Assessment Model for Alaska Description of GUI and Instructions

1.0 Introduction

1.1 Getting Started

The Assessment Model for Alaska (AMAK) provides a graphical interface to the AD Model Builder environment. AMAK is designed to simplify the formation of control and data input files and the creation of standard reports for an age structured stock assessment. AMAK is merely one tool to be used in the exploration of available data for a stock assessment. Make sure you understand how your data were collected and where model assumptions may be inaccurate. Each model assumption should have a biologically reasonable explanation that is supported with your research or is documented in the published literature. Understanding your data and model assumptions prior to entry into AMAK is essential to obtaining interpretable results.

As a professional biologist you should have many methods for exploring biological data. One method for exploring data that works well is to create an excel file containing your data and non-estimated parameters (You may wish to use separate workbooks for non-nested models where data input are different). Use Excel to become familiar with your data. Look for any outliers and recognizable patterns. The more familiar you are with your data, the better assessment you will be able to provide.

Design various model configurations in order to explore different aspects of your data.
You should know which parameters you wish to investigate and whether the data you have are sufficient to adequately explore these parameters. The assumptions made for each model configuration should be well understood and documented prior to running your models. Create a table of model configurations which explains how the model configurations differ from each other and what each model configuration is meant to accomplish. For example, in order to explore fishery selectivity at age you may wish to have a model with a coefficient model for selectivity at age with differing curvature penalties and still others with varying constraints on inter-annual variability.

Once you have a good understanding of your data and some well designed models, you are ready to begin using AMAK.

1.2 AD Model Builder Environment

1.3 Cautions

1.4 The model

AMAK employs an explicit age-structured model with the standard catch equation as the operational population dynamics model (e.g., Fournier and Archibald 1982, Hilborn and Walters 1992, Schnute and Richards 1995). The model components are defined in Tables 1-3 below.

Table 1. Variable descriptions and model specification.

General Definitions	**Symbol/Value**	**Use in Catch at Age Model**
Year index: $i = \{i_f, \ldots., i_l\}$	i	
Number of years	n_y	
First age class in age index	A_f	
Last age class in the age index	A_l	
Age index: $j = \{ A_f, A_f+1, \ldots, A_l-1, A_l\}$	j	
Number of age catagories	n_A	
Mean weight by age *j*	W_j	
Maximum age beyond which selectivity is constant in the selectivity coeficient form	*Maxage*	Selectivity parameterization
Instantaneous Natural Mortality	M	Prior distribution = lognormal (0.3, 0.6^2)
Proportion females mature at age *j*	p_j	Definition of spawning biomass
Sample size for proportion at age *j* in year *i*	T_i	Scales multinomial assumption about estimates of proportion at age
Survey catchability coefficient	q_s	Prior distribution = lognormal (1.0 ,0.2^2)
Stock-recruitment parameters	R_0	Unfished equilibrium recruitment
	h	Stock-recruitment steepness
	σ_R^2	Stock-recruitment variance

