4)=0)
	1	over-dispersion parameter (a) for neg. bin. is estimated
fish_flags(i,44)		grouping flags for estimating neg. bin. variance parameter, and additional zeros parameters if active
fish_flags(i,45)	n	only have a tag-reporting rate deviation if the number of months since the last deviation is $> n$
fish_flags(i,46)	1	turns on estimation of mixture parameters for additional zeros in tag likelihood
age_flag(198)	1	Estimate tag-specific reporting rates

4.5.10 Age-length data

Table 4.12: Key flags settings used with age-length data

Flag	Value	Use
parest_flags(240)	1	Activates the model fit to observed age-length data

4.5.11 Movement Parameters

Table 4.13: Key flags settings used in the estimation of movement parameters

Flag	Value	Use
age_flag(27)	n	$\|n\|/10$ is the penalty weight for movement coefficients either different from zero ($n > 0$) or different from coefficients specified in the *.ini* file ($n < 0$), default = 50 (equivalent to penalty of 5). For $n < 0$
age_flag(28)	1	as for age_flag(27) but for age-specific movement coefficients
age_flag(29)	1	as for age_flag(27) but for non-linear, age-specific movement coefficients
age_flag(68)	1	estimate movement parameters (diff_coffs)
age_flag(69)	1	use movement parameters (diff_coffs)
age_flag(88)	1	estimate age-dependent movement parameters
age_flag(89)	1	use age-dependent movement parameters
age_flag(90)	1	estimate nonlinear age-dependent movement parameters
age_flag(91)	1	use nonlinear age-dependent movement parameters
age_flag(53)	0	frequency of movement same as frequency of recruitment
	n	frequency of movement is one per n recruitment periods

4.5.12 Recruitment, Initial Population, and Scaling

Table 4.14: Key flags for recruitment, initial population and scaling

Flag	Value	Use
age_flag(30)	1	estimate total recruitment deviations (usually on)
age_flag(31)	1	estimate totpop (usually on) for population scaling, note: only active if age_flags(94)=0
age_flag(32)	1	estimate overall population scaling parameter (check if this is turned off in phase 1!)
age_flag(113)	1	estimate parameter for scaling relative init population size and recruitment (usually on)
region_flag(1,n)	1	activate estimation of regional recruitment distribution in region n
age_flag(70)	1	in recruitment computation use parameters of time-series variability in regional recruitment distribution
age_flag(71)	1	estimate parameters of time-series variability in regional recruitment distribution
age_flag(110)	n	if $n > 0$, $n/10$ is the penalty on region rec diffs, default penalty = 0.1
age_flag(72)	n	if $n > 0$, does fit between relative recruitment and environmental correlate using $n/100$ as the penalty weight

Table 4.14: Key flags for recruitment, initial population and scaling

Flag	Value	Use
age_flag(57)	n	the number of recruitments per year is n (see also sec 4.5.5)
age_flag(94)	0	initial populations are free parameters
	1	initial population is $N_{a+1} = N_a \exp(-xM_a)$, where x is age_flag(128)/10
	2	initial population is $N_{a+1} = N_a \exp(-barZ_{a,r})$ where $barZ_{a,r}$ is the total mortality averaged over the first age_flag(95) periods
age_flag(95)	n	n determines the number of initial periods for averaging where age_flag(94) = 2
age_flag(128)	n	n determines 10× the multiplier x where age_flag(94)=1
age_flag(101)	1	turns on recruitment covariate file *.env* must be present
age_flag(102)	1	estimate recruitment/environment correlation parameter region_flags(1,r) 1 estimate average proportion of total recruitment coming from region r
parest_flag(149)	n	if $n > 0$, $n/10$ is the penalty weight for recruitment deviations (recr(iy)), if $n = 0$, the penalty weight is 1.0e-06, if $n < 0$, a self-scaling lognormal recruitment assumption is used
parest_flag(148)	n	if $n > 0$, $n/10$ is the penalty weight for differences between the last two recruitments
age_flag(178)	n	If n=1 the constraint is activated on the sum_product of the regional recruitment proportions and the regional recruitment deviates being equal 1 in each model time period.
parest_flag(400)	n	Exclude the estimation of temporal recruitment deviates for the terminal time periods starting n time periods before the end of the model analysis period.
parest_flags(398)	1	With parest_flags(400) active, fixes the absolute recruitments for the terminal time periods equal to the arithmetic mean of the estimated recruitments; default = 0, is the mean of log(recruitments)
parest_flag(155)	n	if $n > 0$, enable scheme of orthogonal polynomials for estimating recruitment where n is the default number of parameters (polynomial degree + 1) for year, region, season, and region×season recruitment variation. The following table summarizes the use of this and 20 other parest_flags that can be used to manipulate this scheme
age_flag(111)	n	penalty weight for the relative difference for terminal recruitment; default value when n=0 is 0.01, else n determines the weight
age_flag(76)	n	penalty weight for initial population numbers at age curvature; default value when n=0 is 0.1, else n determines the weight
age_flag(5)	1	implement the relative scalar of recruitment to initial population, note: estimated when age_flags(113)=1
age_flag(129)	1	implement autocorrelation in the lognormal random recruitment residuals
age_flag(135)	1	implement autocorrelation in the recruitment deviates of the stock-recruitment relationship
age_flag(136)	1	activate estimation of autocorrelation parameter in the recruitment deviates of the stock-recruitment relationship

Table 4.15: Orthogonal polynomial recruitment flags

	Level of polynomial (Source of Variation)			
	Year	Region	Season	Region-season interaction
Degree+1	pf-155/221	pf-216	pf-217	pf-218
Time period				
Start	pf-183	pf-204	pf-206	pf-208
End	pf-202	pf-210	pf-212	pf-214
Lower bound on degrees				
No. low order coefficients	pf-201	pf-205	pf-207	pf-209
Upper bound on degrees				
No. low order coefficients	pf-203	pf-211	pf-213	pf-215

4.5.13 MSY Stuff

Table 4.16: Key flag settings used to estimate MSY

Flag	Value	Use
age_flag(145)	n	if n is not equal to 0,this turns on stock-recruitment parameters, and if $n > 0$ then n is the penalty weight, else if $n < 0$ then 10^n is the penalty weight.
age_flag(146)	1	estimate stock-recruitment parameters
age_flag(147)	n	number of time periods between spawning and recruitment (default = 1)
age_flag(148)	n	number of years from last time period to compute average F
age_flag(155)	n	number of years from last time period to omit from average F computation
age_flag(149)	0	yield computed in weight (t) 1 yield computed in numbers
age_flag(194)	n	1: include effort deviations in determining average F-at-age when calculating the yield curve. Default is 0 (off).
age_flag(150)	n	if $n > 0$, turns on biomass dynamics estimation and n is the penalty weight
age_flag(151)	1	1: activates estimation of r parameter in Pella-Tomlinson model
age_flag(152)	1	1: activates estimation of K parameter in Pella-Tomlinson model
age_flag(153)	n	n: if $n > 0$, activates prior for steepness parameter in SRR relationship, and $n/10$ is the a parameter of the beta prior distribution
age_flag(154)	n	$n/10$ is the b parameter of the beta prior distribution a=10, b=2 $\Rightarrow$ mode=0.9, sd=0.1; a=13, b=5 $\Rightarrow$ mode=0.75, sd=0.1
age_flag(112)	n	if $n > 0$ use numbers of fish instead of biomass to calculate the catch
age_flag(140)	n	$n > 1$ enables region-specific yield analysis with resolution per Fmult step of $n/100$
age_flag(141)	n	n in the number of Fmult steps in region-specific yield analysis. Default: 200.
age_flag(162)	1	activate fitting of steepness parameter (af163 should be set to zero)
age_flag(163)	n	$n = 0$ stock-recruitment curve parameterized with steepness; $n = 1$, steepness a derived parameter
age_flag(165)	n	$n/100$ (default) or: target for F_{msy}/F or B/B_{msy}. Stockrecruitment parameters, age_flag(145), must be enabled.

Table 4.16: Key flag settings used to estimate MSY

Flag	Value	Use
age_flag(166)	n	$n/100$: penalty weight for F_{msy}/F or B/B_{msy}, default weight = 1000 ($n = 105$)
age_flag(167)	n	$n = 0$ (default) target F_{msy}/F, $n = 1$ target B/B_{msy}, $n = 2$ target SB/SB_{msy}
age_flag(168)	n	$n = 0$ (default) exponent in weighted average for F_{msy}/F, default: 50
age_flag(169)	n	divisor in weighted average for F_{msy}/F, default (for $n = 0$): 100000; $n < 0$) $\Rightarrow$ use Dave's approach.
fish_flags(i,70)	n	scale F of fleet i for equilibrium yield calculations by $n/100$
age_flag(199)	n	$n = 0$ (default) Beverton and Holt stock-recruitment relationship is calculated over the full model analysis period for deriving equilibrium yield estimates, otherwise $n > 0$ specifies the start of the period for the calculation starting n time periods before end of the model analysis period.
age_flag(200)	n	For age_flag(199)> 0, end of period for Beverton and Holt stock-recruitment relationship calculation is n time periods before end of the model analysis period.
age_flag(161)	1	Correct recruitments predicted from the Beverton and Holt stock-recruitment relationship for log-normal bias in the mean. Set = 0 applies no bias correction.
age_flag(182)	1	The Beverton-Holt stock-recruitment relationship is fitted to total "annualised" recruitments and average annual biomass

4.5.14 Fishing Mortality Targets

Table 4.17: Key flag settings used in the estimation of fishing mortality targets

Flag	Value	Use
age_flag(37)	n	average F target is $n/1000$
age_flag(39)	n	penalty weight for F target is 100 * n (default penalty weight = 1000)
age_flag(43)	n	last n years used for annual F target
age_flag(44)	n	$n/100$: annual F target
fish_flags(i,14)	n	restricts F for any individual fishing incident to $n/10$.
age_flag(107)	n	$n > 0$ activates overall exploitation penalty with n the penalty weight and age_flag(108) sets the target
age_flag(108)	n	$n/100$: overall exploitation rate target

4.5.15 Biomass Ratio Targets and Likelihood Profile

Table 4.18: Key flag settings used in the estimation of biomass ratio targets

Flag	Value	Use
age_flag(97)	n	the target biomass ratio (last x years/first x years) is $n/100$
age_flag(98)	n	penalty weight for biomass ratio
age_flag(99)	n	the number of initial and final years used to compute the biomass ratio
parest_flag(346)	n	activates penalty on either target biomass depletion, n=1, or average biomass n=2 for likelihood profile
parest_flag(347)	n	n is the absolute value of the target quantity for the likelihood profile; when parest_flag(346)=1 the target is $n/1000$; when n=0 a report is produced containing the MLE value (without scaling) for the target quantity
parest_flag(348)	n	n is the weight of the penalty on the target quantity for the likelihood profile
age_flag(172)	n	target quantity specified in parest_flag(346) is either total n=0, or adult n>0 biomass
age_flag(173)	n	first time period for defining average terminal biomass for depletion (parest_flag(346)=1) or average biomass (parest_flag(346) =2), counting backwards from end of time series
age_flag(174)	n	last time period for defining average terminal biomass for depletion (parest_flag(346)=1) or average biomass (parest_flag(346) =2), counting backwards from end of time series

4.5.16 Fishery Impact Analysis

Table 4.19: Key flag settings used to parameterize fishery impact analyses

Flag	Value	Use
fish_flags(i,55)	1	disable fleet i (set its catchability to 0) for impact analysis (see 3.8.1). If i = -999, disable all fleets.
age_flag(170)	n	enable a biomass depletion target: ($B_{\text{with fishing}}/B_{\text{without fishing}}$)
age_flag(171)	n	n = 0 (default) unfished (or diminished fishing) calculations based on estimated recruitment trajectories, otherwise calculations are based on estimated stock/recruitment relationship
age_flag(172)	n	n = 0 (default) total biomass depletion, otherwise adult biomass depletion
age_flag(173)	n	first time period for reckoning depletion – counting backwards from end of time series
age_flag(174)	n	last time period for reckoning depletion – counting backwards from end of time series
age_flag(175)	n	$n/1000$ is target depletion level
age_flag(176)	n	penalty on depletion target
age_flag(190)	0	get average recruitment from the SRR (default)
	n	calculate average recruitment for the time period defined by af(190) and af(191)
age_flag(191)	n	with af(190) defines time period for average recruitment

4.5.17 Natural Mortality Estimation

Many flags governing natural mortality put constraints on deviations of natural mortality with age, M' , rather than directly on natural mortality itself, M_a. The M' are the logs of the ratios of natural mortalities at age to mean natural mortality (see page 99).

Table 4.20: Key flag settings used in the estimation of natural mortality

Flag	Value	Use
age_flag(33)	1	estimate zoup$M$
age_flag(73)	1	estimate M_a; $\forall$ age class a
age_flag(109)	n	activates the option for estimating M_a: n=0 estimates M_a as deviations from the mean level; n=1 estimates M_a directly as independent parameters as stored in age_pars(5) in the *.ini* file (or *.par* file); n=2 estimates a spline function; n=3 estimates a Lorenzen function
parest_flags(121)	n	Either: activates the estimation of the spline function with n being the number of nodes and parameters stored in age_pars(5) in the *.ini* file (or *.par* file), note: age_flags(109) must be set to 2; Or when n=2 AND age_flags(109) is set to 3, activates the estimation of the two Lorenzen function parameters
age_flag(77)	n	n is penalty weight on $\Sigma(M_{a-1} - 2M_a + M_{a+1})^2$, default = 25
age_flag(78)	n	n = penalty weight on $\Sigma(M_{a+1} - M_a)^2$, default = 5
age_flag(79)	n	n is penalty weight on $\Sigma(M_a - \text{zoup}M)^2$, default = 10
age_flag(81)	n	n is number of terminal age classes with the same M
age_flag(82)	n	target average M $\equiv$ zoup$M$' is n/100
age_flag(80)	n	n is penalty weight on zoupM'^2, default = 10
age_flag(83)	n	n is minimum age to include in zoupM' , (default = 1)
age_flag(85)	n	n is maximum age to include in zoupM' , (default = nage)
age_flag(84)	n	n is penalty weight for target zoupM' , (default = 0)
age_flag(130)	n	$n > 0$ activates M/K target (natural mortality to von Bertalanfy K) with n/100 the target
age_flag(131)	n	n: first age in average M-at-age for M/K target; defaults to age 1
age_flag(132)	n	n: last age in average M-at-age for M/K target; defaults to oldest age
age_flag(133)	n	n: penalty on M/K target

4.5.18 Projections

Table 4.21: Key flag settings used to parameterize projections

Flag	Value	Use
age_flag(20)	n	n is the number of simulations in stochastic projections (*nsims*)
age_flag(190)	n	n = 0 (default) stock-recruit curve gives recruitment during projection period, otherwise projection period recruitment given by average estimated recruitment during period starting n time periods before end of data series, i.e., the beginning of the projection period defined by fishery data flags in the *.frq* file (page 26).

Table 4.21: Key flag settings used to parameterize projections

Flag	Value	Use
age_flag(191)	n	For age_flag(190)> 0, end of recruitment average is n time periods before end of data series.
age_flag(195)	n	Defines average recruitment determined from the Beverton and Holt stock-recruitment relationship in deterministic projections. n = 0 (default) assumes average recruitment is calculated over the full model analysis period. n = 1 specifies average recruitment according to the period defined by age_flag (190, 191).
age_flag(161)	1	Correct recruitments predicted from the Beverton and Holt stock-recruitment relationship for log-normal bias in the mean. Set = 0 applies no bias correction.
age_flag(183)	1	When age_flag(182)=1 (see 4.5.12), for deterministic projection the annualized Beverton-Holt stock-recruitment relationship prediction is allocated according to the average seasonal recruitment distribution
parest_flag(142)	n	n is last time step for which catch and size data deviations appear in the likelihood function. Useful to set n to the beginning of the projection period to avoid having events during the projection affect parameter estimates.
parest_flag(231)	n	n is the number seed for the random number generator for numbers at age in the first projection years; else if n = 0 a default supplied seed value is used.
parest_flag(232)	n	n = 0 sets the first projection year for stochastic recruitments to 1; else it is set to the last model year.
parest_flag(233)	n	n = 0 sets the last projection year for stochastic recruitments to the last model year.
parest_flag(234)	n	n = 1 activates the placement of randomised effort deviates.
parest_flag(235)	n	n is the standard deviation of the distribution from which random effort deviates are drawn
parest_flag(236)	n	n is the number seed for the random number generator for effort deviates; else if n = 0 a default supplied seed value is used.
parest_flag(237)	n	n = 1 activates the placement of simulated random numbers at age in the first projection year, over *nsims* (age_flag(20)).
parest_flag(238)	n	n = 0 activates the placement of simulated random annual recruitment, over *nsims* (age_flag(20)).
parest_flag(239)	n	n = 1 activates the stochastic recruitments applied as deviates to the predictions of the stock-recruitment relationship in each projection run
parest_flag(241)	1	activates generation of pseudo-observations
parest_flag(242)	1	activates generation of pseudo-observations for estimation model time periods
age_flag(186)	n	n/100 is the standard deviation of the pseudo-observed effort (when projection catch is supplied)
age_flag(187)	n	n/100 is the standard deviation of the pseudo-observed catch (when projection effort is supplied)
parest_flag(353)	1	optimise operating model calculations
age_flag(96)	200	MUST be set to 200 to avoid implausible calculation

4.6 Helpful Utilities

MULTIFAN-CL produces a bewildering array of output files in the directory from which it was run. A typical directory listing might look like the following where most, but not all, of the files were output by MULTIFAN-CL.

Table 4.22: Sample file output by MFCL

$ ls						
00.par	01.par	03.par	04.par	05.par	06a.par	06.par
07.par	08.par	09.par	10.par	11.par	12.par	13.par
14.par	15.par	16.par	17.par	aa	catchequ	catch.rep
check.tmp	contribs	deplabel.tmp	derch.rpt	*doitall*.yft	do_sd	eigv.rpt
error.log	ests.rep	gradient.rpt	junk	length.fit	LL-4A.png	mfcl.cfg
MULTIFAN-CL.ini	nr_tags	plot.rep	tag.rep	tags.err	testequ	tester
tmp.out	tt	variance	weight.fit	xinit.rpt	yft.dep	yft.frq
yft.hes	yft.ini	yft.tag	yft.var			

The aim here is to introduce a few utilities that have been found helpful in sorting out useful information from the output files while a MULTIFAN-CL fit is proceeding or shortly thereafter. Deeper analysis of MULTIFAN-CL output is presented in Chapter 6 where some of these files are discussed in detail.

4.6.1 Dealing with Flags

It is often useful to know how the flags have been set in a *.par* file. Because the flags are not labeled in the *.par* file, it is difficult to find particular flags in a listing of the file.

flags

The utility *flags* helps by listing the flag number, a short flag description, and the value for all flags that are set to other than the default value. Here is a sample:

[ComputerName:try3]$ flags 08.par

INPUT FILE:08.par

Parest Flags

---

12 'growth: avg size of 1st age class' :1

13 'growth: avg size of last age class' :1

14 'growth: est K' :1

15 'growth: SD, size at age' :1

16 'growth: SD, size dependent' :1

32 'control: initial control regime' :2

1 'control: iterationion limit' :1000

50 'control: max gradient limit' :4

146 'control: scale gradient for tot pop.' :25

189 'control: write size freq .fit files' :1

190 'control: write .rep files' :1

141 'sample: likelihood fn selection' :3

Age Flags

---

68 'move: enable', :1

69 'move: est', :1

33 'M: est. avg M', :1

82 'M: target avg M', :20

84 'M: penalty, avg M', :3

57 'recruit: num rec/yr', :1

30 'recruit: est Rtot', :1

113 'recruit: scale N0 and R', :1

70 'recruit: Rrt enable', :1

71 'recruit: Rrt est' :1

110 'recruit: Rrt penalty', :5

94 'N0: strategy', :2

95 'N0: af-94 modifier', :5

146 'MSY: activate SRR', :1

145 'MSY: penalty on SRR pars', :5

147 'MSY: lag betw. spawning and recruitment', :2

148 'MSY: num final yrs for avg F', :5

155 'MSY: yrs to omit for avg F', :1

34 'est effort devs', :1

31 'est totpop', :1

32 'pop scaling param', :1

144 'common wt for catch L to 10000', :10000

Fish Flags : 1 2 3 4 5 6 7 8 9 10 11 12

---

24 'selec: grouping', : 1 1 2 2 3 3 4 4 5 5 5 5

16 'selec: shape', : 1 1 1 1 1 1 1 1 1 1 1 1

3 'selec: 1st age, common sel', : 17 17 17 17 17 17 17 17 17 17 17 17

26 'selec: length-dependent', : 2 2 2 2 2 2 2 2 2 2 2 2

48 'selec: est (obsolete??)', : 1 1 1 1 1 1 1 1 1 1 1 1

1 'q: est avg. q', : 1 1 1 1 1 1 1 1 1 1 1 1

10 ‘q: time series in q’, : 0 0 0 0 1 1 1 1 1 1 1 1

15 ‘q: penalty for q-devs’, : 50 50 50 50 50 50 50 50 50 50 50 50

23 ‘q: time step for q-devs’, : 11 11 11 11 11 11 11 11 11 11 11 11

29 ‘q: grouping for overall q’, : 1 1 1 1 2 2 2 2 3 3 3 3

47 ‘q: arbitrary seasonal q’, : 4 4 4 4 4 4 4 4 4 4 4 4

4 ‘E: est Edevs’, : 2 2 2 2 2 2 2 2 2 2 2 2

55 ‘fishery impact analysis:’, : 1 1 1 1 1 1 1 1 1 1 1 1

pflag, aflag, fflag

These utilities are useful for examining or setting a particular flag when the flag number is known. The general command line for these is:

\{p,a,f\}flag flagno. [=newvalue] parfile

To simply examine flag values the “[=nevalue]” argument is omitted as follows:

[ComputerName:try3]$ pflag 50 08.par

08.par: 4

[ComputerName:try3]$ aflag 145 08.par

08.par: 5

[ComputerName:try3]$ fflag 24 08.par

08.par: 1 1 2 2 3 3 4 4 5 5 5 5

Note that wildcard characters can be used in the *.par* file, as in:

[ComputerName:run4]$ aflag 145 *.par

3G1.par: aflag(145) = 0

3G2.par: aflag(145) = 0

4m1.par: aflag(145) = 0

SRR.par: aflag(145) = 5

homeinE.par: aflag(145) = 5

Flag values can be set with pflag and aflag as in this example which also shows a wildcard in the path to the *.par* file:

[ComputerName:run4]$ aflag 166 =1000000 0[3-6]/Fmsy/j.par

03/Fmsy/j.par: aflag(166) = 1000000

04/Fmsy/j.par: aflag(166) = 1000000

05/Fmsy/j.par: aflag(166) = 1000000

06/Fmsy/j.par: aflag(166) = 1000000

The ability to alter flag values with fflag has not been implemented yet.

flagdiff

This utility facilitates examining how two .par files differ in their flag settings, for example:

[ComputerName:try3]$ flagdiff 08.par 09.par

1c1

< INPUT FILE:08.par

> INPUT FILE:09.par

12c12

< 50 'control: max gradient limit' :4

> 50 'control: max gradient limit' :-6

41a42

> 171 'impact based on SRR' :1

4.6.2 Quick Access to *.par* files

checkafit

While a fit is proceeding through its phases under the direction of a *doitall* file, it is of interest to check on how things are going from time to time. checkafit is a small UNIX-type script that parses portions of one or more *.par* files and lists the final objective function values, the final maximum gradients, and the numbers of active parameters. With an argument of *.par in the directory above, the following is obtained:

[ComputerName:try3]$ checkafit *.par

ID obj grad npar

00.par

01.par 397979.090657947527 18.592977524637 2321

02.par 397367.876420154760 8.392482241683 2299

03.par 400223.872837031668 9.595400425097 3099

04.par 405270.936261841794 8.545998866894 3111

from which we can see that the fit has progressed through phase 4, that there has been some progress in minimizing the objective function, though not much in the last couple of phases. The last gradient value would not be very impressive for a final phase of the fit wherein we would look for a small gradient value ($< 10^{-4}$). However if this were an intermediate phase with convergence criterion set large, then it would be OK.

mfcl.summary

This utility is similar to checkafit in that it parses *.par* files, but in addition to convergence information, it also returns a few parameter estimates as in this example:

[ComputerName:try3]$ mfcl.summary 04.par

04.par

natural mortality coefficient 0.250000000000

The von Bertalanffy parameters

24.122546431390 20.000000000000 40.000000000000

160.349331770499 140.000000000000 200.000000000000

0.100000000000 0 0.300000000000

Variance parameters

6.664684263036 3.000000000000 8.000000000000

0.400000000000 -0.690000000000 0.690000000000

Objective Max. Grad. No. pars

405270.936261841794 8.545998866894 3111

proflist

This utility is designed to return information on MULTIFAN-CL runs with target ratios of F to F_{MSY}, B to B_{MSY}, or B to $B_{\mathrm{no\_fishing}}$ (see age_flags 165 to 167, page 57, and age_flags 170 to 176, page 58). This utility is particularly helpful when making a series of runs to determine a likelihood profile for the target ratio (see page 21). Here is example output:

[ComputerName:BBmsy]$ proflist B*.par

obj grad target realized penalty rezid_obj net_obj

B89.par 1246167.2 0.032270 0.89 0.89085 1e+05 2.293e-02 1246167.152

B94.par 1246169.5 0.094470 0.94 0.94067 1e+05 1.282e-02 1246169.457

B96.par 1246170.2 0.045040 0.96 0.96060 1e+05 9.665e-03 1246170.150

B98.par 1246170.7 0.063840 0.98 0.98052 1e+05 6.947e-03 1246170.740

B100.par 1246171.2 0.095350 1.00 1.00040 1e+05 4.777e-03 1246171.202

B102.par 1246171.6 0.052910 1.02 1.02030 1e+05 2.783e-03 1246171.596

B104.par 1246171.9 0.024950 1.04 1.04030 1e+05 1.548e-03 1246171.863

B105.par 1246172.0 0.093710 1.05 1.05020 1e+05 9.455e-04 1246171.970

B106.par 1246172.0 0.069230 1.06 1.06020 1e+05 5.457e-04 1246172.048

B107.par 1246172.1 0.071540 1.07 1.07010 1e+05 2.530e-04 1246172.104

B108.par 1246172.1 0.098430 1.08 1.08010 1e+05 5.784e-05 1246172.138

B109.par 1246172.1 0.014950 1.09 1.09000 1e+05 6.677e-07 1246172.148

B110.par 1246172.1 0.089900 1.10 1.09990 1e+05 8.368e-05 1246172.134

B111.par 1246172.1 0.044070 1.11 1.10990 1e+05 3.000e-04 1246172.097

B112.par 1246172.0 0.068440 1.12 1.11980 1e+05 6.441e-04 1246172.038

profplot

Another way to follow the progress of a likelihood profile in the making is the utility profplot which displays a plot of the profile embodied in a selection of *.par* files. It requires ImageMagick, which is likely to be part of most Linux distributions. It also uses R functions for access to MULTIFAN-CL output (see section 6.3.2).

