MULTIFAN-CL User's Guide

Pierre Kleiber, John Hampton, and David A. Fournier

April, 2006

Copyright © 2003 by Pierre Kleiber, John Hampton, David A. Fournier.

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided the the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that the english version of this permission notice must be preserved on all copies.

Contents

1 Introduction **1**

2 Overview of MULTIFAN-CL **3**
2.1 What is MULTIFAN-CL? . . . 3
2.2 Some Concepts . . . 4
2.3 What Information is Used? . . . 6

3 Description of Model Processes **8**
3.1 Initial Population . . . 8
3.2 Recruitment . . . 9
3.2.1 Average recruitment and deviations . . . 9
3.2.2 Recruitment by orthogonal polynomials . . . 9
3.2.3 Recruitment by environmental driver . . . 9
3.3 Age and Growth . . . 9
3.4 Catch . . . 10
3.4.1 Catchability . . . 10
3.4.2 Selectivity . . . 11
3.4.3 Fishing effort . . . 13
3.5 Natural Mortality . . . 14
3.6 Movement . . . 15
3.7 Tag-Specific Processes . . . 15
3.7.1 Tag fishery grouping . . . 15
3.7.2 Tag pooling . . . 15
3.7.3 Tag mixing . . . 16
3.7.4 Tag reporting . . . 16
3.8 Biological Reference Points . . . 17
3.8.1 Impact analysis . . . 17
3.8.2 Yield analysis . . . 17
3.8.3 Pella-Tomlinson biomass dynamics . . . 20
3.8.4 Target refrence points . . . 21

4 Running MULTIFAN-CL **22**
4.1 Constructing Input Files . . . 23
4.1.1 The *.frq* file . . . 23
4.1.2 The *.ini* file . . . 29
4.1.3 The *.tag* file . . . 32
4.1.4 The *.par* File . . . 34

4.1.5 The *mfcl.cfg* file . . . 36
4.1.6 The *.env* file . . . 36
4.2 Temporal Structure of Model . . . 36
4.3 Conducting a Fit . . . 37
4.3.1 Creating an initial parameter file (`-makepar`) . . . 38
4.3.2 Command line manipulation of flags (`-switch`) . . . 38
4.3.3 Flag switches in a file (`-file`) . . . 39
4.3.4 Batch execution (*doitall* file) . . . 40
4.4 Projecting the Future . . . 44
4.5 Flag Settings . . . 45
4.5.1 Program Control Flags . . . 46
4.5.2 Catch Estimation . . . 47
4.5.3 Length/weight frequency data . . . 47
4.5.4 Growth Parameter Estimation . . . 47
4.5.5 Catchability Estimation . . . 48
4.5.6 Selectivity Estimation . . . 49
4.5.7 Effort Deviation . . . 50
4.5.8 Tagging Data . . . 50
4.5.9 Movement Parameters . . . 51
4.5.10 Recruitment, Initial Population, and Scaling . . . 52
4.5.11 MSY Stuff . . . 54
4.5.12 Fishing Mortality Targets . . . 55
4.5.13 Biomass Ratio Targets . . . 55
4.5.14 Fishery Impact Analysis . . . 56
4.5.15 Natural Mortality Estimation . . . 56
4.5.16 Projections . . . 57
4.6 Helpful Utilities . . . 58
4.6.1 Dealing with Flags . . . 58
4.6.2 Quick Access to *.par* Files . . . 60
4.6.3 Running in background . . . 61
4.6.4 `screen`: A utility for detaching and reattaching a running MULTI-FAN-CL Process . . . 62
4.6.5 Miscellaneous Other Utilities . . . 65

5 Likelihood Functions 66
5.1 Total Catch Data . . . 66
5.2 Length and Weight Frequency Data . . . 66
5.3 Tagging Data . . . 67
5.4 Penalties and Priors . . . 67
5.4.1 Normal priors . . . 68
5.4.2 Beta priors . . . 69

6 Interpreting Results 70
6.1 Output on Screen . . . 70
6.1.1 What to look for . . . 71
6.1.2 How to intervene . . . 71
6.1.3 When to intervene . . . 72

6.2 Output Files . . . 72
6.2.1 The *.par* File . . . 72
6.2.2 The *plot.rep* file . . . 73
6.2.3 The *length.fit* file . . . 76
6.2.4 The *tag.rep* file . . . 77
6.2.5 The *.var* file . . . 77
6.2.6 The *.dep* file . . . 77
6.2.7 The *.hes* file . . . 77
6.2.8 The *catch.rep* file . . . 77
6.2.9 The *ests.rep* file . . . 77
6.3 Obtaining Graphical Results . . . 77
6.3.1 Java . . . 77
6.3.2 R (or S) . . . 77

A Technical Annex **81**
A.1 Population dynamics model . . . 81
A.1.1 Age-structured dynamics . . . 81
A.1.2 Natural Mortality . . . 82
A.1.3 Movement . . . 82
A.1.4 Growth . . . 83
A.1.5 Calculating length and weight frequencies . . . 83
A.1.6 Fishing Mortality . . . 84
A.1.7 Tagged fish dynamics . . . 85
A.1.8 Aggregate Fishing Mortality . . . 86
A.1.9 Calculating total and spawning biomass . . . 86
A.1.10 Stock Recruitment Relationship . . . 87
A.1.11 Yield Curve . . . 87
A.1.12 Reference Points . . . 88
A.2 Parameter Estimation – The Objective Function . . . 89
A.2.1 Likelihood . . . 89
A.2.2 Priors and penalties . . . 91
A.2.3 Weighting factors, variance, and coefficient of variation . . . 95
A.3 Computational Details . . . 96
A.3.1 Navigating the Source Code . . . 96
A.3.2 Using a debugger . . . 97
A.3.3 Source Code Files . . . 99
A.3.4 Important functions . . . 100
A.3.5 Class descriptions . . . 100

B Complete Flag List **114**
B.1 parest flags . . . 114
B.2 age flags . . . 115
B.3 fish flags . . . 117
B.4 region flags . . . 117
B.5 tag flags . . . 117

Index **118**

List of Tables

5.1 Weighting factors and CV values for normal priors. 68
5.2 Age flag settings and shape of the beta distribution with range 0.2 to 1. . . 69

A.1 gdb and emacs commands. 98
A.2 Example gdb debugging session 99

List of Figures

3.1 Estimated biomass without fishing and under two fishing regimes. 18
3.2 Catch in selected fisheries with and without other fisheries 19
3.3 Ratio of biomass to biomass at MSY 20

A.1 Summarized flow diagram of a MULTIFAN-CL estimation run. 113

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 model[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 infeasible 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 unweildy. 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

[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.

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].

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.

[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 tuna (*Thunnus albacares*) in the western and central Pacific Ocean. Marine and Freshwater Research 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 asssessment 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 accomodated 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. How-

ever MULTIFAN-CL divides the model universe into regions with fish in a particlular 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 infact 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. MULTIFAN-CL 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 controled 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 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 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.

[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.

- 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 25). 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 26). 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 actual numbers of 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 90).

- Tagging data if available, may be incorporated into the analysis and are provided in the *.tag* file (see 4.1.3). 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 model because we assume that the tagged and untagged populations share the same parameters (mortality, growth, movement, etc).

- 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. This is equivalent to setting a uniform prior over the specified range. 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 specific information 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 4.5.10).

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, $-xM_a$, where M_a is the natural mortality by age and x is a multiplier set by age flag 95 (see 4.5.10).

   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, $-\overline{Z}_{a,r} = -(M_a + \overline{F}_{a,r})$, 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).

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. 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.

3.2.2 Recruitment by orthogonal polynomials

3.2.3 Recruitment by environmental driver

Recruitment can also be made sensitive to an environmental driving variable by setting age flags 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 4.1.6).

See section 4.5.10 for details on use of flags mentioned above.

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, $a^\star$, can be modified from von Bertalanffy by activating independent parameters for the average lengths of these younger age classes. The number of age classes so modified is set by parest flag 173, making for an additional $a^\star - 1$ parameters. The values of these parameters are stored in the *.par* file in the 4th row of age_pars (see 4.1.4).

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.

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 cummulative 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 accomodated 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

[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.

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^{β}, 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 effect is activated by setting age flag 157 = 1 (to use the biomass-dependent catchability hypothesis with parameter(s) stored in the 8th row of fishery parameters in the *.par* file) and by setting fish flag 53 = 1 to activate the estimation of the parameter(s).

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.

There are several methods that may be used to parameterize 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 recognises that selectivity is fundamentally a size-based process. In particular, we would like to recognise that age classes that have large overlap in their size distributions should have similar selectivity coefficients. A method implementing such an approach is invoked by fish flag 26. Purely age-based selectivity coefficients are retained if fish flag 26 = 0. An initial parameterization of length-based selectivity, in which the selectivity coefficients for age classes with similar mean lengths are made to be themselves similar, is activated by fish flag 26 = 1. Subsequently, the standard deviations of length at age were also used in the parameterization, 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 and 62.
- 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.

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-cummulative) 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. The penalties are specified for each fishery by way of fish flag 13. 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

[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 prceed with appropriate caution and skepticism.

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 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.

3.5 Natural Mortality

The instantaneous rate of natural mortality (M) consists of an average over age classes and age-specific deviations frrom 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 M/K with age flag 130 giving the target ratio ($\times 100$) and the penalty given by age flag 133 determining $1/(2\sigma^2)$

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

(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 4.5.15 for details. The deviations are estimated in log form and are stored in the *.par* file in the 2nd row of age_pars (see 4.1.4).

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, 2 coefficients for each non-zero element of the movement matrix (see "Movement matrix" - page 25) must be specified in the *.ini* file for each movement period (see "Movement" - page 30). 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 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.

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 recruitmment. The analysis is implemented after a converged fit is completed simply by executing an additonal 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 4.5.14).

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.

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:

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

- 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. 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.

- 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 $\mathcal{A}$ and $\mathcal{B}$ of the beta distribution, which are controlled by age flags 153 and 154 (see 5.4.2).

- 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

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

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_{MSY} (example in Figure 3.3) and F_t/F_{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[5].