Table 2. Variables and equations describing implementation of AMAK

Description	Symbol/Constraints	Key Equation
Survey Abundance index (*s*) by Year	Y_i^s	$\hat{Y}_i^s = q_i^s \sum_{j=1}^{A_l} s_i^s W_{ij} N_{ij}$
Catch biomass by year	C_i	$C_i = \sum_j W_{ij} N_{ij} \frac{F_{ij}}{Z_{ij}} \left(1 - e^{-Z_{ij}}\right)$
Proportion at age *j*, in year *i*	$P_{ij}, \sum_{j=1}^{A_l} P_{ij} = 1.0$	$P_{ij} = \frac{N_{ij} s_{ij}^f}{\sum_{k=1}^{A_l} N_{ik} s_{ik}^f}$
Initial numbers at age ($i = i_1$)	$j = A_f$	$N_{i_1, A_f} = e^{\mu_{R_{i_1}} + \varepsilon_{i_1}}$
	$A_f < j < A_l$	$N_{i_1, j} = e^{\mu_{R_{i_{2-j}}} + \varepsilon_{i_{2-j}}} \prod_{j=1}^{j} e^{-M}$
	$j = A_l$	$N_{i_1, A_l} = N_{i_1, A_l - 1} \left(1 - e^{-M}\right)^{-1}$
Subsequent years ($i > i_1$)	$j = A_f$	$N_{i, A_f} = \frac{S_{i-1} e^{\varepsilon_i}}{\alpha + \beta S_{i-1, A_f}}$
	$A_f < j < A_l$	$N_{i,j} = N_{i-1, j-1} e^{-Z_{i-1, j-1}}$
	$j = A_l$	$N_{i, A_l} = N_{i-1, A_l - 1} e^{-Z_{i-1, A_l - 1}} + N_{i-1, A_l} e^{-Z_{i-1, A_l}}$
Index catchability	μ^s	$q_i^s = e^{\mu^s}$
Instantaneous fishing mortality		$F_{ij} = e^{\mu_f + \eta_j^f + \phi_i}$
Mean fishing effect	μ_f	
Annual effect of fishing in year *i*	$\phi_i, \sum_{i=i_1}^{i_l} \phi = 0$	
Natural Mortality	M	
Total Mortality	Z_{ij}	$Z_{ij} = F_{ij} + M$
Recruitment	μ_{R_i}	

Table 3. Specification of objective function that is minimized (i.e., the penalized negative of the log-likelihood).

Likelihood/penalty component	**Description**	**Notes**
Abundance indices	$L_1 = \lambda_1 \sum_i \left(Y_i^s - \hat{Y}_i^s\right)^2 \frac{1}{2\sigma_i^2}$	Survey abundance
Smoother for selectivities in the selectivity coefficient model	$L_2 = \sum_i \lambda_{22}^l \sum_{j=A_f}^{A_l} \left(\eta_{J+2}^l + \eta_J^l - 2\eta_{J+1}^l\right)$	Smoothness (second differencing), Note: l={s, or f} for suveys and fishery selectivity
Recruitement Regularity	$L_3 = \lambda_3 \sum_{i=i_f}^{i_l} \varepsilon_i^2 \frac{1}{2\sigma_R^2}$	Influences estimates where data are lacking (e.g., if no signal of recruitment strength is available, then the recruitment estimate will converge to median value).
Catch Biomass likelihood	$L_4 = \lambda_4 \sum_{i=i_f}^{i} \ln\left(C_i / \hat{C}_i\right)^2$	
Proportion at age likelihood	$L_5 = \sum_{l,i,j} T_{ij}^l P_{ij}^l \ln\left(\hat{P}_{ij}^l \cdot P \quad \right)$	l={s, f} for survey and fishery age composition observations
Fishig mortality regularity	$L_6 = \lambda_6 \sum_{i=i_f}^{i_l} \phi_i^2$	(relaxed in final phases of estimation)
Priors	$L_7 = \left[\lambda_7 \frac{\ln\left(M / \hat{M}\right)^2}{2 \cdot 0.05} + \lambda_8 \frac{\ln\left(q / \hat{q}\right)^2}{2 \cdot 0.05} \right]$	Prior on natural mortality, and survey catchability
Overall objective function to be minimized	$\dot{L} = \sum_{i=1}^{7} L_i$	

2.0 Control File

2.1 General Parameters Tab

The General Parameters Tab allows user to specify the control file, model notes, and number of projection years. Figure 1 below shows the General Parameters tab with the ModelA10A control and data files selected, "Model A10_A run 1" noted, and projection for five years specified.