4.6.3 Running in background

For longwinded runs it is useful to have MULTIFAN-CL run in the background so that other work can be done while the run proceeds. This is especially so if the MULTIFAN-CL is running on a mainframe machine and the user wishes to log off for the night or weekend leaving MULTIFAN-CL going. The following assumes that MULTIFAN-CL is being run on a LINUX or UNIX-like mainframe, and instructions are issued to it either from a local console or remotely from another computer.

nohup ... &

The nohup command is a standard UNIX utility that allows a process to be started and run independently of the shell that starts it. All output from the process is sent to a file "nohup.out". The following will start a background MULTIFAN-CL analysis contained in a *doitall* file.

[ComputerName:try3]$ nohup *./doitall* &

The "&" at the end of the run string is necessary for restoring control to the shell while the background process continues, and nohup.out allows the shell to exit and the user to log off without disrupting the *./doitall* process.

tail

Because the MULTIFAN-CL output is going to a file, that output can be viewed from any shell that has read access to that file. A convenient way to view it is with another standard UNIX utility, "tail", which lists the last few lines of a file. The -f option causes tail to track the tail end of the file as it evolves:

[ComputerName:try3]$ tail -f nohup.out

condensed.list

Because the output tends to be quite voluminous, it can be difficult to read the salient pieces of it, particularly if the output is produced quickly. The "condensed.list" was designed to filter the output so that only summarized results are listed for each iteration of the function minimizer. Section 6.1.1 gives example filtered output. Sometimes additional information is desired in addition to the bare bones summary. In that case arguments can be given to condensed.list telling it to report more of the output at each major iteration of the minimizer, for example:

[ComputerName:try3]$ tail -f nohup.out | condensed.list alpha beta

.....

Function value -7.8995964e+05; maximum gradient component mag -2.9669e+00

alpha = 3134776.98

beta = 4070.32

.....

Function value -7.8995969e+05; maximum gradient component mag 2.4067e+00

alpha = 3134812.82

beta = 4070.14

.....

Function value -7.8995973e+05; maximum gradient component mag 2.4258e+00

alpha = 3134883.27

beta = 4070.80

.....

In this case condensed.list has displayed output records containing “alpha” and “beta” in addition to its normal filtered output.

4.6.4 screen: A utility for detaching and reattaching a running MULTIFAN-CL Process

One disadvantage of starting a MULTIFAN-CL run with nohup is that it is not then possible for the user to intervene for example with “Ctrl-c q” (see page 74). The screen utility solves that problem by allowing a process to be detached from the shell that started it and subsequently re-attached to that or a different shell – even a new shell after the user has logged off and then on again. Once re-attached, the user can interact with the process through the shell in the normal way. The screen utility has many other features as well. Its only drawback is that its great power makes it difficult to learn. The purpose here is ease the “learning curve” by introducing just those features that have been found particularly useful in running MULTIFAN-CL analyses.

The main thing to understand about the screen utility is that it creates one or more “sockets” for the user which are normally accessible only by that user. Each socket can administer an environment of one or more “screens’ with a shell running in each. Within each screen, the user interacts with the shell as usual. All the normal programs and utilities can be run, and in particular a MULTIFAN-CL analysis can be run either directly or via a *doitall* file with standard output displayed to the screen.

This can perhaps be better explained by running through the following example where the commands to the screen utility are listed prominently and other commands to the shell or to MULTIFAN-CL are mixed with the text: Assume that a user is working on a mainframe computer in a shell with current directory “try3” containing a *.frq* and other files necessary for a MULTIFAN-CL run. The user then enters at the shell prompt:

[ComputerName:try3]$ screen -S mfclrun

A socket with the title “mfclrun” is created. Any name could be chosen, but it’s best that names don’t start with numerals. The user is now in the screen environment, and her physical screen is attached to a “virtual” screen named “screen 0” displaying a shell prompt.

The user then launches a MULTIFAN-CL analysis by calling mfcl with some appropriate run string, and the normal screen output from MULTIFAN-CL scrolls past on the screen. The user then types

Ctrl-a Ctrl-d

and is detached from the screen environment and confronted with a new prompt in the original shell from which screen was invoked, i.e.,

[ComputerName:try3]$ screen -S mfclrun

[ComputerName:try3]$

At that point the user logs off and goes home. The MULTIFAN-CL run continues, still dumping its output to its virtual screen. Later the user logs on remotely from home and enters the following (the directory doesn’t matter):

[ComputerName:homedir]$ screen -r mfclrun

which attaches her home computer screen to screen 0 of her mfclrun socket, and MULTIFAN- CL output scrolls past. She then types

Ctrl-a H

which produces a message on the top of her screen indicating that logfile “screenlog.0” has been created. She then types

Ctrl-a Ctrl-c

which opens a shell prompt in another virtual screen (screen 1) which is now attached to her physical screen. The current directory is try3, same as that of screen 0, and a directory listing would reveal an entry for file "screen.0". She enters

[ComputerName:]$ tail -f screenlog.0 | condensed.list

and begins to see summarized output from MULTIFAN-CL (see 6.1.1). She then types

Ctrl-a Ctrl-d

which detaches her computer from the mfclrun socket. The next morning our intrepid user logs on to the mainframe, again enters

[ComputerName:]$ screen -r mfclrun

and finds herself looking at more summarized output from MULTIFAN-CL in screen 1. She decides that she needs to interrupt the run in order to rescale (see page 75) and types

Ctrl-a Ctrl-a

which switches her to screen 0 where she stops MULTIFAN-CL with Ctrl-c q (see page 74) and issues a new call to mfcl. Typing

Ctrl-a Ctrl-a

toggles her back to screen 1 where screenlog 0 is still being filtered through condensed.list. This switching between virtual screens and detaching and re-attaching goes through several cycles till the analysis is complete. In the process more virtual screens are created with Ctrl-a Ctrl-c which complicates switching screens somewhat (see table below). Screens are discarded by typing

Ctrl-a K

and finally, with only a single virtual screen left, another Ctrl-a K discards the mfclrun socket.

The following table summarizes the screen commands shown above plus a few more helpful commands. Further information on the screen utility can be found in LINUX/UNIX documentation.

Table 4.23: Helpful commands for the screen utility

Outside screen environment:	
screen -ls	list known screen sockets
screen -S *name*	name creates and attaches screen socket *name*
screen -r [*name*]	attach existing screen socket. Need type only enough of name to identify among socket names
screen -dr [*name*]	[name] force attach to existing screen socket which system thinks is already attached (can happen for example with inadvertent disconnect)
Within screen environment:	
Ctrl-a [Ctrl-]d	detach socket
Ctrl-a [Ctrl-]c	create new screen and switch to it
Ctrl-a Ctrl-a	switch to last screen
Ctrl-a [Ctrl-]n	switch to next screen
Ctrl-a [Ctrl-]p	switch to previous screen
Ctrl-a *n*	switch to screen *n*
Ctrl-a Esc	enter scroll mode to allow scrolling in current buffer q exit scroll mode, and move to end of current buffer

Table 4.23: Helpful commands for the screen utility

Outside screen environment:	
Ctrl-a H	toggle logging of current screen output to file *screenlog.n*, where *n* is the current screen number
Ctrl-a K	discard current screen. If only one screen on current socket, exit and discard socket

4.6.5 Miscellaneous Other Utilities

newtry

It is not unusual to conduct a number of fits in the course of a MULTIFAN-CL analysis. To keep a record of all the fitting trials, it is advisable to start each one in a new directory to keep files from being overwritten. newtry facilitates setting up a new directory from an existing one with only the files necessary to begin afresh. Running newtry from within the directory in this example gives the following:

[ComputerName:try3]$ newtry ../try4

[ComputerName:junk]$ cd ../junkk

[ComputerName:try4]$ ls

doitall.yft mfcl.cfg MULTIFAN-CL.ini yft.frq yft.ini yft.tag

The new directory contains only the basic input files plus a copy of the previous *doitall* file. Presumably one or more of these files will be edited in some way so as to conduct the new fit under different conditions of some kind.

It is often the case that the analyst would like to start a new fit from an intermediate phase rather than from the beginning. A useful modification of newtry would be an optional argument to indicate a starting phase so that the new directory would be set up with the appropriate *.par* file(s) in place.

Chapter 5

Likelihood Functions

MULTIFAN-CL fits itself to data by minimizing an objective function consisting of the sum of negative log-likelihood functions of observed data and Bayesian priors on various parameters. This section discusses how to specify and manipulate these various components of the objective function. Mathematical details of the objective function are given in the Technical Annex (see A.2).

5.1 Total Catch Data

A fundamental assumption of MULTIFAN-CL is that the total catch data are observed with relatively little error. The default assumption is that the coefficient of variation (CV) of log-catch residuals is about 0.07 (see below 5.4). This is controlled by age flag 144, which is set automatically to 10000 during the initial phase (the value of the flag is multiplied by 0.01 in the code, thus obtaining the CV of 0.07). The value of the flag may be changed by the user after the initial phase, although this is not recommended.

The facility is available to set the penalty weight for the total catch likelihood on a fishery-by-fishery basis. This is useful if it is suspected that the estimates of total catch for one or more fisheries may be less precise than for the others. Fishery-specific penalties are set using fish flag 56. This flag has the same behavior as age flag 144, i.e., a setting of 10000 assumes a residual CV of 0.07. Note that if fish flag 56 is used, it must be set for all fisheries.

5.2 Length and Weight Frequency Data

Several options are available for the length and/or weight frequency likelihood function (see 4.5.3) through setting parest flag 141 and 139, respectively. We suggest setting this flag to 3, which invokes the full normal probability density, with the observed frequencies used in the variance term. The added constant term related to the number of length-class intervals (see A.2.1) can be adjusted using parest flag 193. The option for the normal likelihood using a student-t probability density function is available (parest flag 141 and 139 = 8). The aim of using this approach was that it may accommodate zero observations and may be self-scaling (i.e. the degrees of freedom may be estimated). Parameter estimation of the student-t probability density function is activated by various flags ([4.5.3]).

Another similar option is a modification to the multinomial probability density function that produces an estimator with good self-scaling properties, while retaining the overdispersion and autocorrelated random effects characteristics of the Gaussian-multinomial. This is called the self-scaling multinomial (M estimator)

with random effects (SSM-RE). In addition to addressing all of the deficiencies in the multinomial, the SSM-RE retains the key multinomial property of being able to deal with observed zero proportions in a completely natural way. Therefore, it does not require modification of the data to remove observed zeros. It is also possible to estimate the strength of the relationship between effective sample size and relative sample sizes. The covariate exponent parameter is estimated by setting fish flags 85 and 86, with the upper bounds on the effective sample size specified by parest flag 336 and 337, for length and weight samples, respectively. The SSM-RE option is activated using parest flag 141 and 139 = 9 ([4.5.3]) and the parameters are stored in fish pars 14 to 21.
It is also possible to apply the self-scaling multinomial method but without the random effects estimation, SSM-noRE. This approach effectively assumes all errors within the estimated effective sample sizes, and is activated using parest flag 141 and 139 = 10 ([4.5.3]). It excludes estimation of the random effects parameters (fish_flags 78, 80, 82, and 84), with no settings for the bounds of these parameters (parest_flags 315, 316, 317 and 370), and no selection of the set of M-estimator coefficients ((parest_flags 338).
Another option that permits estimation of an effective sample size is the Dirichlet Multinomial without random effects estimation (DM_noRE) with two parameters estimated: an exponent for an effective sample size multiplier (λ), and an exponent for a sample size covariate. These are stored in fish_pars(22 and 23), respectively, for length data, and fish_pars(24 and 25), respectively, for weight data. An assumed maximum effective sample size may be specified using parest_flags(342), default is 1000.
For the Robust-Normal, SSM-RE, SSM-noRE and Dirichlet Multinomial likelihood options, the estimated effective sample sizes are calculated and are reported along with the observed sample sizes ([6.3.13]). This facilitates a comparison that illustrates the effect of the self-scaling properties of the latter three options.
Settings 1 and 2 are minimum chi-squared formulations of the likelihood function. Setting 2 scales the constant part of the variance by the inverse of the number of length intervals; setting 1 sets the constant part of the variance to 0.01. The default setting (0) uses the original MULTIFAN formulation of the likelihood (equivalent to setting 1 but with the predicted frequencies used in the variance computation) as given in Fournier et al. (1990). Note that currently, settings 3, 8 or 9 is recommended for weight frequency data.

Aggregated data among sexes or species
Where length- or weight-frequency data is aggregated among sexes or species, an assumption must be made as to how the model predictions of these observations are aggregated accordingly. The default option is to assume a 1:1 ratio, however this may be inappropriate. An alternative is activated by parest_flags(350) set equal to 1, that accounts for the sex- or species-ratio in the predicted numbers caught in the fishing incident associated with the observation, when aggregating the predicted size compositions among sexes or species.

Tail compression

It is possible to aggregate length- and weight-frequencies among particular large and small size class intervals based on a specified percentage, e.g. all lengths can be pooled so that the "plus" length-class contains 0.1% of the total length-frequency. This is sometimes called "tail compression", and addresses the weakness of using the normal or log-normal distribution for a composition data likelihood being its inability to deal with observed proportions equal to zero. The approach "compresses" the distribution, such that the length-intervals having scant or missing observations, such as occurs in the tails (small and large sizes), are aggregated into a single interval comprising a specified total proportion of the observed frequencies. This reduces the incidence of zero observations in the composition data. Activating tail compression is done specific to the likelihood option specified above. For the normal likelihood plus constants, i.e. parest flag 139 and 141 = 3, it is activated by setting parest flag 311 = 1 for the length-frequency data, and parest flag 301 = 1 for the weight-frequency data; and using parest flags 313 and 303 for specifying the proportion in each tail (i.e. the small and large class intervals, not the combined proportion) for the length and weight frequency samples respectively. The formulated proportions are equal to parest flag 313/100 and parest flag 303/100. For the SSM-RE M-estimator, i.e. activated using parest flag 139 and 141 = 9, tail compression is activated by setting parest flag 320 and 330 to the value of the desired minimum number of size class intervals in the tail compressed distributions for length and weight frequency samples, respectively. A penalty weight term is calculated using the predicted proportions in the size class intervals external of the truncated intervals, and

scaled by the effective sample size which has an upper bound specified by parest flag 336 and 337 for length and weight frequency samples, respectively.

Minimum sample size threshold

To avoid size samples having very few observations, and therefore probably a high number of zero size proportions, a minimum threshold sample size can be applied to the length and weight frequency data. This is activated if parest flag 312 > 0 then all length samples having n observations < parest flag 312 are excluded, and if parest flag 302 > 0 all weight samples having n observations < parest flags 302 are excluded. Currently this option is only active for the robust normal likelihood option with tail compression being activated, i.e. if parest flag 311 and 301 = 1.

5.3 Tagging Data

There are several options available for the tag likelihood function, specified by parest flag 111 4.5.8. If the flag is set to 0 or 1, a simple least- squares type function is employed. These functions were employed when the inclusion of tagging data into MULTIFAN-CL was being first developed, and they are not recommended for routine use.

A Poisson likelihood function is employed when parest flag 111 = 2. The Poisson is probably the simplest likelihood function to use for tagging data, and is a useful base model in cases where the identity of tag recaptures with respect to their release group is not maintained over all time periods. One limitation of the Poisson is that the variance of tag recaptures for a particular area-time stratum is determined by the expected value of the recaptures. This results from the various assumptions of a Poisson process, including that tag recapture observations are independent. For most tagging experiments, we expect some degree of clumping or contagion in the pattern of tag recaptures and therefore the assumption of independence is not likely to be satisfied. In such cases, the Poisson variance will underestimate the variance of the data, resulting in over-weighting of the tagging data in the objective function. To deal with this problem, a negative binomial likelihood may be used and is invoked by parest flag 111 = 3. Fishery-specific parameters that scale the variance are now estimated from the data. These parameters are stored in the 4th row of fishery parameters in the *.par* file. The parameterization is such that large values of these scaling parameters make the negative binomial approach the Poisson, whereas small values of the parameters result in higher variance and overdispersion. Estimation of the parameters is activated by setting fish flag 43 = 1 and they may be grouped across fisheries using fish flag 44 as grouping flags.

An improved method for estimating over-dispersion is invoked by parest flag 111 = 4, where the negative binomial variance parameter τ is estimated directly, such that

$$\tau = 1 + e^{fish\_pars(i,4)}$$

Settings for fish flag 43 and 44 are as above, but parest flag 305 = 1 activates the estimation of the parameters (4^{th} row of fishery parameters in the *.par* file). The bounds on the over-dispersion parameters may be defined by parest flag 306. Setting parest flag 306 = 0 specifies the default value for the bounds on the over-dispersion parameters (-5.0 to 5.0), while a non-zero value implements a change to the bounds such that:

lower bound = log(parest flag 306/100.-1.0)

upper bound = log(50.0 * parest flag 306/100. - 1.0)

Therefore, parest flag 306 = 400 produces a lower bound slightly higher than 1 (1.0986) thus approaching the Poisson, with little over-dispersion. The rule of thumb is: the higher the value for the lower bound the greater the over-dispersion.

Relative weighting of the tagging likelihood can be achieved using parest flag 306 such that values larger than 400 impose a higher lower bound on the over-dispersion parameter, and therefore higher variance resulting in lower influence of this data type in the overall objective function (see Appendix A.2.3).

The negative binomial option parest flag 111 = 3 is not recommended as being a reasonable approach.

The negative binomial option parest flag 111 = 4 with no grouping of tags (i.e. -999 32 0) doesn't work as it has not yet been developed. You can achieve the same effect by setting all the grouping to 1, (i.e. -999 32 1).

Another statistical problem that sometimes occurs in tagging experiments is an over representation of zero tag recaptures in the data compared to the predictions of statistical models such as the Poisson and negative binomial. One could imagine several ways that this could occur, including a tendency of fishing operators to not report small numbers of isolated recaptures that might be expected to occur at long times after release when perhaps publicity for the program has wound down. One way to deal with this problem is to allow for added zeros in the probability distribution of the tag recaptures. Essentially, we attempt to estimate the probability of additional zero recaptures by way of fishery-specific parameters that are stored in the 5th row of fishery parameters in the *.par* file. Estimation is activated by fish flag 46 = 1 and fish flag 44 is the grouping flag.

The full technical details of the use of the negative binomial are given in the Technical Annex (Appendix A.2).

5.4 Age-length data

Ageing data from biological sampling for otoliths or other structures used to estimate age, can be formally included in fitting the population model. Presumably these data assist in estimating growth parameters because the readings provide direct observations of the distribution of fish ages within length classes. Typically, these observations are collected from a particular fishing method or fishery using a sampling design stratified in respect of length, and which assumes the observations at age are random within each length class.

The context of age-length data being fitted in the model is activated by parest flag 240, (see 4.5.9).

Using the normal distribution the model growth function predicts mean lengths and standard deviations for each age class from which the predicted distribution of age-at-length in the population is calculated. This is then scaled by the predicted catch age composition, which takes account of the selectivity pattern for the fishery, to derive the predicted distribution of age-at-length for that fishery given the growth estimates of length-at-age.

The observed age composition within each length interval is assumed to be multinomially distributed, and therefore the total negative log-likelihood for a given sample is summated among all length intervals in the sample, and the total over all samples is added to the objective function.

There are two possible approaches for assigning a scalar to the effective sample size. The same effect is achieved by assigning "0" or "1" for all fisheries in the input age-length data file, i.e. the observed sample size is assumed for the effective sample size. Values that are neither 0 nor 1, are multiplied by the observed sample size producing the assumed effective sample size that scales the age-length likelihood and determines the fishery-specific relative weightings.

5.5 Penalties and Priors

The addition of penalties to the likelihood function is a means of adding non-data information to the model. In a Bayesian context, this is equivalent to putting a "prior" on a parameter, or a quantity that is a function

of the parameters. The prior describes the level of certainty we have about a particular variable, and is in effect a prior (i.e., before entering the analysis and being subjected to the data) probability distribution for the parameter. Typically, we use age flags or fish flags to specify the mean of the prior (which might be thought of as the "target" value for the parameter) and a penalty weight that determines the extent to which the objective function is penalized for having a value different to the target. Of course, if the information in the data are consistent with the prior, then the parameter estimate will be close to the target and the penalty will be very small. On the other hand, if there is very little information in the data concerning a particular parameter, then the model estimate will be dominated by the prior and will likely end up at a value very close to the target. It is in this latter case that the application of penalties is very useful – it tends to stabilize the model and avoids the parameter estimates becoming biologically unreasonable. This is not the same as fixing the parameter at the target value, because the probability distribution of the prior is taken into account in the estimation of the covariance matrix of the parameters – if the parameter were simply fixed, its uncertainty would not be recognized in the model. Another possibility is that the data will turn out to be highly informative regarding a particular parameter and that the estimate will differ considerably from the prior. It is here that the analyst needs to look further at the structure of the model to see why an unreasonable estimate has been obtained.

5.5.1 Normal priors

A typical penalty function is of the form $p = w(x - x_{target})^2$, or often $p = w(\log(x/x_{target}))2$ where p is the penalty to be added to the objective function, x is the parameter estimate, x_{target} is the prior mean (or target value) and w is the penalty weighting factor. With this form of penalty function, it is clear that x, or $\log(x)$, is a normally distributed random variable of mean x_{target} and standard deviation $\sigma = 1/\sqrt{\text{zoup2zoup}w}$.

In MULTIFAN-CL the penalty weights w are typically set using age flags or fish flags. The coefficient of variation, CV (σ divided by the mean), is a useful quantity to guide the specification of w. A CV of 0.1 or less would indicate that a variable is fairly well determined, whereas a CV of 0.5 or more would suggest a fairly high degree of uncertainty or variability. Note that many variables, such as effort deviations and catchability deviations, are assumed to deviate logarithmically, i.e., they have multiplicative rather than additive deviations and a mean of 1. In these cases, the CV is approximated by the σ of the logarithmic form of the variable (see A.2.3). To aid in choosing values for penalty weights, a selection of weighting factors and corresponding CV values is given in Table 5.1.

5.5.2 Beta priors

In one case (the steepness of the stock-recruitment relationship) MULTIFAN-CL imposes a non-standard beta distribution prior on the interval 0.2 – 1.0 instead of a normal distribution prior. The shape of the beta prior is set by age flags 153 and 154 (10 times the values of the beta parameters A and B respectively). Table 5.2 shows settings of these age flags to achieve a selection of modes, means, and standard deviations.

Table 5.1: Weighting factors and CV values for normal priors

w — CV	w — CV	w — CV	w — CV
1 — 0.71	7 — 0.27	14 — 0.19	50 — 0.10
2 — 0.50	8 — 0.25	16 — 0.18	100 — 0.07
3 — 0.41	9 — 0.24	20 — 0.16	150 — 0.06
4 — 0.35	10 — 0.22	25 — 0.14	200 — 0.05
5 — 0.32	11 — 0.21	30 — 0.13	500 — 0.03
6 — 0.29	12 — 0.20	40 — 0.11	1000 — 0.02

Table 5.2: Age flag settings and shape of the beta distribution with range 0.2 to 1

age flag 153	age flag 154	mode	mean	sigma	age flag 153	age flag 154	mode	mean	sigma
126	357	0.41	0.41	0.05	368	225	0.70	0.70	0.05
36	86	0.40	0.44	0.10	87	56	0.70	0.69	0.10
12	16	0.40	0.54	0.20	17	14	0.71	0.64	0.20
225	368	0.50	0.50	0.05	357	126	0.80	0.79	0.05
56	87	0.50	0.51	0.10	88	36	0.80	0.77	0.10
14	17	0.49	0.56	0.20	16	12	0.80	0.66	0.20
315	315	0.60	0.60	0.05	269	47	0.90	0.88	0.05
75	75	0.60	0.60	0.10	75	19	0.90	0.84	0.10
15	15	0.60	0.50	0.20	17	11	0.90	0.69	0.20

Setting age flags 153 and 154 both to 10 produces a flat distribution, and setting both less than 10 produces a U-shaped distribution. The derivation of steepness and formulae for calculating the mode, standard deviation, and mean of the beta distribution are given in the Technical Annex (page 97).

Chapter 6

Interpreting Results

6.1 Hessian diagnostic

A MULTIFAN-CL model diagnostic of a non-positive definite Hessian solution may be obtained. The calculations for the covariance matrix is the product of the inverse Hessian and gradients of the dependent variables. This is done because the inverse of the Hessian matrix approximates the variance/covariance matrix of the parameter estimates. Then the Choleski decomposition of the covariance matrix is calculated, and where the determinant has a near-zero value, this indicates a non-positive definite; i.e. the parameters are strongly correlated. If this calculation fails, it can be used as a diagnostic for the matrix not being positive definite.

An improved method uses the singular value decomposition (SVD) to calculate the eigenvalues and eigenvectors of the Hessian. For really difficult, ill-conditioned symmetric matrices, the SVD is more numerically stable. The feature uses the optimized routines in the OpenBLAS dependent libraries and is consequently more efficient than the ADMB decomposition routines, as well as identifying non-positive eigenvalues. This diagnostic is activated using parest_flags(145)=9 [4.5.1] and generates a report file **.svd_report* with the first line being the eigenvalues and subsequent lines being the matrix of the parameter-specific eigenvectors [6.3.14]. The "rule of thumb" diagnostics that an analyst can use to identify a non-positive definite Hessian solution and diagnose the influential parameters are:

- The eigenvalues are in descending order. Check for 0 or negative values that indicate a non-positive definite Hessian, or a parameter that is never used.
- Take the top highest eigenvectors and produce plots for each vector that illustrate the most influential parameters, and sort for the index that links it to the parameter names.
- Do this for all the highest rows and unique parameter names over all the rows, this identifies the most influential parameters in the fit.

For example the minimum, mean and maximum eigenvalues for might be: 2.05617e-11, 46.7, and 1068.7, respectively, indicating a positive-definite solution as there were no zero values. Script is available for plotting each row of the eigenvector matrix so as to identify the most influential parameters, and those causing a failure in this diagnostic, see Appendix [D].