[5]In which case one could argue that such ratios for the period prior to that over which the average

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 symetrical production curve (Schaefer model).

The biomass dynamics penalty is invoked by setting age flag $150 > 0$, with the value

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.

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 refrence points

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 4.5.11, 4.5.12, 4.5.13 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 likeilhood 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 orginal convergence point.

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 10^5 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.

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 therefrom. 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 anaysis 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 an example *.frq* file for yellowfin tuna interspersed with explanatory text. A copy of this or a similar example file can be obtained from <http://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    Year 1 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 sub-model 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
- unused flag
- flag to indicate whether the frequency data are length (0) or age (1) data
- number of recruitments per year assumed in the model
- 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 recrutment 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.

- *.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)

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.

Fishery data flags

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

```
# Data flags
  0 0 0 0 0 1 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 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 0 0
```

[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 26) and in structuring movement and recruitment in the *.ini* file (page 30.

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.

Periodic movement

Aspects of movement behaviour 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 behaviour is further structured by entries in the *.ini* file (see "movement map" – page 30).

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

[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.

- 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 $N_W = 0$, 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.

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 (or -1 if catch is missing)
- effort (or -1 if effort is missing)
- 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 ($N_{\mathrm{L}} = 0$), 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 ($N_{\mathrm{W}} = 0$), these numbers are omitted)

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 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. 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.

4.1.2 The *.ini* file

The *.ini* file supplies information mainly on biological characteristics of the fish. This file is used prior the first phase of an estimation procedure, and the information therein is incorporated into the initial *.par* file (see 4.1.4). 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/>.

Age classes

This fragment of the *.ini* file continues through comment lines to the first 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.

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.

Natural mortality

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).

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 26). Based on the movement weeks defined in the example *.frq* file above, the movement map shown here would define two seasons with one movement behaviour in weeks 41 and 5 of the year and another behaviour in weeks 17 and 29. If it were desired to have different behaviour 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 25). 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:

$$1\to2,\ \ 1\to3,\ \ 2\to1,\ \ 2\to3,\ \ 2\to4,\ \ 3\to1,\ \ 3\to2,\ \ 3\to5,\ \ 4\to2,\ \ 4\to5,\ \ 5\to3,\ \ 5\to4,$$

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

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
```

[3]Note that when there is only one region ($N_R = 1$), there should be one zero entered in the movement section and a single number one entered in the recruitment section of the *.ini* file.

These parameters may be given initial values that are different from the otherwise arbitrary values (zero) assigned by MULTIFAN-CL. The correspondence of model parameters to the various *age pars* is described in the *.par* file (see page 35).

Recruitment

The initial proportion of total recruitment ocurring 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 24).

```
# 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
```

Growth

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
# 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 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 ($\exp(0.69) = 2$).

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

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.

4.1.3 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 $\{N_{\mathrm{r}g}; g = 1 \ldots N_{\mathrm{G}}\}$ 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
#
#
# 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. 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

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.multifan-cl.org/>.

4.1.4 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 Section 4.5.

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

`The parest_flags` – 200 of them in one lengthy record – `vector parest_flags(1, 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 60 flags each – `matrix fish_flags(1, nfsh, 1, 60);`

`tag flags` (if tagging data included) – one record per tag group with 10 flags each – `imatrix tag_flag(1,num_tag_releases,1,10);`

`region control flags` – 10 for each region – `matrix region_flags(1, 10, 1, nregions)`

`percent maturity` – the maturity schedule as given in the *.ini* file. – `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

`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` – (1,10,1,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

Rows 9–10 are unused.

`# seasonal_catchability_pars` – (1,nfsh,1,12) (see 3.4.1).

`# age-class related parameters (age_pars)` – a 10×nage matrix containing by row:

1.
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).

Rows 6–10 are unused. These parameters can be pre-set in the *.ini* file ("Age parameters" – page 30).

`# region parameters` – (region_pars) – a 10×nregions matrix. The first row contains the average over time of the regional recruitment distribution. All other rows are as yet unused.

`# 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)

`# 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)

`# 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 #)

4.1.5 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 alloction. It supplies three numbers:

1. GRADSTACK_BUFFER_SIZE (default, 1000000)
2. CMPDIF_BUFFER_SIZE (default, 56000000)
3. ad_array_mbl_size (default, 25000000)

4.1.6 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 4.5.10 and 3.2). This file supplies the environmental data as a sequence of numbers corresponding to recruitment times in the model.

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 23). 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 behaviour 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 27), 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 26).

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 multplied further. However, if care is taken to keep the various events in phase, the number of nodes will be minimized. Thus the movment 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 preceeding 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]
```

[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.

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`.

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

On progressing from one fitting phase to another, it is usual practice to instruct MULTIFAN-CL 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 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`$_1$ $\cdots$ `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:

I_1		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 $n - 1000 = I_3$
$-n; (100000 \leq n < 100009)$	—	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, skiping calls to mfcl for all the *.par* files accumulated to that point.

[6]For running MULTIFAN-CL in a MS-Windows evnironment 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.

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.

```
#  ------------------------
#  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 satements 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 graph.frq (obs. and pred. LF 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 innerrupt 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.

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 befinning 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 recognised 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.

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 obselete. 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 likelilhood 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 unneccssary, but until this has been investigated it would be better to adhere to it.

Question on fish flag 31 – print selectivity by fleet, length and enable calculation of vector of "cut" selectivities and penalty on first diferences in that vector??

4.5.1 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
	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 55)
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*)
parest_flag(146)	1	scales the gradient for totpop
parest_flag(152)	1	re-scales the gradients to 1
parest_flag(189)	1	write graph.frq for java graphics
parest_flag(190)	1	write *ests.rep* and *plot.rep*
parest_flag(192)	n	keep n terms in the truncated newton minimization scheme, default (0)= 7
parest_flag(197)	1	create new input par file (obsolete)

4.5.2 Catch Estimation

Flag	Value	Use
age_flag(115)	n	SS2-type catch estimation
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

Flag	Value	Use
parest_flag(141)		LF likelihood function
	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
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

4.5.4 Growth Parameter Estimation

Flag	Value	Use
parest_flag(12)	1	estimate mean length of first age class
parest_flag(13)	1	estimate mean length of last age class
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	estimates the first n mean lengths as independent parameters
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(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 Catchability Estimation

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)
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(157)	1	use fish_pars(8) in the model.

4.5.6 Selectivity Estimation

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)	1 2	selectivity is non-decreasing with age 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,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 length-dependent selectivities

	2	uses length-dependent selectivities with sd of length-at-age accounted for (preferable)
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
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
fish_flags(i,61)	n	n is number of nodes for cubic spline selectivity curve
fish_flags(i,62)	n	add n nodes to cubic spline selectivity curve

4.5.7 Effort Deviation

Flag	Value	Use
age_flag(34)	1	turns on effort dev. estimation
fish_flags(i,4)	n	if $n > 1$, turns on effort dev. estimation
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)
age_flag(160)	n	penalty weight for fitting to effort in catch conditioned model

4.5.8 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 likelihood with added zeros for tagging data. fish_pars(5,fi) and fish_pars(6,fi) are the parameters
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, variance (a) for neg. bin. =5 (fish_pars(i,4)=0)
	1	variance (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

4.5.9 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 n	frequency of movement same as frequency of recruitment frequency of movement is one per n recruitment periods
age_flag(158)	n	

4.5.10 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)
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)
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
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(95)/10
	2	initial population is $N_{a+1} = N_a \exp(-\overline{Z}_{a,r})$ where $\overline{Z}_{a,r}$ is the total mortality averaged over the first age_flag(95) periods
age_flag(95)	n	n determines either 10× the multiplier x where age_flag(94)=1 or the number of initial periods for averaging where age_flag(94) = 2
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(177)	n	?????
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 overall, regional, seasonal, and regional×seasonal recruitment variation. The following table summarizes the use of this and 15 other parest flags that can be used to manipulate this scheme.

	Source of Variation			
	Overall	Regions	Seasons	Regions×Seasons
Main Period				
Start	pf-183	pf-204	pf-206	pf-208
End	pf-202	pf-210	pf-212	pf-214
Degree+1	pf-155	pf-216	pf-217	pf-218
Initial Period				
No. low order coefficients	pf-201	pf-205	pf-207	pf-209
Terminal Period				
No. low order coefficients	pf-203	pf-211	pf-213	pf-215

4.5.11 MSY Stuff

Flag	Value	Use
age_flag(145)	n	if $n > 0$, turns on stock-recruitment parameters, and n is the penalty weight.
age_flag(146)	1	estimate stock-recruitment parameters
age_flag(163)	n	$n = 0$ (default), steepness is active parameter of SRR; $n = 1$, steepness a derived parameter
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 1	yield computed in weight (t) 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(158)	n	used to enable region-specific yield analysis
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(163)	1	use steepness as active parameter in stock-recruitment curve
age_flag(165)	n	$n/100$ (default) or: target for F_{msy}/F or B/B_{msy}. Stock-recruitment parameters, age_flag(145), must be enabled.
age_flag(166)	n	$n/100$: penalty weight for F_{msy}/F or B/B_{msy}, default weight = 1000 ($n = 10^5$)
age_flag(167)	n	$n = 0$ (default) target F_{msy}/F, otherwise target B/B_{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.

4.5.12 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

4.5.13 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

4.5.14 Fishery Impact Analysis

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(193)	1	when set, the fishing effort at start is considered in calculating initial conditions for fishery impact analysis
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 recuitment 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 n	get average recruitment from the SRR (default) 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

4.5.15 Natural Mortality Estimation

Many flags governing natural mortality put constraints on deviations of natural mortality with age, M'_a, rather than directly on natural mortality itself, M_a. The M'_a are the logs

of the ratios of natural mortalities at age to mean natural mortality (see page 94).

Flag	Value	Use
age_flag(33)	1	estimate $\overline{M}$
age_flag(73)	1	estimate M_a; $\forall$ age class a
age_flag(109)	1 2	estimate M_a directly instead of as deviations from mean M_a is fixed as given by `age_pars(5)` in the *.ini* file (or *.par* file), see page 30 or 35
age_flag(77)	n	n is penalty weight on $\sum(M_{a-1} - 2M_a + M_{a+1})^2$, default = 25
age_flag(78)	n	n = penalty weight on $\sum(M_{a+1} - M_a)^2$, default = 5
age_flag(79)	n	n is penalty weight on $\sum(M_a - \overline{M})^2$, default = 10
age_flag(81)	n	n is number of terminal age classes with the same M
age_flag(82)	n	target average M $\equiv \overline{M}'$ is $n/100$
age_flag(80)	n	n is penalty weight on $\overline{M}'^2$, default = 10
age_flag(83)	n	n is minimum age to include in $\overline{M}'$, (default = 1)
age_flag(85)	n	n is maximum age to include in $\overline{M}'$, (default = nage)
age_flag(84)	n	n is penalty weight for target $\overline{M}'$, (default = 0)
age_flag(130)	n	$n > 0$ activates M/K target (natural mortality to vonBertalanfy 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

4.5.16 Projections

Flag	Value	Use

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 25).
age_flag(191)	n	For age_flag(190)> 0, end of recruitment average is n time periods before end of data series.
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.

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.

