Modifiable Multistock Model (M3)

Users guide

V1.0 (beta)

9th September 2016

**Tom Carruthers**, UBC t.carruthers@oceans.ubc.ca

**ICCAT GBYP** gbyp@iccat.int

**ICCAT Core Modelling Group**

Laurie Kell, Antonio Di Natale, Doug Butterworth, Harritz Arrizabalaga, Yukio Takeuchi, Sylvain Bonhommeau, Toshi Kitakado, Clay Porch, David Die, Miguel Santos, Paul de Bruyn, Polina Levontin, Richard Hillary.

**Release notes**

* Age based-movement now estimated (v0.18).
* Fishery spool-up to fishing rates in the initial years has been added (v0.18).
* Fishery selectivity reparameterized (v1.0)
* Historical recruitment deviations are now estimated (v1.0)

M3 is currently in its first iternation that is fit for purpose (beta). It has been fitted to a the data and assumptions of the ICCAT trial specification document for ABT. Any comments, bugs or otherwise can be forwarded to t.carruthers@fisheries.ubc.ca.

M3 was compiled using ADMB 11.5 (64 bit) under Windows 10 using mingw64

Table of Contents

[2 Introduction 3](#_Toc448927853)

[3 Model description 3](#_Toc448927854)

[4 Formatting data for input to M3 4](#_Toc448927855)

[5 Running M3 13](#_Toc448927856)

[6 Outputting data and model estimates from M3 13](#_Toc448927857)

[6.1 Reading the report file 13](#_Toc448927858)

[6.2 Reading the parameter and variance-covariance files 18](#_Toc448927859)

[7 Future additions 19](#_Toc448927860)

[8 Appendix A. Model equations 19](#_Toc448927861)

[8.1 Estimated parameters 19](#_Toc448927862)

[8.2 Transition equations 19](#_Toc448927863)

[8.3 Initializing the model 22](#_Toc448927864)

[8.4 Predicting data 23](#_Toc448927865)

[8.5 Likelihood functions, priors and the global objective function 24](#_Toc448927866)

[9 Appendix B. Model code 28](#_Toc448927867)

# Introduction

M3 is a spatial, multistock, subyear, statistical catch-at-length stock assessment model. The model predicts the distribution and movement of individuals among large discrete marine areas (a bulk transfer model, e.g. Taylor et al. 2011).

M3 was designed to be used as an operating model that could be fitted to various data to predict spatial stock structure of a multi-stock fishery (specifically Atlantic bluefin tuna). As such M3 includes several simplifications over a conventional age-structured stock assessment in order to reduce the number of calculations and ensure that the estimation problem is well defined. The two most important simplifications relate to the use of conventional tagging data and the estimation of fishing mortality rates for each fleet in spatio-temporal strata:

(1) M3 uses conventional tagging data only qualitatively to identify other possible migrations that are not observed from electronic tagging (pop-off satellite archival, archival). M3 does not include computationally intensive conditional probability calculations for all of the conventional (spaghetti, floy) tagging data that are considered to be compromised due to highly uncertain and variable reporting rates over time and among fishing fleet types.

(2) The M3 model does not attempt to estimate a fishing mortality rate for each fleet in each spatio-temporal strata (for example the Japanese longline fleet in January 1962 in area 2). Instead partial fishing mortality rates (partial Fs) are calculated by dividing observed catches by a ‘master’ relative abundance index for each spatio-temporal strata. The master relative abundance index is constructed by generalized linear modelling and time-area imputation (e.g. Carruthers et al. 2011) prior to the M3 run. The partial F data (by fleet, time and area) are inputs to the M3 model which then predicts fishing mortality rate by multiplying these partial Fs by a single catchability coefficient by fleet. This greatly reduces the number of estimated parameters (e.g. 4 parameters instead of 8000 for a 50 year, 4 subyear, 10 area model with 4 fleets).

Previous multi-stock models for Atlantic bluefin tuna have made use of catch-at-age composition data. However these data are very sparse and derived by approaches such as cohort slicing and therefore provide inference of fishing mortality rate that may be both biased and imprecise. M3 sidesteps this problem by fitting to length composition data. The additional requirement of this approach is an inverse age-length key (conditional probability of an individual being in a length class l given it is of age class a).

Currently the model estimates age-group specific movement and requires stock of origin data and electronic tagging data (e.g. otolith microchemistry) by age class, space, subyear and year. However real data have not yet been provided that are age-class specific so the only application of M3 thus far was version 0.16 which did not estimate age-group specific movement and was fitted to age-aggregated stock of origin and electronic tagging data.

# Model description

M3 is a fisheries assessment model based on standard equations for age-structured population dynamics (e.g. Quinn and Deriso 1999, Chapter 8) which is common to stock assessment models such as Stock Synthesis 3 (Methot and Wetzel 2013), CASAL (Bull et al. 2012), Multifan-CL (Fournier et al. 1998) and iSCAM (Martell 2015). Similar to these assessment packages, M3 is developed using ADMB (Fournier et al. 2012) for its rapid and robust non-linear estimation performance for problems with relatively large numbers of parameters (i.e. more than 100 parameters).

The conventional age-structured accounting is somewhat more complex for a spatial multi-state model because transitions among states among years (i.e. movement probabilities) must be either specified or estimated. In the case of the M3 model this is further complicated by the additional stratification by subyear that must account for variable duration of subyears and distinguish between subyears in which there is recruitment. A full account of model equations is included in Appendix A (the actual model code, the ADMB template file, is included in Appendix B).

# Formatting data for input to M3

Similarly to other fishery assessment models (e.g. Stock Synthesis, Wetzel and Methot 2011. ISCAM Martell 2016) M3 is implemented in AD Model Builder (Fournier et al. xxx). The data must be formatted into an M3.dat file in order to fit the model. Below is a description of each field of this data file.

1. ny: an integer value, number of model years

55

1. ns: an integer value, number of model subyears (e.g. 4 for a seasonal model)

4

1. np: an integer value, number of model stocks / populations (e.g. 2 for an east-west Atlantic bluefin)

4

1. na: an integer value, number of age classes

35

1. nr: an integer value, number of spatial areas

8

1. nf: an integer value, number of fleets (fishing activities of similar size selectivity)

5

1. nl: an integer value, number of length classes

38

1. nRPTl: an integer value, maximum number of time steps (subyears) that a PSAT can be recaptured (limits unnecessary calculation)

3

1. RPind: a matrix of integers ns rows by nRPTI columns. This matrix is the correct subyear recapture index after col time steps.

1 2 3

2 3 4

3 4 1

4 1 2

1. sdur: a vector of fractions (sum to 1) ns long that is the relative duration of each subyear.

0.25 0.25 0.25 0.25

1. nZeq: an integer value, the number years at the start of the model to calculate mean fishing mortality rates for the calculation of equilibrium biomass prior to the initial year.

5

1. nydist: an integer value, the number annual iterations used to determine stable spatial distribution d of the stock given an estimated movement matrix M (ie to approximate d=Md).

50

1. nyeq: an integer value, the number ‘spool up’ years prior to ny model years over which the stock was subject to equilibrium F.

25

1. ml: a vector of positive real numbers nl long that are the mean length of each length class

25 35 45 55 65 75 85 95 105 115 125 135 145 155 165 175 185 195 205 215 225 235 245 255 265 275 285 295 305 315 325 335 345 355 365 375 385 395

1. RDblock: an integer vector ny long representing the recruitment deviation blocking (e.g. 1,1,1,1,1,2,2,2,2,2,… represents the estimation of two recruitment deviations for the first and second five year block respectively).

1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5 6 6 6 6 6 7 7 7 7 7 8 8 8 8 8 9 9 9 9 9 10 10 10 10 10 11 11 11 11 11

1. nRD: the number of estimated recruitment deviations (matches max(RDblock)).

11

1. iALK: a 4D array (np, ny, na, nl) of fractions (sum to 1 over length classes) of the conditional probability of length class l given age class a.

3.322514e-07 0.001265241 0.1641068 0.7249812 0.1090873 0.0005590728 9.759096e-08 5.802269e-13 1.174987e-19 8.104296e-28 1.903902e-37 1.523424e-48 4.151871e-61 3.85402e-75 1.218515e-90 1.312185e-107 4.812893e-126 6.012627e-146 2.558405e-167 3.707843e-190 1.830292e-214 3.077275e-240 1.762217e-267 3.437156e-296 0 0 0 0 0 0 0 0 0 0 0 0 0 0

7.322951e-11 2.060831e-07 0.0001054372 0.009807099 0.1658373 0.5098227 0.2849386 0.02895199 0.0005348115 1.796049e-06 1.096556e-09 1.217134e-13 2.456074e-18 9.010304e-24 6.009421e-30 7.286533e-37 1.606218e-44 6.436992e-53 4.689825e-62 6.211916e-72 1.495855e-82 6.548604e-94 5.211983e-106 7.541408e-119 1.983796e-132 9.48717e-147 8.248435e-162 1.303772e-177 3.746508e-194 1.957253e-211 1.858927e-229 3.209763e-248 1.007578e-267 5.750154e-288 5.965892e-309 0 0 0

4.572342e-13 5.683686e-10 2.434965e-07 3.595232e-05 0.0018295 0.03208554 0.1939356 0.4039962 0.290047 0.071768 0.006120186 0.0001798746 1.821992e-06 6.360546e-09 7.652684e-12 3.173252e-15 4.534889e-19 2.233573e-23 3.791448e-28 2.218101e-33 4.47228e-39 3.107764e-45 7.442837e-52 6.14328e-59 1.747566e-66 1.713319e-74 5.789141e-83 6.741576e-92 2.705703e-101 3.742573e-111 1.784152e-121 2.93133e-132 1.659851e-143 3.239251e-155 2.178671e-167 5.050218e-180 4.034595e-193 1.110865e-206

1.618478e-14 9.843834e-12 2.82786e-09 3.836961e-07 2.458969e-05 0.0007443113 0.01064124 0.07185647 0.2291794 0.34524 0.2456421 0.08255065 0.0131031 0.0009823447 3.478481e-05 5.8177e-07 4.595672e-09 1.714679e-11 3.021708e-14 2.515118e-17 9.887809e-21 1.836022e-24 1.610244e-28 6.67024e-33 1.30505e-37 1.206005e-42 5.263892e-48 1.085178e-53 1.056649e-59 4.859556e-66 1.055598e-72 1.08302e-79 5.248199e-87 1.201214e-94 1.298572e-102 6.630528e-111 1.599065e-119 1.821462e-128

… (np x ny x na rows)

1. lwa: a vector np long, the a parameter of the length-weight relationship W=aLb

2.95e-05 1.96e-05

1. lwb: a vector np long, the a parameter of the length-weight relationship W=aLb

2.899 3.009

1. len\_age: an array (np x na x ny) of positive real numbers representing the length at age

54.39584 55.07681 76.58744 79.40297 96.88932 101.5688 115.4624 121.7662 132.4539 140.1699 147.9985 156.9392 162.2194 172.2193 175.2294 186.1425 187.1315 198.8292 198.02 210.3892 207.9814 220.9226 217.0945 230.5206 225.4316 239.2662 233.0587 247.2351 240.0364 254.4964 246.4199 261.1128 252.2598 267.1416 257.6024 272.635 262.4901 277.6406 266.9615 282.2017 271.0522 286.3577 274.7945 290.1446

…

1. wt\_age: an array (np x na x ny) of positive real numbers representing the weight at age

54.39584 55.07681 76.58744 79.40297 96.88932 101.5688 115.4624 121.7662 132.4539 140.1699 147.9985 156.9392 162.2194 172.2193 175.2294 186.1425 187.1315 198.8292 198.02 210.3892 207.9814 220.9226 217.0945 230.5206 225.4316 239.2662 233.0587 247.2351 240.0364 254.4964 246.4199 261.1128 252.2598 267.1416 257.6024 272.635 262.4901 277.6406 266.9615 282.2017 271.0522 286.3577 274.7945 290.1446

…

1. Fec: a matrix (np x na) of positive real numbers representing the fecundity at age (sometimes SSB at age)

0.03942355 0.3598737 2.247868 9.799885 27.25873 50.05359 72.14995 93.02628 113.5565 134.1323 154.7556 175.2791 195.5246 215.3283 234.5549 253.0986 270.8815 287.8497 303.9696 319.2248 333.6126 347.1415 359.8287 371.6982 382.779 393.1037 402.7076 411.6271 419.8996 427.5623 434.6523 441.2055 447.2571 452.8408 457.9888

0.02993297 0.1558199 0.5632908 1.661912 4.280973 9.923825 20.92931 40.14014 69.63997 108.98 154.648 201.8202 246.7393 287.6492 324.3137 357.2007 386.9366 414.0713 439.0212 462.0829 483.4656 503.3204 521.762 538.8838 554.7664 569.4832 583.1034 595.6933 607.317 618.0365 627.9115 636.9999 645.3566 653.0344 660.0829

1. steep: a vector np long (0.2-1) representing the steepness of the stock-recruitment relationship for each stock

0.5 0.5

1. spawns: an integer vector np long (0.2-1) representing the subyear in which each stock spawns (must be greater than 1)

2 2

1. canspawn: an binary matrix (np rows by nr columns) representing the areas in which each stock may spawn.