6.2 Output on Screen

For persons experienced in using AD-MODELBUILDER or AUTODIF, the output on the screen from MULTIFAN-CL will be familiar. After a stream of introductory stuff, here is the typical output to the screen (or standard output) from each function evaluation, that is to say, each time the population and fisheries simulation routine runs and the objective function is subsequently calculated: `selectivity pen = 0.00 Av.  F = 0.067044 The fish mort penalty is 1194.576491 after call_penalties pen = 3089.753210 Length frequency data -353667.97 Total catch 1040.98 Total func -349537.23 ========== f eval 21`
Various components of the objective function and its total ("Total func") are shown plus the tally of function evaluations ("f eval"). The above output is shown for every step the minimizer takes along a gradient line. Having determined a minimum value along that line, the minimizer starts a new iteration by establishing a new direction of search at which point additional output is sent to the screen showing the number of parameters being estimated (variables), the maximum gradient magnitude among all the parameters, plus a selection of current parameter values and the gradient of the objective function with respect to each: `230 variables; iteration 15; function evaluation 22 Function value -3.4954e+05; maximum gradient component mag 5.1220e+02`

Table 6.1: Description needed

Var	Value	Gradient	Var	Value	Gradient	Var	Value	Gradient
1	26.38651	-9.75809e+00	2	51.00638	2.45185e+02	3	51.18979	-1.93987e+02
4	-8.56903	-1.32590e-01	5	27.65943	-1.20217e+00	6	29.12218	3.58509e+00
7	204.1072	7.10379e+00	8	179.5699	8.61246e+00	9	222.3152	4.95624e+00
10	208.1248	1.25374e+01	11	214.0105	1.09369e+01	12	236.8858	4.22897e+00
13	239.7104	5.18857e+00	14	256.3488	7.79906e+00	15	263.4407	1.01123e+01
16	257.7195	4.87023e+00	17	256.5459	3.10248e+00	18	255.2254	2.12410e+00
19	252.7260	5.79138e-01	20	250.1414	-8.97963e+00	21	246.9461	-1.36999e+01
22	244.0973	-2.29345e+01	23	239.4727	-3.58491e+01	24	233.3302	1.34997e+00
25	230.4667	2.32083e+00	26	235.1057	1.04368e+00	27	231.5328	4.82628e+00
28	232.5260	4.01869e+00	29	233.4910	3.80338e+00	30	232.5695	4.94573e+00
31	236.8059	2.32086e+00	32	237.8531	2.12982e+00	33	239.3150	8.10424e-01
34	239.9328	-3.62231e-02	35	240.5892	-9.83419e-01	36	240.8852	-1.27347e+00
37	240.5787	-1.95925e+00	38	240.0229	-4.55094e+00	39	239.1912	-3.77607e+00
40	243.5904	-1.48434e+01	41	240.8171	2.30370e+00	42	243.7131	4.33531e+00
43	239.3933	2.56218e+00	44	246.4590	1.78294e+01	45	244.7752	1.40266e+01
46	242.5358	8.27873e+00	47	244.2200	1.11540e+01	48	240.9609	9.32867e+00
49	240.8908	9.95465e+00	50	238.3860	2.82602e+00	51	236.8014	5.14381e+00
52	234.6456	1.45444e+00	53	232.0680	-1.15390e+01	54	229.8253	-1.54616e+01

Note that the parameter values are scaled internally by the fitting procedure. Therefore the values shown are not equivalent to the parameter values as a human user would know them. In any case, the parameters ("variables" in the output) are simply numbered and not given labels that would be familiar to the user.

6.2.1 What to look for

Evidence that the minimizer is making progress in improving the fit of the model is given by the "Function value" and the "maximum gradient component mag". Because there is large amount of other output,

these can be difficult to see, particularly when the output is scrolling quickly. To cope with that a utility, "condensed.list" was created to summarize results (see page 65). Here is some typical output: Function value -6.0376651e+05; maximum gradient component mag -5.9330e+00 ..... Function value -6.0376815e+05; maximum gradient component mag -2.0133e+00 ..... Function value -6.0377007e+05; maximum gradient component mag -2.7934e+00 ..... Function value -6.0377163e+05; maximum gradient component mag 1.9783e+00 ..... A single, salient output record is printed for each iteration of the minimizer, and a dot is printed for each linear step in between. Note that the function value is decreasing slightly with each iteration. The maximum gradient magnitude is also decreasing but not uniformly. It is typical that the maximum gradient magnitude decreases in fits and starts. Sometimes it can increase markedly if the minimizer encounters steeper topography in the objective function, in which case the function value should decrease more quickly for a while.

6.2.2 How to intervene

It is likely that during the course of a model fit, you will want to interrupt the program for one reason or another. The procedure for interrupting differs between Linux and Windows implementations of MULTIFAN-CL. In Linux, interruption is initiated by pressing Ctrl-c, in Windows by pressing q. MULTIFAN-CL will then output the following prompt: press q to quit or c to invoke derivative checker: To terminate the program without saving the current values of the active parameters, simply press Ctrl-c (in fact in Windows this sort of termination can be achieved simply by typing a single Ctrl-c during program execution). To terminate the program saving the current values of active parameters (and any flag settings that may have been changed using the -switch command line sequence), press q <Enter> in response to the above prompt. The optimum point in the program operation is within 0.5 seconds after the screen output "calling gradcalc" is displayed. Current values of active parameters and flag settings are saved in the output *.par* file specified on the command line. The program may then be restarted from this point by re-running MULTIFAN-CL and specifying this output *.par* file as the new input *.par* file in the new command line. The option to invoke the derivative checker is a feature used in debugging. It allows the derivatives of all or specified active parameters with respect to the objective function to be computed using the automatic differentiation code and by finite differences and the derivative values compared.

6.2.3 When to intervene

There are some specific instances in which intervention and other actions are required for troubleshooting purposes. The following is the most common problem that occurs requiring human intervention and remedial action. Problem: The function minimizer is not making progress, i.e., the function value is not changing even though convergence has not been reached (gradients of active parameters are still large). This condition may occur when the function minimizer is in a "bad" part of the parameter space and cannot escape. Solution 1: This problem can often be solved by re-scaling the gradients, as extreme variation in gradients among active parameters can cause problems for function minimizers. Terminate the program, saving the current results using the Ctrl-C q <Enter> sequence as described above. Then re-run the program using the saved output *.par* file as the new input *.par* file and set parest flag 152 to 1 using the command line argument -switch 1 1 152 1. Solution 2: Occasionally, re-scaling the gradients is not sufficient for the function minimizer to be able to escape the black hole that it has fallen into. If the catch likelihood component has become very large, examine the catch fit (in the plot.rep file) and determine which fishery's catch is not being fitted. Then inspect the *.par* file and find the section labeled # average catchability coefficients. If the average catchability for the problem fishery is conspicuously low or high, change it to a more reasonable value, save the file (to a new file name), and re-run the program specifying the modified input *.par* file.

6.3 Output Files

6.3.1 The *.par* file

The *.par* file is mentioned here because it is an output file as well as an input file. It contains all the basic parameter estimates from the fitting phase in which it was produced. Details of the content and format are given above (section 4.1.5).

6.3.2 The *plot.rep* file

The *plot.rep* file contains some structural information about the fit and a lot of information derived from the estimated parameter values. The main purpose of this file is to facilitate graphical presentation of the MULTIFAN-CL results. Here is an annotated listing of an example *plot.rep* file as generated by MULTIFAN-CL version 2.0.2.1 that produces the *plot.rep* version 4.0. Comments on particular sections of the file are provided at the end of this listing.

Table 6.2: Main block

Label	Description and size of following data
# MULTIFAN-CL Viewer	The file header
# 4.0	The current version number
# Frq file = *.frq	The name of the input .frq file
# Input par file = *.par	The name of the input .par file
# Output par file = *.par	The name of the output .par file
# MULTIFAN-CL version number: 2.0.2.1	The version of the MULTIFAN-CL .exe
# Number of time periods	Integer, iyr
# Year 1	Integer, year1
# Number of regions	Integer, ir
# Number of species	Integer, ispp
# Number of age classes	Integer, iage
# Regions species pointer	Integer vector of length ir
# Fishery species pointer	Integer vector of length ifish
# Number of length bins	Integer, ilen
# Number of recruitments per year	Integer, nrec
# Number of fisheries	Integer, ifish
# Fishery selectivity seasons	Integer vector of length ifish
# Fishery selectivity time-blocks	Integer vector of length ifish
# Number of realizations per fishery	Integer vector of length ifish, nr
# Region for each fishery	Integer vector of length ifish
# Time of each realization by fishery (down)	Ifish real vectors of length nr(ifish)
# Mean lengths at age	Real vector of length iage
# SD of length at age	Real vector of length iage
# Mean weights at age	Real vector of length iage
# Natural mortality at age	Real vector of length iage
# Selectivity by age class (across) and fishery (down)	Ifish by iage real matrix
# length bin mid-points	Real vector of length ilen
# length-based selectivity by fishery	Ifish by ilen real matrix
# Catchability by realization (across) by fishery (down)	Ifish real vectors of length nr(ifish)

Table 6.2: Main block

Label	Description and size of following data
# Catchability+effort dev. by realization (across) by fishery (down)	Ifish real vectors of length nr(ifish)
# Fishing mortality by age class (across) and year (down)	Iyear by iage real matrix (each matrix separated by # Species spp tag)
# Fishing mortality by age class (across), year (down) and region (block)	Ir iyear by iage real matrices (each matrix separated by # Region r tag)
# Population number by age (across), year (down) and region	Ir iyear by iage real matrices (each matrix separated by # Region r tag)
# Exploitable population biomass by fishery (across) and year (down)	Iyear by ifish real matrix
# Exploitable population in same units as catch by fishery (across) and year (down)	Iyear by ifish real matrix
# Absolute biomass by region (across) and year (down)	
# Recruitment	Iyear by ir real matrix
# Total biomass	Iyear by ir real matrix
# Adult biomass	Iyear by ir real matrix
# Relative biomass by region (across) and year (down)	Iyear by ir real matrix
# Observed catch by fishery (down) and time (across)	Ifish real vectors of length nr(ifish)
# Predicted catch by fishery (down) and time (across)	Ifish real vectors of length nr(ifish)
# Observed CPUE by fishery (down) and time (across)	Ifish real vectors of length nr(ifish)
# Predicted CPUE by fishery (down) and time (across)	Ifish real vectors of length nr(ifish)

The first two lines are comments indicating the version number of the file, which is important because the format and content of this file has been evolving. The remaining entries provide structural information as indicated by the comments. Note that the "# Number of realizations per fishery" is a vector with one element for each fishery giving the number of fishing records for that fishery in the *.frq* file. The next entry "# Time of each realisation by fishery (down)" is a ragged array with a row for each fishery and variable lengths of the rows as given by the elements of the preceding vector. The section "# Selectivity by age class (across) and fishery (down)" is a matrix[(nfish, time-block, season),nage] with a particular order in respect of the rows depending upon the time-variant selectivity specified for each fishery: - Fishery number - Season - Time-block In case where time-invariant selectivity is specified, the number of rows will be the number of fisheries. However, for fisheries having time-invariant selectivity, e.g. seasonal selectivities (by default 4) and n time-blocks, there will be $4 \times n$ rows of selectivities for this fishery, with the first n rows being for season 1 (among n time-blocks), the next n rows being for season 2 (among n time-blocks), etc. The "# Exploitable population by fishery (across) and year (down)" section is reported dependent upon the selectivity structures that may account for time-blocks and seasons. Where seasonal selectivities are applied, the vector elements of these quantities will be extended to include this dimension. The region-specific matrices in the section labelled "# Absolute biomass by region (across) and year (down)" are indexed in respect of region by the section "# Regions species pointer". Therefore, for the cases of multiple sexes/species, this pointer can be used to identify the quantities in respect of sexes/species.

Table 6.3: Yield block

Label	Description and size of following data
# Yield analysis option: 0=none, 1=Bev&Holt, 2=Pella Tomlinson	Indicates the type of the following yield analysis information

Depending of the value of the yield analysis option there can be 3 different configurations for the yield block:

- If the value is 0, then there is no special data.
- If the value is 1:

Table 6.4: Yield block: option 1

Label	Description and size of following data
# Beverton-Holt stock-recruitment relationship report	Title
# BH autocorrelation 7.421e-02	Estimated or derived autocorrelation coefficient
# alpha = 5.552e+09 beta = 3.697e+06 steepness = 1.502e+03 variance = 3.444e-01	BH-SRR parameters and variance
# Observed spawning biomass	Real vector
# Observed recruitment	Real vector, same length as previous
# Spawning biomass	Real vector
# Predicted recruitment	Real vector, same length as previous
# Beverton-Holt yield analysis report	Subtitle
# MSY	Real
# F multiplier at MSY	Real
# F at MSY	Real
# Total biomass at MSY	Real
# Adult biomass at MSY	Real
# current Total Biomass to Total biomass at MSY	Real
# current Adult Biomass to Adult Biomass at MSY	Real
# Effort multiplier	Real vector
# Equilibrium yield	Real vector, same length as previous
# Equilibrium adult biomass	Real vector, same length as previous
# Equilibrium total biomass	Real vector, same length as previous
# Adult biomass over adult biomass at MSY	Real vector of length iyr
# Total biomass over total biomass at MSY	Real vector of length iyr
# Aggregate F over F at MSY	Real vector of length iyr
# Aggregate F	Real vector of length iyr
# Yield per recruit report	Title
# Effort multiplier	Real vector
# Yield per recruit	Real vector, same length as previous

The "# Beverton-Holt stock-recruitment relationship report" section will have multiple sections for the case of multiple species, with the species index being displayed within the label. For the case of multiple sexes, this section relates only to the female sex, with its species index displayed within the label. In the case of multiple sexes, the equilibrium catch calculations are duplicated over both sexes, and the total equilibrium catch among both is reported in respect of the determinant quantities for the female sex only, e.g. observed

biomass, F multiplier.

- If the value is 2:

Table 6.5: Yield block: option 2

Label	Description and size of following data
# Pella_t yield analysis report	Title
# Yield per recruit report	Title
# Effort multiplier	Real vector
# Yield per recruit	Real vector, same length as previous

Table 6.6: Optional data block: tagging

Label	Description and size of following data
# Tag reporting rates	This label and following data will not appear if analysis does not incorporate tagging data
# Grouping indicator (0 = no grouping, >0 = grouping)	Integer vector of length ifish – if elements are the same, these fisheries are grouped for purpose of tag returns
# Time series variation in reporting rates (0 = no, >0 = yes)	Integer flag, 0=no time series, >0=time series
# tag fish rep group flags	Ifish by ntag+1 integer matrix; where ntag is the number of tagged groups
# Reporting rates by fishery by tag group (no time series variation)	Ifish by ntag+1 real matrix
# No. of time periods associated with tag returns	Integer, ntag
# Time periods associated with grouped tag returns	Real vector of length ntag
# Observed tag returns by time period (across) by fishery groupings (down)	If all grouping flags = 0, this is a ifish by ntag integer matrix otherwise, this is an integer matrix with the number of rows corresponding to the number of unique groups and with ntag columns
# Predicted tag returns by time period (across) by fishery groupings (down)	If all grouping flags = 0, this is a ifish by ntag real matrix otherwise, this is a real matrix with the number of rows corresponding to the number of unique groups and with ntag columns
# Maximum time at liberty	Integer, ilib
# Observed vs predicted tag returns by time at liberty	Two column vectors (first integer, second real) of length ilib

Table 6.7: Optional data block: movement, fishing impact analysis

Label	Description and size of following data
# Movement analysis	Ir real (2*iage+1) by ir matrices – matrices are separated by the label "# Region r"
# Total biomass in absence of fishing	Iyear by ir real matrix
# Adult biomass in absence of fishing	Iyear by ir real matrix

Table 6.7: Optional data block: movement, fishing impact analysis

Label	Description and size of following data
# Exploitable populations in absence of fishing	Iyr by ifish real matrix
# Predicted catch for interaction analysis by fishery (down) and time (across)	Ifish real vectors of length nr(ifish)

6.3.3 The *length.fit* file

Predicted size distributions are output to files called "length.fit" and "weight.fit", depending upon the size data input to the model fit, along with the observed size data to which the predicted distributions should be compared. The two *.fit file types share an essentially an identical format, and a description is provided here in the context of the length frequency data only ("length.fit"), but applies equally to the weight frequency data. These files contain only standard ASCII characters and are line-based. Separators can be space or tabs; the # character is used for commented lines. There may be empty lines that separate the different data blocks, you may also find comment lines anywhere within the file. Note that the "weight.fit" file includes additional commented lines compared to the "length.fit" file, but the format of the values are identical. *VERSION* The very first line of the file is a comment line that displays and identifies the file version number; it supplies no data values. An example follows.

Table 6.8: VERSION

Label	Description and size of following data
# FIT 2	The file header with version number

HEADER First line contains ten values, most of them being unused, e.g. 0 0 HISTOGRAM_NB 0 0 0 0 0 0 0

Table 6.9: HEADER - line 1

Label	Description and size of following data
Elements 1 - 2	Unused
HISTOGRAM_NB	Number of histograms to be displayed (may not be set otherwise by the user)
Elements 4 - 10	Unused

The second line contains constants used to draw histograms of the observed and predicted frequencies. LENGTH_INTERVAL_NB SMALLEST_LENGTH LENGTH_INTERVAL_WIDTH e.g. 95 10 2

Table 6.10: HEADER - line 2

Label	Description and size of following data
LENGTH_INTERVAL_NB	Number of length intervals
SMALLEST_LENGTH	Length of the smallest interval
LENGTH_INTERVAL_WIDTH	The width of length intervals

The third line contains a single integer, being the number of defined fisheries + 1. FISHERIES_NB

Table 6.11: HEADER - line 3

Label	Description and size of following data
FISHERIES_NB	Integer number of fisheries defined + 1; the additional fishery is a summary of all the others

The fourth line contains a vector of length FISHERIES_NB; with each of the first FISHERIES_NB - 1 elements being the number of observed samples for each of the defined fisheries (1 to N), with the last element being equal to the integer value of FISHERIES_NB - 1 (because it is a summary for each fishery): SAMPLE_NB_1 SAMPLE_NB_2 ... SAMPLE_NB_N FISHERIES_NB - 1 e.g. for a model with 33 fisheries defined 0 0 0 0 0 0 0 0 52 0 0 73 73 95 89 90 73 73 68 51 79 76 61 0 0 85 75 14 42 0 12 8 0 33

Table 6.12: HEADER - line 4

Label	Description and size of following data
SAMPLE_NB_X	Integer number of samples for defined fisheries #X and with a final summary "fishery"

The fifth line contains the number of age classes. AGE_CLASS_NB

Table 6.13: HEADER - line 5

Label	Description and size of following data
AGE_CLASS_NB	Integer number of age classes defined

The sixth line contains an index of the species associated with each fishery, being a fisheries-species pointer. FISHERY_SPECIES

Table 6.14: HEADER - line 6

Label	Description and size of following data
FISHERY_SPECIES	Integer vector of length FISHERIES_NB-1

FISHERIES BLOCK

After the file header there are FISHERIES_NB fisheries blocks. Each data block begins with the following line that identifies the fishery to which the data relates. # fishery X

Table 6.15: FISHERIES BLOCK - line 1

Label	Description and size of following data
# fishery X	Header for fishery block number X, fisheries numbering from 1 to FISHERIES_NB - 1
# fishery totals	Header for the final fishery block number FISHERIES_NB, being the summary over all the fisheries 1 to FISHERIES_NB - 1

SAMPLE BLOCK Each fishery data block is made of SAMPLE_NB_X sample blocks. The first line of the sample block is a time identifier for this sample. YEAR MONTH WEEK

Table 6.16: SAMPLE BLOCK - line 1

Label	Description and size of following data
YEAR	Year of the record
MONTH	Month of the record
WEEK	Week of the record

The second line of the sample block contains a real vector, of length AGE_CLASS_NB elements, being the mean lengths at age expressed as a standardised value with respect to the size-class intervals 1 to LENGTH_INTERVAL_NB (the number of size-class intervals). These values are specific to the sex/species relating to the fishery as indicated in the FISHERY_SPECIES vector pointer. They are used to define the position of vertical lines within a histogram chart generated in the Viewer.

Table 6.17: SAMPLE BLOCK - line 2

Label	Description and size of following data
MEAN_LEN_AGE_X	Real vector of length AGE_CLASS_NB; Scaled mean length in each age class X

The third line of the sample block contains the proportion sum, typically having a value = 1.

Table 6.18: SAMPLE BLOCK - line 3

Label	Description and size of following data
PROPORTION_SUM	Real, sum of proportions

The fourth line of the sample block contains the observed proportions in each length interval, being a real vector of length LENGTH_INTERVAL_NB elements.

Table 6.19: SAMPLE BLOCK - line 4

Label	Description and size of following data
OBS_PROPS_LEN_X	Real vector of length LENGTH_INTERVAL_NB; observed proportions in each length interval X

The fifth line of the sample block contains the predicted proportions in each length interval, being a real vector of length LENGTH_INTERVAL_NB elements.

Table 6.20: SAMPLE BLOCK - line 5

Label	Description and size of following data
PRED_PROPS_LEN_X	Real vector of length LENGTH_INTERVAL_NB; model predicted proportions in each length interval X

The following AGE_CLASS_NB lines of the sample block contain a real matrix (AGE_CLASS_NB by LENGTH_INTERVAL_NB) with the elements of each row being the predicted proportion of fish occurring in each particular length interval for the age class.

Table 6.21: SAMPLE BLOCK - lines 6 to (5 + AGE_CLASS_NB)

Label	Description and size of following data
PRED_PROPS_LEN_AGE	Real matrix AGE_CLASS_NB by LENGTH_INTERVAL_NB; model predicted proportions in each length interval for a given age class

6.3.4 The *tag.rep* file

6.3.5 The *.var* file

6.3.6 The *.dep* file

6.3.7 The *.hes* file

6.3.8 The *catch.rep* file

The catch.rep files are text files containing output data from MULTIFAN-CL models, including total annual catches, and fishery-specific annual catches.

These files contain only standard ASCII characters and are value-based. Separators can be space or tabs; the "#" character is used for commented lines. There is not a special header that identifies a file, though comment lines may indicate what's in the file. VERSION 1 *Data blocks* The first line is the following comment: # Total catch by year The second line contains the total of catches in weight over all fisheries (the sum over fishery 1 to fishery Ifish) in each model year N, FtotannN.

Table 6.22: DATA BLOCK - line 2

Label	Description and size of following data
FtotannN	A vector of real numbers of length equal to the number of yars in the model, Iyear (i.e. fishing incident periods, e.g. quarters). Each vector element is the sum total of catches in weight over all fisheries in model year number N

The third line is the following comment: # Catch by year (across) by fishery (down) The next Ifish lines is a matrix of the total annual catches in weight by fishery.

Table 6.23: DATA BLOCK - lines 4 to 3 + Ifish

Label	Description and size of following data
FannN	An Ifish by Iyear real matrix of the total annual catches in weight by fishery (rows). Each row element is the annual catch in weight for the respective fishery in model year N

Careful note should be taken of the multi-species or multi-sex case where by default fisheries are defined for each species/sex. In this case, the values in the catch.rep are as described above for the single species case, i.e. aggregated over all fisheries. Therefore, where the input catches are aggregated among species/sexes, the catch.rep values for annual catch totals will duplicate the model catches for these fishing incidents. This report does not provide the catch totals specific to each species/sex unless all the input catches are dis-aggregated by species/sex.

6.3.9 The *ests.rep* file

6.3.10 The *agelengthresids.dat* file

The negative log-likelihood value for each age-length sample is reported, along with the other objective function components (catch data, tagging data, etc.) to the general output report *test_plot_output* in a section titled # age length likelihood (see 6.3.11). An output report agelengthresids.dat is generated for the detailed examination of the residuals of the age-length likelihoods for each sample. An example of the format is: Fishery 2 Year 2010 Month 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.25 0 0 0 0 0 0 0 0 0.5 0 0 0 0.25 ... 0 0 0 0 0 0 0 0 0 0 0 0.00137273 0.00339324 8.2288e-06 2.19484e-05 ... ... Where the header informs from what fishery, year and month the sample is derived, with three rows being reported for each length interval in the sample: an integer for the sum of the observed proportions (usually = 1), a vector for the observed proportions-at-age, and a vector for the predicted proportions-at-age. Using this information the analyst can directly calculate the residual with respect to a preferred formulation.

6.3.11 The *test_plot_output* file

A report file called "test_plot_output" is produced by default which provides details for creating a table listing the likelihood components making up the total objective function. This is of utility for computing effective samples sizes and other summary statistics related to model fit. This is a powerful diagnostic tool for model development that facilitates the examination of the effects of alternative relative weightings for the input data types and to display conflicts among these data. The report file *test_plot_output* has the following content:

> # BH_steep contribution – the BH-SRR penalty # Effort_dev_penalty_by_fishery – the effort deviates penalty # catchability_dev_penalty_by_fishery – the catchability deviates penalties by fishery # catchability_dev_penalty_by_group – the catchability deviates penalties by fishery grouping # total length component of likelihood for each fishery – the total for each fishery # length-sample components of likelihood for fishery x – by each fishery and realisation sampled # total weight component of likelihood for each fishery – the total for each fishery # weight-sample

components of likelihood for fishery x – by each fishery and realisation sampled # total catch components of likelihood for fishery x – by each fishery and realisation sampled # Tag likelihood by tag release by fishery groups – by release group and fishery grouping within each # age length likelihood – by each sample

6.3.12 The *temporary_tag_report* file

A report file is produced conditional upon parest flag 187 = 1 that is a detailed report of recaptures by release group, fishery, and age. The report is formatted in respect of each release group with rows for the predicted and observed recaptures by region, fishery, year, month, and ages in the columns. The fishery strata are the true fisheries and not the fishery grouping as specified for the tags (fish flags 32).

6.3.13 Size composition estimated effective sample size reports

A report file is produced from the fit to size composition data for the estimated effective sample sizes in respect of the likelihood option selected. The report file names for the length and weight frequency data, respectively, in respect of each option are:

Robust-normal: *lsizemult*; *wsizemult*
SSMULT_noRE and SSMULT_RE: *sizemult*; *wsizemult*
Dirichlet-Multinomial: *dmsizemult*; *wght_dmsizemult*

The format of the report files are the same for the length and weight frequency data but differ in respect of each likelihood option.

For the Robust-normal the columns are:
1. region
2. time period
3. fishing incident
4. fishery number
5. estimated effective sample size
6. observed sample size
7. sum of the ratio SSQ:multinomial variance

For the SSMULT-RE and SSMULT-noRE the columns are:
1. estimated effective sample size
2. ratio observed sample size:average observed sample size
3. observed sample size
4. fishery group number for covariate parameter estimation
5. fishery number

For the Dirichlet-Multinomial-noRE the columns are:
1. fishery group number for covariate parameter estimation
2. assumed upper bound on estimated effective sample size
3. estimated effective sample size
4. lambda parameter value for fishery group
5. sample size covariate parameter value
6. fishery group average estimated effective sample size
7. relative estimated effective sample size value 1
8. relative estimated effective sample size value 2

9. moment estimate of the estimated effective sample size
10. observed sample size

6.3.14 Hessian Diagnostic Report

A report file is produced when the Hessian diagnostic feature is activated [6.1]. The first line is a vector of length *npars* being the eigenvalues in descending order. This is followed by *npars* lines being the matrix of the parameter-specific eigenvectors, with *npars* values in each line.