```
[lomu:try3]$ 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 plot files'                   :1
190 'control: write ests.rep'                     :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 the setting of a particular flag when the flag number is known. Here is an example of each:

```
[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
```

`ch.pflag, ch.aflag, ch.fflag`

These utilities facilitate changing a particular flag in a *.par* file. They are used as follows:

```
[ComputerName:try3]$ ch.pflag flag-no. value name.par

[ComputerName:try3]$ ch.aflag flag-no. value name.par

[ComputerName:try3]$ ch.fflag flag-no. value name.par
```

At this point `ch.fflag` changes the indicated flag for all fleets, but should be re-designed in the future to allow changing for only a designated sub-set of the fleets.

`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 ($< 10^{-4}$). Howvever 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
```

`targcheck`

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 $B_{\text{no_fishing}}$ (see age_flags 165 to 167, page 55, and age_flags 170 to 176, page 56). This utility is particularly helpful when makeing a series of runs to determine a likelihood profile for the target ratio (see page 21). Here is example output:

```
[ComputerName:BBmsy]$ targcheck BB*.par
                      obj           grad target realized penalty resid.pen        net.LL
 BB0.70.par 789880.701288 0.049723469936    0.7 0.701991 3000000 0.2419670 789880.943255
 BB0.80.par 789917.509387 0.098139215007    0.8 0.802923 2000000 0.2660870 789917.775474
 BB0.90.par 789918.466756 0.091428687996    0.9 0.905382 1000000 0.3555110 789918.822267
 BB1.00.par 789944.358042 0.064002285366    1.0 1.008600  500000 0.3664890 789944.724531
 BB1.10.par 789936.656236 0.069560409992    1.1 1.106470  500000 0.1720060 789936.828242
 BB1.20.par 789955.429664 0.065321237183    1.2 1.207790  200000 0.0836848 789955.513349
 BB1.30.par 789946.063038 0.076404383699    1.3 1.276840  200000 0.6463090 789946.709347
 BB1.40.par 789938.575630 0.271294687717    1.4 1.386460 1000000 0.9444360 789939.520066
 BB1.50.par 789923.538875 0.825184407769    1.5 1.489150 2000000 1.0538300 789924.592705
 BB1.60.par 789909.216574 0.338046332232    1.6 1.590760 3000000 1.0054100 789910.221984
 BB1.70.par 789876.286301 1.683403949769    1.7 1.692150 5000000 1.0714200 789877.357721
```

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 contol to the shell while the background process continues, and `nohup.out` allows the shell to exit and the user to log off whithout 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 71). 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 accessable 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 "`screenlog.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 72) and types

```
Ctrl-a Ctrl-a
```

which switches her to `screen 0` where she stops MULTIFAN-CL with `Ctrl-c q` (see page 71) 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.

Outside `screen` environment:	
`screen -ls`	list known screen sockets
`screen -S` *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*`]`	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
`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 intermidiate 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 Bayesean 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. We suggest setting this flag to 3, which invokes the full normal probability density, with the observed frequencies used in the variance term. The use of observed rather than predicted frequencies was suggested by Mark Maunder (IATTC) .... [get Mark to explain rationale]. 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 setting 3 only is available for weight frequency data.

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 funtion 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.

Another statistical problem that sometimes occurs in tagging experiments is an over-representatioon 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 tendancy 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 (page 91).

5.4 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 recognised 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.

5.4.1 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 $\sigma = 1/\sqrt{2w}$.

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.

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

5.4.2 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 $\mathcal{A}$ and $\mathcal{B}$ respectively). Table 5.2 shows settings of these age flags to achieve a selection of modes, means, and standard deviations.

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 145 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 93).

Chapter 6

Interpreting Results

6.1 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
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.

6.1.1 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 utlity, "`condensed.list`" was created to summarize results (see page 62). 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 inbetween. 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.

6.1.2 How to intervene

It is likely that during the course of a model fit, you will want to interupt the program for one reason or another. The procedure for interrupting differs between Linux and Windoze implementations of MULTIFAN-CL. In Linux, interuption is initiated by pressing Ctrl-c, in Windoze 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 Windoze 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. 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 mfcl and specifying the previous output *.par* file as the new input *.par* file.

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.

6.1.3 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.

6.2 Output Files

6.2.1 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 give above (Section 4.1.4).

6.2.2 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:

```
# MULTIFAN-CL Viewer
# 1.9
# Number of time periods
  160
# Year 1
  1962
# Number of regions
  5
# Number of age classes
  20
# Number of recruitments per year
  4
# Number of fisheries
  15
# Number of realizations per fishery
  160  ...   62
# Time of each realization by fishery (down)
  1962.125  ...  2001.875
     .              .
     .              .
  1970.125  ...  2001.875
```

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 is a ragged array with a row for each fishery and variable lengths of the rows as given by the elements of the preceeding vector.

From here on the *plot.rep* file gives mostly information based on parameter estimates, starting with basic population dynamic information tied up with predicting details of abundance and catch over time. To facilitate graphical juxtaposition of predicted and observed values, some observed catch data are reiterated here from the *.frq* file, and CPUEs are calculated therefrom and inserted here as well.

```
# Mean lengths at age
        vector[nage]

# SD of length at age
        vector[nage]

# Mean weights at age
        vector[nage]

# Natural mortality at age
        vector[nage]
```

```
# Selectivity by age class (across) and fishery (down)
        matrix[nage,nfish]

# Catchability by realization (across) by fishery (down)
        ragged array[nfish, nreal[nfish]]

# Catchability+effort dev.  by realization (across) by fishery (down)
        ragged array[nfish, nreal[nfish]]

# Fishing mortality by age class (across) and year (down)
        matrix[nyear, nage]

# Population number by age (across), year (down) and region
# Region 1
        matrix[nyear, nage]

# Region 2
        matrix[nyear, nage]
  .
  .

# Fishing mortality by age class (across), year (down) and region (block)
# Region 1
        matrix[nyear, nage]

# Region 2
        matrix[nyear, nage]
  .
  .

# Exploitable population by fishery (across) and year (down)
        matrix[nyear, nfish]

# Absolute biomass by region (across) and year (down)
# Total biomass
        matrix[nyear, nregion]

# Adult biomass
        matrix[nyear, nregion]

# Relative biomass by region (across) and year (down)
        matrix[nyear, nregion]

# Total biomass in absence of fishing
        matrix[nyear, nregion]

# Observed catch by fishery (down) and time (across)
        ragged array[nfish, nreal[nfish]]

# Predicted catch by fishery (down) and time (across)
        ragged array[nfish, nreal[nfish]]

# Observed CPUE by fishery (down) and time (across)
        ragged array[nfish, nreal[nfish]]

# Predicted CPUE by fishery (down) and time (across)
        ragged array[nfish, nreal[nfish]]
```

Depending on various flag settings the *plot.rep* file may also present material related to

various biological reference point analyses:

`# Yield analysis option:  0=none, 1=Bev&Holt, 2=Pella Tomlinson`
(0, 1, or 2)

`# Beverton-Holt stock-recruitment relationship report`
`# alpha = 5.967e+07 beta = 8.570e+04 steepness = 8.885e-01`
The above is not really a comment line. The three numbers are estimates of the respective parameters.

`# Observed spawning Biomass`
vector[nage-lag] – (lag from age flag 147)

`# Observed recruitment`
vector[nage-lag] – (lag from age flag 147)

`# Spawning Biomass`
vector[100]

`# Predicted recruitment`
vector[100]

`# Beverton-Holt yield analysis report`
`# MSY`
real number

`# F at MSY`
real number

`# Total biomass at MSY`
real number

`# Adult biomass at MSY`
real number

`# Effort multiplier`
vector[66]

`# Equilibrium yield`
vector[66]

`# Equilibrium adult biomass`
vector[66]

`# Equilibrium total biomass`
vector[66]

`# Adult biomass over adult biomass at MSY`
vector[nperiods]

`# Total biomass over total biomass at MSY`
vector[nperiods]

`# Aggregate F over F at MSY`
vector[nperiods]

`# Aggregate F`
vector[nperiods]

`# Yield per recruit report`
`# Effort multiplier`

```
        vector[501]

# Yield per recruit
        vector[501]

# Tag reporting rates
# Grouping indicator (0 = no grouping, >0 = grouping)
        vector[ntaggrps]

# Time series variation in reporting rates (0 = no, >0 = yes)
        an integer

# Reporting rates by fishery (no time series variation)
        vector[nfish]

# No.  of time periods associated with tag returns
        an integer = ntt

# Time periods associated with grouped tag returns
        vector[ntt]

# Observed tag returns by time period (across) by fishery groupings (down)
        matrix[ntaggrps,ntt]

# Predicted tag returns by time period (across) by fishery groupings (down)
        matrix[ntaggrps,ntt]

# Maximum time at liberty
        an integer = ntl

# Observed vs predicted tag returns by time at liberty
        matrix[ntl,2]

. # Movement analysis
# Region 1
        matrix[2*nage+1,nregion]

# Region 2
        matrix[2*nage+1,nregion]
  .
  .
```

6.2.3 The *length.fit* file

Predicted size distributions are output to a file called "length.fit" along with the observed size data to which the predicted distributions should be compared. The first five records give header information as follows. Material outside of parentheses appear in the file as shown. Material within parentheses describes value of a number to appear in the file.