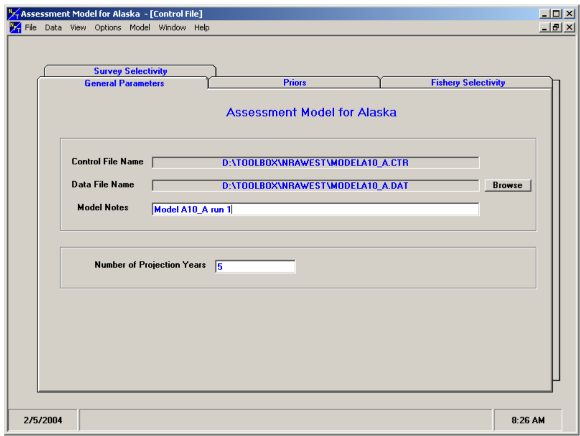


Figure 1 General Parameters tab

2.2 Priors Tab

The Priors tab allows the user to specify the recruitment type, recruitment parameters (steepness and Sigma R), natural mortality, catchability, the precision of the catch biomass estimate, and the years for which the stock recruitment curve will be fitted. It also allows the user to specify in which phase of the model run the parameter will be estimated (2 through 6). A positive phase indicates that the parameter will be used as a prior with the specified precision (CV), while a negative value indicates that the parameter is fixed at the entered value. Estimates of catchability (Q) can also be limited to a range of age classes. Figure 2 below shows a model run with a Beverton-Holt recruitment model selected, a fixed steepness parameter of 0.7, a fixed Sigma R value of 0.5, a prior on natural mortality of 0.3 with a CV of 0.1, catchability parameter (Q) specified at 1.0 for ages 4 to 12, the precision of the catch biomass estimates is set at a CV of 0.05, and the years for fitting the stock recruitment curve is specified as between 1983 and 2003.

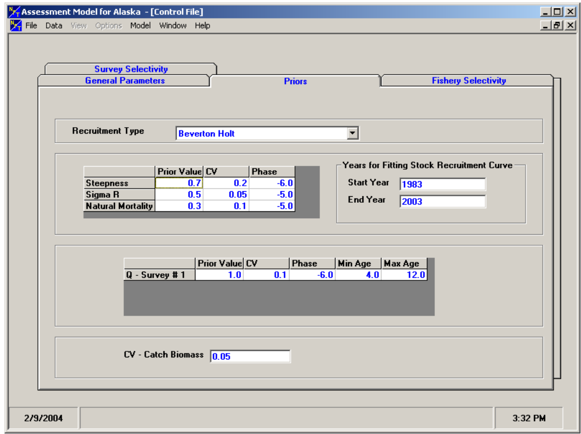


Figure 2 Priors tab

2.2.1 Recruitment

AMAK allows the user to select between three recruitment types: Beverton-Holt, Ricker, and Average.

The toolbox follows the stock-recruitment relationship as by Francis (1992). For the Beverton-Holt form we have:

$$R_i = f(B_{i-1}) = \frac{B_{i-1}e^{\varepsilon_i}}{\alpha + \beta B_{i-1}}$$

where

R_i is recruitment at age 1 in year *i*,

B_i is the biomass of mature spawning females in year *i*,

ε_i is the "recruitment anomaly" for year *i* and $\varepsilon_i \sim N\left(0, \sigma_R^2\right)$,

α, β are stock-recruitment function parameters.

Values for the stock-recruitment function parameters α and β are calculated from the values of R_0 (the number of 0-year-olds in the absence of exploitation and recruitment variability) and the "steepness" of the stock-recruit relationship (*h*). The "steepness" is the fraction of R_0 to be expected (in the absence of recruitment variability) when the mature biomass is reduced to 20% of its pristine level (Francis 1992), so that:

$$\alpha = \frac{\tilde{B}_0\left(\dfrac{1-(h-0.2)}{0.8h}\right)}{R_0}$$

$$\beta = \frac{5h-1}{4hR_0}$$

where

$\tilde{B}_0$ is the total egg production (or proxy, e.g., female spawner biomass) in the absence of exploitation (and recruitment variability) expressed as a fraction of R_0.

Some interpretation and further explanation follows. For steepness equal 0.2, then recruits are a linear function of spawning biomass (implying no surplus production). For steepness equal to 1.0, then recruitment is constant for all levels of spawning stock size. A value of $h = 0.9$ implies that at 20% of the unfished spawning stock size will result in an expected value of 90% unfished recruitment level. Steepness of 0.7 is a commonly assumed default value for the Beverton-Holt form (e.g., Kimura 1988). The same prior distribution for steepness based on a beta distribution as in Ianelli et al. (2001).