0 0 0 0 0 0 0 0 0 1

1 0 1 0 0 0 0 0 0 0

1. Ma: a matrix (np rows by na columns) representing the natural mortality at age of each stock

0.49 0.24 0.24 0.24 0.24 0.2 0.175 0.15 0.125 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

0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14 0.14

1. nCobs: an integer number representing the number of rows of Cobs data (# catch observations)

2754

1. Cobs: an array (nCobs rows by 5 columns) in which each row is a catch observation (column 5) by year (column 1), subyear (column 2), area (column 3), fleet (column 4)

4 2 2 1 687848.1

5 2 2 1 1512417

10 2 2 1 10084.19

11 2 2 1 251642.1

… (nCobs rows)

1. nCPUE: an integer number representing the number of partial F series series

4

1. nCPUEobs: an integer number representing the number of partial F observations

2754

1. CPUEobs: an array (nCPUEobs rows by 6 columns) in which each row is a partial F datum observation (column 6) by year (column 1), subyear (column 2), area (column 3), fleet (column 4) and partial F index number (column 5) matching nCPUE above).

NOTE: For each partial F index (nCPUE), partial Fs are normalized to have a mean of 1 across all time-area strata.

4 2 2 1 1 0.3099188

5 2 2 1 1 0.728963

10 2 2 1 1 0.03535792

11 2 2 1 1 0.8128502

… (nCPUEobs rows)

1. nCLobs: an integer number representing the number of size frequency observations

12197

1. CLobs: an array (nCLobs rows by 6 columns) in which each row is a size frequency observations (number of individuals, column 6) by year (column 1), subyear (column 2), area (column 3), fleet (column 4) and length class (column 5).

44 3 7 1 1 295

47 2 7 3 1 33

49 2 7 3 1 16

26 3 7 4 1 770

… (nCLobs rows)

1. RAI: pass-through data to enable observed vs predicted master index (used to calculate partial Fs) array (nCLobs rows by 6 columns) in which each row is a size frequency observations (number of individuals, column 6) by year (column 1), subyear (column 2), area (column 3), fleet (column 4) and length class (column 5).

44 3 7 1 1 295

1. nI: integer value, the number of fishery-independent relative abundance indices

1

1. nIobs: integer value, the number of fishery-independent relative abundance index observations

34

1. Iobs: a matrix (nIobs rows by 7 columns) of the number of fishery-independent relative abundance index observations (a row per observation). Year (column 1), subyear (column 2), area (column 3), stock (column 4) index number (matching nI, column 5), index type (biomass=1, SSB=2, biomass all stocks=3, column 6), the observed relative abundance index (column 7)

18 2 1 2 1 2 3.321287

19 2 1 2 1 2 6.480129

22 2 1 2 1 2 1.199149

23 2 1 2 1 2 1.748512

… (nIobs rows)

1. nPSAT: number of movements recorded by PSAT tags of known stock of origin

59

1. PSAT: a matrix (nPSAT rows by 6 columns) of recorded PSAT movements of known stock of origin (a row per observation). Stock (column 1), subyear (column 2), time elapsed (subyears) til recapture (column 3), area released (column 4), area recaptured (column 5), number of tags (column 6).

2 1 2 1 1 5

2 2 2 1 2 6

1 1 2 2 2 11

2 1 2 2 2 22

… (nPSAT rows)

1. nPSAT2: integer value, number of movements recorded by PSAT tags of unknown stock of origin

215

1. PSAT2: a matrix (nPSAT2 rows by 4+np columns) of recorded PSAT movements of unknown stock of origin (a row per observation). Subyear (column 1), time elapsed (subyears, column 2), area released (column 3), area recaptured (column 4), probability stock 1 (column 5), probability stock 2 (column 6)….

1 2 7 7 0.99 0.01

2 1 3 4 0.83 0.17

2 1 4 3 0.65 0.35

3 3 5 6 0.98 0.02

… (nPSAT2 rows)

1. nTag: integer value, number of movements recorded by conventional (spaghetti, floy) tags

1568

1. Tag: a matrix (nTag rows by 10 columns) of recorded conventional tag movements (a row per observation). Year released (column 1), subyear released (column 2), area released (column 3), age released (column 4), year recaptured (column 5), subyear recaptured (column 6), area recaptured (column 7), fleet recaptured (column 8), age of recapture (column 9), number of tags (column 10).

2 1 7 4 3 2 2 1 5 10

3 2 7 8 5 2 7 1 10 13

2 4 7 4 6 2 6 1 8 3

1 1 7 4 2 2 7 1 2 8

… (nTag rows)

1. nSOOobs: integer value, number of stock of origin observations

101

1. SOOobs: a matrix (nSOOobs rows by 5 columns) of stock of origin frequency observations (a row per observation). Stock (column 1), year (column 2), subyear (column 3), area (column 4), number of observations (column 5).

2 50 3 5 89

1 50 3 5 8

2 50 4 5 5

2 51 3 5 51

… (nSOOobs rows)

1. nsel: integer value, the number of estimated size selectivities (can be mirrored for multiple fleets if necessary)

3

1. seltype: integer vector, the type of selectivity for each of the nsel selectivities, 2: logistic, 3: Thompson (dome shaped). One of these must be logistic in the current version (0.18)

3 3 2

1. selind: integer vector, which selectivity is assigned to each fleet

1 2 2 3

1. ratiolim: positive real number, 2 position vector with the upper and lower limits on logistic (if any) slope parameter relative to inflection point

0.1 0.4

1. nMP: integer value, number of estimated movement parameters

52

1. nma: integer value, number of movement age classes

3

1. ma: age class assignment by age (np rows, na columns)

1 1 1 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

1 1 1 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

1. nmovind: number of estimated movement parameters minus viscosity

44

1. movind: matrix of integer values (nmovind x 4 columns), the location of estimated movement parameters. Stock (column 1), subyear (column 2), area to (column 3), area from (not used if gravity model, column 4)

1 1 3 999

1 2 3 999

1 3 3 999

1 4 3 999

… (nmovind rows)

1. nmov1: integer value, the number of initial non-estimated movement parameters

8

1. mov1: matrix of integer values, the location of initial non-estimated movement parameters. Stock (column 1), subyear (column 2), area to (column 3), area from (not used if gravity model, column 4)

1 1 2 999

1 2 2 999

1 3 2 999

1 4 2 999

… (nmov1 columns)

1. movtype: integer value, the type of movement parameterization 1:gravity 2:markov matrix

1

1. CobsCV: vector of positive real numbers nf long, the lognormal CV of the observed catches by fleet

0.2 0.2 0.2 0.2

1. CPUEobsCV: vector of positive real numbers nf long, the lognormal CV of the CPUE indices (if complex=1)

0.2 0.2 0.2 0.2

1. IobsCV: vector of positive real numbers nI long, the lognormal CV of the fishery independent indices.

0.2

1. RDCV: positive real number, the CV of the penalty on recruitment deviations.

0.295161

1. R0\_ini: vector of positive real numbers ns long, initial values for R0 (unfished recruitment)

2800000 350000

1. sel\_ini: matrix (nf x nl) of positive real numbers, initial values for length selectivity

0 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

… (nsel rows)

1. selpar\_ini: matrix (nf x 3) of real numbers, the initial values for selectivity parameters

-5 0 -1

-5 0 -1

-1 -1 99

… (nsel rows)

1. lnF\_ini: a vector of positive real values, nCobs long of initial fishing mortality rate estimates

6.907755 -6.907755 -6.907755 -6.907755 -6.907755 -6.907755 … (nCobs values)

1. ilnRD\_ini: matrix ns x na of initial recruitment deviations years=1 ages=2:na

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

0 0 0 0 0 0 0 0 0 0 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. lnRD\_ini: a matrix ns x ny of recruitment deviations after initial year

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

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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. mov\_ini: a 4D array of initial movement probabilities np x ns x nr (area from) x nr (area to)

0 0 0 0 0 0 0 0

0 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571

0 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571

0 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571

0 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571

0 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571

0 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571

… (np x ns x nr rows)

1. qCPUE\_ini: vector of real values nf long that are the initial values for log catchability (used to convert partial Fs to fishing mortality rate or scale CPUE to real abundance)

-4.60517 -4.60517 -4.60517 -4.60517

1. qI\_ini: vector of real numbers nI long that are the initial values for fishery independent catchability

0

1. D\_ini: pass through vector (np long) of simulated (or reference) stock depletion (SSB relative to SSB0)

0.213 0.368

1. complexRD: integer switch for determining whether there should be full estimation of recruitment deviations in each year (1: yes, 0: block recruitment deviations)

0

1. complexF: integer switch for determining whether F’s will be estimated for all years or whether the partial F approach will be used.

0

1. nF: integer value the number of estimated fishing mortality rates (1 if complex F =0)

1

1. debug: integer switch determining whether the model should be run with initial values (for debugging and checking calculations against simulations)(0: no debug, 1: debug)

0

1. verbose: integer switch determining whether the model should print output to the prompt (0:no output, 1:run with printouts)

0

1. datacheck: a value for initially checking whether the data were read in to M3 incorrectly.

991199

# Running M3

The M3 model currently runs as a maximum-likelihood estimator only. It follows that it can be run from the command prompt:

C:/M3/M3

or alternatively from other software such as R (from the R console prompt):

system(“C:/M3/M3.exe”, wait=T, show.output.on.console = T)

# Outputting data and model estimates from M3

The M3 model produces a number of output files, including an M3.rep report file, a maximum likelihood estimate of model parameters M3.par file and an estimated variance-covariance matrix for those parameters M3.cov.

## Reading the report file

The report file can be read from R using the following script

# define a multidimensional array transposition function

tomt<-function(arr){

dim<-new('list')

dims<-dim(arr)

ndims<-length(dims)

for(i in 1:ndims)dim[[i]]<-1:dims[i]

ind<-as.matrix(expand.grid(dim))

out<-array(NA,dims[ndims:1])

out[ind[,ndims:1]]<-arr[ind]

out

}

# define a function for reading ADMB files

ADMBrep<-function(repfile,st,ADMBdim){

tomt(array(scan(repfile,skip=st,nlines=prod(ADMBdim[1:(length(ADMBdim)- 1)])),

ADMBdim[length(ADMBdim):1]))

}

# Create a list of output data

out<-list()

repfile<-“C:/M3/M3.rep"

# (1) np: integer value, the number of populations/stocks

out$np<-scan(repfile,skip=1,nlines=1)

# (2) ny: integer value, the number of model years

out$ny<-scan(repfile,skip=3,nlines=1)

# (3) ns: integer value, the number of subyears

out$ns<-scan(repfile,skip=5,nlines=1)

# (4) nr: integer value, the number of areas

out$nr<-scan(repfile,skip=7,nlines=1)

# (5) nf: integer value, the number of fleets

out$nf<-scan(repfile,skip=9,nlines=1)

# (6) na: integer value, the number of age classes

out$na<-scan(repfile,skip=11,nlines=1)

# (7) nl: integer value, the number of length classes

out$nl<-scan(repfile,skip=13,nlines=1)

# assign values to variables for use below

np<-out$np

ny<-out$ny

ns<-out$ns

nr<-out$nr

nf<-out$nf

na<-out$na

nl<-out$nl

st<-15

#(8) SSB: 3D array (np x ny , ns) of spawning stock biomass estimates

out$SSB<-ADMBrep(repfile,st,ADMBdim=c(np,ny,ns))

st<-st+1+np\*ny

#(9) FL: 5D array (ny x ns x nr x nf x nl) of fishing mortality rate at length estimates

out$FL<-ADMBrep(repfile,st,c(ny,ns,nr,nf,nl))

st<-st+1+ny\*ns\*nr\*nf

# (10) NCobs: integer value, the number of catch observations (pass through data to enable plotting of fit)

out$NCobs<-scan(repfile,skip=st,nlines=1)

st<-st+2

# (11) Cobs: 2 dimensional matrix (NCobs x 5) of catch observations (column 5) by year (column 1), subyear (column 2), area (column 3), fleet (column 4).

out$Cobs<-ADMBrep(repfile,st,c(out$NCobs,5))

st<-st+1+out$NCobs

# (12) Cpred: a 4D array (ny x ns x nr x nf) of predicted catch

out$Cpred<-ADMBrep(repfile,st,c(ny,ns,nr,nf))

st<-st+1+ny\*ns\*nr

# (13) nClobs: integer value, the number of catch at length observations

out$nCLobs<-scan(repfile,skip=st,nlines=1)

st<-st+2

# (14) CLobs: an array (nCLobs rows by 6 columns) in which each row is a size frequency observations (number of individuals, column 6) by year (column 1), subyear (column 2), area (column 3), fleet (column 4) and length class (column 5).