```
0 0 20 0 0 0 0 0 0 0
(no. length bins, ``nbins'') (lower length of 1st bin) (width of bins)
(no. fisheries + 1)
(no. samples in fishery 1) ... (no. samples in last fishery) (no. fisheries)
(no. age classes, ``nages'')
```

The first record is as shown for all *length.fit files*. Its significance is known only unto MULTI-FAN-CL itself (and possibly D. Fournier). The record following it contains three numbers

defining the structure of the sample data. The third record gives one number, which is one plus the number of fisheries defined in the model. The following record gives the number of observed samples for each of the fisheries followed, redundantly enough, by the number of fisheries. The last header record gives the number of age classes.

The header information is followed by the set of observed and predicted sample data for fishery 1, identified by the first record of the set as shown:

```
# fishery 1
```

This is followed by a sample sets organized as follows:

```
(year from start) (month) (week)
(scaled mean length at age 1) . . . . (scaled mean length at age nages)
(sum of observed frequencies in next record)
(obs. frequency of bin 1)   . . . (obs. frequency of bin nbins)
(pred. frequency of bin 1)  . . . (pred. frequency of bin nbins)
[blank record]
 (pred. frequency, age 1, bin 1)     . . .  (pred. frequency, age 1, bin nbins)
             .                                                .
             .                                                .
             .                                                .
(pred. frequency, age nages, bin 1)  . . .  (pred. frequency, age nages, bin nbins)
[blank record]
```

Mean lengths are scaled and shifted for the convenience of a graphics program. More sample sets follow, the number for fishery 1 being given by the first value in the fourth header record above.

6.2.4 The *tag.rep* file

6.2.5 The *.var* file

6.2.6 The *.dep* file

6.2.7 The *.hes* file

6.2.8 The *catch.rep* file

6.2.9 The *ests.rep* file

6.3 Obtaining Graphical Results

6.3.1 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 Jave 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.

6.3.2 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 commercail 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.

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 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.

The following functions 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]]
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, SBtoS-Bmsy, 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

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 spaawning 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

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 incorprated into unix/linux scripts, *medley()* and *quickplot()*, giving very quick access to some of the more cogent MULTIFAN-CL output material.

Appendix A

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:

$$N'_{atr} = \begin{cases} \overline{R}e^{\varphi_t}\alpha_r\gamma_{tr} & ;\ a = 1\ ;\ 1 \leq t \leq T \\ N'_{a,1,r} & ;\ 1 < a < A\ ;\ t = 1 \\ N_{a-1,t-1,r}e^{-Z_{a-1,t-1,r}} & ;\ 1 < a < A\ ;\ 1 < t \leq T \\ N_{a-1,t-1,r}e^{-Z_{a-1,t-1,r}} + N_{a,t-1,r}e^{-Z_{a,t-1,r}} & ;\ a = A\ ;\ 1 < t \leq T \end{cases} \tag{A.1}$$

$$Z_{atr} = \sum_{f \in f_r} F_{atf} + M_a$$

where

- N'_{atr} is the number of age-class a fish at the beginning of time period t in region 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 r after movement has taken place,
- $\overline{R}$ 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, $\overline{R}$, with constraint that $\sum_t \varphi_t = 0$,
- α_r is average recruitment to region r as proportion of $\overline{R}$, $\sum_r \alpha_r = 1$,
- γ_{tr} is additional recruitment deviation at time t and region r, $\prod_t \gamma_{tr} = 1$ and $\sum_r \alpha_r\gamma_{tr} = 1$,
- 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 r,
- 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 4.5.10), the initial abundances $N'_{a,1,r}$ by region are either estimated parameters or they are functions of regional recruitments ($N'_{1,t,r}$) and either natural or total mortalities (M_a or Z_{atr}) averaged over time periods 2 to k:

$$
\begin{aligned}
N'_{a,1,r} &= \begin{cases} \overline{N'}_{1r} \exp\left[-\sum_{a'<a} \widetilde{Z}_{a'r}\right] \;; & 1 < a < A \\ \dfrac{\overline{N'}_{1r}}{1-\exp(-\widetilde{Z}_{ar})} \exp\left[-\sum_{a'<a} \widetilde{Z}_{a'r}\right] \;; & a = A \end{cases} \\
\overline{N'}_{1r} &= \frac{1}{(k-1)} \sum_{t=2}^{k} N'_{1,t,r} \\
\widetilde{Z}_{ar} &= \begin{cases} M_a & ;\ \text{age_flags}(94) = 1 \\ \dfrac{1}{(k-1)} \sum_{t=2}^{k} Z_{atr} & ;\ \text{age_flags}(94) = 2 \end{cases}
\end{aligned}
$$

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:

$$M_a = \overline{M} e^{\varepsilon_a}$$

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).

A.1.3 Movement

Movement is assumed to occur instantaneously at the beginning of each time period. Let $\mathbf{N'}_{at}$ and $\mathbf{N}_{at}$ be pre- and post-movement abundance vectors with elements N'_{atr} and N_{atr} for each region $r = 1, \ldots, R$. Defining ν_a^{rs} as the coefficient of movement from region r to region s at age a, abundance after movement is given by:

$$N_{atr} = N'_{atr} - \left(\sum_{s \neq r} \nu_a^{rs}\right) N_{atr} + \sum_{s \neq r} \nu_a^{sr} N_{ats} \tag{A.2}$$

or in matrix form:

$$\mathbf{B}_a \mathbf{N}_{at} = \mathbf{N'}_{at}$$

where

$$\mathbf{B}_a = \begin{bmatrix} 1 + \sum_{j \neq 1} \nu_a^{j1} & . & . & . & -\nu_a^{R1} \\ . & . & . & . & . \\ -\nu_a^{1i} & . & 1 + \sum_{j \neq i} \nu_a^{ji} & . & -\nu_a^{Ri} \\ . & . & . & . & . \\ -\nu_a^{1R} & . & . & . & 1 + \sum_{j \neq R} \nu_a^{jR} \end{bmatrix}$$

The movement transition is thus

$$\mathbf{N}_{at} = \mathbf{B}_a^{-1}\mathbf{N}'_{at}$$

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 transiton still allows some movement from one to the other within a time step.

Age dependency of the movement coefficients is specified by

$$\nu_a^{rs} = \phi_0^{rs} \exp(\phi_1^{rs}(\kappa_a)^{\phi_2^{rs}})$$

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}, and ϕ_2^{rs}.

A.1.4 Growth

The mean lengths at age μ_a are independent parameters below a specified age $a^\star$ and are constrained by von Bertalanffy growth from age $a^\star$ and older. Thus

$$\mu_a = \begin{cases} L_a \ ; & a < a^\star \\ L_1 + (L_A - L_1)\left[\frac{1-e^{-K(a-1)}}{1-e^{-K(A-1)}}\right] \ ; & a \geq a^\star \end{cases}$$

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

$$\sigma_a = \lambda_1 e^{-\lambda_2\left(1-2\frac{\mu_a - L_1}{L_A - L_1}\right)} \tag{A.3}$$

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_a. 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 ith length interval.

This is an adequate approximation to integrating over the range $(x_i - \omega/2, x_i + \omega/2)$ as long as $\sigma_a > \omega$ (Fournier, 1990). The expected proportion of length i fish in the sample is

$$\mathcal{Q}_{itf}^{\text{pred}} = \sum_{a=1}^{A} p_{atf}$$

where p_{atf} is the proportion of age-class a fish in the sample and is related to predicted catch by

$$p_{atf} = \frac{C_{atf}}{\sum_{a=1}^{A} C_{atf}}$$

A.1.6 Fishing Mortality

Catch in numbers of age-class a fish in time period t by fishery f is given by

$$C_{atf} = \frac{F_{atf}}{Z_{atr_f}} \left[1 - e^{-Z_{atr_f}}\right] N_{atr_f} \tag{A.4}$$

where r_f is the region in which fishery f occurs, where fishing mortality is parameterized as