An example of the report follows.

```
Eigenvalues
1068.67 813.976 665.256 614.74 ...

Eigenvectors
-0.0119025 -0.0776463 -0.092855 ...
6.38932e-05 -3.67808e-05 7.5739 ...
0.0608211 -0.0579217 -0.0124353 ...
0.00186207 0.0655339 0.0650501 ...
...
```

6.4 Obtaining Graphical Results

6.4.1 Java

A graphical viewer of MULTIFAN-CL output written in Java is available on the MULTIFAN-CL web site. It comes with its own documentation –"MULTIFAN-CL Viewer User Manual". It requires a Java interpreter compatible with at least the Java 2 version 1.3 Standard Edition specifications. This viewer is excellent for getting quick visual access to the multitude of information output by MULTIFAN-CL.

6.4.2 R (or S)

For those wishing to use of R, the public domain statistics and graphics program, R functions are available for graphical presentation of MULTIFAN-CL output. They can be easily adapted to S, the commercial counterpart of R. The R functions give a more flexibility than the MULTIFAN-CL viewer in designing graphical presentations for incorporating into computer presentation software or for publication. R is available for both LINUX/UNIX and MS-Windows operating systems and can be obtained free of charge from http://cran.r-project.org. The functions make extensive use of the LINUX/UNIX utility, *awk*, which for MS-Windows is contained in cygwin, available free of charge from http://cygwin.com. The R functions are organized into two libraries, one with functions designed to access the various parts of the MULTIFAN-CL output files, and the other with functions designed to produce a variety of graphics, and making use of the functions in the first set. Both sets are continuously expanding. Therefore this description is necessarily incomplete. Users are advised to get the latest version of the libraries.

6.5 Set 1 – MULTIFAN-CL data access functions

The access functions are supplied in library *getplotstuff*, and are useful for those wishing to create their own graphics. Most of the functions are designed to return R objects from the *plot.rep* and there are functions corresponding to most elements in the file (see 6.2.2. Other access functions retrieve data from the *.var*, giving information on confidence intervals, and other output, and some input, files as well. Documentation of the functions is available with the R documentation system. On most systems, issuing help.start() within an R session will open a web browser with a link to the *getplotstuff* library. Alternatively, issuing ?*function_name* within an R session for any of the functions in the following list will list documentation for that particular function. The list below gives only a short description of the type of R object returned along with dimension information for objects consisting of more than a single number. Note that "lists" serve as ragged arrays for which nested dimension information is given, e.g., "[fishery[time]]" indicates a time series vector for each fishery perhaps with a different number of elements in each vector.

Table 6.24: Functions that access the *plot.rep* file

Function	Object returned	Content
getnpd	number()	Number of time periods
getnyr	number()	Number of years
getyear1	number()	Starting year
getyrs	vector()	Vector of years [year] – for plotting annual data
getimes	vector()	Vector of times [time] – for plotting data at each fishing incident
getnreg	number()	Number of regions
getnages	number()	Number of age classes
getnrec	number()	Number of recruitments per year
getnfish	number()	Number of fisheries
getnreal	vector()	Number of realizations [fishery]
getrtimes	list()	Time of each realization [fishery[time]]
getlen.age	vector()	Mean lengths [age]
getlen.age.sd	vector()	SD of length at [age]
getwt.age	vector()	Mean weights [age]
getM.age	vector()	Natural mortality [age]
getselect	matrix()	Selectivity [fishery, age]
getqlist	list()	Catchability [fishery[time]]
getq	data.frame()	Catchabilities [time,c(fisheries,time)]
getqedlist	list()	Catchability+effort dev. [fishery[time]]
getqed	data.frame()	Catchability+effort dev. [time,c(fisheries,time)]
getFya	matrix()	Fishing mortality [year, age class]
getFyar	array()	Fishing mortality [year, age class, region]
getNyar	array()	Population number [year, age class, region]
getNya	matrix()	Population number [year, age class]
getNyr	matrix()	Population number [year, region]
getNexp	matrix()	Exploitable population [time, fishery]
getBabs	matrix()	Absolute biomass [time, region]
getBadult	matrix()	Adult biomass [time, region]
getBrel	matrix()	Relative biomass [time, region]
getBnof	matrix()	Total biomass in absence of fishing [time, region]
getColist	list()	Observed catch [fishery[time]]
getCo	data.frame()	Observed catch [time,c(fisheries,time)]
getCplist	list()	Predicted catch [fishery[time]]

Table 6.24: Functions that access the *plot.rep* file

Function	Object returned	Content
getCp	data.frame()	Predicted catch in each [time,c(fisheries,time)]
getCPUEolist	list()	Observed CPUE [fishery[time]]
getCPUEo	data.frame()	Observed CPUE [time,c(fisheries,time)]
getCPUEplist	list()	Predicted CPUE [fishery[time]]
getCPUEp	data.frame()	Predicted CPUE [time,c(fisheries,time)]
getFmsy	number()	F at MSY
getBmsy	number()	Biomass at MSY
getSBmsy	number()	Adult biomass at MSY
getFtoFmsy	vector()	Ratio of F to F at MSY [time]
getBtoBmsy	vector()	Ratio of biomass to biomass at MSY [time]
getSBtoSBmsy	vector()	Ratio of spawning biomass to spawning biomass at MSY [time]
getMSY	number()	MSY
getYieldAnal	list()	Yield analysis report [Fmult[n], Yeq[n], SBeq[n], Beq[n], MSY, Fmsy, Bmsy, SBmsy, SBtoSBmsy[time], BtoBmsy[time], FtoFmsy[time], F[time]] – vectors Fmult, Yeq, SBeq, and Beq give projections of yield, spawning and total biomass with increasing F multiplier. Length of vectors, n, depends on how many Fmult steps it takes to bring yield to zero. Fmsy, Bmsy, SBmsy, SBtoSBmsy, BtoBmsy, FtoFmsy contain same data as correspondingly named functions above.
getYPR	list()	Yield per recruit report Fmult[n],YPR[n]
getBHSR	list()	Beverton-Holt stock-recruitment report [alpha, beta, slope, spawnB[n], recrut[n]]
gettagretbydate	data.frame()	Observed and predicted tag returns by calendar time
gettagretbyliberty	data.frame()	Observed and predicted tag returns by time at liberty

Table 6.25: The following functions access the *.var* file

Function	Object returned	Content
getF2Fmsy.sd	data.frame()	Ratio of F to Fmsy by time period with upper and lower confidence bounds
getB2Bmsy.sd	data.frame()	Ratio of biomass to biomass at MSY by time period with upper and lower confidence bounds
getSB2SBmsy.sd	data.frame()	Ratio of spawning biomass to spawning biomass at MSY by time period with upper and lower confidence bounds
getRecrel.sd	data.frame()	Relative recruitment with confidence bounds
getRec.sd	data.frame()	Absolute recruitment with confidence bounds
getBrelt.sd	data.frame()	Relative biomass with confidence bounds
getBabst.sd	data.frame()	Absolute biomass with confidence bounds
getyield.sd	data.frame()	Yield curve with confidence bounds
getM.sd	data.frame()	Natural mortality at age with confidence bounds

6.6 Set 2 – MULTIFAN-CL graphics plotting functions

The plotting functions are supplied in a library *doplotstuff*. These functions are evolving at an even greater pace than the data access functions. Therefore the documentation is not as complete. However, the documentation is available by issuing help.start() within an R session. Two functions of note are *plotmedley()* and *quickplot()*. These functions make a composit plots using other plotting functions. They are a quick way to get a visual impression of intermediate, or final, results of a MULTIFAN-CL analysis. They are not as thorough in this regard as the MULTIFAN-CL Viewer, but they have been incorporated into unix/linux scripts, *medley()* and *quickplot()*, giving very quick access to some of the more cogent MULTIFAN-CL output material.

Appendix A

Appendix: Technical Annex

A.1 Population dynamics model

A.1.1 Age-structured dynamics

The dynamics of untagged fish within a region are governed by the following equations:

Table A.1: Equations that govern the dynamics of untagged fish within a region

$$N'_{\text{atr}} = \begin{cases} Re^{\varphi \text{t}}\alpha_r\gamma_{\text{tr}} & ;\ a = 1; 1 \leq t \leq T \\ N'_{a1r} & ;\ 1 < a < A; t = 1 \\ N_{a-1,t-1,r}e^{-Z_{a-1,t-1,r}} & ;\ 1 < a < A; 1 < t \leq T \\ N_{a-1,t-1,r}e^{-Z_{a-1,t-1,r}} + N_{a,t-1,r}e^{-Z_{a,t-1,r}} & ;\ a = A; 1 \leq t \leq T \end{cases} \tag{A.1}$$

where

N'_{atr} is the number of age-class a fish at the beginning of time period t in region r r before movement has taken place,

N_{atr} is the number of age-class a fish at the beginning of time period t in region after movement has taken place,

R is temporal average of spatially aggregated recruitment,

φ_t determines a multiplicative deviation, e^{φ_t}, at time t of spatially aggregated recruitment from average recruitment, R, with constraint that $\sum_t \varphi_t = 0$,

α_r is average recruitment to region r as proportion of R, $\sum_r \alpha \text{r} = 1$,

γ_{tr} is additional recruitment deviation at time t and region r, $\prod_t \gamma_{\text{tr}} = 1$ and $\sum_r \alpha_r\gamma_{\text{tr}} = 1$,

A is total number of age classes,

T is total number of time periods,

Z_{atr} is instantaneous rate of total mortality of age-class a in time period t in region,

F_{atf} is instantaneous rate of fishing mortality of age-class a in time period t by fishery f,

f_r is the set of fishery indices of fisheries occurring in region r, and

M_a is instantaneous rate of natural mortality of age-class a.

Depending on setting of age flag 94 (see section 4.5.12), the initial abundances N'_{atr} by region are either estimated parameters or they are functions of regional recruitments (N'_{atr}) and either natural or total mortalities (M_a or Z_{atr}) averaged over time periods 2 to k:

$$N'_{\text{atr}} = \begin{cases} N'_{1r} \exp\left[-\sum_{a'<a} \tilde{Z}_{a'r}\right] & \& \quad ; \quad 1 < a < A \\ \frac{N'_{1r}}{1-\exp\left(-\tilde{Z}_{\text{ar}}\right)} \exp\left[-\sum_{a'<a} \tilde{Z}_{a'r}\right] & \& \quad ; a = A \end{cases}$$

$$N'_{1r} = \frac{1}{(k-1)} \sum_{t=2}^{k} N'_{1tr}$$

$$\tilde{Z}_{\text{ar}} = \begin{cases} M_a & \& \quad ; \quad \text{age\_flags}(94) = 1 \\ \frac{1}{(k-1)} \sum_{t=2}^{k} Z_{\text{atr}} & \& \quad ; \quad \text{age\_flags}(94) = 2 \end{cases}$$

A.1.2 Natural Mortality

Natural mortality at age, M_a, is actually parameterized as an average natural mortality modified by age-specific deviations as follows:

$$M_a = M e^{\varepsilon_a}$$

where the mean of ε_a is constrained to zero. The deviations can either be fixed (default = 0) or estimated depending on flag settings (see section 3.5).

Functional forms for natural mortality at age are parameterized either by a spline function or a formulation as per Lorenzen[12]:

$$M_a = c\,(l_a)^b$$

Where l_a is the mean length of age class a. Now the mean length is parameterized by the von Bertalanffy (see A.1.4). To achieve numerical stability, the dimension of the mean length in the von Bertalanffy function has been "undimensionalised" by parameterizing it as:

$$l_a = d + (1-d)\left[\frac{1 - exp(-K(a-1))}{1 - exp(-K(A-1))}\right]$$

where

L_1 is the mean length of the first age class,

L_A is the mean length of the oldest age class, and

$d = \frac{L_1}{L_A}$

A.1.3 Movement

Movement is assumed to occur instantaneously at the beginning of each time period. Let $N^{'}_{\text{at}}$ and N_{at} be pre- and post-movement abundance vectors with elements $N^{'}{}_{\text{atr}}$ and N_{atr} for each region $r = 1, \cdots, R$. Defining ν_a^{rs} as the coefficient of movement from region r to region s at age a, abundance after movement is given by:

$$N_{\text{atr}} = N^{'}_{\text{atr}} - \left(\sum_{s \neq r} \nu_a^{\text{rs}} \right) N_{\text{atr}} + \sum_{s \neq r} \nu_a^{\text{sr}} N_{\text{ats}}$$

or in matrix form:

$$\text{B}_a \text{N}_{\text{at}} = \text{N}^{'}_{\text{at}}$$

where

$$\text{B}_a = \begin{bmatrix} 1 + \sum_{j \neq 1} \nu_a^{j1} & . & . & . & -\nu_a^{1i} \\ . & . & . & . & . \\ -\nu_a^{1i} & . & 1 + \sum_{j \neq 1} \nu_a^{j1} & . & -\nu_a^{1i} \\ . & . & . & . & . \\ -\nu_a^{1i} & . & . & . & 1 + \sum_{j \neq 1} \nu_a^{j1} \end{bmatrix}$$

The movement transition is thus

$$\text{N}_{\text{at}} = \text{B}_a^{-1} \mathbf{N}^{'}{}_{\text{at}}$$

Note that this is a fully implicit transition which guarantees numerical stability even with large rates of movement (Press et al., 1988). Also, even though ν_a^{rs} and ν_a^{sr} are normally set to zero if region r and s do not share a border, the implicit transition still allows some movement from one to the other within a time step.

Age dependency of the movement coefficients is specified by

$$\nu_a^{\text{rs}} = \phi_0^{\text{rs}} \exp\left(\phi_1^{\text{rs}} \left(\kappa_a \right)^{\phi_2^{\text{rs}}} \right)$$

where κ_a is age scaled from -1 to +1, i.e. $\kappa_a = (2a - A - 1)/(A - 1)$. Movement will either increase, stay constant, or decrease with age depending on whether ϕ_1^{rs} is greater than, equal to, or less than zero, and ϕ_2^{rs} adds non-linearity to the age dependency.

Movement parameters estimated by model fit are ϕ_0^{rs}, ϕ_1^{rs}, ϕ_2^{rs}.

A.1.4 Growth

The mean lengths at age μ_a are independent parameters below a specified age a^* and are constrained by von Bertalanffy growth from age a^* and older. Thus

$$\mu_a = \begin{cases} L_a & \&; a < a^* \\ L_1 + (L_A - L_1)\left[\frac{1-exp(-K(a-1))}{1-exp(-K(A-1))}\right] & ; a \geq a^* \end{cases}$$

where

L_1 is the mean length of the first age class,

L_A is the mean length of the oldest age class, and

K is the von Bertalanffy growth coefficient.

Following Fournier et al. (1990,1998) the standard deviation of length-at-age is assumed to be a function of length

$$\sigma_a = \lambda_1 \exp\left[-\lambda_2\left(1 - 2\frac{\mu_a - L_1}{L_A - L_1}\right)\right]$$

where λ_1 is the standard deviation at an intermediate age and λ_2 the length-dependent trend. $\lambda_2 = 0$ implies length independence.

A.1.5 Calculating length and weight frequencies

It is assumed that the length measurements input to MULTIFAN-CL for fishery f are a random sample of the catch of that fishery in period t. It is further assumed that the lengths of age-class a fish are normally distributed about a mean length μ_a with standard deviation σ. Then the probability that the length of an age-class a fish selected at random lies in length interval i is approximated by

$$P(i|\mu_a, \sigma_a) \approx \frac{\omega}{\sqrt{2\pi}\sigma_a} \exp\left[\frac{-(x_i - \mu_a)^2}{2\sigma_a^2}\right]$$

where

ω is the size of the length intervals, and

x_i is the midpoint of the i^{th} length interval.

This is an adequate approximation to integrating over the range $(x_i - \frac{\omega}{2}, x_i + \frac{\omega}{2})$ as long as $\sigma_a > \omega$ (Fournier, 1990). The expected proportion of length i fish in the sample is

$$\mathrm{Q}_{\mathrm{itf}}^{\mathrm{pred}} = \sum_{a-1}^{A} p_{\mathrm{atf}}$$

where p_{atf} is the proportion of age-class a fish in the sample and is related to predicted catch by

$$\mathbf{p}_{\mathrm{atf}} = \frac{\mathbf{C}_{\mathrm{atf}}}{\sum_{\mathbf{a=1}}^{\mathbf{A}} \mathbf{C}_{\mathrm{atf}}}$$

A.1.6 Fishing Mortality

Catch in numbers of age-class a fish in time period t by fishery f is given by

(A.4)

where r_f is the region in which fishery f occurs, where fishing mortality is parameterized as

(A.5)

and where

$s_{a\ f}$ is the selectivity coefficient of fishery f for age-class a fish,

$q_{t\ f}$ is the catchability coefficient for fishery f in time period t,

B is a biomass index for region r and time period t,

$\beta_t\ r$ is the parameter for effect of biomass on catchability (default= 0),

$E_{t\ f}$ is the fishing effort of fishery f in time period t, and

ζ is the parameter for effect of effort on catchability (default= 1),

$\varepsilon_{t\ f}$ represents transient deviations in effort.

A.1.7 Selectivity

The fishery-specific, age-specific selectivities, s_{af}, can be constrained to be non-decreasing with age and they can also be constrained to be constant with age from a given age class. Furthermore, the s_{af} can either be estimated directly or they can be parameterized as a function of a vector of N^L length-specific selectivities $\mathbf{s}'_f$ as follows: Define a function

where μ_a is the mean length and σ_a is the standard deviation of length for age class a. Then $x(l)$ is the proportional distance of length l along the interval from the average length of the youngest age class minus its standard deviation, μ_1 - σ_1, to the average length of the oldest age class plus its standard deviation, μ_A + σ_A. The inverse function is

$$l(x) = x(\mu_A + \sigma_A) + (1 - x)(\mu_1 - \sigma_1)$$

Age-specific selectivities are then given by

$$s_{af} = \int_0^1 N(\mu_a, \sigma_a, l(x)) S(\boldsymbol{s}'_f, x) dx$$

where $N(\mu, \sigma, l)$ is the normal density function of l with mean μ and standard deviation σ, and $S(\mathbf{s}'_f, x)$ is selectivity as a function of x on the interval [0—1]. S is defined by the parameters $\mathbf{s}'_f$, and is realized by a modified interpolation between those parameters which represent selectivity on N^{L} -1 equal intervals along the interval [(μ_1 -σ_1) — (μ_A +σ_A)]. The interpolation is modified to assure differentiability of the objective function with respect to the $\mathbf{s}'_f$ and the integration is carried out numerically on ten discrete intervals centered on $x(\mu_a)$ from $x(-2\sigma_a)$ to $x(+2\sigma_a)$. Choice of selectivity parameterization is governed by the setting of various fish flags detailed in section 4.5.7.

A.1.8 Catchability

Time series structure in catchability is allowed by:

$q_t + \delta_{t,f} = q_{t,f}\ e^{\eta t\ f}$ (A.6)

where the $e^{\eta t\ f}$ represent cumulative changes in catchability assumed to occur at regular intervals δ_t, set by fish flag 10. Within year catchability is also allowed to vary seasonally with either a sinusoidal pattern with two parameters or an arbitrary pattern with up to 12 multiplicative deviates:

(A.7)

where

q'_{tf} is the catchability before seasonal adjustment,

ν_t is an integer denoting the quarter of the year pertaining to time period t,

$c^1{}_f$ is a fishery-specific amplitude parameter, and

$c^2{}_f$ is a fishery-specific phase parameter, and

$c^i{}_f$ is one of up to 12 seasonal pattern deviates at evenly spaced intervals within a year.

A.1.9 Tagged fish dynamics

The dynamics of tagged fish are similar to those of untagged fish, and the parameterization is shared for the most part between the two. The major modifications for tags are that recruitment is replaced by tag releases and that tags are grouped into release cohorts of tagged fish, indexed by c and released at age $a^{rel}{}_c$, in time interval $t^{rel}{}_c$, and region $r^{rel}{}_c$. The cohorts maintain their separate identity through the age class before apool after which the tagged fish enter a pooled group signified by cohort index c^*. The number of tagged fish by cohort, time, and region prior to movement is given by

where

$a(c,\ t) = a^{rel}{}_c + t - t^{rel}{}_c$

relates historical time to the age of a particular cohort, and where

$g_t = \{\mathrm{c} \mid a^{rel}{}_c + t - t^{rel}{}_c = a^{\mathrm{pool}}\}$

is the set of indices of cohorts that have just graduated to tag pools at time t. In general, fishing mortality is assumed to be the same for tagged and untagged fish. However, for some time after release, this may not be the case until the tagged fish have effectively mixed with the untagged fish that they represent. Thus fishing mortality of tagged fish by release cohort, age, time and fishery is given by

where n^{mix} is a number of time periods to allow for mixing. During the mixing period the fishing mortality of tags is set so that the catch of tags is effectively determined from the observed tag returns $R^{\mathrm{Tobs}}{}_{cat\ f}$ with a correction for the tag return rate $X_{t\ f}$ for fishery f at time t.

Movement of tagged fish is entirely analogous to untagged fish as in Eqn. A.2. Thus

Assigning ages to tagged fish at length

The observed length of each tag is converted to a probability distribution over ages. The age distribution of the catch is not used for this currently, but rather the growth function is used to provide the probability-at-age.

First, for case of a single sex/species, let p_j be the predicted distribution of the catch at age j. For the k^{th} tagged fish of length l_k we compute the probabilities q_{kj} , that a fish of length l_k has age j. This is the probability that a fish of a particular length has a particular age, for every tagged fish k and is taken from

the estimated growth function (mean and variance of length-at-age). So, at present, MULTIFAN-CL uses q_{kl} in the model (i.e. the growth function predictions are used directly). Currently the model assigns tagged fish to the two age classes adjacent to that of the corresponding mean length-at-age.

Using the normal distribution the growth function predicts mean lengths μ_j and standard deviations σ_j for each age class j. Then if length is l let

$$q_{\mathrm{lj}} = \frac{e}{\sigma_j}^{\frac{-(\mu_j - l)^2}{2\sigma_j^2}}$$

Taking the inverse of the growth function predictions by summating over all ages j within each length l provides the distribution of ages in each length interval. The probability that a fish of length l having age j can be set as

$$\rho_{\mathrm{lj}} = \frac{q_{\mathrm{lj}}}{\sum_{j=1}^{m} q_{\mathrm{lj}}}$$

Where within each length interval the probabilities q_{lj} over all ages $j = 1$ to m is summated, and from ρ_{lj} the probabilities at age can be calculated.

The sex- or species-disaggregated model performs all dynamic calculations for each sex/species separately, including the tagged populations for each. For cases where a tag release group consists of all sexes/species aggregated, is therefore necessary to apportion the tag releases among the sexes/species. This is done according to the model estimate of the untagged population numbers by sex/species in the region and time period in which the tag release group occurred.

Tag release groups consisting of data combined over both sexes/species were treated to have associated recaptures that were also combined over both sexes/species; i.e. for these release groups, the sex/species is unknown for both the releases and recaptures. For a given fishing incident, the number of recaptures at length from the tagged population is predicted, $R_{\mathrm{ct\ f}}^{\mathrm{Tpred}}$, and in the case of multiple sexes/species they are predicted in respect of each $\boldsymbol{i}$: $R_{\mathrm{ict\ f}}^{\mathrm{Tpred}}$ from the tagged populations for each sex/species. Where observed tag recaptures are available in respect of sex/species these predictions are input directly to the objective function likelihood term for each respective sex/species. Where observed tag recaptures are available only for all sexes/species combined, fitting to these recapture data (observations) combined over all sexes/species therefore required that the predictions of recaptures from the corresponding release group must be combined before calculating the tagging likelihood.

A.1.10 Aggregate Fishing Mortality

For calculating yield curves and reference points applied to the stock as a whole, it is necessary to calculate fishing mortality aggregated over fisheries (and therefore over regions). This is computed as a weighted average over regions of fishing mortalities by time by age.

(A.8)

It is useful to define a baseline fishing mortality against which to compare projected results with alternate fishing regimes. We define the baseline fishing mortality as the average of aggregate fishing mortality over a given time interval τ_1 to τ_2 as follows:

where F_{at} is given by equation A.8

A.1.11 Calculating total and spawning biomass

The time series of regional total and spawning biomass is given by

where w_a is the average weight and ${p^{mat}}_a$ the proportion mature for age class a fish. The proportion mature at age is given as a fixed input vector. Average weight is given by

where μ_a is the mean length and σ_a is the standard deviation of length for age class a fish, and where $N(\mu,\sigma,l)$ is the normal density function of l with mean μ and standard deviation σ. The integration is carried out numerically on disrete 0.5 cm length intervals from 3.5 below to 3.5 above the mean length μ_a.

A.1.12 Stock Recruitment Relationship

We assume a Beverton-Holt stock-recruitment relationship (SRR) between spawning biomass at time t aggregated over regions ${B^s}_t = \sum_r {B^s}_{tr}$ and recruitment in all regions R_t+t_{lag} after a time lag t_{lag}. Thus

(A.9)

where α is the maximum recruitment at large spawning stock and $ß$ is the spawning stock at which recruitment is reduced to half of α. The recruitment lag is fixed, but α and $ß$ are estimated by penalizing deviations (see section 4.5.12) between recruitment predicted by the SRR and recruitment as used in Eqn. A.1.

A.1.13 Yield Curve

The yield curve is the projected catch in numbers or weight as a function of a fishing mortality multiplier, x. It depends on projected total biomass at equilibrium, $\sim{B}^{t}(x)$, which can be calculated from equilibrium recruitment, $\sim R$, and an equilibrium biomass per recruit function derived as follows.

Combining some multiple of baseline fishing mortality and natural mortality at age, M_a into a total mortality schedule we define an abundance-at-age per recruit function $\varphi_a(x)$

(A.10)

where x is a fishing mortality multiplier. Further define age-aggregate functions Φ^{t} and Φ^{s} for total and spawning biomass per recruit

(A.11)

where w_a is mean weight at age a, and ma is the proportion spawning at age a. By aggregating across age cohorts, the functions, $\Phi^{t}(x)$ and $\Phi^{s}(x)$, implicitly assume that the mortality-at-age schedule is unchanging and therefore that equilibrium conditions hold. Thus they account for equilibrium age distribution as determined by the mortality-at-age schedule to give total and spawning biomass per recruit at equilibrium.

Assuming the SSR given in Eqn. A.9 holds at equilibrium we can say

$$\widetilde{R}(x) = \frac{\alpha\widetilde{B}(x)}{\beta + \widetilde{B}^{S}}$$

where the time indexing is dropped because equilibrium is assumed. A second relationship between $\sim{B}^{s}(x)$ and $\sim{B}^{s}$ is available from the spawning biomass per recruit, $\Phi^{s}(x)$ (Eqn. A.11),

$\sim{B}^{s}(x) = \sim R(x)\Phi^{s}(x)$ (A.13)

Solving the above two equations and two unknowns for $\sim R(x)$ gives

$$\widetilde{R}(x) = \alpha - \frac{\beta}{\Phi^S(x)}$$

Equilibrium abundance-at-age can now be calculated from the abundance per recruit function $\varphi_a(x)$ (Eqn. A.10).