out$CLobs<-ADMBrep(repfile,st,c(out$nCLobs,6))

st<-st+1+out$nCLobs

# (15) CLtotpred: an array (ny x ns x nr x nf x nl) of total catch (numbers) by length class

out$CLtotpred<-ADMBrep(repfile,st,c(ny,ns,nr,nf,nl))

st<-st+1+ny\*ns\*nr\*nf

# (16) mov: a 4D array (np x ns x nr x nr) of estimated movement probabilities

out$mov<-ADMBrep(repfile,st,c(np,ns,nr,nr))

st<-st+1+np\*ns\*nr

# (17) sel: a matrix of estimated selectivities

out$sel<-ADMBrep(repfile,st,c(nf,nl))

st<-st+1+nf

# (18) RAI: a 3D array (nr x ns x ny) of observed relative abundance (pass through – e.g the master index - for evaluating fit)

out$RAI<-ADMBrep(repfile,st,c(nr,ns,ny))

st<-st+1+nr\*ns

# (19) ml: vector (nl long) of mean length by length class

out$ml<-scan(repfile,skip=st,nlines=1)

st<-st+2

# (20) VB: a 4D array (ny x ns x nr x nf) of vulnerable biomass estimates

out$VB<-ADMBrep(repfile,st,c(ny,ns,nr,nf))

st<-st+1+ny\*ns\*nr

# (21) B: a 3D array (ny x ns x nr) of biomass estimates

out$B<-ADMBrep(repfile,st,c(ny,ns,nr))

st<-st+1+ny\*ns

# (22) N: a 5D array (np x ny x ns x na x nr) of estimated stock numbers

out$N<-ADMBrep(repfile,st,c(np,ny,ns,na,nr))

st<-st+1+np\*ny\*ns\*na

# (23) lwa: a vector np long of length- weight parameters a (W=aLb)

out$lwa<-scan(repfile,skip=st,nlines=1)

st<-st+2

# (24) lwa: a vector np long of length- weight parameters b (W=aLb)

out$lwb<-scan(repfile,skip=st,nlines=1)

st<-st+2

# (25) len\_age: a 3D array (ny x na x np) of length at age

out$len\_age<-ADMBrep(repfile,st,c(ny,na,np))

st<-st+1+ny\*na

# (26) wt\_age: a 3D array (ny x na x np) of weight at age

out$wt\_age<-ADMBrep(repfile,st,c(ny,na,np))

st<-st+1+ny\*na

# (27) nMP: integer value, the number of estimated movement parameters

out$nMP<-scan(repfile,skip=st,nlines=1)

st<-st+2

# (28) nmovind: integer value, the position of non viscosity (if gravity model) estimated movement parameters (if movtype = 2 the markov matrix is estimated and nMP = nmovind)

out$nmovind<-scan(repfile,skip=st,nlines=1)

st<-st+2

# (29) movind: a matrix (nmovind x 4) of the position of estimated movement parameters. Stock (column 1), subyear (column 2), area to (column 3), area from (not used if gravity model, column 4)

out$movind<-ADMBrep(repfile,st,c(out$nmovind,4))

st<-st+1+out$nmovind

# (30) nmov1: the number of initial movement parameters that are not estimated (rows must sum to one so there is n-1 degrees of freedom)

out$nmov1<-scan(repfile,skip=st,nlines=1)

st<-st+2

# (31) mov1: matrix of integer values, the location of initial non-estimated movement parameters. Stock (column 1), subyear (column 2), area to (column 3), area from (not used if gravity model, column 4)

out$mov1<-ADMBrep(repfile,st,c(out$nmov1,4))

st<-st+1+out$nmov1

# (32) movtype: integer value, the type of movement model (1:gravity, 2:markov –fully specified)

out$movtype<-scan(repfile,skip=st,nlines=1)

st<-st+2

# (33) M\_age: matrix (np x na) of natural mortality rate at age

out$M\_age<-ADMBrep(repfile,st,c(np,na))

st<-st+1+np

# (34) h: vector np long of the steepness of the stock recruitment curve

out$h<-scan(repfile,skip=st,nlines=1)

st<-st+2

# (35) RDblock: a vector ny long indicating the recruitment deviation blocking

out$RDblock<-scan(repfile,skip=st,nlines=1)

st<-st+2

# (36) mat\_age: a matrix (np x na) of fraction mature at age

fec<-ADMBrep(repfile,st,c(np,na))

out$mat\_age<-fec/t(out$wt\_age[1,,])

st<-st+1+np

# (37) nsel: integer value, the number of different selectivities that are estimated

out$nsel<-scan(repfile,skip=st,nlines=1)

st<-st+2

# (38) seltype: vector of integer values nsel long that represents the selectivity type (2: logistic 3: Thompson)

out$seltype<-scan(repfile,skip=st,nlines=1)

st<-st+2

# (39) selind: a vector nf long, which fleet is using what selectivity

out$selind<-scan(repfile,skip=st,nlines=1)

st<-st+2

# (40) spawns: an integer vector ns long representing the subyear in which spawning occurs

out$spawns<-scan(repfile,skip=st,nlines=1)

st<-st+2

# (41) iALK: a 4D array (np x ny x na x nl) that is the inverse age-length key (conditional probability of being in length class l given age class a P(l|a).

out$iALK<-ADMBrep(repfile,st,c(np,ny,na,nl))

st<-st+1+np\*ny\*na

# (42) lnqs: a vector nf long that is the estimated log catchability of each fleet

out$lnqs<-scan(repfile,skip=st,nlines=1)

st<-st+2

# (43) nZeq: integer value, the number of initial model years of fishing mortality rate that are averaged to calculate equilibrium F

out$nZeq<-scan(repfile,skip=st,nlines=1)

st<-st+2

# (44) nydist: integer value, the number of years that movement M is iterated to calculate the initial spatial population distribution d, to converge on the condition d=Md.

out$nydist<-scan(repfile,skip=st,nlines=1)

st<-st+2

#(45) nyeq: integer value, the number of years before model year 1 that equilibrium F is assumed to have been taken.

out$nyeq<-scan(repfile,skip=st,nlines=1)

st<-st+2

# (46): datacheck, single value, has the output data been read incorrectly?

out$datacheck<-scan(repfile,skip=st,nlines=1)

## Reading the parameter and variance-covariance files

The easiest approach to reading these files is provided here in an R function read.fit() which is adapted from code of Anders Nielson. The code operates on the M3 root directory and automatically writes the MLE parameter estimates of the M3.par file to a list position ‘est’ and the variance-covariance matrix to the position ‘cov’. The list also contains positions for the objective function value ‘nlogl’ and parameter names ‘names’.

read.fit<-function(file="C:/M3"){

ret<-list()

parf<-paste(file,"M3.par",sep="/")

parfile<-as.numeric(scan(parf, what='', n=16, quiet=TRUE)[c(6,11,16)])

ret$nopar<-as.integer(parfile[1])

ret$nlogl<-parfile[2]

ret$maxgrad<-parfile[3]

cfile<-paste(file,'M3.cor', sep='/')

lin<-readLines(cfile)

ret$npar<-length(lin)-2

ret$logDetHess<-as.numeric(strsplit(lin[1], '=')[[1]][2])

sublin<-lapply(strsplit(lin[1:ret$npar+2], ' '),function(x)x[x!=''])

ret$names<-unlist(lapply(sublin,function(x)x[2]))

ret$est<-as.numeric(unlist(lapply(sublin,function(x)x[3])))

ret$std<-as.numeric(unlist(lapply(sublin,function(x)x[4])))

ret$cor<-matrix(NA, ret$npar, ret$npar)

corvec<-unlist(sapply(1:length(sublin), function(i)sublin[[i]][5:(4+i)]))

ret$cor[upper.tri(ret$cor, diag=TRUE)]<-as.numeric(corvec)

ret$cor[lower.tri(ret$cor)] <- t(ret$cor)[lower.tri(ret$cor)]

ret$cov<-ret$cor\*(ret$std%o%ret$std)

return(ret)

}

Output<-read.fit(“C:/M3”)

# Future additions

Version 1.01 may include:

* Dynamic calculation of inverse age-length key
* Fractional movement model (ie perfectly mixed just redistributes individuals with no viscosity)
* Conditional stock of origin assignment to electronic tagging observations of unknown stock of origin based on model estimated stock composition by time-area strata (what is the probability of stock 1 given this path through spatio-temporal strata?)
* A switch to calculate recruitment based on SSB in the spawning area only versus stock wide

# Appendix A. Model equations

## Estimated parameters

The majority of parameters estimated by the model relate to movement probabilities and annual recruitment deviations (Table 1). The number of estimated parameters can be reduced substantially by limiting estimation to only those movements that have been recorded or are considered credible. For example, given a quarterly time step (e.g. Jan-Mar, April-Jun etc.) and the spatial definitions of the 2015 data preparatory meeting (Anon. 2015, Figure 1), an evaluation of conventional tagging for Atlantic bluefin tuna data reveals that less than 80 parameters of the 224 possible are required to characterize all of the possible movements recorded by these tagging data.

## Transition equations

The standard age-structured equations are complicated somewhat by the subyear temporal structure in which ageing and recruitment occur in a particular subyear. In this version of the model, spawning occurs for all stocks in a subyear *ms*, after subyear 1 (this is also likely to be the case in any final model fitted to bluefin tuna data since spawning in the Mediterranean and Gulf of Mexico is thought to occur after a period of movement early in the year).

Numbers of individuals *N*, for stock *s*, in a model year *y*, in the first subyear *m*=1, age class *a*, and area *r* are calculated from individuals that have moved , in the previous year, final subyear *nm*, of the same age class subject to combined natural and fishing mortality rate *Z*:

1)

where total mortality rate is calculated from annual natural mortality rate *M,* divided by the fraction of the year represented by the subyear *tm*, and fishing mortality rate *F*, summed over all fleets *f*.

2)

Fishing mortality rate at age is derived from fishing mortality rate by length class *FL* and the conditional probability of fish being in length class *l,* given age *a* (an inverse age-length key, LAK).:

3)

The fishing mortality rate at length is calculated from an index of fishing mortality rate *I*, an estimated catchability coefficient *q* and a length selectivity ogive *s*, by fleet:

4)

Selectivity is calculated by the Thompson (1994) ogive and an estimate of mean length *L* of an age class *l*:

5)

In the spawning subyear *ms*, aging and recruitment occur:

6)

Recruitment is currently assumed to occur in user-specified spawning area for each stock *rs*. Recruitment is assumed to follow a Beverton-Holt function of spawning stock biomass *SSB* in the defined spawning areas *rs* relative to unfished spawning stock biomass *SSB0* and is subject to annual recruitment deviations *R*, for each stock.

7)

where *h* is the steepness parameter (fraction of unfished recruitment at 1/5 unfished spawning stock biomass) and spawning stock biomass is calculated from moved stock numbers in the subyear prior to spawning subyear *ms*, in spawning area *rs*, weight of individuals at age *w*, and the fraction of individuals mature at age *mat*:

8)

where weight is calculated from length at age *l*:

9)

and fraction mature at age is assumed to be a logistic function of age with parameters for the age at 50% maturity *γ*, and slope *ϑ*:

10)

Stock numbers for subyears that are not the first subyear of the year and are not the spawning subyear are calculated:

11)

In each subyear, after mortality and recruitment, fish are moved according to a Markov transition matrix *mov* that represents the probability of a fish moving from area *k* to area *r* at the end of the subyear *m*:

12)

The movement matrix is calculated from a log-space matrix *lnmov* and a logit model to ensure each row *k*, sums to 1:

13)

Movements from an area *k* to an area *r* that are considered not to be credible (e.g. from the Eastern Mediterranean to the Gulf of Mexico) are assigned a large negative number (essentially zero movement). For each area *k,* from which individuals can move, the first possible value is assigned a value of zero; subsequent possible movements are assigned an estimated parameter *ψ* (since rows must sum to 1 there is one less degree of freedom):

14)

This movement formulation limits estimation to only those movements that are possible given the data (e.g. consistent with observed tagging data).

## Initializing the model

Compared with spatially aggregated models, initialization is more complex for spatial models, particularly those that may need to accommodate movement by age and include regional spawning and recruitment. The solution used here is to iterate the transition equations above (Equations 1, 6, 7, 11, 12) given zero fishing mortality until the spatial distribution of stock numbers converges for each of the subyears.

Prior to this iterative process an initial guess at the spatial and age structure of stock numbers is made using the estimated movement matrix and natural mortality rate at age *M*:

15)

It typically takes between 50 and 100 iteration years of unfished conditions for stock numbers to converge to within 1/10 of a percent of the previous iteration. To ensure stability of the estimation, a fixed number of iterations is defined by the user.

## Predicting data

For each fleet *f*, total predicted catches in weight , are calculated from the Baranov equation:

16)

Similarly predicted catches in numbers at age *CAA*, is given by:

17)

This can be converted to a prediction of total catches in numbers by length class *CAL* using a stock specific inverse age-length key, *LAK*:

18)

The model predicts spawning stock biomass indices , that are standardized to have a mean of 1 for each stock over the total number of years *ny*:

19)

The model predicts vulnerable biomass indices , by fleet that are standardized to have a mean of 1 for each fleet:

20)

Where vulnerable biomass *V* is calculated:

21)

The model predicts stock of origin composition of catches , from predicted catch numbers at age:

22)

## Likelihood functions, priors and the global objective function

Table 2, summarizes the likelihood functions for the various data types. A log-normal likelihood function was assumed for total catches by fleet. The log-likelihood was calculated:

23)

Similarly the log-likelihood component for indices of vulnerable biomass and spawning stock biomass were calculated:

24)

25)

The length composition data are assumed to be distributed multinomially. In traditional stock assessment settings catch composition data may often dominate the likelihood function due to the large number of observations. This is exacerbated by a failure to account for non-independence in size composition samples. There are two possible solutions: (1) manually specify the effective sample size (ESS) of length-composition samples or (2) use a multinomial likelihood function that includes the conditional maximum likelihood estimate of the ESS (perhaps even a freely estimated ESS, S. Martell personal communication). In this version of the code, ESS is user-specified.