$$F_{atf} = s_{af} q_{tf} B_{r_f}^{\beta} E_{tf}^{\zeta} e^{\varepsilon_{tf}} \tag{A.5}$$

and where

s_{af}	is the selectivity coefficient of fishery f for age-class a fish,
q_{tf}	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_{tf}	is the fishing effort of fishery f in time period t, and
ζ	is the parameter for effect of effort on catchability (default= 1),
ε_{tf}	represents transient deviations in effort.

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

$$x(l) = \frac{l - (\mu_1 - \sigma_1)}{(\mu_A + \sigma_A) - (\mu_1 - \sigma_1)}$$

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, $\mu_1 - \sigma_1$, to the average length of the oldest age class plus its standard deviation, $\mu_A + \sigma_A$. The inverse function is

$$l(x) = x(\mu_A + \sigma_A) + (1 - x)(\mu_1 - \sigma_1)$$

Age-specific selectivities are then given by

$$s_{af} = \int_0^1 \mathcal{N}(\mu_a, \sigma_a, l(x)) \mathcal{S}(\mathbf{s}'_f, x) dx$$

where $\mathcal{N}(\mu, \sigma, l)$ is the normal density function of l with mean μ and standard deviation σ, and $\mathcal{S}(\mathbf{s}'_f, x)$ is selectivity as a function of x on the interval [0—1]. $\mathcal{S}$ is defined by the parameters $\mathbf{s}'_f$, and is realized by a modified interpolation between those parameters which represent selectivity on $N^{\mathrm{L}}-1$ equal intervals along the interval $[(\mu_1-\sigma_1)$ — $(\mu_A+\sigma_A)]$. 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 4.5.6.

Catchability

Time series structure in catchability is allowed by:

$$q_{t+\delta t,f} = q_{tf} e^{\eta_{tf}} \tag{A.6}$$

where the $e^{\eta_{tf}}$ represent cumulative changes in catchability assumed to occur at regular intervals δt, 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:

$$q_{tf} = \begin{cases} q'_{tf} \exp(c^1_f \sin[2\pi(\frac{\upsilon_t}{4} - c^2_f)]) \ ; & \text{fish flag } 27 > 0 \\ q'_{tf} c^i_f \ ; & \text{fish flag } 47 > 0 \end{cases} \tag{A.7}$$

where

q'_{tf}	is the catchability before seasonal adjustment,
υ_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.1.7 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 a^{pool} after which the tagged fish enter a pooled group signified by cohort index $c^\star$. The number of tagged fish by cohort, time, and region prior to movement is given by

$$N^{\mathrm{T}'}_{ctr} = \begin{cases} 0 \ ; & t < \min(t^{\mathrm{rel}}_c) \\ N^{\mathrm{rel}}_c \ ; & t = t^{\mathrm{rel}}_c \ ; \ r = r^{\mathrm{rel}}_c \ ; \ c \neq c^\star \\ N^{\mathrm{T}}_{c,t-1,r} e^{-Z^{\mathrm{T}}_{c,t-1,r}} \ ; & t^{\mathrm{rel}}_c < t < t^{\mathrm{rel}}_c + a^{\mathrm{pool}} - a^{\mathrm{rel}}_c \ ; \ c \neq c^\star \\ N^{\mathrm{T}}_{c^\star,t-1,r} e^{-Z^{\mathrm{T}}_{c^\star,t-1,r}} + \sum_{b \in g_t} N^{\mathrm{T}}_{b,t-1,r} e^{-Z^{\mathrm{T}}_{b,t-1,r}} \ ; & t < \min(t^{\mathrm{rel}}_c) \ ; \ c = c^\star \end{cases}$$

$$\begin{aligned} Z^{\mathrm{T}}_{ctr} &= \sum_{f \in f_r} F^{\mathrm{T}}_{ctf} + M_{a(c,t)} \ ; \ c \neq c^\star \\ Z^{\mathrm{T}}_{c^\star tr} &= \mathrm{fn}_1(F_{atf} \mid f \in f_r \ ; \ a \geq a^{\mathrm{pool}}) + \mathrm{fn}_2(M_a \mid a \geq a^{\mathrm{pool}}) \end{aligned}$$

where

$$a(c,t) = a_c^{\text{rel}} + t - t_c^{\text{rel}}$$

relates historical time to the age of a particular cohort, and where

$$g_t = \{c \mid a_c^{\text{rel}} + t - t_c^{\text{rel}} = a^{\text{pool}}\}$$

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

$$F_{ctf}^{\text{T}} = \begin{cases} \text{fn}_3(R_{ctf}^{\text{Tobs}}, X_{tf}) \ ; & t_c^{\text{rel}} \leq t < t_c^{\text{rel}} + n^{\text{mix}} \\ F_{a(c,t),t,f} \ ; & t \geq t_c^{\text{rel}} + n^{\text{mix}} \end{cases}$$

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_{catf}^{Tobs} with a correction for the tag return rate X_{tf} for fishery f at time t.

Movement of tagged fish is entirely analogous to untagged fish as in Eqn. A.2. Thus

$$\begin{aligned} N_{ctr}^{\text{T}} &= N_{ctr}^{\text{T}'} - \left(\textstyle\sum_{s \neq r} \nu_a^{rs}\right) N_{ctr}^{\text{T}'} + \textstyle\sum_{s \neq r} \nu_a^{sr} N_{cts}^{\text{T}'} \\ N_{tr}^{\text{P}} &= N_{tr}^{\text{P}'} - \left(\textstyle\sum_{s \neq r} \nu_a^{rs}\right) N_{tr}^{\text{P}'} + \textstyle\sum_{s \neq r} \nu_a^{sr} N_{ts}^{\text{P}'} \end{aligned}$$

A.1.8 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.

$$F_{at} = \frac{\sum_r \left[N_{atr} \sum_{f \in f_r} F_{atf}\right]}{\sum_r N_{atr}} \tag{A.8}$$

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:

$$\overline{F}_a = \frac{1}{\tau_2 - \tau_1 + 1} \sum_{t=\tau_1}^{\tau_2} F_{at}$$

where F_{at} is given by equation A.8

A.1.9 Calculating total and spawning biomass

The time series of regional total and spawning biomass is given by

$$B_{tr}^{\text{t}} = \sum_{a=1}^{A} N_{atr} w_a \ ; \ B_{tr}^{\text{s}} = \sum_{a=1}^{A} N_{atr} w_a p_a^{\text{mat}}$$

where w_a is the average weight and p_a^{mat} 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

$$w_a = \int_0^\infty \mathcal{N}(\mu_a, \sigma_a, l) l^b dl$$

where μ_a is the mean length and σ_a is the standard deviation of length for age class a fish, and where $\mathcal{N}(\mu, \sigma, l)$ 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.1.10 Stock Recruitment Relationship

We assume a Beverton-Holt stock-recruitment relationship (SRR) between spawning biomass at time t aggregated over regions $B_t^{\text{s}} = \sum_r B_{tr}^{\text{s}}$ and recruitment in all regions $R_{t+t_{\text{lag}}}$ after a time lag t_{lag}. Thus

$$R_{t+t_{\text{lag}}} = \frac{\alpha B_t^{\text{s}}}{\beta + B_t^{\text{s}}} \tag{A.9}$$

where α is the maximum recruitment at large spawning stock and β is the spawning stock at which recruitment is reduced to half of α. The recruitment lag is fixed, but α and β are estimated by penalizing deviations (see page 92) between recruitment predicted by the SRR and recruitment as used in Eqn. A.1.

A.1.11 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, $\widetilde{B^{\text{t}}}(x)$, which can be calculated from equilibrium recruitment, $\widetilde{R}$, 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 $\phi_a(x)$

$$\phi_a(x) = \begin{cases} 1 \ ; & a = 1 \\ \exp\left(-\sum_{j=1}^{a-1}(M_j + x\overline{F}_j)\right) \ ; & a \in \{2 \ldots A-1\} \\ \exp\left(-\sum_{j=1}^{A-1}(M_j + x\overline{F}_j)\right)\left[\dfrac{1}{1-\exp\left(-(M_A + x\overline{F}_A)\right)}\right] \ ; & a = A \end{cases} \tag{A.10}$$

where x is a fishing mortality multiplier. Further define age-aggregate functions Φ^{t} and Φ^{s} for total and spawning biomass per recruit

$$\begin{aligned} \Phi^{\text{t}}(x) &= \textstyle\sum_{a=1}^{A} \phi_a(x) w_a \\ \Phi^{\text{s}}(x) &= \textstyle\sum_{a=1}^{A} \phi_a(x) w_a m_a \end{aligned} \tag{A.11}$$

where w_a is mean weight at age a, and m_a is the proportion spawning at age a. By aggregating across age cohorts, the functions, $\Phi^{\text{t}}(x)$ and $\Phi^{\text{s}}(x)$, 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 the SSR given in Eqn. A.9 holds at equilibrium we can say

$$\widetilde{R}(x) = \frac{\alpha \widetilde{B^{\mathrm{s}}}(x)}{\beta + \widetilde{B^{\mathrm{s}}}} \tag{A.12}$$

where the time indexing is dropped because equilibrium is assumed. A second relationship between $\widetilde{B^{\mathrm{s}}}(x)$ and $\widetilde{B^{\mathrm{s}}}$ is available from the spawning biomass per recruit, $\Phi^{\mathrm{s}}(x)$ (Eqn. A.11),

$$\widetilde{B^{\mathrm{s}}}(x) = \widetilde{R}(x)\Phi^{\mathrm{s}}(x) \tag{A.13}$$

Solving the above two equations and two unknowns for $\widetilde{R}(x)$ gives

$$\widetilde{R}(x) = \alpha - \frac{\beta}{\Phi^{\mathrm{s}}(x)} \tag{A.14}$$