To have the critical value for the stock-recruitment function (steepness, h) on the same scale for the Ricker model, we begin with the parameterization of Kimura (1990):

$$R_i = f(B_{i-1}) = \frac{B_{i-1} e^{a\left(1-\frac{B_{i-1}}{\tilde{B}_0}\right)+\varepsilon_i}}{\phi_0}$$

It can be shown that the Ricker parameter a maps to steepness as:

$$h = \frac{e^a}{e^a + 4}\text{, and therefore}\quad \alpha = \ln\left(\frac{-4h}{h-1}\right)$$

so that the prior used on h can be implemented in both the Ricker and Beverton-Holt stock-recruitment forms. Here the term ϕ_0 represents the equilibrium unfished spawning biomass per-recruit.

$$\phi_0 = \frac{\tilde{B}_0}{R_0}$$

The average model is simply a stochastic process about the (geometric) mean recruitment (μ_R) for all specified years. In this case the μ_R is an independently estimated parameter.

$$R_i = \mu_R e^{\varepsilon_i}$$

2.2.2 Survey Selectivity

The Survey Selectivity tab allows the user to specify survey selectivity parameters including selectivity type and parameters for fitting survey selectivity. The user can select between three selectivity models for each survey: selectivity coefficient, logistic selectivity, and double logistic selectivity.

<u>Selectivity Coefficient</u>

In the survey selectivity coefficient model the selectivity relationship is modeled with a smoothed non-parametric relationship that can take on any shape (with penalties controlling the degree of change and curvature specified by the user). Selectivity is conditioned so that the mean value over all ages will be equal to one. Selectivity can be allowed to vary annually in a random walk process with specified levels of constraint per year by specifying the year-specific coefficient of variation of the process error term.

$s_j^s = e^{\eta_j^s}$, where $j \le$ *maxage*, and

$s_j^s = e^{\eta_{\max age}^s}$, where $j >$ *maxage* and $\eta_j^s, \sum_{j=1}^{A_l} \eta_j^s = 0$

Figure 3 shows a selectivity coefficients model configured such that the curve asymptotes after 12 years, the selectivity model is fit in phase 4, there is a curvature penalty with a CV of 0.2, and a decrease selectivity CV of 200 (This setting means that there is no effective penalty on a decrease).

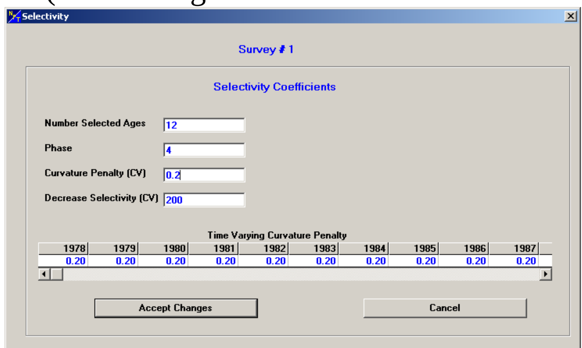


Figure 3 Selectivity Coefficients model parameters for Survey #1

<u>Logistic</u>

A two parameter logistic equation is used;

$$s_j^{'} = \left(\frac{1}{1 + e^{-\beta_1 (j - \alpha_1)}} \right)$$

$$s_j = s_j^{'} / \max_j \left(s_j^{'} \right)$$

where a_1 = inflection age, β_1 = slope at the inflection age. Selectivity can be allowed to vary annually in a random walk process with specified levels of constraint per year by specifying the year-specific coefficient of variation of the process error term on both estimated parameters. Figure 4 shows an example of the AMAK logistic selectivity model specifications where the parameters are estimated in phase 4 with priors on the slope (β_1) of 0.4 and inflection age (a_1) of 9. The logistic model parameters are allowed to vary every year within a CV of 0.2 as specifice in the "Time Varying Selectivity Specification" fields.