The log-likelihood component for length composition data is calculated:

26)

Where the model predicted fraction of catch numbers in each length class *p*, is calculated:

27)

Similarly the log-likelihood component for PSAT tagging data of known stock of origin (SOO), released in year *y*, subyear *m*, area *r* and recaptured in year *y2,* subyear *m2*, and area *k* is calculated:

28)

where recapture probabilities *θ,* are calculated by repeatedly multiplying a distribution vector *d*, by the movement probability matrix *mov*. For example for a tag released on a fish of stock 1 in year 2, subyear 3, and area 4, the probability of detecting the tag in year 3, subyear 2 for the various areas is calculated:

29)

where

30)

The log-likelihood component for PSAT tagging data of unknown stock of origin PSATu, is currently weighted according to the compound probability that a fish is of a particular stock given the track history for that tag. For example for a tag *t*, tracked in series of years *yi*, subyears *mi*, and regions *ri*, the weight *w,* of that tag for a specific stock is calculated:

31)

This is simply the product of fractions of that stock in those time-area strata divided by the product of the fractions of other stocks in those time-area strata. An alternative approach would be to compare the relative probabilities of the observed movements among the stocks although it is unclear whether this circularity (PSAT data are a primary source of information regarding movement) could lead to estimation problems.

The weighted likelihood function is similar to that of the stocks of known origin but includes the appropriate weighting term for each tag

32)

The log-likelihood component for stock of origin data *SOO* was also calculated assuming a multinomial distribution:

33)

In addition to these likelihood functions for observed data, priors may be placed on the steepness parameter *h*, of the stock recruitment relationship and a factor *Mfac*, multiplied by the user specified natural mortality rate-at-age schedule *Minit*.

34)

The factor applied to the natural mortality rate-at-age schedule is assumed to be lognormally distributed according to user specified mean and standard deviation parameters.

35)

Steepness is parameterized by a logit model constrained between 0.2 and 1:

36)

In the logit-1 space, a normal prior is adopted for this transformed steepness , parameter that includes user specified mean , and standard deviation , parameters. The corresponding log-likelihood component is:

37)

The global objective function *OBJT*, to be minimized is the summation of the weighted , likelihood components:

38)

# Appendix B. Model code

// ===========================================================================================================================================

//

// Modifiable Multistock Model (M3)

//

// v0.17 (alpha)

//

// 5st February 2016

//

// ICCAT GBYP, Tom Carruthers UBC

//

// ===========================================================================================================================================

//

// -- About M3 --

//

// M3 is a spatial, multistock, subyear, statistical catch-at-length stock assessment model. M3 was designed to be used as an operating model

// that could be fitted to various data to predict spatial stock structure of a multi-stock fishery (specifically Atlantic bluefin tuna). In

// this regard M3 includes several simplifications over a conventional age-structured stock assessment in order to reduce calculations and

// ensure that the estimation problem is well defined.

//

// M3 is currently in a developmental (alpha) stage. Any comments, bugs or otherwise can be forwarded to t.carruthers@fisheries.ubc.ca.

//

// M3 was compiled using ADMB 11.5 (64 bit) under Windows 8 using mingw64

//

// -- Acknowledgements --

//

// Model structure and assumptions follow the recommendations of the ICCAT MSE core modelling group.

//

// ------------------------------------------------------------------------------------------

// ---------------------------- ICCAT CORE MODELLING GROUP ----------------------------

// Joe Powers, Laurie Kell, Antonio Di Natale, Doug Butterworth, Harritz Arrizabalaga,

// Yukio Takeuchi, Sylvain Bonhommeau, Toshi Kitakado, Clay Porch, David Die,

// Miguel Santos, Paul de Bruyn, Polina Levontin, Richard Hillary

// ------------------------------------------------------------------------------------------

//

// M3 uses Steve Martell's stats.cxx library (publically available as part of the iSCAM package, https://github.com/smartell/iSCAM)

//

//

// -- Some conventions --

//

// 1) Matrices are arranged in order of population, year, season

// 2) Timesteps include subyears and are indexed to year and subyear by yind and sind, respectively

// 3) Stock separation is not explicit in the data except for stock of origin, some PSAT (not PSAT2) and some conventional tags

// 4) Movement by subyear is the movement into an area in that subyear: models runs F(y-1) M(y-1), mov(y), F(y), M(y), mov(y+1),...

// 5) Recapture probabilities are 100% assigned to the release area in first time step after which they are affected by movement e.g. for

// a 5 area model (0,1,0,0,0) (0.05,0.7,0.1,0.1,0.5),...

// 6) First subyear of first year is for initialisation and there is no recorded fishing

// 7) Rows in movement matrix with no recorded movements are assigned 1/nareas probability to avoid the vanishing fish problem (which

// would occur if all areas were assigned zero probability)

//

//

// -- To do minor --

//

// \* infleclim is no longer used in the code below - update model writing functions etc

// \* Investigate speed improvements from assigning data predictions to vector and using vectorized likelihoods

//

//

// -- To do major --

//

// \* Dynamic age-length key based on linear interpolation between discrete growth groups

// \* Conditional stock of origin by PSAT tag track according to SOO data

// \* Fractions (zero viscosity) movement model

// \* Predicted stock of origin by fleet type

// \* Spawning stock is SSB only in spawning area

//

//

// -- Change log since v0.16 (ie post CMG Monterey Jan 2016) --

//

// \* Fractional movement model (26-1-2016)

// \* Movement by age class (28-1-2016)

// \* Spool-up with equilibrium Z over first nZeq years (29-1-2016)

// \* Lag Rec-SSB by one year (29-1-2016)

// \* Plus group calculation (29-1-2016)

// \* Recapture probability by age (1-2-2016)

// \* Fixed bug with spool up that ignored age structures greater than 1 (5-2-2016)

// \* Added pass-through of equilibrium F constants nZeq, nydist, nyeq (report section)

//

// ==========================================================================================================================================

DATA\_SECTION

// -- Model Dimensions --

init\_int ny; // Number of years

init\_int ns; // Number of sub-year time steps

init\_int np; // Number of stocks

init\_int na; // Maximum age

init\_int nr; // Number of areas

init\_int nf; // Number of fleets

init\_int nl; // Number of length classes

init\_int nRPT; // Maximimum number of repapture probability time-steps (seasons)

init\_matrix RPTind(1,ns,1,nRPT); // The correct subyear index for timesteps after initial subyear e.g. for release in s=2: 2,3,4,1,2,3...

init\_vector sdur(1,ns); // Subyear duration e.g. 0.3,0.1,0.2,0.4

init\_int nZeq; // Number of years at the start of the model to calculate equilibrium Z from (mean of nZep years)

init\_int nydist; // Number of years over which initial stock distribution is calculated (prior to spool up)

init\_int nyeq; // Number of spool-up years over which the stock is subject to nZeq, used to define equilibrium conditions

init\_vector ml(1,nl); // Mean length in each length class (used for calculating selectivity only)

init\_vector RDblock(1,ny); // Recruitment blocks for simple recruitment estimation

init\_int nRD; // Number of estimated recruitment dev parameters per stock

// -- Growth --

init\_4darray ALK(1,np,1,ny,1,na,1,nl); // Age-Length Key by stock

init\_vector lwa(1,np); // Length-weight conversion factor a w=al^b

init\_vector lwb(1,np); // Length-weight conversion factor b w=al^b

init\_3darray len\_age(1,ny,1,na,1,np); // Length at age (pass through for independently calculating biomass from .rep file)

init\_3darray wt\_age(1,ny,1,na,1,np); // Weight at age (pass through for independently calculating biomass from .rep file)

// -- Maturity --

init\_matrix Fec(1,np,1,na); // SSB by age by stock (previously a fecundity calculation)

init\_vector steep(1,np); // Steepness of the stock-recruitment curve

// -- Spawning --

init\_vector spawns(1,np); // The subyear in which the stock spawns

// -- Natural Mortality rate --

init\_matrix Ma(1,np,1,na); // Instantaneous natural mortality rate at age by stock

// -- Fishery data --

init\_int nCobs; // Number of catch observations by year, subyear, area, fleet

init\_matrix Cobs(1,nCobs,1,5); // Catch observations

init\_int nCPUE; // Number of CPUE series used / partial F's

init\_int nCPUEobs; // Number of relative abundance index observations by year, subyear, area, fleet, index No. (nCPUE)

init\_matrix CPUEobs(1,nCPUEobs,1,6); // Observed relative index observations

init\_int nCLobs; // Number of Catch-at-length observations

init\_matrix CLobs(1,nCLobs,1,6); // Catch-at-length observations by year, subyear, area, fleet, length category, N

init\_3darray RAI(1,nr,1,ns,1,ny); // The real relative abundance index from the simulation / cpue index (this is pass-through data so that output fit can be summarized)

// -- Fishery independent indices --

init\_int nI; // Number of fishery independent indices

init\_int nIobs; // Number of relative abundance index observations by timestep, area

init\_matrix Iobs(1,nIobs,1,7); // Observed relative index observations year, subyear, area, stock, index number (nI), type (1) biomass (2) SSB, index

// -- PSAT tags --

init\_int nPSAT; // Number of PSAT recapture events (known SOO)

init\_matrix PSAT(1,nPSAT,1,7); // PSAT release-recapture events from stock, subyear, timestep, from-area, to-area, N

init\_int nPSAT2; // Number of PSAT recapture events (not known SOO)

init\_matrix PSAT2(1,nPSAT2,1,5+np); // PSAT2 release-recapture events from subyear, timestep, from-area, to-area, prob p=1, p=2...

// -- Conventional tags --

init\_int nTag; // Number of conventional tag observations

init\_matrix Tag(1,nTag,1,10); // Tag release-recpature from Year/subyear/area/age to Year/subyear/area/fleet/age, N

// -- Stock of origin --

init\_int nSOOobs; // Number of stock of origin observations by stock, year, subyear, area, fleet, N

init\_matrix SOOobs(1,nSOOobs,1,5); // Stock of origin data

// -- Selectivity controls --

init\_int nsel; // Number of estimated selectivities

init\_ivector seltype(1,nsel); // 2: logistic, 3: Thompson dome-shaped

init\_vector selind(1,nf); // Which selectivity is assigned to each fleet

init\_vector ratiolim(1,2); // Limits on logistic slope parameter relative to inflection point

//init\_vector infleclim(1,2); // Limits on modal selectivity

// -- Movement estimation --

init\_int nMP; // Number of movement parameters estimated

init\_int nma; // Number of movement age-classes

init\_matrix ma(1,np,1,na); // Movement age classes by age for each stock

init\_int nmovind; // Number of non-residency paramters estimated

init\_matrix movind(1,nmovind,1,5); // Index of estimable movements (stock, subyear, area from, area to)

init\_int nmov1; // Number of unestimated zero (logspace) movements

init\_matrix mov1(1,nmov1,1,5); // Index of first non-estimated movement parameter (fixed to zero)

init\_int movtype; // 1: gravity (nr), 2: Markov ((nr-1) x nr), 3: fractional (nr-1, no viscosity)

// -- Observation errors --

init\_vector CobsCV(1,nf); // Catch observation error

init\_vector CPUEobsCV(1,nCPUE); // CPUE observation error

init\_vector IobsCV(1,nI); // Fishery independent index obs error (only if complexF = 1)

// -- Priors --

init\_number RDCV; // Recruitment deviation penalty

// -- Likelihood weights --

init\_int nLHw; // Number of likelihood weights

init\_vector LHw(1,nLHw); // Likelihood weights (1 catch, 2 cpue, 3 FIindex, 4 Lcomp, 5 SOO, 6 PSAT, 7 PSAT2, 8 RecDev, 9 mov, 10 sel)

// -- Initial values --

init\_vector R0\_ini(1,np); // Unfished recruitment

init\_matrix sel\_ini(1,nf,1,nl); // Selectivity

init\_matrix selpars\_ini(1,nf,1,3); // Selectivity parameters

init\_vector lnF\_ini(1,nCobs); // Effort (complexF=0) or log fishing mortality rate (complexF=1)

init\_matrix ilnRD\_ini(1,np,2,na); // Recruitment deviations (initial)

init\_matrix lnRD\_ini(1,np,1,ny); // Recruitment deviations

init\_5darray mov\_ini(1,np,1,ns,1,na,1,nr,1,nr); // Movement parameters

init\_vector lnqCPUE\_ini(1,nCPUE); // q estimates for CPUE fleets

init\_vector lnqI\_ini(1,nI); // q estimates for Fish. Ind. Indices

// -- Misc --

init\_int complexRD; // 0: estimate five year blocks of recruitment 1: estimate all annual recruitment deviations

init\_int complexF; // 0: Estimate a q by fleet using C/index as a covariate of effort (nf params e.g. 5) 1: Estimate an F for every Cobs (nCobs params e.g. 1200)

init\_int nF; // nCobs if complexF = 1, or 1 if complexF=0

init\_int debug; // 1 = run with initial values

init\_int verbose; // 1 = print handy text

init\_number datacheck; // Validates length of dat file read, 991199

// -- Integer definitions --

int mi; // Movement index used only in the gravity model to keep track of where the viscosity parameters are