Equilibrium abundance-at-age can now be calculated from the abundance per recruit function $\phi_a(x)$ (Eqn. A.10).

$$\widetilde{N}_a(x) = \widetilde{R}(x)\phi_a(x)$$

from which we get catch-at-age in numbers

$$\begin{aligned} \widetilde{C^{\mathrm{N}}}_a(x) &= \tfrac{\overline{F}_a}{M_a+\overline{F}_a}(\widetilde{N}_a(x) - \widetilde{N}_{a+1}(x)) \ ; \ a \in \{1 \ldots A-1\} \\ \widetilde{C^{\mathrm{N}}}_A(x) &= \tfrac{\overline{F}_A}{M_A+\overline{F}_A}\widetilde{N}_A(x) \end{aligned}$$

Yield in numbers and weight is then given by

$$\widetilde{Y^{\mathrm{N}}}(x) = \sum_{a=1}^{A} \widetilde{C^{\mathrm{N}}}_a(x) \ ; \ \widetilde{Y^{\mathrm{W}}}(x) = \sum_{a=1}^{A} \widetilde{C^{\mathrm{N}}}_a(x) w_a$$

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 $x_i \in \{0, 0.1, 0.2, 0.3, \ldots 50.0\}$. Note that the yield curve can go into negative territory at high levels of x even though it is biologically unreasonable.

A.1.12 Reference Points

If $x^\star$ is the value of x that maximizes the yield curve, then the maximum sustainable yield, MSY, in biomass or numbers is given by $\widetilde{Y^{\mathrm{W}}}(x^\star)$ or $\widetilde{Y^{\mathrm{N}}}(x^\star)$ respectively. $x^\star$ 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 $x^\star$ can be chosen simply as the element of the series $\{0, 0.1, 0.2, 0.3, \ldots 50.0\}$ corresponding to the maximum element, $\max(y_i)$, of the series of yields in the yield curve. However a continuous version of $x^\star$ can be estimated as a weighted average of the x_i:

$$x^\star = \frac{\sum_i x_i \exp(y_i/\mathcal{Y})^{\mathcal{X}}}{\sum_i \exp(y_i/\mathcal{Y})^{\mathcal{X}}}$$

where the constant, $(\mathcal{Y})$, 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, $\mathcal{X}$, 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,

$$\widetilde{B^{\mathrm{t}}}(x) = \widetilde{R}(x)\Phi^{\mathrm{t}}(x) \tag{A.15}$$

MULTIFAN-CL calculates the following additional stock assessment reference points:

- $B^{\mathrm{t}}_{\mathrm{MSY}} = \widetilde{B^{\mathrm{t}}}(x^\star)$ – equilibrium total biomass at MSY
- $B^{\mathrm{s}}_{\mathrm{MSY}} = \widetilde{B^{\mathrm{s}}}(x^\star)$ – equilibrium spawning biomass at MSY
- $F_{\mathrm{MSY}} = \widetilde{Y^{\mathrm{W}}}(x^\star)/\widetilde{B^{\mathrm{t}}}(x^\star)$ – 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:

$$\frac{B^{\mathrm{t}}_t}{B^{\mathrm{t}}_{\mathrm{MSY}}} \ ; \ \frac{B^{\mathrm{s}}_t}{B^{\mathrm{s}}_{\mathrm{MSY}}} \ ; \ \frac{F_t}{F_{\mathrm{MSY}}}$$

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.

Catch data

The contribution to negative log-likelihood from the discrepancy between observed and predicted catch is

$$\Theta^{\mathrm{C}} = p^{\mathrm{C}} \sum_t \sum_f \left[\log(1 + C^{\mathrm{obs}}_{tf}) - \log(1 + C^{\mathrm{pred}}_{tf})\right]^2 \ ; \ t, f \in \{u, g \mid C^{\mathrm{obs}}_{ug} \neq -1\}$$

where

$$C^{\mathrm{pred}}_{tf} = \begin{cases} \sum_{a=1}^{A} C_{atf} \ ; & f \in \{\text{fisheries with catch in numbers}\} \\ \sum_{a=1}^{A} C_{atf} w_a \ ; & f \in \{\text{fisheries with catch in weight}\} \end{cases}$$

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.

Size sample data

The contribution to the negative log-likelihood function from the discrepancy between observed and predicted proportions at size ($\mathcal{Q}^{\text{obs}}_{itf}$ and $\mathcal{Q}^{\text{pred}}_{itf}$) in the sample data is given by the following robustified formulation:

$$\begin{aligned}\Theta^{\text{L}} = \; & 0.5 \sum_i \sum_t \sum_f \log\left[2\pi(\xi_{itf} + 1/I)\right] + I \sum_t \sum_f \log(\tau_{tf}) \\ & - \sum_i \sum_t \sum_f \log\left[\exp\left(-\frac{(\mathcal{Q}^{\text{obs}}_{itf} - \mathcal{Q}^{\text{pred}}_{itf})^2}{2(\xi_{itf} + 1/I)\tau^2_{tf}}\right) + 0.001\right]\end{aligned}$$

where

$$\begin{aligned}\xi_{itf} &= \mathcal{Q}^{\text{obs}}_{itf}(1 - \mathcal{Q}^{\text{obs}}_{itf}) \\ \tau_{tf} &= P^{\text{L}} / \min(1000, S_{tf})\end{aligned}$$

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 $\min(1000, S_{tf})$ 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 P_{L} recognizies the 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 $\mathcal{Q}^{\text{obs}} = 0$, rendering the model less sensitive to occasional observations of low probablilty. The constant 0.001 provides additional robustness to the estimation.

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_{tf}. Predicted returns by tag chohort, age, time and fishery are given by

$$R^{\text{Tpred}}_{ctf} = \frac{F^{\text{T}}_{ctf} X_{tf}}{Z^{\text{T}}_{ctr}} \left[1 - e^{-Z^{\text{T}}_{ctr}}\right] N^{\text{T}}_{ctr} \;\; ; \;\; t \geq t^{\text{rel}}_c + n^{\text{mix}}$$

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:

$$R^{\text{Tpred}}_{ctg} = \sum_{f \in f_g} R^{\text{Tpred}}_{ctf} \;\; ; \;\; R^{\text{Tobs}}_{ctg} = \sum_{f \in f_g} R^{\text{Tobs}}_{ctf}$$

where f_g is the set of fishery indices for fisheies in group g.

There are five alternate formulations for the contribution of tagging data to the negative log-likelihood function. The choice is governed by parest flag 111 (section 4.5.8).

Least squares

$$\Theta^{\text{T}} = \sum_{ctg} \frac{(R^{\text{Tobs}}_{ctg} - R^{\text{Tpred}}_{ctg})^2}{R^{\text{Tpred}}_{ctg} + 0.01}$$

Robust least squares

$$\Theta^{\mathrm{T}} = \sum_{ctg} \log \left[\exp \left(-\frac{(R_{ctg}^{\mathrm{Tobs}} - R_{ctg}^{\mathrm{Tpred}})^2}{R_{ctg}^{\mathrm{Tpred}} + 0.01} \right) + 0.01 \right]$$

Poisson

For the poisson distribution the probability of a vector of observations $\boldsymbol{x}$ with corresponding expected values λ_i is

$$P(\boldsymbol{x}) = \prod_i \frac{\lambda_i^{x_i} e^{-\lambda_i}}{x_i!}$$

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:

$$\Theta^{\mathrm{T}} = \sum_{ctg} R_{ctg}^{\mathrm{Tpred}} - \sum_{ctg} R_{ctg}^{\mathrm{Tobs}} \log(R_{ctg}^{\mathrm{Tpred}} + 10^{-10}) + \sum_{ctg} \log(\Gamma(R_{ctg}^{\mathrm{Tobs}} + 1)$$

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

$$P(\boldsymbol{x}) = \prod_i \left[\frac{\Gamma(\lambda_i \beta_i + x_i)}{\Gamma(\lambda_i \beta_i)\Gamma(x_i + 1)} \left(\frac{\beta_i}{\beta_i + 1} \right)^{\lambda_i \beta_i} \left(\frac{1}{\beta_i + 1} \right)^{x_i} \right] \tag{A.16}$$

where λ is the expected value for each observation, and the variance for each observation is

$$\sigma^2(x_i) = \lambda_i \left(1 + \frac{1}{\beta_i} \right)$$

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:

$$\begin{aligned} \Theta^{\mathrm{T}} = \sum_{ctg} \quad & \big[\log \Gamma(\beta_{ctg} R_{ctg}^{\mathrm{Tpred}} + R_{ctg}^{\mathrm{Tobs}}) - \log \Gamma(\beta_{ctg} R_{ctg}^{\mathrm{Tpred}}) - \log \Gamma(R_{ctg}^{\mathrm{Tobs}} + 1) \\ & + \beta_{ctg} R_{ctg}^{\mathrm{Tpred}} \log(\beta_{ctg}) - (\beta_{ctg} R_{ctg}^{\mathrm{Tpred}} + R_{ctg}^{\mathrm{Tobs}}) \log(\beta_{ctg} + 1) \big] \end{aligned}$$

Negative bionomial with added zeroes

[– under construction –]

A.2.2 Priors and penalties

Many of the multitude of estimated parameters are constrained to some degree by prior distributions or penalties.

Effort deviations

The contribution of effort deviations, ε_{tf} in Eqn. A.5, to the objective function is formulated by the following prior:

$$\Theta^{\mathrm{E}} = -\sum_t \sum_f \log\left[\exp(-p_f^{\mathrm{E}}\sqrt{E_{tf}}\varepsilon_{tf}^2) + 0.05\exp(-\frac{1}{5}p_f^{\mathrm{E}}\sqrt{E_{tf}}\varepsilon_{tf}^2)\right]$$