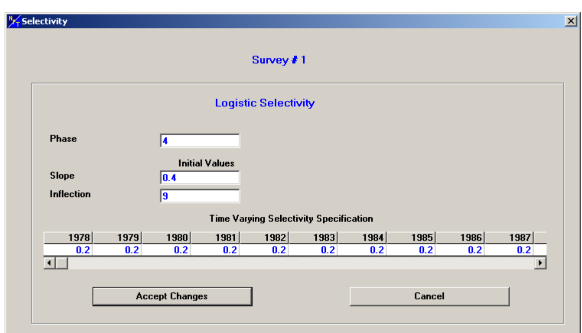


Figure 4 Logistic Selectivity model parameters for Survey #1

Double Logistic

A four parameter double logistic equation is used;

$$s_j' = \left(\frac{1}{1+e^{-\beta_1(j-\alpha_1)}}\right)\left(1-\frac{1}{1+e^{-\beta_2(j-\alpha_2)}}\right)$$
$$s_j = s_j' / \max_j \left(s_j'\right)$$

where a_1 = inflection age, β_1 = slope at the inflection age fro the ascending logistic part of the equation, and a_2 , β_2 = the inflection age and slopw of the descending logistic part. Selectivity can be allowed to vary annually in a random walk process with specified levels of constraint per year by specifying the year-specific coefficient of variation of the process error term on all four estimated parameters. Figure 5 shows an example of the AMAK double logistic selectivity model specifications where the parameters are estimated in phase 4 with priors on the ascending slope (β_1) of 0.2, an ascending inflection age (a_1) of 6, a descending slope (β_2) of 0.9 and a descending inflection age (a_2) of 11. The parameters are allowed to vary every year within a CV of 0.20 as specified in the "Time Varying Selectivity Specification" fields.

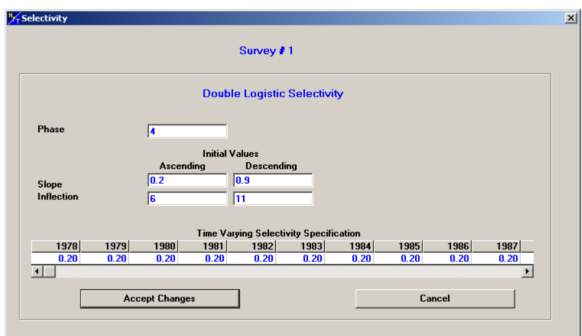


Figure 5 Double Logistic Selectivity model parameters for Survey #1

2.2.3 Fishery Selectivity

Similar to the Survey Selectivity tab, the Fishery Selectivity tab allows the user to specify fishery selectivity parameters including selectivity type and the parameters for fitting fishery selectivity to the data. The user can select between three selectivity models for each fishery: selectivity coefficient, logistic selectivity, and double logistic selectivity. The parameterization of the logistic and double logistic models is described above and is the same as that of the Survey Selectivity models.

Selectivity Coefficient

The Fishery selectivity coefficient model is modeled with a smoothed non-parametric relationship that can take on any shape (with penalties controlling the degree of change and curvature specified by the user). Selectivity is conditioned so that the mean value over all ages will be equal to one. Selectivity can be allowed to vary annually in a random walk process with specified levels of constraint per year by specifying the year-specific coefficient of variation of the process error term.

This mean that fishery selectivity in year *i* and at age *j* is $s_{ij}^{f} = e^{\eta_j^f}$ where j ≤ maxage, and

$s_{ij}^{f} = e^{\eta_{\max age}^f}$ where j > maxage. η_{ij}^{f}, $\sum_{j=A_f}^{A_l} \eta_{ij} = 0$ in years when time variation is allowed, and

$\eta_{i,j}^{f} = \eta_{i-1,j}^{f}$ in years when selectivity is constant over time ($i \neq$ change year).

3.0 Input Data

The input data section is the part of the GUI application that allows the user to create the .dat file for AD-model builder. The numbers of survey and fishery estimates of catch, biomass, as well as biological attributes, can be specified and historic data can be entered into the model here.

3.1 General Parameters

The General Parameters tab allows the user to enter the first and last year for which data are available, the first and last age classes to be used in the model estimations, the number of fisheries, and the number of survey time series available. Once these are set by clicking on the “Set” button the user may enter the number years for which there are age composition data available per fishery and the number of years for which there are indices of biomass and age composition data available for each survey time series. In Figure 6 the first and last year for which data are available are 1978 and 2003 respectively, the first and last age classes are 1 and 15 respectively, and there is 1 fishery and 1 survey time series specified. In the “Number of Years With Data” field in Figure 3 there are 8 years of fishery age composition (Fishery # 1 - Age Composition Data), 5 years of survey indices of biomass (Survey # 1 - Estimated Data) and 3 years of age composition data available for the survey (Survey # 1 - Age Composition Data).