PARAMETER\_SECTION

// -- Estimated parameters --

init\_bounded\_vector lnR0(1,np,10.,16.5,1); // Unfished recruitment

init\_bounded\_matrix selpar(1,nsel,1,seltype,-4.,3.,1); // Selectivity parameters

//init\_bounded\_matrix ilnRD(1,np,2,na,-1.,1.,2); // Recruitment deviations (years prior to initial year)

init\_bounded\_matrix lnRD(1,np,1,nRD,-2.,2.,1); // Recruitment deviations

init\_bounded\_vector movest(1,nMP,-5.,5.,1); // Movement parameters

init\_bounded\_vector lnqCPUE(1,nCPUE,-8.,-2.3,1); // q estimates for CPUE fleets UB mu F=0.3

init\_bounded\_vector lnqI(1,nI,-2.3,2.3,1); // q estimates for Fish. Ind. Indices

// -- Objective function values --

objective\_function\_value objG; // Global objective function value

number objC; // Catch observations

number objCPUE; // Standardized cpue indices

number objI; // Fishery independent indices

number objCL; // Length composition

number objSOO; // Stock of origin

number objRD; // Recruitment deviations

number objmov; // Priors on movement

number objPSAT; // PSAT tags with certain stock of origin

number objPSAT2; // PSAT tags with uncertain stock of origin

// -- Transitions --

5darray N(1,np,1,ny,1,ns,1,na,1,nr); // Stock numbers

matrix NLA(1,na,1,nl); // Stock numbers by age and length

matrix surv(1,np,1,na); // Survival

3darray SSB(1,np,1,ny,1,ns); // Spawning stock biomass

3darray SSBi(1,np,1,ny,1,ns); // Spawning stock biomass index

matrix SSBdist(1,np,1,nr); // A temporary vector storing the spatial distribution of spawning biomass

matrix spawnr(1,np,1,nr); // The fraction of spawning in each area

4darray VB(1,ny,1,ns,1,nr,1,nf); // Vulnerable biomass

3darray B(1,ny,1,ns,1,nr); // Biomass

vector R0(1,np); // Unfished recruitment

vector SSB0(1,np); // Unfished spawning stock biomass

vector SSBpR(1,np); // Unfished spawning stock biomass per recruit

matrix iRD(1,np,2,na); // Recruitment deviations (years prior to initial year)

matrix RD(1,np,1,ny); // Recruitment deviations

5darray CTL(1,np,1,ny,1,ns,1,nr,1,nl); // Catch at length

5darray CTA(1,np,1,ny,1,ns,1,nr,1,na); // Catch at age

// -- Exploitation rates --

vector F(1,nF); // Estimated fishing mortality rate

5darray FAT(1,np,1,ny,1,ns,1,nr,1,na); // Population level F at Age Total

5darray FL(1,ny,1,ns,1,nr,1,nf,1,nl); // Fishing mortality rate at length

4darray FT(1,ny,1,ns,1,nr,1,nl); // Total fishing mortality rate at length all fleets

5darray Z(1,np,1,ny,1,ns,1,na,1,nr); // Total mortality rate at age

4darray Zeq(1,np,1,ns,1,na,1,nr); // Equilibrium Z calculation

vector qCPUE(1,nCPUE); // Catchability of CPUE indices CPUE=qVB

vector qI(1,nI); // Catchability of fishery independent indices (1) I=qB (2) I=qSSB

// -- Selectivity --

matrix msel(1,nsel,1,nl); // Master (estimated) length selectivities

matrix sel(1,nf,1,nl); // Fleet selectivities assigned from master

vector spar(1,3); // Vector of transformed selectivity parameters

// -- Growth --

matrix wl(1,np,1,nl); // Weight at length group

// -- Movement --

5darray movcalc(1,np,1,ns,1,nma,1,nr,1,nr); // Movement calculation matrix

5darray movm(1,np,1,ns,1,nma,1,nr,1,nr); // Master movement matrix by movement age class

5darray mov(1,np,1,ns,1,na,1,nr,1,nr); // Movement matrix by age

// -- Tagging --

6darray RecapP(1,np,1,ns,1,nma,1,nRPT,1,nr,1,nr); // Recapture probabilities

// -- Temporary arrays --

3darray stemp(1,np,1,ns,1,nr); // Temporary season index

vector sind(1,nRPT); // Temporary season index for calculation of recapture probabilities

vector NLtemp(1,nl); // Stores a temporary vector of numbers at length (efficiency)

// -- Observations --

5darray CWpred(1,np,1,ny,1,ns,1,nr,1,nf); // Catches by fleet (weight) and stock

4darray CWtotpred(1,ny,1,ns,1,nr,1,nf); // Total catches by weight

4darray CNpred(1,np,1,ny,1,ns,1,nr); // Catches by all fleets (numbers) and stock

6darray CLpred(1,np,1,ny,1,ns,1,nr,1,nf,1,nl); // Catches by fleet (weight) and stock

5darray CLtotpred(1,ny,1,ns,1,nr,1,nf,1,nl); // Catch composition by fleet

5darray CLtotfrac(1,ny,1,ns,1,nr,1,nf,1,nl); // Catch composition by fleet fraction

//sdreport\_number R0(1); // Requirement of mceval for producing posterior estimates of model parameters/variables

PROCEDURE\_SECTION

if(debug)cout<<"datacheck: "<<datacheck<<endl; // Were the data the correct length?

calcSurvival(); // Calculate survival

calcMovement(); // Calculate movement

calcSelectivities(); // Calculate selectivities

assignPars(); // Assigns estimates of R0, F, iRD, RD, qCPUE, qI

if(debug==1) assignInits(); // Overwrite R0, sel, F, iRD, RD, mov, qCPUE, qI to simulated values

calcF(); // Calculate fishing mortality rate at age / length

initModel(); // Initialize the model (numbers / catches in first year)

//calcDynALK(); // Dynamically calculate inverse age-length key based on predicted fishing mortality rate

calcTransitions(); // Move/kill/reproduce fish over model years

calcRecaptureProb(); // Calcualte recapture probabilities

calcObjective(); // Calculate objective function

if(verbose)simsam(); // Print out simulated values versus estimated values for each function evaluation

if(debug==1) exit(1); // Exit prior to first function evaluation if in debugging mode

FUNCTION assignPars

{

// -- Assign estimated parameters --

R0=mfexp(lnR0); // Assign unfished recruitment

qCPUE=mfexp(lnqCPUE); // Assign catchability for CPUE indices

if(complexRD){ // If annual recruitment deviations are to be estimated

RD=mfexp(lnRD)/mean(mfexp(lnRD)); // Ensure mean 1 recruitment deviations

}

else{ // If blocks of recruitment deviations are to be estiamted

for(int pp=1;pp<=np;pp++){ // Loop over stocks

for(int yy=1;yy<=ny;yy++){ // Loop over years

int ry = RDblock(yy); // Find the correct reference block for this year

RD(pp,yy)=mfexp(lnRD(pp,ry)); // Assign recruitment deviation accordingly

}

RD(pp)/=mean(RD(pp)); // Ensure recruitment deviations sum to 1

} // End of stocks

} // End of recruitment estimation type (complexRD)

//iRD=mfexp(ilnRD\_ini); // Assign initial recruitment deviations

qI=mfexp(lnqI); // Assign fishery independent catchabilities

for(int pp=1;pp<=np;pp++){ // Loop over stocks

wl(pp)=lwa(pp)\*pow(ml,lwb(pp)); // Calcualte weight at length

} // End of stocks

if(debug)cout<<"--- Finished assignPars ---"<<endl;

}

FUNCTION assignInits

{

// -- Assign initial (starting) values --

R0=R0\_ini; // Assign unfished recruitment

//iRD=mfexp(ilnRD\_ini); // Assign initial recruitment deviations

RD=mfexp(lnRD\_ini); // Assign recruitment deviations

if(complexF)F=mfexp(lnF\_ini); // Assign fishing mortality rates

sel=sel\_ini; // Assign selectivitiese

mov=mov\_ini; // Assign movement

qCPUE=mfexp(lnqCPUE\_ini); // Assign CPUE index catchability

qI=mfexp(lnqI\_ini); // Assign fishery independent index catchability

if(debug)cout<<"--- Finished assignInits ---"<<endl;

}

FUNCTION calcSurvival

{

// -- Calculate survival --

for(int pp=1;pp<=np;pp++){ // Loop over stocks

surv(pp,1)=1.; // Survival to age 1 is 100%

for(int aa=1;aa<=(na-1);aa++){ // Loop over age classes

surv(pp,aa+1)=exp(-sum(Ma(pp)(1,aa))); // Calculate survivial

} // End of age classes

} // End of stocks

if(debug)cout<<"--- Finished calcSurvival ---"<<endl;

}

FUNCTION calcMovement

{

// -- Movement modelling -----------

for(int pp=1;pp<=np;pp++){ // Loop over stocks

for(int ss=1;ss<=ns;ss++){ // Loop over subyears

movcalc(pp)(ss)=-10.; // Set all movements to be unlikely (logit space)

} // End of subyears

} // End of stock

switch(movtype){ // What type of movement model?

case 1: // -- Gravity model ---------------------------------------------------------------------

for(int mm=1;mm<=nmov1;mm++){ // Set the first possible movement to be fixed at zero

int pp=mov1(mm,1); // Stock

int ss=mov1(mm,2); // Subyear

int aa=mov1(mm,3); // Movement age class

int tr=mov1(mm,4); // To area

for(int fr=1;fr<=nr;fr++){ // Loop over from-areas

movcalc(pp,ss,aa,fr,tr)=0.; // Set to zero

} // End of from-areas

} // End of first possible movement

for(int mm=1;mm<=nmovind;mm++){ // Loop over estimated movement params (first np\*ns\*nma are residency viscosity parameters)

int pp=movind(mm,1); // Stock

int ss=movind(mm,2); // Subyear

int aa=movind(mm,3); // Movement age class

int tr=movind(mm,4); // To area

for(int fr=1;fr<=nr;fr++){ // Loop over from-areas

movcalc(pp,ss,aa,fr,tr)=movest(mm+np\*ns\*nma); // Assign estimated parameter

} // End of from-areas

} // End of estimated movements

mi=0;

for(int pp=1;pp<=np;pp++){ // Loop over stocks

for(int ss=1;ss<=ns;ss++){ // Loop over subyears

mi+=1; // Keep track of viscosity parameter number

for(int aa=1;aa<=nma;aa++){ // Loop over age classes

for(int rr=1;rr<=nr;rr++){ // Loop over areas

movcalc(pp,ss,aa,rr,rr)+=mfexp(movest(mi)); // Add viscosity

}

} // End of age class

} // End of subyear

} // End of stock

break; // End of gravity model

case 2: // -- Fully prescribed (Markov) movement matrix ------------------------------------------

for(int mm=1;mm<=nmov1;mm++){ // Set the first possible movement to be fixed at zero

int pp=mov1(mm,1); // Stock

int ss=mov1(mm,2); // Subyear

int aa=mov1(mm,3); // Movement age class

int fr=mov1(mm,4); // From area

int tr=mov1(mm,5); // To area

movcalc(pp,ss,aa,fr,tr)=0.; // Set to zero

}

for(int mm=1;mm<=nMP;mm++){ // Assign all other logit space movement parameters to the mov array

int pp=movind(mm,1); // Stock

int ss=movind(mm,2); // Subyear

int aa=mov1(mm,3); // Movement age class

int fr=movind(mm,4); // From area

int tr=movind(mm,5); // To area

movcalc(pp,ss,aa,fr,tr)=movest(mm); // Set to estimated parameter

}

break; // End of fully prescribed (Markov) movement matrix

case 3: // -- Fractional model (essentially gravity model with no viscosity parameter) -----------

for(int mm=1;mm<=nmov1;mm++){ // Set the first possible movement to be fixed at zero

int pp=mov1(mm,1); // Stock

int ss=mov1(mm,2); // Subyear

int aa=mov1(mm,3); // Movement age class

int tr=mov1(mm,4); // To area

for(int fr =1;fr<=nr;fr++){ // Loop over from-areas

movcalc(pp,ss,aa,fr,tr)=0.; // Set to zero

} // End of from-areas

} // End of first possible movement

for(int mm=1;mm<=nmovind;mm++){ // Other possible movements are set to estimated parameters

int pp=movind(mm,1); // Stock

int ss=movind(mm,2); // Subyear

int aa=mov1(mm,3); // Movement age class

int tr=movind(mm,4); // To area

for(int fr =1;fr<=nr;fr++){ // Loop over from-areas

movcalc(pp,ss,aa,fr,tr)=movest(mm); // Assign estimated parameter

} // End of from-areas

} // End of possible movement

break; // End of fractional movement model

} // End of types of movement models case

// -- Logit transformation ----------------------------------------------

movm=exp(movcalc); // Make positive

for(int pp=1;pp<=np;pp++){ // Stock

for(int ss=1;ss<=ns;ss++){ // Subyear

for(int aa=1;aa<=nma;aa++){ // Movement age class

for(int fr = 1; fr<=nr;fr++){ // From area

movm(pp)(ss)(aa)(fr)/=sum(movm(pp)(ss)(aa)(fr)); // Normalize to sum to 1 (inv logit)