where p_f^{E} 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 4.5.7) 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_f^{E} results in small deviation from the observed effort whereas a relatively small value of p_f^{E} allows the model greater flexibility to deviate from the observed effort values if this results in a better fit to the data.

Catchability deviations

Catchability deviations (η_{tf} in Eqn. A.6) are assumed to have normal prior distributions, their contribution to the objective function being

$$\Theta^{\mathrm{QD}} = \sum_f p_f^{\mathrm{QD}} \sum (\mu_f^{\mathrm{QD}} - \eta_{tf})^2$$

Seasonal catchability

$$\Theta^{\mathrm{QS}} = \sum_f p^{\mathrm{QS}} c_{1f}^2$$

Kalman filter deviations

[– under construction –]

Recruitment deviations

The contribution to the objective function from the spatially aggregated recruitment deviations φ_t is given by

$$\Theta^{\mathrm{R1}} = p^{\mathrm{R1}} \sum_t \varphi_t^2 + 0.1 \sum_{t=1}^{T-4} (\varphi_{t+4} - \varphi_t)^2$$

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 men zero. The contribution of this prior to the objective funciton is

$$\Theta^{\mathrm{R2}} = p^{\mathrm{R2}} \sum_t \sum_r [\log(\gamma_{tr})^2].$$

Stock-recruitment relationship

If estimation of stock-recruitment parameters is activated, then a spatially aggregated estimate of recuitment is given by Equ. A.9. A penalty for deviations of this recruitment estimate from spatially aggregated recruitment in Eqn. A.1, namely $\overline{R}\log(\varphi_t)$ gives the following contribution to the objective function:

$$\Theta^{\mathrm{R3}} = p^{\mathrm{R3}} \sum_{t=l+1}^{T} [\log(\overline{R}\varphi_t) - \log(R'_t)]^2)$$

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

$$\widetilde{R}(0) = \alpha - \frac{\beta}{\Phi^{\mathrm{s}}(0)} \tag{A.17}$$

and the corresponding unfished equibrium biomass is then

$$\widetilde{B^{\mathrm{s}}}(0) = \alpha\Phi^{\mathrm{s}}(0) - \beta \tag{A.18}$$

From Eqn. A.12, recruitment at 20% of unfished equilibrium spawning biomass, would be

$$R_{20\%} = \frac{\alpha\frac{1}{5}\widetilde{B^{\mathrm{s}}}(0)}{\beta + \frac{1}{5}\widetilde{B^{\mathrm{s}}}(0)} \tag{A.19}$$

which with Eqn. A.18 gives

$$R_{20\%} = \frac{\alpha(\alpha\Phi^{\mathrm{s}}(0) - \beta)}{4\beta + \alpha\Phi^{\mathrm{s}}(0)} \tag{A.20}$$

The required ratio for steepness as defined above, $R_{20\%}/\widetilde{R}(0)$, comes out to

$$S = \frac{\alpha\Phi^{\mathrm{s}}(0)}{4\beta + \alpha\Phi^{\mathrm{s}}(0)} \tag{A.21}$$

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

$$\Theta^{\mathrm{R4}} = -\log\left(\frac{(S-0.2)^{\mathcal{A}-1}(1.0-S)^{\mathcal{B}-1}}{(1.0-0.2)^{\mathcal{A}+\mathcal{B}-1}\int_0^1 x^{\mathcal{A}-1}(1-x)^{\mathcal{B}-1}dx}\right)$$

where $\mathcal{A}$ and $\mathcal{B}$ are shape parameters of the beta distribution and are controlled by age flag 153 and 154 (see 4.5.11 and 5.4.2). The mode, mean, and standard deviation of the beta prior are given by

$$\begin{aligned}
\mathrm{Mo}_{\mathrm{beta}} &= 0.2 + 0.8\tfrac{\mathcal{A}-1}{\mathcal{A}+\mathcal{B}-2} \\
\mu_{\mathrm{beta}} &= 0.2 + 0.8\tfrac{\mathcal{A}}{\mathcal{A}+\mathcal{B}} \\
\sigma_{\mathrm{beta}} &= 0.8\tfrac{1}{(\mathcal{A}+\mathcal{B})}\sqrt{\tfrac{\mathcal{A}\mathcal{B}}{\mathcal{A}+\mathcal{B}+1}}
\end{aligned}$$

Movement coefficients

The movement parameters, $\phi_0^{x,y}$ and $\phi_1^{x,y}$, have a normally distributed prior with mean zero. Their contribution to the objective function is

$$\Theta^{\mathrm{D}} = p^{\mathrm{D}} \left[\sum_{x,y} (\phi_0^{x,y})^2 + (\phi_1^{x,y})^2 \right].$$

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

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

$$\Theta^{\mathrm{X}} = \sum_f p_f^{\mathrm{X}} (X_f - \mu_f)^2$$

with fishery specific weighting factors, p_f^{X}, controlled by fish flags 35 and means, μ_f, controlled by fish flags 36 (see 4.5.8).

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

$$\Theta^{\mathrm{S}} = \sum_f \left[p_f^{\mathrm{S1}} \sum_{a=1}^{A-2} (s_{af} - 2s_{a+1,f} + s_{a+2,f})^2 + p_f^{\mathrm{S2}} \sum_{a=1}^{A-3} (s_{af} - 3s_{a+1,f} + 3s_{a+2,f} - s_{a+3,f})^2 \right]$$

where the fishery specific weighting factors, p_f^{S1} and p_f^{S2} are controlled by fish flags 41 and 42 (see 4.5.6).

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

$$\Theta^{\mathrm{M}} = p^{\mathrm{M1}} \sum_{a=1}^{A-2} (M'_a - 2M'_{a+1} + M'_{a+2})^2 + p^{\mathrm{M2}} \sum_{a=1}^{A-1} (M'_a - M'_{a+1})^2 + p^{\mathrm{M3}} \sum_{a=1}^{A} (M'_a - \overline{M'})^2 + p^{\mathrm{M4}} (\overline{M'})^2$$

where the M'_a are the log of the natural mortalities at age divided by the mean thereof, i.e.

$$M'_a = \log \left[\frac{M_a}{\frac{1}{A} \sum_{a=1}^{A} M_a} \right]$$

and where

$$\overline{M'} = \frac{1}{A} \sum_{a=1}^{A} M'_a$$

The weighting factors, p^{M1}, p^{M2}, p^{M3}, and p^{M4}, are controlled by age flags 77, 78, 79, and 80 respectively (see 4.5.15).

Fishing mortality at MSY

The fishing mortality multiplier at MSY, $x^\star$, is the ratio of fishing mortality at MSY to a nominal aggregate fishing mortality. $x^\star$ 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

$$\Theta^{\text{F}} = p^{\text{F}} \log(x^\star / x_{\text{targ}})^2$$

see 4.5.11.

A.2.3 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

$$L_x = \prod_i \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2\sigma^2}\left(\log(x_i) - \log(\hat{x}_i)\right)^2}$$

where $\hat{x}_i$ is some predictor of x_i. Ignoring the constant term $\frac{1}{2\pi\sigma^2}$ the negative log-likelihood becomes

$$-logL_x = \frac{1}{2\sigma^2} \sum_i \left(\log(x_i) - \log(\hat{x}_i)\right)^2$$

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

$$\sigma \approx \frac{1}{\sqrt{2p}} \tag{A.22}$$

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

$$\frac{e^{(\log(\mu)+\sigma)} - \mu}{\mu} = \frac{e^{\log(\mu)}e^{\sigma} - \mu}{\mu} = \frac{\mu e^{\sigma} - \mu}{\mu} = e^{\sigma} - 1$$

which by Taylor expansion is

$$[1 + \sigma + \frac{\sigma^2}{2!} + \dots\,] - 1 \approx \sigma$$

Similarly, an instance of x_i one standard deviation below its mean leads to

$$\frac{\mu - e^{(\log(\mu)-\sigma)}}{\mu} = 1 - e^{-\sigma} = 1 - [1 - \sigma + \frac{\sigma^2}{2!} - \dots\,] \approx \sigma$$

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

$$p \approx \frac{1}{2\sigma^2} \approx \frac{1}{2\text{CV}^2}$$

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 entrée 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. 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 ojective 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...

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.

[1] In certain circumstances not shown in the flow diagram, fcomp is called by other parts of the code than the minimizer.

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 Windoze 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 sprinkeled 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.

Borland debugger

[– under construction –]

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:

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-x3 and then A-xgdbC_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 main`C_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 -makepar`C_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.

[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.

Table A.1: 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*

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.

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 deciferable 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".

Table A.2: 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)
```