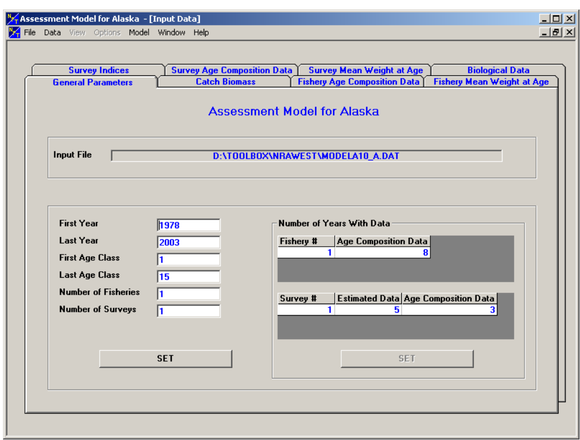


Figure 6 General Parameters tab

3.2 Survey Indices

The survey indices tab allows users to enter data on the survey indices by year. There are four fields required for each survey indices; Year, Month, Estimate and STD Dev. The program creates a row for each “estimated data” specified on the General Parameters Tab. The Year is the year of the survey, Month is the month the survey was conducted (if the survey was conducted over more than one month use the mean month), Estimate is the biomass index (this is a weight estimate and the units must agree with the catch estimates on the Catch Estimates tab), and ST Dev is the standard deviation of the survey biomass estimate (units must be the same as the Estimate units). Figure 7 demonstrates entries of biomass estimates for surveys from 1991 through 2002.

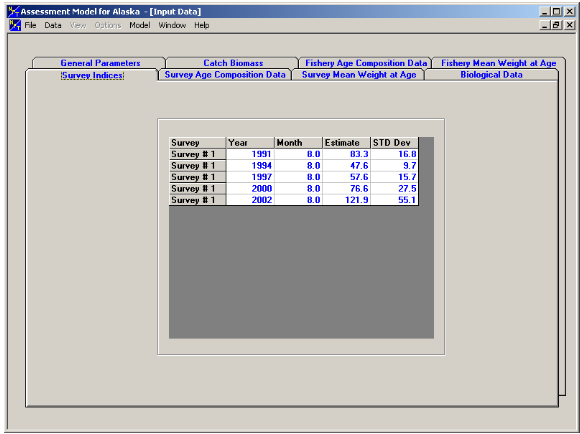


Figure 7 Survey Indices tab

3.3 Survey Age Composition Data

The Survey Age Composition Data tab allows users to enter age composition data by year and age class for each survey where age composition data is available. There are fields for year, sample size, and proportion of biomass per year for each age. The sum of all age proportions should add to one for each year. Since the survey age composition is fitted as a multinomial distribution a sample size is required for each year. The sample size determines the influence the estimate will have on the model fit. Figure 8 illustrates data entry for three years worth of survey age composition estimates, age classes greater than age 4 cannot be seen in this figure, but are accessible in the application by using the scroll bars on the bottom of the tab.

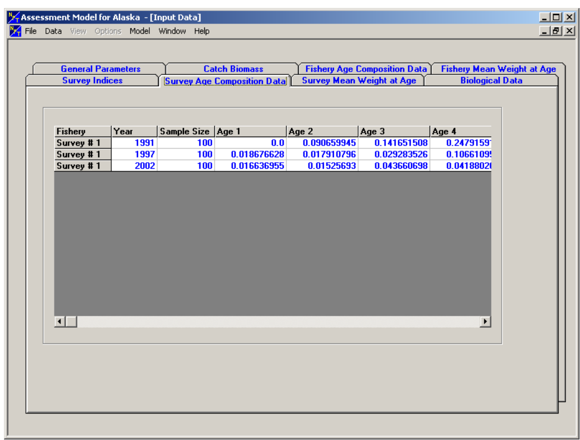


Figure 8 Survey Age Composition tab

3.4 Survey Mean Weight at Age

The Survey Mean Weight at Age tab allows users to enter estimates of mean weight at age in kilograms for every year from the first to the last year and every age class from the first to the last age class as specified on the General Parameters tab. These data are required and every year and age class must have an entry in order for the model to function properly. Figure 9 illustrated the Survey Mean weight at age tab.