} // End of area

} // End of movement age class

} // End of subyear

} // End of stock

// -- Transcode this to an age structured movement matrix -----------

for(int pp=1;pp<=np;pp++){ // Stock

for(int ss=1;ss<=ns;ss++){ // Subyear

for(int aa=1;aa<=na;aa++){ // Age

int aam=ma(pp,aa); // Retrieve relevant movement age class

mov(pp)(ss)(aa)=movm(pp)(ss)(aam); // Assign movement age class to age

} // End of age

} // End of subyear

} // End of stock

if(debug)cout<<"--- Finished calcMovement ---"<<endl;

}

FUNCTION calcSelectivities

{

// Selectivity calculations =======================================================

// -- Master selectivities (can be mirrored across fleets) --

for(int ss=1;ss<=nsel;ss++){ // Loop over estimated selectivities

switch(seltype(ss)){ // Cases match number of estimated parameters for simplicity

case 2: // Logistic selectivity

spar(2)=ml(nl)\*(0.1+0.7\*mfexp(selpar(ss,2))/(1+mfexp(selpar(ss,2)))); // Inflection point (2) as a fraction of largest length I(0.1|0.8)

spar(1)=spar(2)\*(0.01+0.49\*(mfexp(selpar(ss,1))/(1+mfexp(selpar(ss,1))))); // Logistic slope (1) as fraction of inflection point (2) I(0.01|0.5)

for(int ll=1;ll<=nl;ll++){ // Loop over length classes

msel(ss,ll)=1/(1+mfexp((spar(2)-ml(ll))/spar(1))); // Logistic selectivity function

} // End of length classes

break; // End of logistic selectivity

case 3: // Thompson dome-shaped selectivity

spar(1)=0.2\*mfexp(selpar(ss,1))/(1+mfexp(selpar(ss,1))); // Dome-shape parameter I(0|0.2)

spar(2)=0.1+(0.6\*mfexp(selpar(ss,2))/(1+mfexp(selpar(ss,2)))); // Precision as the ratio of the inflection point I(0.1|0.7)

spar(3)=ml(nl)\*(0.1+(0.8\*mfexp(selpar(ss,3))/(1+mfexp(selpar(ss,3))))); // Inflection point as a fraction of largest length I(0.15|0.9)

for(int ll=1;ll<=nl;ll++){ // Loop over length classes

msel(ss,ll)=(1/(1-spar(1)))\*pow(((1-spar(1))/spar(1)),spar(1)) \* mfexp(spar(2)\*spar(1)\*(spar(3)-ml(ll)))/(1+mfexp(spar(2)\*(spar(3)-ml(ll)))); // Thompson selectivity function

} // End of length classes

break; // End of Thompson selectivity

}

}

// -- Fleet specific selectivities --

for(int ff=1;ff<=nf;ff++){

int si=selind(ff); // Find correct master index

sel(ff)=msel(si); // Map master selectivities onto fleet-specific selectivities

}

if(debug)cout<<"--- Finished calcSelectivities ---"<<endl;

}

FUNCTION calcF

{

double tiny=1E-10;

FL.initialize(); // Fishing mortality rate at length = 0

FT.initialize(); // Total fishing mortality rate = 0

FAT.initialize(); // Total fishing mortality rate at age = 0

Z.initialize(); // Total mortality = 0

Zeq.initialize(); // Equilibrium Z = 0

if(complexF){ // -- Estimate a fishing mortality rate for each catch observation ----------------

for(int i=1; i<=nCobs; i++){ // Only calculate F's when catches are observed

int yy=Cobs(i,1); // Year

int ss=Cobs(i,2); // Subyear

int rr=Cobs(i,3); // Region

int ff=Cobs(i,4); // Fleet

FL(yy)(ss)(rr)(ff)= sel(ff)\*F(i); // Assign estimated fishing mortality rate-at-length to array

} // End of catch observations

}else{ // -- Alternatively use an index of fishing effort -------------------------------

for(int i=1;i<=nCPUEobs;i++){

int yy=CPUEobs(i,1); // Year

int ss=CPUEobs(i,2); // Subyear

int rr=CPUEobs(i,3); // Region

int ff=CPUEobs(i,4); // Fleet

FL(yy)(ss)(rr)(ff)= sel(ff)\*CPUEobs(i,6)\*qCPUE(ff); // Calculate fishing mortality rate at length

}

}

for(int yy=1; yy<=ny;yy++){ // Loop over years

for(int ss=1;ss<=ns;ss++){ // Loop over seasons

for(int rr =1;rr<=nr;rr++){ // Loop over areas

for(int ll=1;ll<=nl;ll++){ // Loop over length bins

FT(yy,ss,rr,ll)=tiny; // Avoids division by zero problems

for(int ff=1;ff<=nf;ff++){ // Loop over fleets

FT(yy,ss,rr,ll)+=FL(yy,ss,rr,ff,ll); // Summation of F-at-length

} // End of fleets

} // End of length classes

} // End of areas

} // End of subyears

} // End of years

// -- Calculate F at age --

for(int pp=1;pp<=np;pp++){ // Loop over stocks

for(int yy=1;yy<=ny;yy++){ // Loop over years

for(int ss=1;ss<=ns;ss++){ // Loop over subyears

for(int rr=1;rr<=nr;rr++){ // Loop over areas

FAT(pp)(yy)(ss)(rr)=FT(yy)(ss)(rr)\*trans(ALK(pp)(yy))+tiny; // Calculate fishing mortality rate at age, F(a) = sigma(l) P(l|a)\*F(l)

for(int aa=1;aa<=na;aa++){ // Loop over ages

Z(pp)(yy)(ss)(aa)(rr)=FAT(pp)(yy)(ss)(rr)(aa)+(Ma(pp,aa)\*sdur(ss)); // Calculate total mortality rate

if(yy<=nZeq){ // Equilibrium Z calculation

Zeq(pp)(ss)(aa)(rr)+=Z(pp)(yy)(ss)(aa)(rr); // Sum up equilibrium Z

}

} // End of ages

} // End of areas

} // End of subyears

} // End of years

} // End of stocks

Zeq/=nZeq; // Divide by number of equilibrium Z years to calculate mean equilibrium Z

if(debug)cout<<"--- Finished calcF ---"<<endl;

}

FUNCTION initModel

{

double tiny=1E-10; // Define a small number for ensuring that log(0) doesn't happen

N.initialize(); // Stock numbers = 0

B.initialize(); // Stock biomass = 0

SSB.initialize(); // Spawning stock biomass = 0

SSBdist.initialize(); // Spawning distribution = 0

SSB0.initialize(); // Unfished spawning stock biomass = 0

SSBpR.initialize(); // SSB per recruit = 0

CWtotpred.initialize(); // Total catch (weight) = 0

CWpred.initialize(); // Catch (weight) = 0

CNpred.initialize(); // Catch (numbers) = 0

CLpred.initialize(); // Catch (length class) = 0

CLtotpred.initialize(); // Total catch (length class) = 0

CLtotfrac.initialize(); // Total catch fractions (length class) = 0

CTA.initialize(); // Temporary catch at age = 0

VB.initialize(); // Vulnerable biomass = 0

for(int pp=1;pp<=np;pp++){

SSB0(pp)=sum(elem\_prod(surv(pp)\*R0(pp),Fec(pp))); // Unfished Spawning Stock Biomass

SSBpR(pp)=SSB0(pp)/R0(pp); // Unfished SSB per recruit

}

for(int pp=1;pp<=np;pp++){ // Loop over stocks

// -- Initial guess at stock distribution -------------------------------------------

stemp(pp)(ns)=1./nr; // Distribute a fish evenly over areas

for(int ii=1;ii<=nydist;ii++){ // Loop over 10 years

for(int ss=1;ss<=ns;ss++){ // Loop over subyears

if(ss==1){ // Take bits of fish from previous years final subyear

stemp(pp)(1)=stemp(pp)(ns)\*mov(pp)(ss)(na); // Move them assuming mature movement

} // End of 'if first subyear'

else{ // Take bits of fish from this years' previous subyear

stemp(pp)(ss)=stemp(pp)(ss-1)\*mov(pp)(ss)(na); // Move them assuming mature movement

} // End of 'if not first subyear'

} // End of subyears

} // End of years

// -- Spool up to equilibrium given total mortality rate over first nZeq years -----------

for(int rr=1;rr<=nr;rr++){ // Loop over areas

for(int aa=1;aa<=na;aa++){

N(pp,1,ns,aa,rr)=R0(pp)\*surv(pp,aa)\*stemp(pp,ns,rr); // Stock numbers are spatial distribution multiplied by Survival and R0

SSB(pp,1,ns)+=N(pp,1,ns,aa,rr)\*Fec(pp,aa); // Spawning stock biomass

} // End of subyears

} // End of areas

for(int yy=1;yy<=nyeq;yy++){ // Loop over equilibrium years

for(int ss=1;ss<=ns;ss++){ // Loop over seasons

if(ss==1){ // First subyear isn't a spawning season ------------------------------------------------

for(int aa=1;aa<=na;aa++){ // Loop over age classes

N(pp)(1)(1)(aa)=elem\_prod(mfexp(-Zeq(pp)(ns)(aa)),N(pp)(1)(ns)(aa))\*mov(pp)(1)(aa); // Mortality then move

for(int rr=1;rr<=nr;rr++){ // Loop over areas

SSB(pp,1,1)+=N(pp,1,1,aa,rr)\*Fec(pp,aa); // SSB is summed (+=) over age classes and areas (sum())

} // End of areas

} // End of ages

}else{ // Could be a spawning season ------------------------------

if(ss==spawns(pp)){ // If a spawning season...-------------------------------------------------------------------

for(int rr=1;rr<=nr;rr++){ // Loop over areas

SSBdist(pp,rr)=0.; // Reset SSB distribution counter

} // End of areas

for(int aa=1;aa<=na;aa++){ // Loop over age classes

N(pp)(1)(ss)(aa)=elem\_prod(mfexp(-Zeq(pp)(ss-1)(aa)),N(pp)(1)(ss-1)(aa))\*mov(pp)(ss)(aa); // Mortality then move

for(int rr=1;rr<=nr;rr++){ // Loop over areas

SSB(pp,1,ss)+=(N(pp,1,ss,aa,rr)\*Fec(pp,aa)); // SSB is summed (+=) over age classes and areas (sum())

SSBdist(pp,rr)+=(N(pp,1,ss,aa,rr)\*Fec(pp,aa)); // The distribution of SSB among areas

} // End of areas

} // End of age classes

for(int rr=1;rr<=nr;rr++){ // Loop over areas

spawnr(pp,rr)=SSBdist(pp,rr)/sum(SSBdist(pp)); // Calculate spawning fraction of SSB

}

N(pp)(yy)(ss)(na)+=N(pp)(yy)(ss)(na-1); // Plus group

for(int aa=(na-1);aa>=2;aa-=1){ // Loop down age classes from plusgroup(ns)-1

N(pp)(yy)(ss)(aa)=N(pp)(yy)(ss)(aa-1); // Age fish

} // End of ages

N(pp)(1)(ss)(1)=spawnr(pp)\*(0.8\*R0(pp)\*steep(pp)\*SSB(pp,1,ss))/

(0.2\*SSBpR(pp)\*R0(pp)\*(1-steep(pp))+(steep(pp)-0.2) \* SSB(pp,1,ss)); // Recruitment

}else{ // Not a spawning season ----------------------------------------------------------------

for(int aa=1;aa<=na;aa++){ // Loop over age classes

N(pp)(1)(ss)(aa)=elem\_prod(mfexp(-Zeq(pp)(ss-1)(aa)),N(pp)(1)(ss-1)(aa))\*mov(pp)(ss)(aa); // M then move

for(int rr=1;rr<=nr;rr++){ // Loop over areas

SSB(pp,1,ss)+=(N(pp,1,ss,aa,rr)\*Fec(pp,aa)); // SSB is summed (+=) over age classes and areas (sum())

} // End of areas

} // End of ages

} // End of 'not a spawning season'

} // End of 'could be a spawning season?'