$\sim N_a(x) = \sim R(x)\varphi_a(x)$

from which we get catch-at-age in numbers

Table A.2: Description needed

$\sim C^N{}_a(x)$	=	$barF_a$	$(\sim N_{\ a}(x)$ - $\sim N_{\ a+1}(x))$; $a \in \{1. \ . \ .$ A-1$\}$
$\sim C^N{}_A(x)$	=	$\frac{M_a \ + \ F_a}{barF_A \ \ M_A \ + \ F_A}$	$\sim N_{\ a}(x)$

Yield in numbers and weight is then given by

The yield curve is a plot of a series of yields, y_i, in weight or numbers against a corresponding series of fishing mortality multipliers $x_i \in \{0, 0.1, 0.2, 0.3, \ . \ . \ . \ 50.0\}$. Note that the yield curve can go into negative territory at high levels of x even though it is biologically unreasonable.

A.1.14 Reference Points

If x^* is the value of x that maximizes the yield curve, then the maximum sustainable yield, MSY, in biomass or numbers is given by $\sim Y \ W \ (x^*)$ or $\sim Y \ N \ (x^*)$ $\sim Y \ N \ (x^*)$ respectively. x^* is a reference point in itself being a measure of the ratio of equilibrium fishing mortality at MSY to "base" fishing mortality, the average fishing mortality over a number of (usually recent) time periods. The value of x^* can be chosen simply as the element of the series $\{0, 0.1, 0.2, 0.3, \ . \ . \ . \ 50.0\}$ corresponding to the maximum element, $\max(y_i)$, of the series of yields in the yield curve. However a continuous version of x^* can be estimated as a weighted average of the x_i:

Table A.3: Description needed

x^*	=	$\frac{\sum_i \ x_i \ \exp(y_i/Y)^X}{\sum_i \ \exp(y_i/Y)^X}$

where the constant, (Y), defaults to 10^5 but can be set by age flag 169. It should be larger than $\max(y_i)$, and for best accuracy should not be more than $10 \times \max(y_i)$. The exponent, X, in the weighting factors is large (defaults to 50 and is settable by age flag 168) to assure that the x_i near the summit of the curve are heavily weighted over those further away, and particularly over those in regions where the yield curve is negative.

Given equilibrium recruitment as determined above and Eqn. A.13 for equilibrium spawning biomass plus the analogous equation for total biomass at equilibrium,

$\sim B^{\mathrm{t}}(x) = \sim R(x)\Phi^{\mathrm{t}}(x)$ (A.15)

MULTIFAN-CL calculates the following additional stock assessment reference points:

- $B^{\mathrm{t}}{}_{\mathrm{MSY}} = \sim B^{\mathrm{t}} \ (x^*)$ – equilibrium total biomass at MSY

- $B^{s}_{MSY} = \sim B^{s}(x^*)$ – equilibrium spawning biomass at MSY
- $F_{MSY} = \sim Y\ W\ (x^*)/\sim B^{t}\ (x^*)$ – fishing mortality at MSY.

Finally, time series of total biomass, spawning biomass, and fishing mortality are calculated with respect to their respective values at MSY:

Table A.4: Description needed

$B^{t}{}_{t}$	;	Bst	;	F_t
Bt_{MSY}		Bs MSY		FMSY

A.2 Parameter Estimation – The Objective Function

Parameters are estimated by searching for the set of parameter values that minimizes an objective function consisting of the sum of several parts. Some of the parts are formulated as the negative log-likelihood of the observed data given a specific set of parameter values. Other parts consist of prior distributions on parameters or penalties that serve to constrain the parameter estimates.

A.2.1 Likelihood

Three sets of observed data contribute to the likelihood portion of the objective function: catch data, length frequency data, and tagging data.

A.2.2 Catch data

The contribution to negative log-likelihood from the discrepancy between observed and predicted catch is

$\Theta^{C} = p^{C}\sum t\sum f[\log(1 + C^{obs}{}_{tf}) - \log(1 + C^{pred}{}_{tf})]^2\ ;\ t,f \in \{u,gzapC^{obs}{}_{ug} \neq -1\}$

where

and p^{C} is a weighting factor determined by prior assumption about the accuracy of the observed total catch data. Note that in the input data, observed catch is set to -1 for fishing incidents with unknown catch, and such incidents are excluded from the likelihood function.

A.2.3 Size sample data

Robust normal

The contribution to the negative log-likelihood function from the discrepancy between observed and predicted proportions at size ($Q^{obs}{}_{it\,f}$ and $Q^{pred}{}_{it\,f}$) in the sample data is given by the following robustified formulation:

$$\Theta^{L} = 0.5\sum_{i}\sum_{t}\sum_{f}\log\left[2\pi(\xi_{itf} + \frac{1}{I})\right] + I\sum_{t}\sum_{f}log(\tau_f)$$

$$-\sum_{i}\sum_{t}\sum_{f}\log\left[\exp\left(-\frac{\left(Q^{obs}_{itf} - Q^{pred}_{itf}\right)^2}{2\left(\xi_{itf} + \frac{1}{I}\right)\tau_{tf}}\right) + 0.001\right]$$

where

$\xi_{it\,f} = \mathrm{Q^{obs}}_{it\,f}$ (1 - $\mathrm{Q^{obs}}_{it\,f}$)

$\tau_{t\,f} = \mathrm{P^L}/$ min(1000, $\mathrm{S}_{t\,f}$)

and

S_{tf} is the size of the size-frequency sample taken from fishery f in time period t,

I is the number of size intervals in the samples, and

P_L is a multiplier set by fish flags 49 (default = 10).

The term min(1000, $\mathrm{S}_{t\,f}$) reduces the influence of very large sample sizes by assuming that sample sizes >1000 are no more accurate than sample sizes of 1000. The multiplier $\mathrm{P_L}$ recognizes the variance of real size-frequency samples is almost certainly much greater than truly random samples of a given size. The constant $1/I$ ensures that the variance term is not zero when $\mathrm{Q^{obs}} = 0$, rendering the model less sensitive to occasional observations of low probability. This constant may be adjusted using parest_flags(193), such that it is: parest_flags(193)/100)/I. The constant 0.001 provides additional robustness to the estimation.

Self-scaling Multinomial with random effects

The probability density function for the multinomial distribution is given by

$$\frac{\Gamma(N+1)}{\prod_i \Gamma(n_i+1)} \prod p_i^{n_i} \tag{A.1}$$

In (A.1) i indexes a set of catagories or bins, n_i is the number of observations ocurring in the i'th bin, p_i is the probability of an observations ocurring in the i'th bin, $N = \sum_i n_i$ and $\Gamma(x)$ is the gamma function which for integer values of x satisfies the equation

$$\Gamma(x+1) = x! \tag{A.2}$$

The multinomial is modified to become

$$\frac{\Gamma(N^{\psi(N,\eta)} + \phi(N,\theta))}{\prod_{i'} \Gamma(N q_{i'} + 0.1)} \tag{A.3}$$

where i' means sum over those i for which the proportions $q_i > 0$ and $\psi(N,\eta)$, $\phi(N,\theta)$ are functions of N and parameters (η_0, η_1), $(\theta_0, \theta_1, \ldots, \theta_4)$ where the η_i and θ_i have been chosen that maximize the expression and produces the correct estimated value for N using a training set of simulated compositiomnal data.

The use of an M estimator for the multinomial enables one to attempt to estimate a parameter which describes the strength of the relationship between effective sample size and relative observed sample sizes. Let S_i be the sample size estimate for the $i'th$ sample and

$$\bar{S} = \frac{1}{n} \sum_i S_i \tag{A.4}$$

be the average sample size where n is the number of samples. Let N be the parameter which represents the average effective sample size. Assume that the effective sample size, N_i, for the $i'th$ sample satisfies the relationship.

$$N_i = N \left(\frac{S_i}{\bar{S}} \right)^{\alpha} \tag{A.5}$$

where α is an estimated parameter which determines the strength of the relationship. If $\alpha = 0$ the model has detected no information in the relative sample size data. If $\alpha = 1$ the model has detected that the relative sample size data contains perfect information about the relative sample size.

Dirichlet Multinomial without random effects

The probability mass function of a set of observations having a Dirichlet-multinomial (DM) distribution can be reparameterised if we set $\alpha_k = \lambda p_k$ so that it appears more like the multinomial.

$$\frac{\Gamma(\lambda)}{\Gamma(N+\lambda)} \prod_{k=1}^{K} \frac{\Gamma(x_k + \lambda p_k)}{\Gamma(\lambda p_k)} \tag{A.6}$$

For our purposes it is useful to write the above PMF in terms of the observed compositions

$$(q_1, q_2, \ldots, q_K) \text{ where } q_i = \frac{x_i}{N} \text{ and } \sum_{k=1}^{K} x_k \tag{A.7}$$

so that the above equation becomes

$$\frac{N!}{(Nq_1)! \cdots (Nq_K)!} \frac{\Gamma(\lambda)}{\Gamma(N+\lambda)} \prod_{k=1}^{K} \frac{\Gamma(Nq_k + \lambda p_k)}{\Gamma(\lambda p_k)} \tag{A.8}$$

One can view the Dirichlet multinomial N as an upper bound on the effective sample size. This can be fixed at some appropriate value denoted by $\tilde{N}$, say $\tilde{N} = 1000$. The variance of q_k is equal to

$$\sigma^2_{q_k} = \frac{p_k(1-p_k)}{\tilde{N}} \frac{\tilde{N}+\lambda}{1+\lambda} \tag{A.9}$$

or

$$\sigma^2_{q_k} = \frac{p_k(1-p_k)}{\tilde{N}\frac{1+\lambda}{\tilde{N}+\lambda}} \tag{A.10}$$

So that in the DM $\tilde{N}\frac{1+\lambda}{\tilde{N}+\lambda}$ plays the same role that N plays in the multinomial M estimator. The observed sample size N is taken from the size data and a relative sample size is calculated in respect of the mean N,

$$\tilde{n}_i = \frac{N_i}{\bar{N}} \tag{A.11}$$

and the DM effective sample size multiplier is,

$$\tilde{m}_i = \tilde{n}_i^c \tag{A.12}$$

where c is the sample size covariate exponent, and the effective sample size multiplier is,

$$\lambda_i = e^d \tilde{m}_i \tag{A.13}$$

where d is the sample size multiplier exponent.

A.2.4 Tagging data

Predicted tag returns are calculated for times following the mixing period. The calculation is analogous to that for catch in Eqn. A.4 with a modification for tag return rate $X_{t\,f}$. Predicted returns by tag chohort, age, time and fishery are given by

Table A.5: Description needed

$\mathrm{R}^{\mathrm{Tpred}}{}_{ctf}$	=	$F^{T}{}_{\mathrm{ct\ f}} X_{\mathrm{t\ f}}$	$[1 - e^{-Z}\mathrm{T}ctr] N^{\mathrm{T}}{}_{ctr}$	$; t > t^{\mathrm{rel}}{}_{c} + n^{\mathrm{mix}}$
		$\mathrm{Z}^{\mathrm{T}}{}_{ctr}$		

with F^{T}, and Z^{T} defined in section A.1.7. For entry of tagging data into the objective function, observed and predicted tag returns by various fisheries can be grouped as follows:

where f_g is the set of fishery indices for fisheries in group g.

There are five alternate formulations for the contribution of tagging data to the negative loglikelihood function. The choice is governed by parest flag 111 (section 4.5.8).

Least squares

Robust least squares

Poisson

For the poisson distribution the probability of a vector of observations $\boldsymbol{x}$ with corresponding expected values λ_i is

Table A.6: Description needed

$\mathrm{P}(\boldsymbol{x}) = \Pi i$	$\lambda^{xi}{}_{i}\ e^{-\lambda}\mathrm{i}$	
	$\mathrm{x}_i!$	

Taking the negative of the log and substituting the predicted tag returns for the λ_i and observed returns for the x_i gives the objective function contribution to according to the poisson assumption:

$\Theta^{\mathrm{T}} = \Sigma ctg R^{\mathrm{Tpred}}{}_{ctg} - \Sigma ctg R^{\mathrm{Tobs}}{}_{ctg} \log(R^{\mathrm{Tpred}}{}_{ctg} + 10^{-10}) + \Sigma ctg \log(\Gamma(R^{\mathrm{Tobs}}{}_{ctg} + 1)$

with 10^{-10} in the middle term in case of predicted returns of zero.

Negative bionomial

Because tag returns are not wholly independent events it it likely that the variance in tag return data is probably larger than would be the case with the poisson distribution wherein the variance is equal to the expected value. An alternative is provided in which the variance can be larger than the expected value. This is the negative bionomial distribution under which the probability for a vector of observations $\boldsymbol{x}$ is

Table A.7: Description needed

$P(\boldsymbol{x}) = \Pi i$	[	$\Gamma(\lambda_i\beta_i + x_i)$	(	β_i	$)\lambda i \beta i$	(	1	$)xi]$	(A.16)
		$\Gamma(\lambda_i\beta_i)\Gamma(x_i + 1)$		$\beta_i + 1$			$\beta_i + 1$		

where λ is the expected value for each observation, and the variance for each observation is

$\sigma^2(x_i) = \lambda_i\ (1 + 1/\beta_i)$

Thus $1+1/\beta$ is the degree to which the variance exceeds the expected value, and as β get large the negative binomial approaches the poisson distribution.

Taking the negative of the log and substituting the predicted tag returns for the λ_i and observed returns

for the x_i in equation A.16 gives the objective function contribution according to the negative bionomial assumption:

Relative weighting of the tagging likelihood can be achieved using parest flag 306 such that values larger than 400 impose a higher lower bound on the over-dispersion parameter τ of the negative binomial distribution. The higher variance imposed results in lower influence of this data type in the overall objective function.

The negative binomial variance parameter τ is estimated directly, such that

$$\tau = 1 + e^{fish\_pars(i,4)}$$

Specific to fishery or fishery group i and overdispersion is:

$$a = \frac{x}{\tau - 1}$$

$$ap = log(a + x)$$

So the negative binomial log-likelihood is:

$$\Theta^T = \left(a * ap + x^{\text{obs}} * ap\right) - \left(a * \log\left(a\right) + x^{\text{obs}} x\right) - \Gamma\left(a + x^{\text{obs}}\right) + \Gamma\left(x^{\text{obs}} + 1\right) + \Gamma\left(a\right)$$

A non-zero value for parest_flags(306) implements a change to the bounds such that:

$$lower\ bound = log\left(\frac{parest\_flags(306)}{100} - 1\right)$$

$$upper\ bound = log\left(50 \times \frac{parest\_flags(306)}{100} - 1\right)$$

Therefore, parest flag 306 = 400 produces a lower bound slightly higher than 1 (1.0986) and therefore approaching the Poisson, with little over-dispersion, and higher values imposes greater over-dispersion, consequently higher variance, and hence lower relative influence.

Negative bionomial with added zeroes

[– under construction –]

A.2.5 Age-length data

The ageing data from biological sampling for otoliths provide direct observations of the distribution of fish ages within length classes. Typically, these observations, c_{ljm} are collected from a particular fishing method or fishery m using a sampling design stratified in respect of length, and which assumes the observations at age j are random within each length class l.

Using the normal distribution the model growth function predicts mean lengths μ_j and standard deviations σ_j for each age class j. Then if length is l let

$$q_{\text{lj}} = \frac{e}{\sigma_j}^{\frac{-(\mu_j - l)^2}{2\sigma_j^2}}$$

The predicted catch age composition, p_{jm}, (that takes account of the selectivity pattern) for fishery m is used to derive the predicted distribution of age-at-length for that fishery given the growth estimates of length-at-age

$$\rho_{\mathrm{ljm}} = \frac{p_{\mathrm{jm}} q_{\mathrm{lj}}}{\sum_k p_{\mathrm{km}} q_{\mathrm{lk}}}$$

The observed age composition within each length interval is assumed to be multinomially distributed, and therefore the negative log-likelihood for length interval l is

$$-\sum_j \vartheta_{\mathrm{lm}} \mu_{\mathrm{ljm}} \log(\rho_{\mathrm{ljm}}) \varepsilon_m$$

Where, $\mu_{\mathrm{ljm}}$is the observed proportions age-at-length, $\vartheta_m$is the effective sample size for fishery m, with the total for fishery m is summated among all length intervals, and $\varepsilon_m$is an effective sample size scalar for the samples collected from fishery m. The total likelihood for an age-length is the sum over the length intervals in the sample.

The observed sample size by length interval and fishery is:

$$\vartheta_{\mathrm{ml}} = \sum_j c_{\mathrm{ljm}}$$

A.2.6 Priors and penalties

Many of the multitude of estimated parameters are constrained to some degree by prior distributions or penalties.

A.2.7 Effort deviations

The contribution of effort deviations, ε_{tf} in Eqn. A.5, to the objective function is formulated by the following prior:

where ${p^{\mathrm{E}}}_f$ is a weighting factor for fishery f, and the left hand term in the log expression is a robustified normal distribution with zero mean and fishery specific variance. The ε_{tf} are also weighted by the square root of the effort which ensures that observations at very low effort have relatively little impact on the objective function through the effort deviations. The penalties are set by fish flags 13 (see section 4.5.8) and the expected standard deviation ${\sigma^{\mathrm{E}}}_f$ in the effort deviations is approximated at average effort level by Eqn. A.22. A relatively high value of ${p^{\mathrm{E}}}_f$ results in small deviation from the observed effort whereas a relatively small value of ${p^{\mathrm{E}}}_f$ allows the model greater flexibility to deviate from the observed effort values if this results in a better fit to the data.

A.2.8 Catchability deviations

Catchability deviations (η_{tf} in Eqn. A.6) are assumed to have normal prior distributions, their contribution to the objective function being

A.2.9 Catchability – biomass-dependent effect

When the hypothesis is applied for a biomass-dependent effect on catchability (see section 3.4.1), a normal prior is calculated in respect of scalars of the parameter estimates and the initial population biomass over the year for which average fishing mortality is calculated.

$$\Theta^{\mathrm{QCD}} = p^{\mathrm{QCD}} \sum_{t} \sum_{r} \left(\mu_{\mathrm{tr}}^{\mathrm{QCD}} - B_{\mathrm{tr}}\right)^2$$

A.2.10 Seasonal catchability

A.2.11 Kalman filter deviations

[– under construction –]

A.2.12 Recruitment deviations

The contribution to the objective function from the spatially aggregated recruitment deviations ϕ_t is given by

where the weight p^{R1} governs the variance of normally distributed prior, modified by the second term above which introduces a small amount of auto correlation in the recruitment deviations.

The time-series deviations from the average spatial distribution of recruitment, γ_{tr}, are assigned weak prior distributions of mean zero. The contribution of this prior to the objective function is

A.2.13 Stock-recruitment relationship

If estimation of stock-recruitment parameters is activated, then a spatially aggregated estimate of recruitment is given by Equ. A.9. A penalty for deviations of this recruitment estimate from spatially aggregated recruitment in Eqn. A.1, namely $zoupR\log(\phi_t)$ gives the following contribution to the objective function:

Steepness

The SRR parameters are further constrained by a prior placed on steepness S defined as the ratio of recruitment at 20% of unfished equilibrium biomass to recruitment at unfished equilibrium biomass. Unfished equilibrium recruitment from Eqn. A.14 (setting x to zero) is

$\sim R(0) = \alpha - \quad \beta \quad$ (A.17)

$\Phi^{\mathrm{s}}(0)$

and the corresponding unfished equibrium biomass is then

$\sim B^{\mathrm{s}}(0) = \alpha\Phi^{\mathrm{s}}(0) - \beta$ (A.18)

From Eqn. A.12, recruitment at 20% of unfished equilibrium spawning biomass, would be

$R_{20\%} = \alpha\ 1/5\ \sim \mathrm{B}^{s}(0)$ (A.19)

$\beta + 1/5\ \sim B^{\mathrm{s}}(0)$

which with Eqn. A.18 gives

$R_{20\%} = \alpha(\ \alpha\Phi^{s}(0) - \beta\ )$ (A.20)

$4\beta + \alpha\Phi^{\mathrm{s}}(0)$

The required ratio for steepness as defined above, $R_{20\%}/\ \sim R(0)$, comes out to

$S = \quad \alpha\Phi^{s}(0) \qquad$ (A.21)

$4\beta + \alpha\Phi^{s}(0)$

The prior distribution on S is a non-standard beta on the range [0.2 – 1.0]. The contribution to the objective function is therefore

where A and B are shape parameters of the beta distribution and are controlled by age flag 153 and 154 (see sections 4.5.13 and 5.5.2). The mode, mean, and standard deviation of the beta prior are given by

A.2.14 Movement coefficients

The movement parameters, ${\varphi^{x,y}}_0$ and ${\varphi^{x,y}}_1$, have a normally distributed prior with mean zero. Their contribution to the objective function is

with the effect that in the absence of information in the data concerning movement into or out of a region, the movement coefficients will tend towards zero. The weighting factor, p^{D}, is controlled by

A.2.15 Tag reporting rates

Tag-reporting rates are given normally distributed priors with fishery specific means and penalty weights. Their contribution to the objective function is thus

with fishery specific weighting factors, ${p^{\mathrm{X}}}_f$, controlled by fish flags 35 and means, μ_f , controlled by fish flags 36 (see section 4.5.9).

A.2.16 Selectivity curvature

Smoothness in the selectivity curves is regulated by penalties on the second and third differences of the selectivity-at-age coefficients. The contribution to the objective function is

where the fishery specific weighting factors, ${p^{\mathrm{S1}}}_f$ and ${p^{\mathrm{S2}}}_f$ are controlled by fish flags 41 and 42 (see section 4.5.7).

A.2.17 Spline selectivity penalty

A penalty is calculated in respect of the mean of the spline selectivities such that it is subscripted in respect of fishery, seasons, and time-blocks. It is calculated when deriving the spline functions.

$$\Theta^{\mathrm{SPL}} = p^{\mathrm{SPL}} s^{2}_{\mathrm{fbs}}$$

A.2.18 Generic selectivity penalty

Selectivities may be given normally distributed priors conditional upon parest_flags(74) > 0, that adds a penalty (the weight is specified by the value assigned to the flag) in respect of the selectivity parameters. This improves stability in estimating logistic and double-normal selectivities particularly in the initial stages of estimation. Optionally, a different penalty weight can be assigned among the fisheries using fish_flags(i,72). This will supercede the setting in parest_flags(74).

$$\Theta^{\mathrm{SLD}} = \sum_{\mathrm{fbs}} p^{\mathrm{SLD}} \left(s_{\mathrm{fbs}} - s_{\mathrm{fbs}} \right)^2$$

A.2.19 Natural mortality rates

The natural mortality rates, M_a, are constrained by smoothing penalties and a penalty to avoid extreme deviations from the mean. The contribution to the objective function is

where the M'_a are the log of the natural mortalities at age divided by the mean thereof, i.e.

and where

The weighting factors, p^{M1}, p^{M2}, p^{M3}, and p^{M4}, are controlled by age flags 77, 78, 79, and 80 respectively (see section 4.5.17).

A.2.20 Fishing mortality at MSY

The fishing mortality multiplier at MSY, x^*, is the ratio of fishing mortality at MSY to a nominal aggregate fishing mortality. x^* can be given a target value, x_{targ} with associated penalty weight, p^{F}, according to age flags 165 and 166. The resulting contribution to the objective function is

$\Theta^{\mathrm{F}} = p^{\mathrm{F}} \log(x^*/x_{\mathrm{targ}})^2$

see 4.5.13.

A.2.21 Weighting factors, variance, and coefficient of variation

Many parts of the objective function include a weighting (or penalty) factor. These factors should each be set by the user to reflect the variability in the variable involved in the particular part of the objective function. Assuming approximately log-normal deviations for these variables, for variable x with mean μ and variance σ^2 the likelihood of a set of x values is

where $\hat{x}_i$ is some predictor of x_i. Ignoring the constant term $1/2\pi\sigma^2$ the negative log-likelihood becomes

Most parts of the objective function have a similar form, that is, a factor p times a sum of squared deviations of some sort. Therefore putting p in for $1/(2\sigma^2)$ the standard deviation σ is at least approximated by

(A.22)

To see that the coefficient of variation, CV, of the unlogged variable is approximately the standard deviation σ of the log form of the variable, note that an instance of x_i one standard deviation above its mean would be

$x_i = e^{(\log(\mu)+\sigma)}$

and the CV would be

$\mathrm{e}^{(\log(\mu)+\sigma)} - \mu = \mathrm{e}^{\log(\mu)}\mathrm{e}^{\sigma} - \mu = \mu\mathrm{e}^{\sigma} - \mu = e^{\sigma} - 1$

$\mu\ \mu\ \mu$

which by Taylor expansion is

$[1 + \sigma + \sigma^2/2! + \ldots] - 1 \approx \sigma$

Similarly, an instance of x_i one standard deviation below its mean leads to

$\mu - \mathrm{e}^{(\log(\mu)-\sigma)} = 1 - e^{-\sigma} = 1 - [1 - \sigma + \sigma^2/2! - \ldots] \approx \sigma$

μ

So, ignoring second order and higher terms, the CV of x is approximately σ. Thus a weight for a term of the objective function can be set according to prior assumption about the variance or the CV of x by

$p \approx 1/2\sigma^{\ 2} \approx 1/2\mathrm{CV}^2$

e.g., for $p = 100$, CV ≈ 0.07.

A.3 Computational Details

Although we will attempt to keep the documentation in this User's Guide as up-to-date as we can, it is inevitable that the documentation will lag behind continuing developments in MULTIFAN-CL. Thus the only completely authoritative documentation is the source code itself. However, finding one's way around in the source code is daunting even for those well versed in the C++ language. Therefore we provide this section not only for those curious about computational approaches in MULTIFAN-CL but also as an entree into the code for those wishing to delve into details not (yet) covered by the User's Guide.

A.3.1 Navigating the Source Code

The flow of events in a MULTIFAN-CL run is summarized in Figure A.1 (page 164). A complete flow diagram of MULTIFAN-CL would be much more intricate.

In describing the computational flow, we will begin with the innermost part, the "fcomp" routine and work outward. The fcomp routine is where simulation of the fish population and fishing is conducted. Here parameters are used which are meaningful to the biological or fishery aspects of the model, such as natural mortality and catchability. Not all of these "functional" parameters are operative in a MULTIFAN-CL run. Which ones are operative depends on the settings of control flags that enable, or disable, various features of the model. Among the operative parameters will be a subset consisting of those which are being estimated. The values of these "active" parameters can be different each time the flow of computation runs through the fcomp routine. So fcomp's first job is to determine those values. They are passed to fcomp from its calling routine in the "X-vector". Which element of the X-vector goes with which functional parameter is determined by settings of yet more control flags. Values of non-active parameters are determined elsewhere and are available to fcomp by way of....