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)
```

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.

A.3.3 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 orgainizing 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.3.4 Important functions

ADMB functions

[– under construction –] Functions in the ADMB library called by MULTIFAN-CL code to be documented here.

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.3.5 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.

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 "∃" 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 7	
parest 8	
parest 9	
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 σ of size at age [4.5.4]
parest 16	size dependent σ [4.5.4]
parest 17	
parest 18	
parest 19	
parest 20	∃(sub-phase number during intial 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 [4.5.8]
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	
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 [4.5.8]
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	
parest 140	∃
parest 141	likelihood function selection [4.5.3]
parest 142	∃
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 [4.5.10]
parest 149	penalty for recruitment devs [4.5.10]
parest 150	∃
parest 151	∃
parest 152	rescale gradients to 1 [4.5.1]
parest 153	∃
parest 154	
parest 155	enable orthogonal polynomial for recruitment [4.5.10]
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.: years at start [4.5.10]
parest 184	
parest 185	
parest 186	
parest 187	
parest 188	
parest 189	write plot files [4.5.1]
parest 190	write ests.rep [4.5.1]
parest 191	∃
parest 192	no. terms in newton minimization [4.5.1]
parest 193	
parest 194	
parest 195	
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.: degree at start [4.5.10]
parest 202	orthog. poly.: years at end [4.5.10]
parest 203	orthog. poly.: degree at end [4.5.10]
parest 204	orthog. poly.: years at start, regions [4.5.10]
parest 205	orthog. poly.: degree at start, regions [4.5.10]
parest 206	orthog. poly.: years at start, seasons [4.5.10]
parest 207	orthog. poly.: degreeat start, seasons [4.5.10]
parest 208	orthog. poly.: years at start, regXsea [4.5.10]
parest 209	orthog. poly.: degree at start, regXsea [4.5.10]
parest 210	orthog. poly.: years at end, regions [4.5.10]
parest 211	orthog. poly.: degree at end, regions [4.5.10]
parest 212	orthog. poly.: years at end, seasons [4.5.10]
parest 213	orthog. poly.: degree at end, seasons [4.5.10]
parest 214	orthog. poly.: years at end, regXsea [4.5.10]
parest 215	orthog. poly.: degree at end, regXsea [4.5.10]
parest 216	orthog. poly.: degree, regions [4.5.10]
parest 217	orthog. poly.: degree, seasons [4.5.10]
parest 218	orthog. poly.: degree, regionsXseasons [4.5.10]
parest 219	
parest 220	
parest 221	
parest 222	
parest 223	
parest 224	
parest 225	
parest 226	
parest 227	
parest 228	
parest 229	
parest 230	
parest 231	
parest 232	
parest 233	
parest 234	
parest 235	
parest 236	
parest 237	
parest 238	
parest 239	
parest 240	
parest 241	
parest 242	
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 260	
parest 271	
parest 272	
parest 273	
parest 274	
parest 275	
parest 276	
parest 277	
parest 278	
parest 279	
parest 270	
parest 281	
parest 282	
parest 283	
parest 284	
parest 285	
parest 286	
parest 287	
parest 288	
parest 289	
parest 280	
parest 291	
parest 292	
parest 293	
parest 294	
parest 295	
parest 296	
parest 297	
parest 298	
parest 299	
parest 290	
parest 291	
parest 292	
parest 293	
parest 294	
parest 295	
parest 296	
parest 297	
parest 298	
parest 299	
parest 300	

B.2 age flags

Flag	Action/Meaning/Status
age 1	
age 2	
age 3	
age 4	
age 5	
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	
age 21	
age 22	
age 23	
age 24	
age 25	
age 26	
age 27	∃
age 28	
age 29	
age 30	est tot recruit [4.5.10]
age 31	est totpop [4.5.10]
age 32	totpop fixed or not [4.5.1, 4.5.10]
age 33	est. M [4.5.15]
age 34	est. effort devs [4.5.7]
age 35	∃
age 36	∃
age 37	avg. F target [4.5.12]
age 38	∃
age 39	penalty on F target [4.5.12]
age 40	∃
age 41	∃
age 42	∃
age 43	no. terminal years for F target [4.5.12]
age 44	annual F target [4.5.12]
age 45	∃
age 46	∃
age 47	∃
age 48	∃
age 49	∃
age 50	∃
age 51	∃
age 52	∃
age 53	movement frequency [4.5.9]
age 54	∃
age 55	∃
age 56	∃
age 57	month doubling [4.5.5 4.5.10]
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 [4.5.9]
age 69	activate movement pars [4.5.9]
age 70	activate time series of reg recruitment [4.5.10]
age 71	est time series of reg recruitment [4.5.10]
age 72	recruitment and environment [4.5.10]
age 73	est age-dep M [4.5.15]
age 74	∃
age 75	∃
age 76	∃
age 77	penalty on 2nd derivative of M_a [4.5.15]
age 78	penalty on 1st derivative of M_a [4.5.15]
age 79	penalty on $\sum(M_a - \overline{M})^2$ [4.5.15]
age 80	penalty on $\overline{M}'^2$ [4.5.15]
age 81	terminal ages with the same M [4.5.15]
age 82	target $\overline{M}'$ [4.5.15]
age 83	min age in $\overline{M}'$ [4.5.15]
age 84	penalty on target $\overline{M}'$ [4.5.15]
age 85	max age in $\overline{M}'$ [4.5.15]
age 86	∃
age 87	∃
age 88	activate age-dep movement pars [4.5.9]
age 89	est. age-dep movement pars [4.5.9]
age 90	activate non-linear age-dep movement pars [4.5.9]
age 91	est. non-linear age-dep movement pars [4.5.9]
age 92	catch errors [4.5.1]
age 93	∃
age 94	starting pop strategy [4.5.10]
age 95	modifier for age_flags(94) [4.5.10]
age 96	tag pooling [4.5.8]
age 97	target biomass ratio [4.5.13]
age 98	penalty on biomass ratio [4.5.13]
age 99	no. inital and final yeas for calc biomass ratio [4.5.13]
age 100	neg. binomial for tag likelihood [4.5.8]
age 101	activate *.env* file [4.5.10]
age 102	est environmental correlation par [4.5.10]
age 103	in code but commented out
age 104	enable implicit q devs [4.5.5]
age 105	∃
age 106	∃
age 107	penalty on overall exploitation [4.5.12]
age 108	overall exploitation target [4.5.12]
age 109	whether $\log(M)$ at age by deviations from mean or direct [4.5.15]
age 110	penalty on region_rec_diffs [4.5.10]
age 111	∃
age 112	numbers or biomass for MSY calcs [4.5.11]
age 113	scaling init. pop and rectuitment [4.5.10]
age 114	
age 115	SS2-type catch estimation
age 116	
age 117	
age 118	
age 119	
age 120	?????????
age 121	?????????
age 122	?????????
age 123	?????????
age 124	?????????
age 125	
age 126	
age 127	
age 128	
age 129	
age 130	M/K target [4.5.15]
age 131	age1 for M/K target [4.5.15]
age 132	age2 for M/K target [4.5.15]
age 133	penalty on M/K target [4.5.15]
age 134	
age 135	
age 136	
age 137	
age 138	
age 139	
age 140	enable region-specific yield anal. and set Fmult resolution [4.5.11]
age 141	default no. Fmult steps in region-specific yiel anal.
age 142	
age 143	∃
age 144	common wt for catch L to 10000 [4.5.2]
age 145	penalty on stock-recruit pars [4.5.11]
age 146	activate stock-recruit [4.5.11]
age 147	lag betw. spawning and recruitment [4.5.11]
age 148	years from last year for avg. F [4.5.11]
age 149	yield in wt or numbers [4.5.11]
age 150	penalty for biomass dynamics est [4.5.11]
age 151	activate r in Pella-Tomlilnson [4.5.11]
age 152	activate K in Pella-Tomlilnson [4.5.11]
age 153	a of beta prior in SRR [4.5.11]
age 154	b of beta prior in SRR [4.5.11]
age 155	years from last year to omit from avg F [4.5.11]
age 156	use fish_pars 7 in model [4.5.5]
age 157	use fish_pars 8 in model [4.5.5]
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 [4.5.7]
age 161	
age 162	
age 163	select alternate parameters for SRR
age 164	
age 165	target for Fmsy/F or B/Bmsy [4.5.11]
age 166	penalty weight for Fmsy/F or B/Bmsy [4.5.11]
age 167	target Fmsy/F or B/Bmsy [4.5.11]
age 168	exponent in weighted average for Fmsy/F [4.5.11]
age 169	divisor in weighted average for Fmsy/F [4.5.11]
age 170	enable a biomass depletion target [4.5.14]
age 171	unfished calculations by estimated recruitment or SRR [4.5.14]
age 172	total or adult biomass depletion [4.5.14]
age 173	first time period for reckoning depletion [4.5.14]
age 174	last time period for reckoning depletion [4.5.14]
age 175	100X target depletion level [4.5.14]
age 176	penalty on depletion target [4.5.14]
age 177	??????
age 178	
age 179	
age 180	
age 181	
age 182	
age 183	
age 184	
age 185	∃
age 186	
age 187	
age 188	
age 189	
age 190	
age 191	
age 192	∃
age 193	fishery impact analysis [4.5.14]
age 194	include effort devs in yield analysis [4.5.11]
age 195	
age 196	
age 197	
age 198	

age 199
age 200

B.3 fish flags

Flag	Action/Meaning/Status
fish 1	est. avg. q [4.5.5]
fish 2	∃
fish 3	1st age of common selectivity [4.5.6]
fish 4	est. effort devs [4.5.7]
fish 5	∃
fish 6	∃
fish 7	∃
fish 8	∃
fish 9	∃
fish 10	time series in q [4.5.5]
fish 11	selectivity bias, 1st age [4.5.4]
fish 12	∃
fish 13	penalty for effort devs [4.5.7]
fish 14	limit on F per fishing incident [4.5.12]
fish 15	penalty for q-devs [4.5.5]
fish 16	selectivity shape [4.5.6]
fish 17	∃
fish 18	∃
fish 19	no. ages for sel_dev_coffs [4.5.6]
fish 20	!
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 [4.5.5]
fish 24	grouping for selectivity [4.5.6]
fish 25	∃
fish 26	length-dependent selectivities [4.5.6]
fish 27	sinusoidal seasonal q [4.5.5]
fish 28	grouping for seasonal q [4.5.5]
fish 29	grouping for common catchability deviations[4.5.5]
fish 30	∃
fish 31	∃
fish 32	grouping for tag recaptures [4.5.8]
fish 33	est. tag reporting rate [4.5.8]
fish 34	grouping for tag reporting rate [4.5.8]
fish 35	penalty for tag reporting rate prior [4.5.8]
fish 36	tag reporting rate prior [4.5.8]
fish 37	est. time series of tag reporting rate [4.5.8]
fish 38	σ_2 for Kalman filt. effort devs [4.5.7]
fish 39	σ_2 for Kalman filt. q devs [4.5.5]
fish 40	∃
fish 41	penalty weight for second difference selectivity smoothing [4.5.6]
fish 42	penalty weight for third difference selectivity smoothing [4.5.6]
fish 43	σ^2 for neg. bin., see parest_flags(111) [4.5.8]
fish 44	grouping for neg. bin. [4.5.8]
fish 45	fishery specific catch wts [4.5.2]
fish 46	est. of mixture pars in tag likelihood [4.5.8]
fish 47	arbitrary seasonal q [4.5.5]
fish 48	selectivity est. (obsolete??) [4.5.6]
fish 49	length sample size [4.5.3]
fish 50	weight sample size [4.5.3]
fish 51	effect of effort on q [4.5.5]
fish 52	grouping for fish_flag 51 [4.5.5]
fish 53	effect of abund. on q [4.5.5]
fish 54	grouping for fish_flag 53 [4.5.5]
fish 55	turn off fisheries for impact analysis [4.5.14]
fish 56	penalty to make selectivity a non decreasing function of age [4.5.6]
fish 57	functional form for selectivity [4.5.6]
fish 58	
fish 59	
fish 60	grouping for common initial catchability[4.5.5]
fish 61	number of nodes for cubic spline selectivity [4.5.6]
fish 62	number of nodes for cubic spline selectivity [4.5.6]
fish 63	
fish 64	
fish 65	∃
fish 66	variable q, implicit method
fish 67	
fish 68	
fish 69	
fish 70	
fish 71	
fish 72	
fish 73	
fish 74	
fish 75	
fish 76	
fish 77	
fish 78	
fish 79	
fish 80	
fish 81	
fish 82	
fish 83	
fish 84	
fish 85	
fish 86	
fish 87	
fish 88	
fish 89	
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
reg 1	recruitment distribution by reg [4.5.10]
reg 2	
reg 3	
reg 4	
reg 5	
reg 6	
reg 7	
reg 8	
reg 9	
reg 10	

B.5 tag flags

Flag	Action/Meaning/Status
tag 1	set number of tag mixing periods [4.5.8]
tag 2	
tag 3	
tag 4	
tag 5	
tag 6	
tag 7	
tag 8	
tag 9	
tag 10	

Index

abundance per recruit, 87
attach/detach, 62

biological reference points, 17
biomass, 86
biomass per recruit, 87

coefficient of variation, 95

debugging, 97

equilibrium recruitment, 88

fishery impact, 17
flow diagram, 113
Fmsy, 95

graphical output, 77

interrupting a run, 71

likelihood function, 66, 89
    catch data, 89
    sample data, 90
    tag data, 67, 90
        negative bionomial, 91
        poisson, 91
likelihood profile, 21

MSY, 88

natural mortality, 82
negative bionomial, 91
nodes, 37

objective function, 89
output
    files, 72
    screen, 70

poisson, 91
population dynamics, 81
prior distributions
    beta, 18, 69, 93
    normal, 68
projection period, 3, 26, 28

reference points, 88

screen utility, 62
    command summary, 65
selectivity, 84
standard output, 70
steepness, 18, 93
stock-recruitment relationship, 18, 87, 93

tag data, 32, 90
tag dynamics, 15, 85
targets, 21

weighting factors, 95

yield curve, 87