} // End of subyears

} // End of year

// -- Initial year catch calculations ----------------------------------------------------------

for(int rr=1;rr<=nr;rr++){ // Loop over areas

for(int ss=1;ss<=ns;ss++){ // Loop over seasons

CTA(pp)(1)(ss)=trans(elem\_prod(

elem\_prod(N(pp)(1)(ss),mfexp(Z(pp)(1)(ss))-1),

elem\_div(trans(FAT(pp)(1)(ss)),Z(pp)(1)(ss)))); // total catch at age in first year

CTL(pp)(1)(ss)(rr)=CTA(pp)(1)(ss)(rr)\*ALK(pp)(1); // Catch at length is catch at age \* inverse age length key

for(int aa=1;aa<=na;aa++){ // Loop over age class

NLA(aa)=N(pp)(1)(ss)(aa)(rr)\*ALK(pp)(1)(aa); // Temporarily store numbers at length

} // End of age class

NLtemp=colsum(NLA); // Numbers at length (sum over age classes)

B(1,ss,rr)+=sum(elem\_prod(NLtemp,wl(pp))); // Biomass summed over stocks

for(int ff=1;ff<=nf;ff++){ // Loop over fleet types

VB(1,ss,rr,ff)+=sum(elem\_prod(elem\_prod(NLtemp,wl(pp)),sel(ff))); // Vulnerable biomass summed over stocks

} // End of fleets

for(int ff=1;ff<=nf;ff++){ // Loop over fleets

for(int ll=1;ll<=nl;ll++){ // Loop over length classes

CLpred(pp)(1)(ss)(rr)(ff)(ll)=CTL(pp)(1)(ss)(rr)(ll)\*(FL(1)(ss)(rr)(ff)(ll)/FT(1)(ss)(rr)(ll)); // Catch at length by stock // Length composition catches

CLtotpred(1)(ss)(rr)(ff)(ll)+=CLpred(pp)(1)(ss)(rr)(ff)(ll); // Total predicted length composition of catches

} // End of length classes

CWpred(pp,1,ss,rr,ff)=sum(elem\_prod(CLpred(pp)(1)(ss)(rr)(ff),wl(pp))); // Total catch weight by fleet and stock

CWtotpred(1,ss,rr,ff)+=CWpred(pp,1,ss,rr,ff); // Total predicted catch weight by fleet

CLtotfrac(1)(ss)(rr)(ff)=CLtotpred(1)(ss)(rr)(ff)/(sum(CLtotpred(1)(ss)(rr)(ff))+tiny); // Fraction of catch at length class

} // End of fleets

CNpred(pp,1,ss,rr)=sum(CTL(pp)(1)(ss)(rr)); // total predicted catches (numbers) of all fleets

} // End of subyears

} // End of areas

} // End of stocks

if(debug)cout<<"--- Finished initModel ---"<<endl;

}

FUNCTION calcTransitions

{

// Order of calculations Catch(y-1), M(y-1), move(y), Catch(y), M(y), mov(y+1)...

double tiny=1E-10; // Create a small constant to avoid the log(0) error.

for(int pp=1;pp<=np;pp++){ // Loop over stocks

for(int yy=2;yy<=ny;yy++){ // Loop over years

for(int ss=1;ss<=ns;ss++){ // Loop over seasons

if(ss==1){ // First subyear isn't a spawning season ----------------------------------------------------------

for(int aa=1;aa<=na;aa++){ // Loop over age classes

N(pp)(yy)(1)(aa)=elem\_prod(mfexp(-Z(pp)(yy-1)(ns)(aa)),N(pp)(yy-1)(ns)(aa))\*mov(pp)(1)(aa); // Mortality then move

for(int rr=1;rr<=nr;rr++){ // Loop over areas

SSB(pp,yy,1)+=N(pp,yy,1,aa,rr)\*Fec(pp,aa); // SSB is summed (+=) over age classes and areas (sum())

} // End of areas

} // End of ages

}else{ // Could be a spawning season ---------------------------------------------------------------------

if(ss==spawns(pp)){ // If a spawning season...-------------------------------------------------------------------

for(int rr=1;rr<=nr;rr++){ // Loop over areas

SSBdist(pp,rr)=0.; // Reset SSB distribution counter

} // End of areas

for(int aa=1;aa<=na;aa++){ // Loop over age classes

N(pp)(yy)(ss)(aa)=elem\_prod(mfexp(-Z(pp)(yy)(ss-1)(aa)),N(pp)(yy)(ss-1)(aa))\*mov(pp)(ss)(aa); // Mortality then move

for(int rr=1;rr<=nr;rr++){ // Loop over areas

SSB(pp,yy,ss)+=(N(pp,yy,ss,aa,rr)\*Fec(pp,aa)); // SSB is summed (+=) over age classes and areas (sum())

SSBdist(pp,rr)+=(N(pp,yy,ss,aa,rr)\*Fec(pp,aa)); // Calculate distribution of SSB over areas

} // End of areas

} // End of age classes

for(int rr=1;rr<=nr;rr++){ // Loop over areas

spawnr(pp,rr)=SSBdist(pp,rr)/sum(SSBdist(pp)); // Calculate spawning fraction

} // End of areas

N(pp)(yy)(ss)(na)+=N(pp)(yy)(ss)(na-1); // Plus group calculation

for(int aa=(na-1);aa>=2;aa-=1){ // Loop down age classes

N(pp)(yy)(ss)(aa)=N(pp)(yy)(ss)(aa-1); // Age fish

} // End of age classes

N(pp)(yy)(ss)(1)=spawnr(pp)\*RD(pp,yy)\*(0.8\*R0(pp)\*steep(pp)\*SSB(pp,yy-1,ss))/

(0.2\*SSBpR(pp)\*R0(pp)\*(1-steep(pp))+(steep(pp)-0.2) \* SSB(pp,yy-1,ss)); // Recruitment (SSB lag 1 year)

}else{ // Not a spawning season ----------------------------------------------------------------

for(int aa=1;aa<=na;aa++){ // Loop over age classes

N(pp)(yy)(ss)(aa)=elem\_prod(mfexp(-Z(pp)(yy)(ss-1)(aa)),N(pp)(yy)(ss-1)(aa))\*mov(pp)(ss)(aa); // M then move

for(int rr=1;rr<=nr;rr++){ // Loop over areas

SSB(pp,yy,ss)+=(N(pp,yy,ss,aa,rr)\*Fec(pp,aa)); // SSB is summed (+=) over age classes and areas (sum())

} // End of areas

} // End of ages

} // Not a spawning season

} // Could be a spawning season?

CTA(pp)(yy)(ss)=trans(elem\_prod(

elem\_prod(N(pp)(yy)(ss),mfexp(Z(pp)(yy)(ss))-1),

elem\_div(trans(FAT(pp)(yy)(ss)),Z(pp)(yy)(ss)))); // Calculate catch at age

for(int rr=1;rr<=nr;rr++){ // Loop over areas

CTL(pp)(yy)(ss)(rr)=CTA(pp)(yy)(ss)(rr)\*ALK(pp)(yy); // Calculate catch at length

for(int aa=1;aa<=na;aa++){ // Loop over age classes

NLA(aa)=N(pp)(yy)(ss)(aa)(rr)\*ALK(pp)(yy)(aa); // Temporarily store numbers at length

} // End of age classes

NLtemp=colsum(NLA); // Calculate numbers by length class (sum over ages)

B(yy,ss,rr)+=sum(elem\_prod(NLtemp,wl(pp))); // Biomass summed over stocks

for(int ff=1;ff<=nf;ff++){ // Loop over fleets

VB(yy,ss,rr,ff)+=sum(elem\_prod(elem\_prod(NLtemp,wl(pp)),sel(ff))); // Vulnerable biomass summed over stocks

} // End of fleets

for(int ff=1;ff<=nf;ff++){ // Loop over fleets

for(int ll=1;ll<=nl;ll++){ // Loop over length classes

CLpred(pp)(yy)(ss)(rr)(ff)(ll)=CTL(pp)(yy)(ss)(rr)(ll)\*(FL(yy)(ss)(rr)(ff)(ll)/FT(yy)(ss)(rr)(ll)); // Catch at length by stock // Length composition catches

CLtotpred(yy)(ss)(rr)(ff)(ll)+=CLpred(pp)(yy)(ss)(rr)(ff)(ll); // Total predicted length composition of catches

} // End of length classes

CLtotfrac(yy)(ss)(rr)(ff)=CLtotpred(yy)(ss)(rr)(ff)/(sum(CLtotpred(yy)(ss)(rr)(ff))+tiny); // Calculate catch fractions by length class

CWpred(pp,yy,ss,rr,ff)=sum(elem\_prod(CLpred(pp)(yy)(ss)(rr)(ff),wl(pp))); // Total catch weight by fleet by stock

CWtotpred(yy,ss,rr,ff)+=CWpred(pp,yy,ss,rr,ff); // Total (over ages and stocks) predicted catches by fleet

} // End of fleets

CNpred(pp,yy,ss,rr)=sum(CTL(pp)(yy)(ss)(rr)); // Total predicted catch numbers (all fleets)

} // End of areas

} // End of subyear

} // End of year

SSBi(pp)=SSB(pp)/(sum(SSB(pp))/(ny\*ns)); // Calculate SSB index normalized to 1

} // End of stock

if(debug)cout<<"--- Finished calcTransitions ---"<<endl;

}

FUNCTION calcRecaptureProb

{

for(int pp=1;pp<=np;pp++){ // Stocks

for(int ss=1; ss<=ns;ss++){ // Subyears

int s2 = RPTind(ss,2); // Retrieve the correct subyear for timestep tt and release subyear ss

for(int aa=1; aa<=nma; aa++){ // loop over movement age classes

for(int rr=1;rr<=nr;rr++){ // Regions

RecapP(pp)(ss)(aa)(1)(rr)=0.; // Set the area vector to all zeros

RecapP(pp,ss,aa,1,rr,rr)=1.; // Recapture probability is 100% for same area in the same timestep

//for(int tt=2;tt<=nRPT;tt++){ // Timesteps (incremental subyears)

RecapP(pp)(ss)(aa)(2)(rr)=RecapP(pp)(ss)(aa)(1)(rr)\*mov(pp)(s2)(aa); // Recalculate recapture probability in next timestep given movement

//} // timestep (nRPt)

}

} // End of areas

} // End of subyears

} // End of stocks

if(debug)cout<<"--- Finished calcRecaptureProb ---"<<endl;

}

FUNCTION calcObjective

{

objG.initialize(); // Global

objC.initialize(); // Catch data

objCPUE.initialize(); // Standardized cpue indices

objI.initialize(); // Fishery independent indices

objCL.initialize(); // Length composition

objSOO.initialize(); // Stock of origin

objRD.initialize(); // Recruitment deviations

objmov.initialize(); // Priors on movement

objPSAT.initialize(); // PSAT tags certain stock of origin

objPSAT2.initialize(); // PSAT tags w uncertain stock of origin

dvariable LHtemp; // Temporary store of the calculated likelihood values

double tiny=1E-10; // Create a small constant to avoid log(0) error

// -- Catch observations --

for(int i=1;i<=nCobs;i++){ // Loop over catch observations

int yy=Cobs(i,1); // Year

int ss=Cobs(i,2); // Subyear

int rr=Cobs(i,3); // Region

int ff=Cobs(i,4); // Fleet

LHtemp=dnorm(log(CWtotpred(yy,ss,rr,ff)+tiny),log(Cobs(i,5)+tiny),CobsCV(ff)); // Log-normal LHF

objC+=LHtemp\*LHw(1); // Weighted likelihood contribution

objG+=LHtemp\*LHw(1); // Weighted likelihood contribution

//cout<<"y="<<yy<<" s="<<ss<<" r="<<rr<<" f="<<ff<<" Cobs="<<Cobs(i,5)<<" Cpred="<<CWtotpred(yy,ss,rr,ff)<<endl;

}

//exit(1);

if(debug)cout<<"--- \* Finished Catch LHF ---"<<endl;

// -- CPUE observations --

dvariable CPUEtemp;

if(complexF){ // The CPUE indices are contributing to the likelihood function

for(int i=1;i<=nCPUEobs;i++){ // Loop over catch rate indices

int yy=CPUEobs(i,1); // Year

int ss=CPUEobs(i,2); // Subyear

int rr=CPUEobs(i,3); // Region

int ff=CPUEobs(i,4); // index ID

int CPUEi=CPUEobs(i,5); // q index

CPUEtemp=VB(yy,ss,rr,ff)\*qCPUE(CPUEi); // Calculate vulnerable biomass

LHtemp=dnorm(log(CPUEtemp+tiny),log(CPUEobs(i,6)+tiny),CPUEobsCV(CPUEi)); // Log-normal LHF

objCPUE+=LHtemp\*LHw(2); // Weighted likelihood contribution

objG+=LHtemp\*LHw(2); // Weighted likelihood contribution

}

if(debug)cout<<"--- \* Finished CPUE LHF ---"<<endl;

}

// -- Fishery independent indices --

dvariable Itemp; // Create a dummy variable cor calculating a normalilized (mean 1) index

for(int i=1; i<=nIobs;i++){ // Loop over fishery - independent indices

int yy=Iobs(i,1); // Year

int ss=Iobs(i,2); // Subyear

int rr=Iobs(i,3); // Region

int pp=Iobs(i,4); // stock

int ii=Iobs(i,5); // q index (often stock for SSB types)

int tt=Iobs(i,6); // Type

//cout<<"y="<<yy<<" s="<<ss<<" r="<<rr<<" q index ="<<ii<<" type="<<tt<<endl;

switch(tt){

case 1: // Biomass

Itemp= B(yy,ss,rr)\*qI(ii); // Predicted index

break;

case 2: // SSB

Itemp=qI(ii)\*(SSB(pp,yy,ss)/mean(extract\_row(trans(SSB(pp)),ss))); // Predicted index normalized to mean 1

break;

}

LHtemp=dnorm(log(Itemp+tiny),log(Iobs(i,7)+tiny),IobsCV(ii)); // Log-normal LHF

objI+=LHtemp\*LHw(3); // Weighted likelihood contribution

objG+=LHtemp\*LHw(3); // Weighted likelihood contribution

}

if(debug)cout<<"--- \* Finished FI index LHF ---"<<endl;

// -- Length composition data --

for(int i=1;i<=nCLobs;i++){ // Loop over catch at length observations

int yy=CLobs(i,1); // Year

int ss=CLobs(i,2); // Subyear

int rr=CLobs(i,3); // Region

int ff=CLobs(i,4); // Fleet type

int ll=CLobs(i,5); // Length class

//cout<<"y="<<yy<<" s="<<ss<<" r="<<rr<<" f="<<ff<<" l="<<ll<<" CLobs="<<CLobs(i,6)<<" CLpred="<<CLtotpred(yy,ss,rr,ff,ll)<<endl;