Having determined the values of its operative parameters, fcomp conducts a simulation, and then calculates the objective function suitably modified by various priors and penalties. It then returns to the calling routine, the minimizer,[1] passing back the value of the objective function to the calling routine along with the gradient, which is a vector of partial derivatives of the objective function with respect to the X-vector.

The job of the minimizer is to find the values of elements in the X-vector such that the objective function, is minimized. The minimizer knows nothing about the functional parameters. It also knows nothing about details of the objective function except that when it passes values of the X-vector to the fcomp routine, it gets back a value of the objective function as well as the gradient, which is a vector of partial derivatives of the objective funcction with respect to the active parameters. On the basis of that information, it either adjusts the values of the X-vector and calls fcomp again or it quits depending on whether the maximum gradient value is less than some threshold or the number of calls to fcomp, the iteration count, has reached some limit.

mean-lengths-bound check.... Setting gradient structure....

Quit flag and parest flag 30.... Setting control structure...

[1] In certain circumstances not shown in the flow diagram, fcomp is called by other parts of the code than the minimizer.

Finally we get to the first box in the diagram which deals with setting up the spatial, temporal, and biological structure of the population dynamic model, determining which processes will be enabled in the model and hence which of the many parameters will be operative, and determining which of the operative parameters will be active.

A.3.2 Using a debugger

Perhaps the best way to follow the flow of MULTIFAN-CL is to run it with a debugger. A debugger is also useful when trying to find what is wrong with the input files when MULTIFAN-CL quits with a cryptic error message. In using a debugger, there are two, somewhat unfortunate, choices. For working in the μsoft Windows environment, Borland in the past supplied a good, user friendly, debugger. But that version of Borland's debugger is no longer available, and its updates are not as satisfactory. In the Linux world, on the other hand, we have gdb, the GNU debugger, which is very powerful, but lacks user friendliness.

Regardless of which debugger you are using, note that there are a variety of consistency checks sprinkled throughout the code, and that in case of error, they all finish up by calling an exit routine, ad boundf(). It is useful to set a break point at that routine. Then when the program stops at that point, you can follow back in the stack to find where the check error occurred.

A.3.3 Borland debugger

[– under construction –]

A.3.4 GNU debugger

As we mentioned, the GNU debugger lacks user friendliness. However, for those willing to run it within the emacs (or xemacs) editor, the facility of use is considerably improved. Therefore we offer here some hints on running a gdb session in the emacs environment. Some familiarity with emacs is assumed.

First some notation conventions for keyboard entries:

Table A.8: Description needed

C_r	—	Enter (carriage return)
A-x	—	Alt-x
C-x	—	Ctrl-x
C-xat	—	Ctrl-x and then a and t *while still pressing* Ctrl
C-xo	—	Ctrl-x and then o *after releasing* Ctrl
	—	space
PgDn, PgUp	—	Page Down, Page Up

It is useful to run gdb with a vertically split emacs window. This is accomplished by keying *C-x*3 and then *A-x*gdbC_r. This will produce the message: Run gdb (like this): gdb in the "Minibuf" at the bottom of the emacs window. Key in mfclC_r, and you will be placed in a gdb session in the left hand window with a (gdb) prompt. Before starting a run of mfcl, it is most useful to set a break point. To break at the beginning of mfcl execution, enter break mainC_r at the prompt. See Table A.1 for other ways to set break points.

To start a run enter run followed by the desired command string, for example:

run yft.frq yft.ini 00.par -makeparC_r

This will put the source file nnewlan.cpp in an emacs buffer in the right hand window positioned with a triangular pointer at the first breakpoint. As gdb moves through execution of mfcl this pointer will follow the source code (more or less[2]), bringing up other source files as execution steps into code in those files.

There are numerous commands available to control gdb. Unfortunately, commands for the same action can differ depending on which window has current focus, and some are available only in one or the other of the windows. Table A.1 gives a selection of commonly used commands some of which are useful in setting breakpoints and stepping through the code. Moving the cursor, scrolling windows, and toggling the window focus can be accomplished by key entries or by the mouse, which can be also be used to manipulate the relative window sizes by clicking and dragging in the horizontal grey bar immediately below the vertical bar separating the windows.

Table A.9: gdb and emacs commands

gdb window	source window	action
n	*C-xan*	advance execution to next source code statement
s	*C-xas*	like n but if current statement calls a subroutine, step into the subroutine
c	*C-xar*	continue execution to next breakpoint
C-ct	*C-xat*	set a temporary breakpoint at emacs position in source buffer
	C-x	set a breakpoint at emacs position in source buffer
break *xxx*		set a breakpoint, where *xxx* can be the name of a subroutine or the name and line number of a source file, e.g., nnewlan.cpp:84
delete		erase all breakpoints
F6	F6	toggle window focus
A-PgUp	*A-PgUp*	scroll other window up
A-PgDn	*A-PgDn*	scroll other window down
↑	↑	move cursor up
↓	↓	move cursor down
C-↑		step through command history
p *x*		print value of variable *x*
ptype *y*		print structure of variable *y*

Stepping through the code with n and s is a good way to follow the flow of the program, but when halted at a breakpoint it is also useful to examine the contents of program variables. Table A.2 is a schematic view of a gdb session in which a breakpoint is set, and the program run, whereupon it halts at that breakpoint shown by the pointer in the source code window. The program may be about to write to a file whose name is stored in string variable full_output_parfile_path. The next user command reveals that the file name is 00.par as is should be (from the run command). The next user command examines the structure of fsh.bd_coff which is a component of class variable fsh. Much of the resulting output in the gdb window has been deleted in the example, but what is there shows that fsh.bd_coff has a component *va as well as a minimum and maximum index; so it evidently is something related to a vector of pointers to doubles. The decipherable part of the stuff printed by the next user command shows the index to range from 1 to 3. The next user command prints the first element of fsh.bd_coff cast to (double &) giving a value of 30. The final user command shows a trick for printing all three elements of the vector. The gdb debugger is full of such tricks waiting to be discovered, much like goodies in an Easter egg hunt. Try a Google search with "gdb debugger guide help tricks".

[2]MULTIFAN-CL is usually compiled with a high optimization level to maximize speed and efficiency. This means that the object code no longer corresponds exactly to the source code. If this is annoying, a debug version of MULTIFAN-CL can be compiled with optimization turned off.

A.3.5 Error Exit

MULTIFAN-CL can be compiled in optimized mode or in safe mode. The safe mode checks array indices against array boundaries and exits if an overrun occurs, producing an error message such as "matrix bound exceeded." In redoing the offending MULTIFAN-CL run in the debugger, it is useful to place a breakpoint at ad_boundf(int i) in source code file nnewlan.cpp. That way the program will suspend before running to completion thereby allowing the debugger to trace back through the sequence of function calls leading to ad_boundf. This should give a clue as to what caused the array overrun.

Table A.10: Portion of a gdb debugging session within the emacs editor. Blue colored text is entered by user.

gdb window
`(gdb) break newl5.cpp:88`
Breakpoint 1 at 0x8088775: file newl5.cpp, line 88. (gdb)
run simdat.frq simdat.ini 00.par -makepar
Starting program: /home/pkleiber/mfcl-feb25-03/gmult
Breakpoint 1, fcomp(dvar_len_fish_stock_history cons
dvar_vector const&, int, int, dvector const&, int, i
print_switch=1, _gbest=@0xbfffe2c0,
gradient_switch=0, pq_flag=0x0) at newl5.cpp:88
(gdb) p &full_output_parfile_path[1]
$1 = (const unsigned char *) 0x81e1fb0 "00.par"
(gdb) ptype fsh.vb_coff
type = class dvar_vector {
private:
double_and_int *va;
int index_min;
int index_max;
arr_link *link_ptr;
vector_shapex *shape;
.
< text deleted >
.
}
(gdb) p fsh.vb_coff
$2 = {va = 0x4946b4a8, index_min = 1, index_max = 3,
link_ptr = 0x825ac28, shape = 0x825ac48}
(gdb) p (double &)fsh.vb_coff[1]
$3 = (double &) @0x4946b4b0: 30
(gdb) p (double[3] &)fsh.vb_coff[1]
$4 = (double (&)[3]) @0x4946b4b0: {30, 100,
0.29999999999999999}
(gdb)

Table A.11: Portion of a gdb debugging session within the emacs editor. Continuted from previous table. Source code window.

source code window

```
double fcomp(const dvar_len_fish_stock_history& _f
const dvar_vector& _x, int nvar,int print_switch
int gradient_switch,ivector * pq_flag)
{
dvar_vector& x=(dvar_vector&) _x;
dvector& gbest=(dvector&) _gbest;
dvar_len_fish_stock_history& fsh=(dvar_len_fish_
//cout << "Entered fcomp" << endl;
char tmp_file_names[19][13]={"tmpout.1","tmpout.
"tmpout.5","tmpout.6","tmpout.7","tmpout.8","t
"tmpout.11","tmpout.12","tmpout.13","tmpout.14
"tmpout.16","tmpout.17","tmpout.18","tmpout.19
HERE
dvariable f=0.0;
//cout << "entered fcomp" << endl;
greport("beginning fcomp do_every");
dvariable mean_length_constraints=0.0;
fsh.generate_report=print_switch;
f+=fsh.reset(x);
if (fsh.parest_flags(197))
{
par_ofstream ofs(full_output_parfile_path);
ofs << fsh;
exit(0);
}
//cout << "constraint penalty = " << f << endl;
HERE
dvariable ffpen=0.0;
// cout << " before do_every" << endl;
ffpen=0.0;
fsh.do_everything_calc(ffpen,pq_flag);
if (pvm_switch ==0 || pvm_switch ==1)
```

A.3.6 Source Code Files

There are approximately 100 source code files. Familiarity with the content of these files is useful in navigating the code. Here are a few which we have gotten around to annotating. We hope to expand the list at we find time, perhaps organizing them in functional groupings. One problem is that the source files tend to proliferate with updated versions of MULTIFAN-CL; so the list will probably never be complete.

nnewlan.ccc – Contains main() plus routines for dealing with the command line and setting things up.

optmatch.ccc – Routines to recognize command line options.

lbselclc.ccc – Contains routines to manipulate selectivities, including calculation of selectivity at length and for smoothing selectivity at age based on variance in length at age.

mfexp.ccc – Upper bounded exp function

A.3.7 Important functions

A.3.8 ADMB functions

[– under construction –] Functions in the ADMB library called by MULTIFAN-CL code to be documented here.

A.3.9 MULTIFAN-CL functions

set_value – (x, y, v, ii, fmin, fmax, fpen, s)

set_value_inv – (x, y, v, ii, fmin, fmax, fpen, s)

boundp – (xx, fmin, fmax, fpen, s)

boundpin –

fcomp –

reset –

-- etc. --

A.3.10 Class descriptions

```
// The class fish_stock_history contains all the relevant data and parameters which
// determine the exploitation of the stock the data are ordered by fishing period.
class dvar_fish_stock_history
{
public:

     double likecomponent[10];
     int generate_report;
     int month_1; // the month in which recruitment occurs
     int frq_file_version;
     ivector parest_flags; // the old control flags from the standard multifan
     // model so they can be accessed from the class
     // members The standard flags from the old multifan
     // model
     ivector age_flags; // The new flags for the age structured part of the
     // model
     imatrix fish_flags; // Fishery specific control flags
     imatrix data_fish_flags; // Fishery specific control flags
     imatrix old_fish_flags; // Fishery specific control flags
     imatrix tag_flags; //
     // fish_flags(i,1) :  the number of age classes with selecitvity in
     // fishery i
```

```
int nage; // The number of age classes
dvar4_array nrsurv;
dvar_vector biomass;
dvar_matrix biomass_by_region;
dvar_matrix rel_biomass_by_region;
dvar_vector catch_biomass;
dvar_vector catch_numbers; // JH 27/03/02
dvar3_array F_by_age_by_year_by_region;
dvar_matrix F_by_age_by_year;
dvar5_array nrfm;
dvar4_array nrtm;
// fisheries grouping stuff used in catchability calculations
imatrix gfish_ptr;
ivector num_grouped_fish_times;
int ngroups;
imatrix global_fishing_periods; // need for grouping effort
// for cobbs-douglas
// production function
ivector effective_len_size;
ivector effective_weight_size;
dmatrix grouped_effort;
dmatrix grouped_fish_time;
dmatrix grouped_between_times;
dvar_matrix grouped_catchability;
imatrix gfish_index;
dvar_matrix grouped_catch_dev_coffs;
int nyears; // The number of years over which fishing occurred
int num_tag_releases; // the number of sets of releases of tagged fish
int min_tag_year;
ivector tag_region; // the region in which each tag set was released
ivector tag_year; // the year in which each tag set was released
ivector true_tag_year; // the year in which each tag set was released
ivector tag_month; // the month in which each tag set was released
ivector true_tag_month; // the month in which each tag set was released
ivector itr; // the number of differenct periods in which tags were
// returned from that tag group
ivector itind; // the first realization with tags present
ivector initial_tag_year; // the each tag set was
// released[---????---]
```

```
ivector terminal_tag_year; // the each tag set was
// released[---????---]
imatrix initial_tag_period; // the each tag set was
// released[---????---]
imatrix terminal_tag_period; // the each tag set was
// released[---????---]
imatrix initial_tag_recruitment_period; // the each tag set
// was released [---????---]
imatrix initial_tag_release_by_length; // the tag release data
// by length intervals
dvar_matrix initial_tag_release_by_age; // the tag release data by
// length intervals
i3_array tag_recaptures_by_length;
int tag_shlen; // The number of fisheries
double tag_filen; // The number of fisheries
int tag_nlint; // The number of fisheries
int first_time; // used as date offset for renumbering data dates
int month_factor; // used as date offset for renumbering data dates
int num_fisheries; // The number of fisheries
ivector minttp;
ivector min_init_tag_period;
ivector maxttp;
ivector minimum_initial_tag_period;
dvector pmature;
int num_recruitment_periods;
int initial_recruitment_count;
ivector num_fish_periods; // The number of fishing periods
int num_fish_data_recs; // The total number of fishery
// data records
int num_regions; // The number of fishery data records
imatrix num_fish_incidents; // The number of fishing incidents which
// occurred during each fishing period in each
// region
imatrix recruitment_period; //
i3_array num_tagfish_incidents; // The number of fishing incidents which
// occurred during each fishing period in
// each region for each tag group
i3_array num_alltagfish_incidents; // The number of fishing incidents which
// occurred during each fishing period in
```

```
// each region for each tag group
imatrix num_pooledtagfish_incidents; // The number of fishing incidents
// which occurred during each fishing
// period in each region
ivector num_fish_times; // the number of times a particular fishery occurred
ivector fishing_period; // the fishing period during which an instance of a
// fishery occurs
dvar_matrix effort_by_fishery;
ivector fishing_region; // the fishing period during which an instance of a
// fishery occurs
ivector fishing_incident; // the fishing_incident corresponding to an
// instance of a fishery
i3_array parent; // Point to the fishery containing a fishing incident
imatrix year; // The year during which the fishing incident occurred
imatrix movement_period; // The year during which the fishing incident
// occurred [---????---]
int year1; // The first year a fishing incident occurred
imatrix realization_period; // The fishing period in
// which each realization of a fishery
// occurred
imatrix realization_incident; // The fishing incident to which each
// realization of a fishery corresponded
imatrix realization_region;// The fishing incident to which each realization
// of a fishery corresponded
i3_array header_record_index; // The index of the (sorted) header record
// which corresponds to a particular fishing
// period and fishing incident
dvar_matrix region_pars;
imatrix region_flags;
dvar_vector gml;
dvar_vector predicted_yield_bh;
dvar_vector predicted_eqbio_bh; // JH 21/03/02 - equil.  adult biomass
dvar_vector predicted_eqtotbio_bh; // JH 21/03/02 - equil.  total biomass
dvar_vector tb_ratio; // JH 27/03/02 - tot biomass over TB at MSY
dvar_vector ab_ratio; // JH 27/03/02 - spn biomass over AB at MSY
dvar_vector F_ratio; // JH 27/03/02 - aggregate F over F at MSY
dvector predicted_yield_bh_x;
dvar_vector predicted_yield_pt;
dvector predicted_yield_pt_x;
```

```
dvar_vector predicted_recruitment_bh;
dvector predicted_recruitment_bh_x;
dvector region_area;
dvar_matrix region_rec_diffs;
dvar_matrix region_rec_diff_coffs;
dvar_vector region_rec_diff_sums;
dvar_matrix fraction; // The fraction of the total natural mortality occurring
// during this fishing period
ivector regmin;
ivector regmax;
dvariable totpop; // population size scaling parameter
dvariable recmean;
dvariable initmean;
dvariable rec_init_diff; // diff level between rec and init pop
dvar_matrix D; // moves the fish around
dvar3_array Dad; // moves the fish around
imatrix Dflags; // moves the fish around
dvar_vector diff_coffs; // parameters in the diffusion matrix
dvar_vector diff_coffs2; // parameters in the diffusion matrix
dvar_vector diff_coffs3; // parameters in the diffusion matrix
dvar_vector xdiff_coffs; // parameters in the diffusion matrix
dvar_vector xdiff_coffs2; // parameters in the diffusion matrix
dvar_vector xdiff_coffs3; // parameters in the diffusion matrix
dvar_vector recr; // the relative recruitment levels
ivector rec_times; // the relative recruitment levels
dvector rec_covars; // environmenital factors affecting recruitment
dvar_vector tmprecr; // the relative recruitment levels
dvar_vector avail_coff; // orthogonal polys which determine [---???---]
dvar_vector orth_recr; // orthogonal polys which determine the relative
// recruitment levels
dmatrix recr_polys;
dvar_matrix initpop;// the relative inital population at age levels
dvar_vector tmpinitpop; // the relative inital population at age levels
dvar_vector actual_recruit; // The total number of fish recruiting to the
// population each year
dvar_vector actual_init; // The initial number of fish in the population in
// year 1
dvar3_array num_fish; // The number of fish in each age class in the population
// at the beginning of each fishing period
```

```
dvar3_array num_fish0; // The number of fish in each age class in the
// population at the beginning of each fishing period in
// the absence of fishing
dvar3_array total_num_fish; // The number of fish in in the population at the
// beginning of each fishing period
dvar4_array tagnum_fish; // The number of fish in each tag group in age class
// in the population at the beginning of each fishing
// period
dvar3_array pooled_tagnum_fish; // The number of fish in each tag group in age
// class in the population at the beginning of
// each fishing period
dvar3_array epooled_tagnum_fish_recr; // The number of fish in each tag group
// in age class in the population at the
// beginning of each fishing period
dmatrix tag_release_by_length;
dvar_matrix lbsel;
dvar3_array pooledtagN; // The number of fish in each age class in the
// population at the beginning of each Year
dvar4_array tagN; // The number of fish in each age class in the population at
// the beginning of each ‘Year
dvar3_array tagrelease; // The number of fish in each age class in the
// population at the beginning of each Year
dvar3_array N; // The number of fish in each age class in the population at the
// beginning of each year N(region, year, age)
dvar3_array N0; // The number of fish in each age class in the population at
// the beginning of each Year in the absence of fishing
// N0(region, year, age)
dvar3_array exp_N; // The number of fish in each age class in the population at
// the beginning of each Year
dvar4_array catch; // The number of fish in the catch from each age class,
// fishing period, and each fishery
dvar4_array mean_weights_at_age; // The number of fish in the catch from each
// age class, fishing period, and each fishery
dvar4_array exp_catch; // The number of fish in the catch from each age class,
// fishing period, and each fishery
dvar5_array tagcatch; // The number of tagged fish in the catch from each age
// class, fishing period, and each fishery
dvar4_array pooled_tagcatch; // The number of tagged fish in the catch from
// each age class, fishing period, and each
```

```
// fishery
dvar5_array obstagcatch; // The number of tagged fish in the catch from each
// age class, fishing period, and each fishery
dvar4_array pooledobstagcatch; // The number of tagged fish in the catch from
// each age class, fishing period, and each
// fishery
dvar5_array obstagcatch1;
d4_array tot_tag_catch; // The number of tagged fish in the catch from each
// age class, fishing period, and each fishery
d3_array region_tot_tag_catch; // The number of tagged fish in the catch
// from each age class, fishing period, and
// each fishery
d3_array pooledtot_tag_catch; // The number of tagged fish in the catch from
// each age class, fishing period, and each
// fishery
d5_array obstagcatch_by_length; // The number of tagged fish in the catch
// from each age class, fishing period, and
// each fishery
d4_array pooledobstagcatch_by_length; // The number of tagged fish in the
// catch from each age class, fishing
// period, and each fishery
dvar3_array tot_catch; // The total number of fish in the catch fishing period,
// and each fishery
dvar4_array prop; // The number of proportion in the catch from each age class,
// fishing period, and each fishery
dvar4_array incident_sel; // The selectivity parameters for each fishing
// incident
dvar3_array mean_incident_sel; // The selectivity parameters for each fishing
// incident
dvar3_array delta2; // The random variations in selectivity
dmatrix between_times; // The times between realizations of a fishery
imatrix imp_back; // How far back to go in implicit robust mean calculations
// for catchability
imatrix imp_forward; // How far forward to go in implicit robust mean
// calculations for catchability
dvar_matrix fishery_sel; // The selectivity parameters for each fishery
dvar_matrix selcoff; // The selectivity parameters for each fishery
dvar3_array survival; // The survival rate for each fishing period
dvar4_array fish_mort; // The fishing mortality rate for each fishing
```

```
// period, fishing incident and age class
dvar4_array fish_mort_calcs; // The fishing mortality rate for each fishing
// period, fishing incident and age class
dvar3_array tot_mort; // The total mortality rate for each fishing period
dvar3_array tot_mort0; // The total mortality rate for each fishing period
dvar_matrix nat_mort; // The annual mortality rate for each year and age class
dvar3_array catchability; // The catchability by fishing period by fishing
// incident
dvar3_array FF; // The catchability by fishing period by fishing incident
d3_array effort; // the fishing effort by fishing period by fishing incident
imatrix month; // the month of the year during which each fishery occurs
imatrix true_month; // the month of the year during which each fishery
// occurs
imatrix true_week; // the month of the year during which
// each fishery occurs
imatrix true_year; // the month of the year during which
// each fishery occurs
imatrix week; // The week of the month during which each fishery occurs
dvariable nat_mort_coff; // determines the natural mortality
double catch_init; // Initial value for the catchability
dvar_vector q0; // overall catchability coefficient
dvar_vector q1; // time dependent trend in catchability coefficient
dvar_matrix effort_dev_coffs;
imatrix zero_catch_flag;
dvar_matrix catch_dev_coffs; // determine random walk deviations in catchability
// time trend
dvar_matrix rep_dev_coffs; // determine random walk deviations in reporting
rate
// time trend
dvar3_array rep_rate; // Tag reporting rate by region, fishing period, fishing
// incident
dvar_matrix implicit_catchability; //
dvar_matrix fish_pars; // extra fisheries related parameters
dvar_matrix seasonal_catchability_pars; // explicit seasonal catchability as
// opposed to trig function
imatrix seasonal_catchability_pars_index;
dmatrix seasonal_catchability_pars_mix;
dvar_matrix age_pars; // extra age-class related parameters
dmatrix biomass_index; // extra fisheries related parameters
```

```
dvar3_array effort_devs; // deviations in effort fishing mortality relationship
dvar3_array sel_dev_coffs; //determine deviations in selectivity
dvar4_array sel_devs; //determine deviations in selectivity
dvar_vector corr_wy; // determines within year correlation in sel deviations
dvar_vector corr_by; // determines between year correlation in sel deviations
dvar_vector corr_wc; // determines within cohort correlation in sel deviations
dvar_vector corr_eff; // determines // The data
d3_array obs_tot_catch;
dmatrix obs_region_tot_catch;
d4_array obs_catch; // The number of fish in the catch from each age class,
// fishing period, and each fishery
d4_array obs_prop; // The estimated proportion in the catch from each age
// class, fishing period, and each fishery
dvar_matrix qmu;
dvar_matrix qvar;
dmatrix qstudent;
void do_grouped_between_time_calculations(void);
void allocate_optional_stuff(void);
void do_the_diffusion_yf(int year,dvar_matrix& DINV);
void fishing_mortality_calc(void);
void fishing_mortality_calc(d3_array&);
dvar_fish_stock_history(int ng,int nfp, ivector& nfi,
int nfsh, int nyrs);
dvar_fish_stock_history(int ng,int nfp, ivector& nfi,
int nfsh, int nyrs, ivector& flags);
void catch_equations_calc_implicit(dvar_vector& sv,
dvariable& ffpen);
void catch_equations_calc_implicit_mc(dvar_vector& sv,
dvariable& ffpen);
void seasonal_catchability_calc(void);
void fishery_selectivity_calc(void);
void incident_selectivity_calc(void);
void get_initial_population(dvar_vector& sv);
void get_initial_tag_population(dvar_vector& sv, int it);
void do_the_diffusion(int year,dvar_vector& sv,
dvar3_array& N, ofstream * pofs=NULL);
void do_the_diffusion(int it,int year,dvar_vector& sv,
dvar3_array& N, ofstream * pofs=NULL);
void get_exp_N(void);
```

```
void incident_selectivity_calc(d3_array&);
void incident_selectivity_calc(d3_array&,dvar_vector&,
dvar_vector&);
void proportion_at_age_calc(void);
void total_fish_mort_and_survival(int it,int ip,int yr,
const dvar_vector & tmp);
void fishery_selcoff_init();
void rep_rate_devs_calc(void);
void do_everything_calc();
void do_everything_calc(d3_array&);
void do_everything_calc(d3_array& len_sample_size,
dvar_vector& vb_coff,dvar_vector& var_coff,dvar_vector& sv,
dvariable&);
void get_initial_parameter_values();
void do_newton_raphson(int ir,int ip,dvariable& ffpen);
void transform_in();
void transform_out();
//have_data_this_year(void);
dvariable have_data_this_year(int it,int ir,int& ip,int cy);
void print_tag_accounting_info(void);
void calculate_tag_catches(int it);
void have_no_data_this_year(int it,int ir,int cy);
void initial_population_profile_calc();
void natural_mortality_calc(void);
void natural_mortality_calc2(void);
//void catchability_init();
void natural_mortality_init();
void catchability_calc(void);
void catchability_devs_calc(void);
void set_zero_catch_flag(void);
void get_observed_total_catch();
void catch_equations_calc(dvar_vector& sv);
void xcatch_equations_calc(dvar_vector& sv);
void catch_equations_calc_movement(dvar_vector& sv);
dvariable tag_catch_equations_calc(dvar_vector& sv);
void tag_catch_equations_calc_mc(dvar_vector& sv);
void catch_equations_calc_avail();
dvariable reset(dvar_vector& x,int& ii);
dvariable reset(dvar_vector& x,int& ii,
```

```
d3_array& len_sample_size);
void sel_dev_all_comp();
void fmcomp(void);
void effort_devs_calc();
void do_fish_mort_intermediate_calcs(void);
void do_fish_mort_intermediate_calcs(int ir,int ip);
void albacore_control_switches(void);
void set_control_switches(void);
void xinit(dvector& x);
void xinit(dvector& x,int& ii);
void xinit(dvector& x,int& ii,d3_array& len_sample_size);
void xinit(dvector& x,int& ii,d3_array& len_sample_size,
ofstream& xof);
void do_recruitment_period_calculations(void);
void get_initial_tag_recruitment_period(void);
int nvcal(d3_array& len_sample_size);
void get_initial_age_structure(void);
void dvar_fish_stock_history::get_initial_age_structure(
const dvariable& totpop,dvar_vector& sv);
dvar_fish_stock_history(int ntg,int nregions, int ng,ivector& nfp,
imatrix& nfi,int nfsh,int nyrs,ivector& fl,
ivector& par_fl, ivector& nft, ivector& _regmin,
ivector& _regmax, ivector& _dataswitch,imatrix& _Dflags);
void dvar_fish_stock_history::
get_initial_age_structure_equilibrium(void);
dvar_matrix dvar_fish_stock_history::
get_equilibrium_survival_rate(void);
void print_tagging_fit_info(ofstream& of);
void print_movement_report(ofstream& of);
dvariable tag_catch_equations_calc_pooled(dvar_vector& sv);
void get_initial_tag_fishery_realization_index(void);
void do_newton_raphson_for_tags(int it,int ir,int ip,
dvariable& ffpen);
void allocate_some_tag_stuff(void);
void pooled_tag_catch_equations_calc(dvar_vector& sv);
void grouped_catchability_calc(void);
void do_grouped_tags_between_time_calculations(void);
dvariable robust_kalman_filter_for_catchability(void);
void effort_multiplier_for_cobb_douglas_production_function(void);
```

```
     void read_recruitment_env_data(adstring& root);
     void print_tag_return_by_time_at_liberty(ofstream& of);
     void explicit_seasonal_catchability_calc(void);
     dvariable normalize_seasonal_catchability(void);

}; // end class dvar_fish_stock_history
class dvar_len_fish_stock_history:  public dvar_fish_stock_history
{
public:

     ivector nlintv;
     //dmatrix freq;
     dvector fmid;
     dvector wmid;
     dvar4_array mean_length; (reg,pd,flt,age)
     dvar3_array mean_length_yr;
     dvar3_array mean_weight_yr;
     double len_wt_coff;
     dvar_matrix length_sel;
     dvar3_array relative_bio;
     dvar3_array len_dist;
     dvar3_array RB;
     d3_array ORB;
     dvar4_array vars;
     dvar4_array sdevs;
     int nlint; // no.  of length bins
     double shlen; // bottom of 1st length bin
     double filen; // width of length bins
     int nwint; // no.  of weight bins
     double wshlen; // bottom of 1st weight bin
     double wfilen; // width of weight bins
     dvar_vector vb_coff;
     dvar_vector vb_bias;
     ivector common_vb_bias;
     dvar_vector common_vb_bias_coffs;
     dvar_vector sv;
     dvar_vector growth_dev;
     dvar_vector var_coff;
     dvar4_array tprob;
```

```
     dvar4_array wtprob;
     dvar_matrix lengthbsel;
     d4_array len_freq;
     d4_array wght_freq;
     d3_array len_sample_size;
     d3_array wght_sample_size;
     double fmin1;
     double fmax1;
     double fminl;
     double fmaxl;
     double vmin1;
     double vmax1;
     double vminl;
     double vmaxl;
     double rhomin;
     double rhomax;
     int nfmbound;
     ivector ifper;
     ivector ifinc;
     ivector iffish;
     ivector iageclass;
     dvector fmmin;
     dvector fmmax;
     dvector pdown;
     dvector pup;
     friend dvariable
     objective_function(dvar_len_fish_stock_history& fsh);

public:

     void big_fish_catch(void);
     void do_everything_calc(dvariable&);
     void mean_length_calc(void);
     void mean_length_calcx(void);
     void variance_calc(void);
     void predicted_frequency_calc(void);
     void fast_pred_frequency_calc(void);
     void fast_weight_pred_frequency_calc(void);
     void fast_pred_frequency_calc_len_based(void);
     dvariable reset(dvar_vector& x);
```

```
int nvcal(dvector&x);
friend par_uostream& operator << (par_uostream& pof,
dvar_len_fish_stock_history& fsh);
friend par_ofstream& operator << (par_ofstream& pof,
dvar_len_fish_stock_history& fsh);
dvariable robust_fit(void);
dvariable fmeanpen(void);
void obs_length_moments_calc(
dvar_matrix& obs_catch_moment1,dvar_matrix& obs_catch_moment2);
void pred_length_moments_calc(dvar_matrix& pred_catch_moment1,
dvar_matrix& pred_catch_moment2);
void main_length_calcs(dvar_vector& tprobb,dvar_vector& propp,
dvar_vector& mean_len,dvar_vector& sigg);
void main_length_calcs2(dvar_vector& tprobb,
dvar_vector& propp,dvar_vector& mean_len,dvar_vector& sigg,
dvar_vector& relative_num_at_length,dvar_vector& length_sel,
dvar_vector& nnum_fish,dvar_vector& llen_dist,int fi);
void main_length_calcs_len_based(dvar_vector& tprobb,
dvar_vector& propp, dvar_vector& mean_len,
dvar_vector& sigg, dvar_vector& nnum_fish,
dvar_vector& llendist, int fi);
void calculate_the_biomass_by_region(int ir,int i);
dvariable fit_tag_returns_parallel(void);
void main_length_calcs_print(uostream& ofs,
dvar_vector& propp,dvar_vector& mean_len,
dvar_vector& sigg);
dvector get_mean_length(int k)
{
return value(mean_length( fishing_region(k),fishing_period(k),
fishing_incident(k)) );
}
dvector get_props(int k)
{
return value(prop( fishing_region(k),fishing_period(k),
fishing_incident(k) ));
}
dvector get_vars(int k)
{
return value(vars(fishing_region(k),fishing_period(k),
```

```
fishing_incident(k) ));
}
d4_array vb_length_calc(void);
dvector vb_length_calc(int,int,int);
void print_pred_frequencies(ofstream& ofs);
void print_pred_frequencies(uostream& ofs);
dvector main_length_calcs_print(ofstream& ofs,dvar_vector& propp,
dvar_vector& mean_len,dvar_vector& sigg);
int nvcal(void);
dvariable dvar_len_fish_stock_history::fit_tag_returns(void);
dvariable dvar_len_fish_stock_history::
fit_tag_returns_sqrt(void);
void read_tagging_data(adstring& root,int);
void set_control_switches(void);
void set_shark_control_switches(void);
void set_Yukio_control_switches(void);
void set_gridsearch_control_switches(void);
void albacore_control_switches(void);
void xinit(dvector& x);
dvar_len_fish_stock_history(dvar_fish_stock_history& fsh,
int _nlint, double& _shlen,double& _filen );
dvar_len_fish_stock_history(dvar_fish_stock_history& fsh,
int _nlint, double& _shlen,double& _filen,i3_array& x,
i3_array& u,int _nwint);
void cons_convert_tag_lengths_to_age(void);
void var_convert_tag_lengths_to_age(void);
void cfast_pred_frequency_calc(dvar_len_fish_stock_history& cfsh);
void set_some_flags(ivector& dataswitch);
void cons_observed_tag_catch_calc(void);
void observed_tag_catch_calc(int direction_flag);
void observed_tag_catch_by_length_calc(void);
void observed_tags_by_age_from_length(void);
void observed_tags_by_age_from_length_pooled(void);
void tot_tags_catch(void);
void print_tag_data(int ir,int ip,ofstream &ofs);
dvariable fit_pooled_tag_returns(void);
dvariable grouped_tag_reporting_rate_penalty(void);
dvariable biomass_dynamics_pt(void);
void yield_analysis_pt(ofstream * pof);
```

```
     void calculate_the_biomass(void);
     void calculate_the_biomass_by_region(void);
     void calculate_the_catch_biomass(void);
     void calculate_the_catch_numbers(void); // JH 27/03/02
     void get_fishing_mortality_by_age_by_year_by_region(void);
     void get_fishing_mortality_by_age_by_year(void);
     void calculate_the_mean_weights_at_age(void);
     void mean_lengths_by_year_calc(void);
     void mean_weights_by_year_calc(void);
     void get_equilibrium_structure_for_yield(void);
     dvariable fit_tag_returns_mix(void);
     dvariable fit_pooled_tag_returns_mix(void);
     void tag_returns_report(const ofstream &);
     void main_weight_calcs(dvar_vector& tprobb,dvar_vector& propp,
     dvar_vector& mean_len,dvar_vector& sigg);
     void setup_some_stuff(double tmp_len_wt_coff, double_wshlen,
     double _wfilen,int nwint, int _num_fisheries,
     imatrix _dff,par_cifstream * _pinfile,
     int _month_1,int _first_time,int _mfactor);
     void print_pred_wght_frequencies(ofstream& ofs);
     dvector main_wght_calcs_print(ofstream& ofs,
     dvar_vector& propp,dvar_vector& mean_len,dvar_vector& sigg);
     void set_effective_length_and_weight_sizes(void);

}; // end class dvar_len_fish_stock_history
class fishery_catch_at_age_record_array
{
// this class is intended to hold an array of FISHERY catch at age
// records in such a form that they can be read in sequentially from
// a user's FRQ file.  Then they will probably be sorted to be used in
// a fish_stock_history structure.

     int nage;
     int index_min;
     int index_max;
     fishery_catch_at_age_record * ptr;
     void allocate(int nage)
     {
     for (int i=indexmin();i<=indexmax();i++)
```

```
    {
    (*this)(i).allocate(nage);
    }
    }
    void allocate(fishery_header_record_array& fhra,int ng);

public:

    int indexmin(){return index_min;}
    int indexmax(){return index_max;}
    int size(){return index_max-index_min+1;}
    int numage(){return nage;}
    fishery_catch_at_age_record& operator [] (int i)
    {
    #ifdef SAFE_ARRAYS
    check_index(index_min,index_max,i,
    " fishery_catch_at_age_record& operator [] (int i)");
    #endif
    return ptr[i];
    }
    fishery_catch_at_age_record& elem(int i)
    {
    #ifdef SAFE_ARRAYS
    check_index(index_min,index_max,i,
    " fishery_catch_at_age_record& elem(int i)");
    #endif
    return ptr[i];
    }
    fishery_catch_at_age_record& operator () (int i)
    {
    #ifdef SAFE_ARRAYS
    check_index(index_min,index_max,i,
    " freq_record& operator [] (int i)");
    #endif
    return ptr[i];
    }
    fishery_catch_at_age_record_array(int min,int max,int ng)
    {
    nage=ng;
    index_min=min;
```

```
    index_max=max;
    int sz=size();
    ptr=new fishery_catch_at_age_record [sz];
    if (ptr==NULL)
    {
    cerr << "Error allocating memory in fishery_freq_record_array"<<endl; exit(21);
    }
    ptr-=indexmin();
    allocate(nage);
    }
    friend ostream& operator <<
    (ostream& ofs, fishery_catch_at_age_record_array& fcara);
    fishery_catch_at_age_record_array(fishery_header_record_array& fhra, int ng);
    ~fishery_catch_at_age_record_array()
    {
    ptr+=indexmin();
    delete [] ptr;
    ptr=NULL;
    }
    void sort(void);

}; // end class fishery_catch_at_age_record_array
```