LHtemp=(-CLobs(i,6)\*log(CLtotfrac(yy,ss,rr,ff,ll)+tiny)); // Multinomial LHF

objCL+=LHtemp\*LHw(4); // Weighted likelihood contribution

objG+=LHtemp\*LHw(4); // Weighted likelihood contribution

}

if(debug)cout<<"--- \* Finished length composition LHF ---"<<endl;

// -- Stock of origin data --

dvariable SOOpred; // Calculated predicted fraction of each stock of origin (there aren't that many stock of origin observations so it involves less computation)

dvariable SOOtot; // For storing the sum of SOO numbers

SOOtot.initialize(); // SSOtot = 0

for(int i=1;i<=nSOOobs;i++){ // Loop over stock of origin observations

int pp=SOOobs(i,1); // Population

int yy=SOOobs(i,2); // Year

int ss=SOOobs(i,3); // Subyear

int rr=SOOobs(i,4); // Region

SOOtot=0.; // Reset sum

for(int pp=1;pp<=np;pp++){

SOOtot+=CNpred(pp,yy,ss,rr);

}

SOOpred=CNpred(pp,yy,ss,rr)/(SOOtot+tiny); // Calculate predicted fraction

LHtemp=(-SOOobs(i,5)\*log(SOOpred+tiny)); // Multinomial LHF

objSOO+=LHtemp\*LHw(5); // Weighted likelihood contribution

objG+=LHtemp\*LHw(5); // Weighted likelihood contribution

}

if(debug)cout<<"--- \* Finished SOO LHF ---"<<endl;

// -- PSAT tagging --

for(int i=1;i<=nPSAT;i++){ // Tags with certain stock of origin

int pp=PSAT(i,1); // Population

int ss=PSAT(i,2); // Subyear

int aa=PSAT(i,3); // Movement age class

int tt=PSAT(i,4); // Time at liberty (subyears)

int rr=PSAT(i,5); // Region from

int r2=PSAT(i,6); // Region to

LHtemp=(-PSAT(i,7)\*log(RecapP(pp,ss,aa,tt,rr,r2)+tiny)); // Multinomial LHF

objPSAT+=LHtemp\*LHw(6); // Weighted likelihood contribution

objG+=LHtemp\*LHw(6); // weighted likelihood contribution

}

if(debug)cout<<"--- \* Finished PSAT known SOO LHF ---"<<endl;

for(int i=1;i<=nPSAT2;i++){ // Individual tags with uncertain stock of origin

int ss=PSAT2(i,1); // Year

int aa=PSAT2(i,2); // Movement age class

int tt=PSAT2(i,3); // Subyear

int rr=PSAT2(i,4); // Region from

int r2=PSAT2(i,5); // Region to

for(int pp=1;pp<=np;pp++){

LHtemp=(-log(RecapP(pp,ss,aa,tt,rr,r2)+tiny)\*PSAT2(i,5+pp)); // Multinomial LHF

objPSAT2+=LHtemp\*LHw(7); // Weighted likelihood contribution

//objG+=LHtemp\*LHw(7); // Weighted likelihood contribution

}

}

if(debug)cout<<"--- \* Finished PSAT unknown SOO LHF ---"<<endl;

// -- Recruitment deviations --

for(int pp=1;pp<=np;pp++){ // Loop over stocks

/\*for(int aa=2;aa<=na;aa++){

LHtemp=dnorm(ilnRD(pp,aa),0.,RDCV); // Initial age-structure deviations (currently disabled)

objRD+=LHtemp\*LHw(8); // Weighted likelihood contribution

objG+=LHtemp\*LHw(8); // Weighted likelihood contribution

}\*/

for(int yy=1;yy<=nRD;yy++){ // Loop over years (or blocks of recruitment deviations if complexRD=0)

LHtemp=dnorm(lnRD(pp,yy),0.,RDCV); // Recruitment deviations

objRD+=LHtemp\*LHw(8); // Weighted likelihood contribution

objG+=LHtemp\*LHw(8); // Weighted likelihood contribution

}

}

// -- Movement parameters ---

for(int mm=1;mm<=nMP;mm++){

LHtemp=dnorm(mfexp(movest(mm)),-5.,5.); // Weak prior (very low movement)

objmov+=LHtemp\*LHw(9); // Weighted likelihood contribution

objG+=LHtemp\*LHw(9); // Weighted likelihood contribution

}

for(int i=1;i<=nsel;i++){

objG+=dnorm(selpar(i),0,2.)\*LHw(10); // Weak prior on selectivity

}

if(debug)cout<<"--- \* Finished rec dev penalty ---"<<endl;

//objG+=dnorm(lnF(1),log(0.1),2); // Temporary fix to allow F estimation to be simplied to a master index and partial F approach (complexF = 0)

if(debug)cout<<"--- Finished calcObjective ---"<<endl;

if(verbose)cout<<"Catch LHF "<<objC<<endl; // Report catch likelihood component

if(verbose)cout<<"CPUE LHF "<<objCPUE<<endl; // Report CPUE likelihood component

if(verbose)cout<<"FI index LHF "<<objI<<endl; // Report FI index likelihood component

if(verbose)cout<<"Length comp LHF "<<objCL<<endl; // Report catch at length likelihood component

if(verbose)cout<<"SOO LHF "<<objSOO<<endl; // Report stock of origin likelihood component

if(verbose)cout<<"PSAT LHF "<<objPSAT<<endl; // Report PSAT likelihood component

if(verbose)cout<<"PSAT uSOO LHF "<<objPSAT2<<endl; // Report PSAT2 likelihood component

if(verbose)cout<<"Rec dev LHF "<<objRD<<endl; // Report Rec dev likelihood component

if(verbose)cout<<"Global objective "<<objG<<endl; // Report Global objective function

}

FUNCTION simsam

{

// If working with simulated data do some printing

cout<<"R0 sim = "<<log(R0\_ini)<<endl; // Simulated R0

cout<<"R0 sam = "<<log(R0)<<endl; // Estimated R0

cout<<"sel sim f1= "<<sel\_ini(1)<<endl; // Simulated selectivity fleet 1

cout<<"sel sam f1= "<<sel(1)<<endl; // Estimated selectivity fleet 1

cout<<"sel sim f2= "<<sel\_ini(2)<<endl; // Simulated selectivity fleet 2

cout<<"sel sam f2= "<<sel(2)<<endl; // Estimated selectivity fleet 2

//cout<<"RDi sim= "<<exp(ilnRD\_ini(1))<<endl; // Simulated initial recruitment deviations

//cout<<"RDi sam= "<<iRD(1)<<endl; // Estimated initial recruitment deviations

cout<<"RD sim= "<<exp(lnRD\_ini(1))<<endl; // Simulated recruitment deviations

cout<<"RD sam= "<<RD(1)<<endl; // Estimated recruitment deviations

cout<<"mov sim p1 s1= "<<endl; // Simulated movement probabilities for stock 1 in subyear 1

cout<<mov\_ini(1)(1)<<endl; // Simulated movement probabilities for stock 1 in subyear 1

cout<<"mov sam p1 s1= "<<endl; // Estimated movement probabilities for stock 1 in subyear 1

cout<<mov(1)(1)<<endl; // Estimated movement probabilities for stock 1 in subyear 1

cout<<"mov sim p2 s1= "<<endl; // Simulated movement probabilities for stock 2 in subyear 1

cout<<mov\_ini(2)(1)<<endl; // Simulated movement probabilities for stock 2 in subyear 1

cout<<"mov sam p2 s1= "<<endl; // Estimated movement probabilities for stock 2 in subyear 1

cout<<mov(2)(1)<<endl; // Estimated movement probabilities for stock 2 in subyear 1

cout<<"qCE sim= "<<exp(lnqCPUE\_ini)<<endl; // Simulated catchabilities

cout<<"qCE sam= "<<qCPUE<<endl; // Estimated catchabilities

cout<<"qI sim= "<<exp(lnqI\_ini)<<endl; // Simulated FI index catchability

cout<<"qI sam= "<<qI<<endl; // Estimated FI index catchability

}

REPORT\_SECTION

{

report <<"np, number of stocks"<<endl;

report <<np<<endl;

report <<"ny, number of years"<<endl;

report <<ny<<endl;

report <<"ns, numnber of subyears"<<endl;

report <<ns<<endl;

report <<"nr, number of areas"<<endl;

report <<nr<<endl;

report <<"nf, number of fleets"<<endl;

report <<nf<<endl;

report <<"na, number of age classes"<<endl;

report <<na<<endl;

report <<"nl, number of length classes"<<endl;

report <<nl<<endl;

report <<"SSB (p,y,s) spawning stock biomass"<<endl;

report <<SSB<<endl;

report <<"Fishing mortality rate at length FL (y s r f l)"<<endl;

for(int yy=1;yy<=ny;yy++){

report <<FL(yy)<<endl;

}

report <<"nCobs"<<endl;

report <<nCobs<<endl;

report <<"Cobs (y-s-r-f) observed catches (weight)"<<endl;

report <<Cobs<<endl;

report <<"CWtotpred (y,s,r,f) predicted catches (weight)"<<endl;

for(int yy=1;yy<=ny;yy++){

report <<CWtotpred(yy)<<endl;

}

report <<"nCLobs"<<endl;

report <<nCLobs<<endl;

report <<"CLobs (y-s-r-f-l) observed catch at length (numbers)"<<endl;

report <<CLobs<<endl;

report <<"CLtotpred (y,s,r,f,l)"<<endl;

for(int yy=1;yy<=ny;yy++){

report <<CLtotpred(yy)<<endl;

}

report <<"mov (p,s,r,r) Markov movement matrix"<<endl;

report <<mov<<endl;

report <<"sel (f,l) selectivity by fleet and length class"<<endl;

report <<sel<<endl;

report<<"RAI (r,s,y)"<<endl;

report<<RAI<<endl;

report<<"ml (l) mean length of each length bin"<<endl;

report<<ml<<endl;

report<<"VB (y,s,r,f) vulnerable biomass by fleet"<<endl;

report<<VB<<endl;

report<<"B (y,s,r) biomass"<<endl;

report<<B<<endl;

report<<"N (p,y,s,a,r)"<<endl;

for(int pp=1;pp<=np;pp++){

report <<N(pp)<<endl;

}

report<<"lwa (p) length-weight paramter alpha"<<endl;

report<<lwa<<endl;

report<<"lwb (p) length-weight paramter beta"<<endl;

report<<lwb<<endl;

report<<"len\_age (y,a,p) length at age (pass through) used to calculate iALK and B independently"<<endl;

report<<len\_age<<endl;

report<<"wt\_age (y,a,p) weight at age (pass through) used to calculate iALK and B independently"<<endl;

report<<wt\_age<<endl;

report<<"nMP"<<endl;

report<< nMP <<endl; // Number of movement parameters estimated

report<<"nmovind"<<endl;

report<<nmovind<<endl;

report<<"movind (nmovind, 4)"<<endl;

report<<movind<<endl;

report<<"nmov1"<<endl;

report<<nmov1<<endl;

report<<"mov1(nmov1,4)"<<endl; // Index of first non-estimated movement parameter (fixed to zero)

report<<mov1<<endl;

report<<"movtype"<<endl; // 1: gravity, 2: markov

report<<movtype<<endl;

report<<"Ma (p,a)"<<endl;

report<<Ma<<endl;

report<<"steep (p)"<<endl;

report<<steep<<endl;

report<<"RDblock (y)"<<endl;

report<<RDblock<<endl;

report<<"Fec (p,a)"<<endl;

report<<Fec<<endl;

report<<"nsel number of estimated selectivities"<<endl;

report<<nsel<<endl;

report<<"seltype (nsel): logistic, 3: Thompson dome-shaped"<<endl;

report<<seltype<<endl;

report<<"selind (f) which selectivity is assigned to each fleet"<<endl;

report<<selind<<endl;

report<<"spawns (p) the subyear for spawning"<<endl;

report<<spawns<<endl;

report<<"ALK (p,y,a,l) inverse age-length key p(l|a)"<<endl;

for(int pp=1;pp<=np;pp++){

report <<ALK(pp)<<endl;

}

report<<"lnqCPUE (f) the estimated log qs"<<endl;

report<<lnqCPUE<<endl;

report<<"nZeq number of initial years to average over to get equilibrium Z"<<endl;

report<<nZeq<<endl;

report<<"nydist number of year iterations to get initial spatial distribution"<<endl;

report<<nZeq<<endl;

report<<"nyeq number of initial year iterations used to calculation initial stock size and distribution"<<endl;

report<<nyeq<<endl;

report<<"datacheck"<<endl;

report<<datacheck<<endl;

}

RUNTIME\_SECTION

maximum\_function\_evaluations 5000

convergence\_criteria 1.e-3

TOP\_OF\_MAIN\_SECTION

arrmblsize = 70000000;

gradient\_structure::set\_GRADSTACK\_BUFFER\_SIZE(1.e7);

gradient\_structure::set\_CMPDIF\_BUFFER\_SIZE(1.e7);

gradient\_structure::set\_MAX\_NVAR\_OFFSET(5000);

gradient\_structure::set\_NUM\_DEPENDENT\_VARIABLES(5000);

GLOBALS\_SECTION

#include <admodel.h>

#include "stats.cxx"

//#include <fstream>

//ofstream nodesout("nodes.cha");