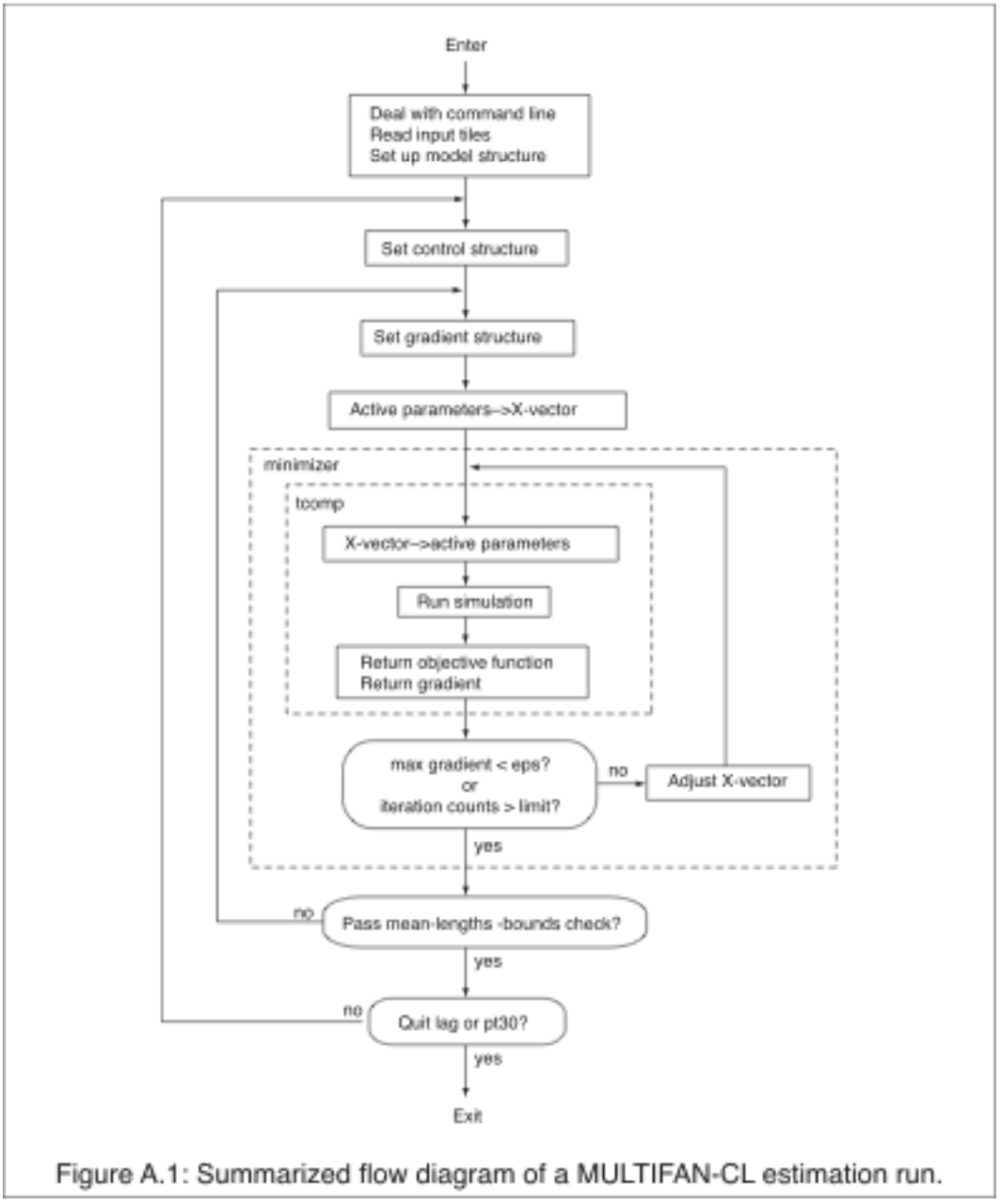


Figure A.1: Summarized flow diagram of a MULTIFAN-CL estimation run.

A.4 Hessian calculation in parallel

The following R scripts facilitate the parallel calculation of the Hessian and the derivatives of dependent variables that can typically be computationally intensive. The scripts submit the parallel calculations over a cluster of workstations with the independent jobs being managed among the workstations by condor software (http://www.cs.wisc.edu/condor). However, this is not essential and the scripts can be modified to assign parts of the calculation to specific workstations. The positions of the parameter sections to be calculated in parallel are controlled by parest_flags(223) and parest_flags(224).

1. Script to create the doitall and condor submit files

```
# set up job parameters
win <- (.Platform$OS.type=="windows")
if (win) {
dir <- "L:\\condor_mfcl\\hessian\\"
} else
{dir<-"/home/mfcl/condor_mfcl/hessian/"}
nsplit <- 20
frq.file <- "alb.frq"
ini.file <- "alb.ini"
tag.file <- "alb.tag"
par.file <- "estall.par"
your.email <- "simonh@spc.int"
# set up standard parameters
if (win) {
template.dir <- "L:\\condor_mfcl\\hessian\\"
} else {
template.dir<-"/home/mfcl/condor_mfcl/hessian/"
}
setwd(dir)
npars <- scan(par.file,skip = grep("# The number of parameters", readLines(par.file)),
nlines=1, quiet=TRUE)
doitall.template <- paste(c(template.dir,"condor_doitall_hess.template"),collapse="")
doitall.file <- "condor_doitall_hess.doit"
sub.template <- paste(c(template.dir,"sub.hessian.template"),collapse="")
sub.file <- "sub.hessian.sub"
f.mfcl.lin <- paste(c(template.dir,"mfclo32.lin"),collapse="")
f.mfcl.win <- paste(c(template.dir,"mfclo32.exe"),collapse="")
f.mfcl.cfg <- paste(c(template.dir,"mfcl.cfg"),collapse="")
f.exe.bat <- paste(c(template.dir,"mfcl.$$(Opsys).bat"),collapse="")
arg2 <- (1:nsplit)*ceiling(npars/nsplit)
arg1 <- arg2
arg1[2:nsplit] <- arg2[1:(nsplit-1)]+1
arg2[nsplit] <- npars
arg1[1] <- 1
```

```
# write out new doitall file

runline <- paste(c(" ./mfclo32",frq.file,par.file,"junk -switch 3 1 145 1 1 223 $1 1 224
$2"),collapse=" ")

a <- readLines(doitall.template)

out <- file(doitall.file, "w")

cat(a, "\n", file=out, sep="\n", append=TRUE)

cat(runline, "\n", file=out, sep="", append=TRUE)

close(out)

# write new submit file

a <- readLines(sub.template)

out <- file(sub.file, "w")

cat(a, "\n", file=out, sep="\n", append=F)

exeline <- paste(c("executable =", f.exe.bat),collapse="")

cat(exeline,"\n", file=out, sep="", append=TRUE)

tifline1 <- paste(c("TRANSFER_INPUT_FILES=",f.mfcl.lin),collapse="")

tifline <- paste(c(tifline1,f.mfcl.win,f.mfcl.cfg,frq.file,ini.file,tag.file,par.file,
doitall.file),collapse=",")

emline <- paste(c("notify_user=",your.email),collapse="")

cat(emline,"\n", file=out, sep="", append=TRUE)

cat(tifline,"\n", file=out, sep="", append=TRUE)

for (i in 1:nsplit)

{

line1 <- paste(c("arguments=",arg1[i]," ",arg2[i]),collapse="")

cat(line1, "\n", file=out, sep="", append=TRUE)

cat("queue","\n", file=out, sep="", append=TRUE)

}

close(out)

# submit condor run

# system(paste(c("c:\\cygwin\\bin\\dos2unix.exe",doitall.file),collapse=" "))

if (win) {

system(paste(c("c:\\cygwin\\bin\\dos2unix.exe",doitall.file),collapse=" "))

system(paste(c("condor_submit",sub.file),collapse=" "))

}

#if (!win) system("export PATH=$PATH:/opt/condor-6.8.8/bin")

if (!win) system(paste(c("/opt/condor-6.8.8/bin/condor_submit",sub.file),collapse=" "))
```

1. Template doitall and condor.sub files

a. doitall

```
#!/bin/sh
export PATH=$PATH:$ADTMP1:/usr/local/lib/
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
cd $ADTMP1
set
```

a. condor.sub

```
# MFCL & R always run in vanilla universe.  Other option is standard universe - Linux only, needs
# recompiling with condor.  We haven't used it yet.
universe = vanilla
# The batch file that calls cygwin/bash and runs the doitall.  Files must be in submit directory.  Maybe
# standardize to say I:/condor/mfcl.$$(Opsys).bat
executable = mfcl.$$(Opsys).bat
# Necessary at least for jobs that run on a windows PC
getenv = true
# Can set initialdir if running multiple jobs, though these directories have to be set up beforehand
#initialdir = sim.$(Cluster).$(Process)
# Each submission goes to a $(Cluster).  A single submission can have multiple processes (e.g.  "queue
# 10" would give 10 processes)
# Error messages, log file, and the output.  Output records all the useful screen output from MFCL.
error = $(Cluster).$(Process).condor_mfcl.err
log = $(Cluster).$(Process).condor_mfcl.log
output = $(Cluster).$(Process).condor_mfcl.out
# bring everything back when it's done
should_transfer_files = YES
# Requirements are flexible
Requirements = (OpSys == "LINUX" || OpSys =="WINNT51" || OpSys =="WINNT60") && \
(Arch=="INTEL" || Arch=="X86_64") && \
```

```
((memory > 1200 && ((name!= "slot1@simonh_pc.noumea.spc.local" || KeyboardIdle > 1800) &&
machine!="betty.spc.int")) || \
(name== "slot1@SIMONH_LT.noumea.spc.local" || name=="slot1@peters_pc.noumea.spc.local"))
# rename results to ID the cluster and process
transfer_output_remaps = \
"test.par = $(Cluster).$(Process).out.par ; \
doitall.condor = $(Cluster).$(Process).doitall.alb ; \
length.fit = $(Cluster).$(Process).length.fit ; \
weight.fit = $(Cluster).$(Process).weight.fit ; \
plot.rep = $(Cluster).$(Process).plot.rep"
# bring it all back whether or not it succeeds
when_to_transfer_output = ON_EXIT_OR_EVICT
# set priority, default is 0
priority=0
```

1. Script to collate results

```
fdat <- c(1,871,1741)
nfiles <- length(fdat)
fdat2 <- paste(fdat,fdat,sep="_")
fnames <- as.character(interaction("alb.hes",fdat2,sep=""))
# set up job parameters
win <- (.Platform$OS.type=="windows")
if (win)
dir <- "C:\\Documents and Settings\\simonh\\My Documents\\test\\" else
dir<-"/home/mfcl/condor_mfcl/hessian/"
setwd(dir)
file1 <- paste(c(dir,fnames[1]),collapse="")
outfile <- paste(c(dir,"test.hes"),collapse="")
# open the first file
con1r <- file(file1, open="rb") # opens binary file for reading
con2w <- file(outfile, open="wb") # opens binary file for writing
close(con2w)
con2w <- file(outfile, open="ab") # opens binary file for appending
# read the first file
size<- readBin(con1r, "integer", n = 3)
```

```
npar <- size[1]
nrows <- size[3] - size[2] + 1
a <- matrix(nrow=nrows,ncol=npar)
prow <- 0
for (i in 1:nrows)
{
prow <- prow+1
a[i,] <- readBin(con1r, "double", n = npar)
writeBin(a[i,],con2w)
}
close(con1r)
ff=2
for (ff in 2:nfiles)
{
con <- file(fnames[ff], open="rb") # opens binary file for reading
s2 <-readBin(con, "integer", n = 3)
a <- matrix(nrow=nrows,ncol=npar)
for (i in 1:min(c(nrows,npar-nrows*(ff-1))))
{
prow <- prow+1
tmp <- readBin(con, "double", n = npar)
a[i,] <- tmp
writeBin(a[i,],con2w)
}
close(con)
}
close(con2w)
res <- matrix(nrow=npar,ncol=npar)
con <- file("test.hes", open="rb") # opens binary file for reading
for (i in 1:npar)
{
res[i,] <- readBin(con, "double", n = npar)
}
```

Appendix B

Complete Flag List

Where a flag appears in the flag lists in section 4.5, a shortened description is given here along with a reference back to that section. Flags flagged with a "9" are also used in the MULTIFAN-CL code but are not (yet) listed in section 4.5. Those flagged with a "!", are found only in code that is presently commented out. Those not flagged at all are not found anywhere in the code.

B.1 parest flags

Flag Action/Meaning/Status
parest 1 function evaluation limit [4.5.1]
parest 2
parest 3
parest 4
parest 5
parest 6
parest 7
parest 8
parest ∃
parest 10
parest 11
parest 12 avg size of 1st age class [4.5.4]
parest 13 avg size of last age class [4.5.4]
parest 14 estimate K [4.5.4]
parest 15 generic _ of size at age [4.5.4]
parest 16 size dependent _ [4.5.4]
parest 17
parest 18
parest 19
parest 20 ∃ (sub-phase number during initial phase of fit)
parest 21 ∃
parest 22
parest 23 ∃
parest 24
parest 25
parest 26
parest 27
parest 28
parest 29
parest 30 ∃
parest 31 !
parest 32 initial control regime [4.5.1]
parest 33 upper bound for tag rep. rate [4.5.9]
parest 34 ∃
parest 35
parest 36
parest 37
parest 38
parest 39
parest 40
parest 41 tot weight fit only [4.5.1]
parest 42
parest 43
parest 44
parest 45
parest 46
parest 47
parest 48
parest 49
parest 50 maximum gradient target [4.5.1]
parest 51
parest 52
parest 53
parest 54
parest 55
parest 56
parest 57
parest 58
parest 59
parest 60
parest 61
parest 62
parest 63
parest 64
parest 65
parest 66
parest 67
parest 68
parest 69
parest 70
parest 71
parest 72
parest 73

parest 74 generic selectivity penalty [4.5.7]
parest 75 ∃
parest 76
parest 77
parest 78
parest 79
parest 80
parest 81
parest 82
parest 83
parest 84
parest 85
parest 86
parest 87
parest 88
parest 89
parest 90
parest 91
parest 92
parest 93
parest 94
parest 95
parest 96
parest 97
parest 98
parest 99
parest 100 ∃
parest 101 ∃
parest 102
parest 103
parest 104
parest 105
parest 106
parest 107
parest 108
parest 109
parest 110 ∃
parest 111 likelihood type for tags [4.5.9]
parest 112
parest 113
parest 114
parest 115
parest 116
parest 117
parest 118
parest 119
parest 120 ∃
parest 121
parest 122
parest 123
parest 124
parest 125
parest 126
parest 127
parest 128
parest 129
parest 130
parest 131
parest 132
parest 133
parest 134
parest 135
parest 136
parest 137
parest 138
parest 139 WF likelihood function selection [4.5.3]
parest 140 ∃
parest 141 LF likelihood function selection [4.5.3]
parest 142 last time step for catch/size devs [4.5.18]
parest 143 ∃
parest 144 ∃
parest 145 hessian and standard dev. report [4.5.1]
parest 146 scale gradient for tot pop. [4.5.1]
parest 147
parest 148 penalty for diff. betw. last 2 recruitments [4.5.12]
parest 149 penalty for recruitment devs [4.5.12]
parest 150 ∃
parest 151 ∃
parest 152 rescale gradients to 1 [4.5.1]
parest 153 ∃
parest 154
parest 155 enable orthogonal polynomial for recruitment [4.5.12]
parest 156 ∃
parest 157 density dep. growth [4.5.4]
parest 158 ∃
parest 159 ∃
parest 160 ∃
parest 161 ∃
parest 162 ∃
parest 163 ∃
parest 164 ∃
parest 165 ∃
parest 166 ∃
parest 167 ∃
parest 168 rescale age in growth function [4.5.4]
parest 169 rescalling a fn of length [4.5.4]
parest 170 ∃
parest 171 power term on age [4.5.4]
parest 172 ∃
parest 173 1st n lengths are indep. parameters [4.5.4]
parest 174 alt. VB growth parameterization – gmlxx
parest 175 rel. to age pars(4) (growth devs) – gml
parest 176 smooths gml & presumably age pars(4)
parest 177
parest 178
parest 179
parest 180 ∃
parest 181 ∃
parest 182 penalty wt. for length estimation [4.5.4]
parest 183 orthog. poly.: start time period, year [4.5.12]
parest 184 estimate (parest flag 173) parameters [4.5.4]
parest 185
parest 186 write fishmort and plotq0.rep [4.5.1]
parest 187 write temporary_tag_rep [4.5.1]
parest 188 write ests.rep and tag.rep [4.5.1]
parest 189 write .fit files [4.5.1]
parest 190 write .rep files [4.5.1]
parest 191 write simulation report files [4.5.1]
parest 192 no. terms in newton minimization [4.5.1]
parest 193 constant for normal size likelihood [4.5.3]
parest 194
parest 195 insert *check_numbers* in the *.par* [4.5.1]
parest 196
parest 197 new input par file (obsolete) [4.5.1]
parest 198 ∃
parest 199 ∃
parest 200 .par file version number
parest 201 orthog. poly.: lower bound degree, year [4.5.12]
parest 202 orthog. poly.: end time period, year [4.5.12]
parest 203 orthog. poly.: upper bound degree, year [4.5.12]

parest 204 orthog. poly.: start time period, region [4.5.12]
parest 205 orthog. poly.: lower bound degree, region [4.5.12]
parest 206 orthog. poly.: start time period, season [4.5.12]
parest 207 orthog. poly.: lower bound degree, season [4.5.12]
parest 208 orthog. poly.: start time period, region×season [4.5.12]
parest 209 orthog. poly.: lower bound degree, region×season [4.5.12]
parest 210 orthog. poly.: end time period, region [4.5.12]
parest 211 orthog. poly.: upper bound degree, regions [4.5.12]
parest 212 orthog. poly.: end time period, season [4.5.12]
parest 213 orthog. poly.: upper bound degree, seasons [4.5.12]
parest 214 orthog. poly.: end time period, region×season [4.5.12]
parest 215 orthog. poly.: upper bound degree, region×season [4.5.12]
parest 216 orthog. poly.: degree, region [4.5.12]
parest 217 orthog. poly.: degree, season [4.5.12]
parest 218 orthog. poly.: degree, region×season [4.5.12]
parest 219
parest 220
parest 221 orthog. poly.: degree, year [4.5.12]
parest 222
parest 223 start position parallel Hessian calculation [A.4]
parest 224 end position parallel Hessian calculation [A.4]
parest 225
parest 226 use Richards growth par
parest 227 estimate Richards growth par
parest 228
parest 229 parallel calculation of derivatives dep. variables
parest 230 parallel calculation of derivatives dep. variables
parest 231 random seed for numbers-at-age in simulations [4.5.18]
parest 232 first projection year for stochastic recruitments in simulations [4.5.18]
parest 233 last projection year for stochastic recruitments in simulations [4.5.18]
parest 234 activates randomised effort deviates in simulations [4.5.18]
parest 235 standard deviation of stochastic effort deviates in simulations [4.5.18]
parest 236 random seed for effort deviates in simulations [4.5.18]
parest 237 activates stochastic numbers-at-age in simulations [4.5.18]
parest 238 activates stochastic recruitments in simulations [4.5.18]
parest 239 activates stochastic recruitment deviates in simulations [4.5.18]
parest 240 activates fit to age-length data [4.5.10]
parest 241 activates generate pseudo-observations [4.5.18]
parest 242 activates generate pseudo-observations in estimation periods [4.5.18]
parest 243
parest 244
parest 245
parest 246
parest 247
parest 248
parest 249
parest 250
parest 251
parest 252
parest 253
parest 254
parest 255
parest 256
parest 257
parest 258
parest 259
parest 260
parest 261
parest 262
parest 263
parest 264
parest 265
parest 266
parest 267
parest 268
parest 269
parest 270
parest 271
parest 272
parest 273
parest 274
parest 275
parest 276
parest 277
parest 278
parest 279
parest 280 estimate WF log(var) for normal student-t [4.5.3]
parest 281
parest 282 estimate WF log(dof) for normal student-t [4.5.3]
parest 283
parest 284
parest 285
parest 286
parest 287
parest 288
parest 289
parest 290 estimate LF log(var) for normal student-t [4.5.3]
parest 291
parest 292 estimate LF log(dof) for normal student-t [4.5.3]
parest 293
parest 294
parest 295
parest 296
parest 297
parest 298
parest 299
parest 300 estimate WF tot_exp for normal student-t [4.5.3]
parest 301 activate WF tail compression [4.5.3]
parest 302 minimum threshold WF sample size [4.5.3]
parest 303 proportion for WF tail compression [4.5.3]
parest 304
parest 305 estimate tag neg.binomial variance [4.5.9]
parest 306 bounds for tag neg.binomial variance [4.5.9]
parest 307
parest 308
parest 309
parest 310 estimate LF tot_exp for normal student-t [4.5.3]
parest 311 activate LF tail compression [4.5.3]
parest 312 minimum threshold LF sample size [4.5.3]
parest 313 proportion for LF tail compression [4.5.3]
parest 314
parest 315 SSM-RE bound for length rho correlation [4.5.3]
parest 316 bounds for SSM-RE log-length/weight variance [4.5.3]
parest 317 SSM-RE bounds length/weight samp.sz covariate [4.5.3]
parest 318

parest 319
parest 320 SSM-RE tail compression min.length size [4.5.3]
parest 321
parest 322
parest 323 estimate selectivity deviate coefficients [4.5.7]
parest 324
parest 325
parest 326
parest 327
parest 328
parest 329
parest 330 SSM-RE tail compression min.weight size [4.5.3]
parest 331
parest 332
parest 333
parest 334 SSM-RE u.bound log-length N variance [4.5.3]
parest 335 SSM-RE u.bound log-weight N variance [4.5.3]
parest 336 SSM-RE u.bound length samp.sz covariate [4.5.3]
parest 337 SSM-RE u.bound weight samp.sz covariate [4.5.3]
parest 338 SSM-RE M-estimator coefficients selection [4.5.3]
parest 339
parest 340
parest 341
parest 342 DM_noRE assumed maximum sample size [4.5.3]
parest 343
parest 344
parest 345
parest 346 target average or depleted biomass [4.5.15]
parest 347 target depletion level [4.5.15]
parest 348 penalty weight on target biomass [4.5.15]
parest 349
parest 350 use catch sex-ratio for aggregating multi-sex or multi-species size composition data [4.5.3]
parest 351 objective function minimising method [4.5.1]
parest 352
parest 353 optimise simulation model calculations [4.5.18]
parest 354
parest 355
parest 356
parest 357
parest 358
parest 359
parest 360
parest 361
parest 362
parest 363
parest 364
parest 365
parest 366
parest 367
parest 368
parest 369
parest 370 SSM-RE bound for weight rho correlation [4.5.3]
parest 371
parest 372
parest 373
parest 374
parest 375
parest 376
parest 377
parest 378
parest 379
parest 380
parest 381
parest 382
parest 383
parest 384
parest 385
parest 386
parest 387
parest 388
parest 389
parest 390
parest 391
parest 392
parest 393
parest 394
parest 395
parest 396
parest 397
parest 398 Terminal recrs. are arithmetic mean [4.5.12]
parest 399
parest 400 Excl. terminal recr.s estimation [4.5.12]

B.2 age flags

Flag Action/Meaning/Status
age 1
age 2
age 3
age 4
age 5 activate rec_init_diff scalar [4.5.12]
age 6
age 7
age 8
age 9 ∃
age 10
age 11
age 12
age 13
age 14
age 15
age 16
age 17
age 18
age 19
age 20 no. simulations in stochastic projections [4.5.18]
age 21
age 22
age 23
age 24
age 25
age 26
age 27 penalty weight on movement [4.5.11]
age 28 penalty weight on movement-at-age [4.5.11]
age 29 penalty weight on movement-at-age [4.5.11]
age 30 est tot recruit [4.5.12]
age 31 est totpop [4.5.12]
age 32 totpop fixed or not [4.5.1 and 4.5.12]
age 33 est. M [4.5.17]
age 34 est. effort devs [4.5.8]
age 35 maximum bound on effort deviates [4.5.8]
age 36 maximum bound on selectivity deviates [4.5.7]
age 37 avg. F target [4.5.14]
age 38 ∃
age 39 penalty on F target [4.5.14]
age 40 ∃

age 41 ∃
age 42 ∃
age 43 no. terminal years for F target [4.5.14]
age 44 annual F target [4.5.14]
age 45 ∃
age 46 ∃
age 47 ∃
age 48 ∃
age 49 ∃
age 50 ∃
age 51 ∃
age 52 ∃
age 53 movement frequency [4.5.11]
age 54 ∃
age 55 ∃
age 56 ∃
age 57 month doubling [4.5.6 and 4.5.12]
age 58 ∃
age 59 ∃
age 60 ∃
age 61 ∃
age 62 ∃
age 63 ∃
age 64 ∃
age 65 ∃
age 66 ∃
age 67 ∃
age 68 estimate movement pars [4.5.11]
age 69 activate movement pars [4.5.11]
age 70 activate time series of reg recruitment [4.5.12]
age 71 est. time series of reg recruitment [4.5.12]
age 72 recruitment and environment [4.5.12]
age 73 est. age-dep M [4.5.17]
age 74 ∃
age 75 ∃
age 76 penalty on initial population curvature [4.5.12]
age 77 penalty on 2nd derivative of M_a [4.5.17]
age 78 penalty on 1st derivative of M_a [4.5.17]
age 79 penalty on $\sum(M_a$ - $zoupM)^2$ [4.5.17]
age 80 penalty on $zoupM$'^{2} [4.5.17]
age 81 terminal ages with the same M [4.5.17]
age 82 target $zoupM$' [4.5.17]
age 83 min age in $zoupM$' [4.5.17]
age 84 penalty on target $zoupM$' [4.5.17]
age 85 max age in $zoupM$' [4.5.17]
age 86 ∃
age 87 ∃
age 88 activate age-dep movement pars [4.5.11]
age 89 est. age-dep movement pars [4.5.11]
age 90 activate non-linear age-dep movement pars [4.5.11]
age 91 est. non-linear age-dep movement pars [4.5.11]
age 92 catch errors [4.5.1]
age 93 ∃
age 94 starting pop strategy [4.5.12]
age 95 initial periods for average total mortality [4.5.12]
age 96 tag pooling [4.5.9]
age 97 target biomass ratio [4.5.15]
age 98 penalty on biomass ratio [4.5.15]
age 99 no. inital and final yeas for calc biomass ratio [4.5.15]
age 100 neg. binomial for tag likelihood [4.5.9]
age 101 activate *.env* file [4.5.12]
age 102 est. environmental correlation par [4.5.12]
age 103 in code but commented out
age 104 enable implicit q devs [4.5.6]
age 105 ∃
age 106 ∃
age 107 penalty on overall exploitation [4.5.14]
age 108 overall exploitation target [4.5.14]
age 109 whether log(M) at age by deviations from mean or direct [4.5.17]
age 110 penalty on region rec diffs [4.5.12]
age 111 ∃
age 112 numbers or biomass for MSY calcs [4.5.13]
age 113 scaling init. pop and rectuitment [4.5.12]
age 114
age 115 SS2-type catch estimation
age 116 Maximum fishing mortality [4.5.2]
age 117
age 118
age 119
age 120 ?????????
age 121 ?????????
age 122 ?????????
age 123 ?????????
age 124 ?????????
age 125 use biomass-dependent catchability [4.5.6]
age 126 estimate region_pars(2) [4.5.6]
age 127 penalty for estimating region_pars(2) [4.5.6]
age 128 multiplier for M for init_pop calculation [4.5.12]
age 129 autocorrelated recruitment deviates [4.5.12]
age 130 M/K target [4.5.17]
age 131 age1 for M/K target [4.5.17]
age 132 age2 for M/K target [4.5.17]
age 133 penalty on M/K target [4.5.17]
age 134
age 135 autocorrelated stock-recruitment deviates [4.5.12]
age 136
age 137
age 138
age 139
age 140 enable region-specific yield anal. and set Fmult resolution [4.5.13]
age 141 default no. Fmult steps in region-specific yield anal. [4.5.13]
age 142
age 143 ∃
age 144 common wt for catch L to 10000 [4.5.2]
age 145 penalty on stock-recruit pars [4.5.13]
age 146 activate stock-recruit [4.5.13]
age 147 lag betw. spawning and recruitment [4.5.13]
age 148 years from last year for avg. F [4.5.13]
age 149 yield in wt or numbers [4.5.13]
age 150 penalty for biomass dynamics est [4.5.13]
age 151 activate r in Pella-Tomlilnson [4.5.13]
age 152 activate K in Pella-Tomlilnson [4.5.13]
age 153 a of beta prior in SRR [4.5.13]
age 154 b of beta prior in SRR [4.5.13]
age 155 years from last year to omit from avg F [4.5.13]
age 156 use fish pars 7 in model [4.5.6]
age 157
age 158 was region-specific yield anal. now available for other use
age 159 ∃
age 160 penalty for fitting to effort in catch conditioned model [4.5.8]
age 161 activate SRR log-normal bias correction [4.5.13]
age 162 activate steepness fit in SRR [4.5.13]
age 163 select alternate parameters for SRR [4.5.13]
age 164
age 165 target for $F_{\rm msy}/F$ or $B/B_{\rm msy}$ [4.5.13]

age 166 penalty weight for F_{msy}/F or B/B_{msy} [4.5.13]
age 167 target F_{msy}/F or B/B_{msy} [4.5.13]
age 168 exponent in weighted average for F_{msy}/F [4.5.13]
age 169 divisor in weighted average for F_{msy}/F [4.5.13]
age 170 enable a biomass depletion target [4.5.16]
age 171 unfished calculations by estimated recruitment or SRR [4.5.16]
age 172 total or adult biomass depletion [4.5.15] [4.5.16]
age 173 first time period for reckoning depletion or average [4.5.15] [4.5.16]
age 174 last time period for reckoning depletion or average [4.5.15] [4.5.16]
age 175 100X target depletion level [4.5.16]
age 176 penalty on depletion target [4.5.16]
age 177 ??????
age 178 constraint on regional recruitments [4.5.12]
age 179
age 180
age 181
age 182 annualised SRR fit [4.5.13]
age 183 allocate annual SRR by season [4.5.18]
age 184
age 185 ∃
age 186 st.dev pseudo-observed effort [4.5.18]
age 187 st.dev pseudo-observed catch [4.5.18]
age 188 convert maturity at length to age [4.5.5]
age 189
age 190 time period for avg recr. in projection [4.5.18] [4.5.16]
age 191 time period for avg recr. in projection [4.5.18] [4.5.16]
age 192 ∃
age 193 shared selectivity at length among sexes [4.5.7]
age 194 include effort devs in yield analysis [4.5.13]
age 195 activates SRR prediction for af(190, 191) [4.5.18]
age 196
age 197
age 198 activate release group reporting rates [4.5.9]
age 199 start time period for yield calculation [4.5.13]
age 200 end time period for yield calculation [4.5.13]

B.3 fish flags

Flag Action/Meaning/Status
fish 1 est. avg. q [4.5.6]
fish 2 ∃
fish 3 1st age of common selectivity [4.5.7]
fish 4 est. effort devs [4.5.8]
fish 5 ∃
fish 6 ∃
fish 7 ∃
fish 8 ∃
fish 9 ∃
fish 10 time series in q [4.5.6]
fish 11 selectivity bias, 1st age [4.5.4]
fish 12 ∃
fish 13 penalty for effort devs [4.5.8]
fish 14 limit on F per fishing incident [4.5.14]
fish 15 penalty for q-devs [4.5.6]
fish 16 selectivity shape [4.5.7]
fish 17 ∃
fish 18 ∃
fish 19 no. ages for sel_dev_coffs [4.5.7]
fish 20 penalty weight on sel_dev_coffs estimation [4.5.7]
fish 21 seasonal growth – under construction [4.5.4]
fish 22 grouping for sel. bias [4.5.4]
fish 23 time step for q-devs [4.5.6]
fish 24 grouping for selectivity [4.5.7]
fish 25
fish 26 length-dependent selectivities [4.5.7]
fish 27 sinusoidal seasonal q [4.5.6]
fish 28 grouping for seasonal q [4.5.6]
fish 29 grouping for common catchability deviations [4.5.6]
fish 30 ∃
fish 31 ∃
fish 32 grouping for tag recaptures [4.5.9]
fish 33 est. tag reporting rate [4.5.9]
fish 34 grouping for tag reporting rate [4.5.9]
fish 35 penalty for tag reporting rate prior [4.5.9]
fish 36 tag reporting rate prior [4.5.9]
fish 37 est. time series of tag reporting rate [4.5.9]
fish 38 σ^2 for Kalman filt. effort devs [4.5.8]
fish 39 σ^2 for Kalman filt. q devs [4.5.6]
fish 40 ∃
fish 41 penalty weight for second difference selectivity smoothing [4.5.7]
fish 42 penalty weight for third difference selectivity smoothing [4.5.7]
fish 43 σ^2 for neg. bin., see parest flags(111) [4.5.9]
fish 44 grouping for neg. bin. [4.5.9]
fish 45 fishery specific catch wts [4.5.2] and tag reporting rate deviations [4.5.9]?????
fish 46 est. of mixture pars in tag likelihood [4.5.9]
fish 47 arbitrary seasonal q [4.5.6]
fish 48 selectivity est. (obsolete??) [4.5.7]
fish 49 length sample size [4.5.3]
fish 50 weight sample size [4.5.3]
fish 51 effect of effort on q [4.5.6]
fish 52 grouping for fish flag 51 [4.5.6]
fish 53 effect of biomass on q [4.5.6]
fish 54 grouping for fish flag 53 [4.5.6]
fish 55 turn off fisheries for impact analysis [4.5.16]
fish 56 penalty to make selectivity a non decreasing function of age [4.5.7]
fish 57 functional form for selectivity [4.5.7]
fish 58
fish 59
fish 60 grouping for common initial catchabilityn [4.5.6]
fish 61 number of nodes for cubic spline selectivity [4.5.7]
fish 62 number of nodes for cubic spline selectivity [4.5.7]
fish 63
fish 64
fish 65 fat-tailed Cauchy distrib. for effort devs [4.5.8]
fish 66 time-varying effort wt (set internally) [4.5.8]
fish 67 SSM-RE log-length N variance [4.5.3]
fish 68 SSM-RE and DM_noRE parameters fishery grouping [4.5.3]
fish 69 DM_noRE length sample size multiplier [4.5.3]
fish 70 scale F of individual fleets for equilibrium yield calculations [4.5.13]
fish 71 time-blocks in time-variant selectivity [4.5.7]
fish 72 fishery-specific generic selectivity penalty [4.5.7]
fish 73
fish 74 seasons in time-variant selectivity [4.5.7]
fish 75 age classes for zero selectivity-at-age [4.5.7]
fish 76 SSM-RE log-weight N variance [4.5.3]
fish 77 SSM-RE and DM_noRE parameters fishery grouping [4.5.3]
fish 78 SSM-RE length rho correlation [4.5.3]

fish 79 multi-sex shared eff_devs [4.5.8]
fish 80 SSM-RE weight rho correlation [4.5.3]
fish 81
fish 82 SSM-RE log-length variance [4.5.3]
fish 83
fish 84 SSM-RE log-weight variance [4.5.3]
fish 85 SSM-RE length sample size covariate [4.5.3]
fish 86 SSM-RE weight sample size covariate [4.5.3]
fish 87 DM_noRE weight sample size multiplier [4.5.3]
fish 88 DM_noRE weight sample size covariate [4.5.3]
fish 89 DM_noRE length sample size covariate [4.5.3]
fish 90
fish 91
fish 92
fish 93
fish 94
fish 95
fish 96
fish 97
fish 98
fish 99
fish 100

B.4 region flags

Flag Action/Meaning/Status
The region_flags are a matrix having 10 rows and the number of columns is equal to the number of regions. The settings for the flags can be made as in the following note, via the -switch options, or in the input *.ini* file. It may be preferable to specify their values as -switch options if it is desired to activate the estimation of parameters in a particular phase of a model fitting procedure.

reg 1 activate estimation of recruitment distribution by region [4.5.12]
reg 2
reg 3
reg 4
reg 5
reg 6
reg 7
reg 8
reg 9
reg 10

Note: -99999 applies to all regions. Flags have values relating to the 10 rows: -100009 $\leq n \leq$ -100000 , where the first row is identified by -100000 and the last row by -100009. Region-specific values are specified as follows (using example for region flag 1,in row 1): the flag is -100000, followed by the flag number (being the region), and followed by the flag value (0 or 1), e.g. -100000 1 1 for region 1, activating estimation of recruitment distribution for region 1.

B.5 tag flags

Flag Action/Meaning/Status
tag 1 set number of tag mixing periods [4.5.9]
tag 2
tag 3
tag 4
tag 5
tag 6
tag 7
tag 8
tag 9
tag 10
Note: -9999 applies to all tag groups Tag group-specific values are: -10000 =< n =< -99999; i.e. value = -9999 – n. For example for release group 5, the value is –(9999 + 5) = -10004

Appendix C

Likelihood Profile - bash script

When calculating likelihood profiles in respect of a derived model quantity such as average absolute biomass or biomass depletion, see section 3.8.6, it is necessary to run an algorithm external of MULTIFAN-CL that estimates solutions for each element of a range of scalars in respect of the chosen model quantity. This algorithm may be run using a Linux bash script as follows.

```
Pen1=100000
Pen2=1000000
Pen3=10000000
Reps1=15
Reps2=25
Reps3=25
Reps4=1000
Reps5=100
Reps6=500
Af173=150
Af174=5
Prog="directory_path"/mfclo64
Frq=bet.frq
Initp=bio100.par
# Function that runs each set of evaluations within the for loop
function call_mf1 ()
{
    Temp=`bc -l <<< "$5*$6/100"`
    Target=`printf "%.0f" $Temp`
    if [ ! -f $4 ];
    then
```

```
        echo "file " $4 " does not exist"
        $1 $2 $3 $4 \
        -switch 10 2 32 1 1 187 0 1 188 0 -999 55 0 1 1 $8 1 346 2 1 347 $Target 1 348 $7 2 173 $9 2 174 $10
    else
        echo "file " $4 " exists already"
    fi
}
# get the MLE for average biomass
M0=0
MLE=0
if [ ! -f avg_bio ];
then
    call_mf1 $Prog $Frq $Initp bio${M0}a.par $M0 $MLE $Pen1 $Reps1 $Af173 $Af174
else
    echo "file avg_bio exists"
fi
M1=‘cat avg_bio‘
MLE=‘printf "%.0f" $M1‘
echo "The MLE for average biomass is " $MLE
# for loop over range of scalars
for Mult in 90 80 70 60 50
do
    call_mf1 $Prog $Frq $Initp bio${Mult}a.par $Mult $MLE $Pen1 $Reps1 $Af173 $Af174
    call_mf1 $Prog $Frq bio${Mult}a.par bio${Mult}b.par $Mult $MLE $Pen2 $Reps2 $Af173 $Af174
    call_mf1 $Prog $Frq bio${Mult}b.par bio${Mult}.par $Mult $MLE $Pen3 $Reps3 $Af173 $Af174
    call_mf1 $Prog $Frq bio${Mult}.par bio${Mult}final.par $Mult $MLE $Pen3 $Reps4 $Af173 $Af174
    call_mf1 $Prog $Frq bio${Mult}final.par bio${Mult}finalx.par $Mult $MLE $Pen3 $Reps5 $Af173 $Af174
    call_mf1 $Prog $Frq bio${Mult}finalx.par bio${Mult}finaly.par $Mult $MLE $Pen3 $Reps6 $Af173
$Af174
    call_mf1 $Prog $Frq bio${Mult}finaly.par bio${Mult}finalz.par $Mult $MLE $Pen3 $Reps4 $Af173
$Af174
    mv test_plot_output test_plot_output_${Mult}
    Initp=bio${Mult}finalz.par
done
```

Appendix D

Hessian diagnostic - R script

When calculating the singular value decomposition (SVD) diagnostic of the Hessian solution, the ouput file **.svd_report* is generated containing the eigenvalues and eigenvectors. The following script is useful in interpreting this report in respect of a positive or non-positive definite Hessian solution, and in identifying the most influential parameters.

```
# Read report file of eigenvalues and eigenvectors
setwd("c:/Testing/MTHRD/skj2016_var_err/")
# - firstly the eigenvalues
svd_eig <- scan("skj.svd_report",skip=1,nlines=1)
summary(svd_eig)
npar <- length(svd_eig)
svd_eig[c((npar-40):npar)]
hist(svd_eig,breaks=40)
#
# Read eigen_vectors for each parameter in respect of the other
# parameters
svd_v <- read.table("skj.svd_report",skip=4)
dim(svd_v)
# input parameter names for the model
pnms <- read.table("xinit.rpt",header=FALSE,row.names = NULL)
#
# Check consistency
if(length(pnms[,2]) != dim(svd_v)[2])
print("ERROR: xinit.rpt dimensions inconsistent with *.svd_report")
stop()

# make unique parameter names for eigenvector matrix
pnms2 <- cbind(pnms,paste(pnms[,1],pnms[,2],sep="_"))
# assign row and column names according to the parameters
dimnames(svd_v)[[1]] <- pnms2[,3]
dimnames(svd_v)[[2]] <- pnms2[,3]
#
# Plot of each row - but initially examine the first row
i <- 1
plot(c(1:dim(svd_v)[1]),svd_v[i,], main="Parameter-specific eigenvector", xlab="Parameter", ylab="Eigenvector
value")
```

```
absmat <- cbind(c(1:dim(svd_v)[2]),as.vector(as.numeric(abs(svd_v[1,]))))
sabsmat <- absmat[order(absmat[,2]),]
# Largest contributing params:
parlarg <- sabsmat[c((npar-40):npar),1]
# get the parameter names
pars_infl <- pnms2[pnms2[,1]%in%parlarg,3]
#Summary script for loop content for examining subsequent rows put the
# following within a loop over npar
#
# plot(c(1:dim(svd_v)[1]),abs(svd_v[4,]))
# absmat <- cbind(c(1:dim(svd_v)[2]),as.vector(as.numeric(abs(svd_v[4,]))))
# sabsmat <- absmat[order(absmat[,2]),]
# pnmlarg <- sabsmat[c((npar-40):npar),1]
# pars_infl <- pnms2[pnms2[,1]%in%pnmlarg,3]
# pars_infl
```

Index