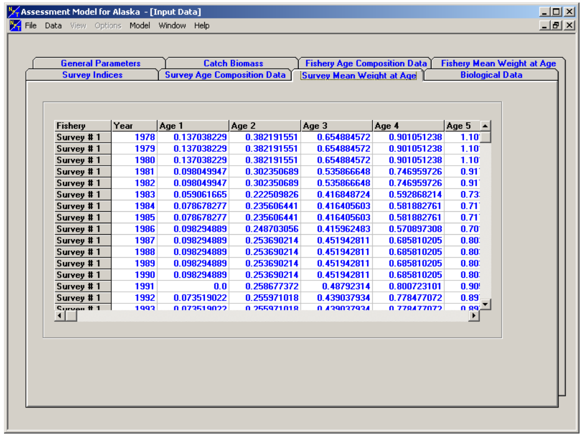


Figure 9 Survey Mean Weight at Age Tab

3.5 Catch Biomass

The Catch Biomass tab allows users to enter catch biomass estimates from each fishery specified on the General Parameters tab by year. An estimate must be entered for each year from the first to the last year. The units used must be the same as that used for the survey indices. Figure 10 illustrates data entry on the Catch Biomass tab.

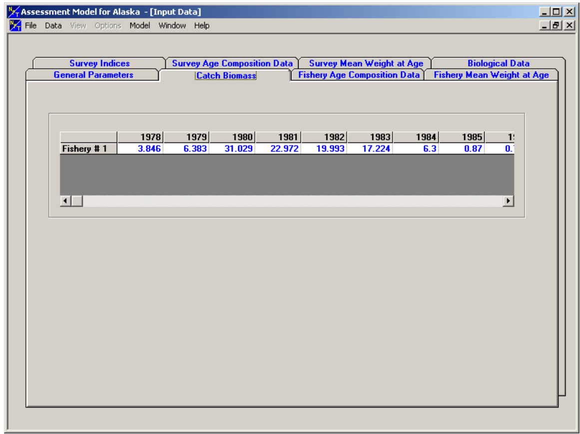


Figure 10 Catch Biomass Tab

3.6 Fishery Age Composition Data

The Fishery Age Composition Data tab allows users to enter age composition data by year and age class for each fishery where age composition data is available. There are fields for year, sample size, and proportion of biomass per year for each age. The sum of all age proportions should add to one for each year. Since the fishery age composition is fitted as a multinomial distribution a sample size is required for each year. The sample size determines the influence the estimate will have on the model fit. Figure 11 illustrates data entry for three years worth of survey age composition estimates, age classes greater than age 4 cannot be seen in this figure, but are accessible in the application by using the scroll bars on the bottom of the tab.

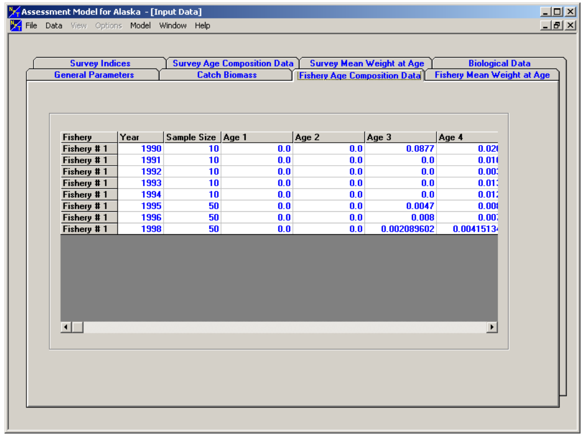


Figure 11 Fishery Age Composition Tab

3.7 Fishery Mean Weight at Age

The Fishery Mean Weight at Age tab allows users to enter estimates of mean weight at age in kilograms for each fishery for every year from the first to the last year and every age class from the first to the last age class as specified on the General Parameters tab. These data are required and every year and age class must have an entry in order for the model to function properly. Figure 12 illustrated the Fishery Mean weight at age tab.

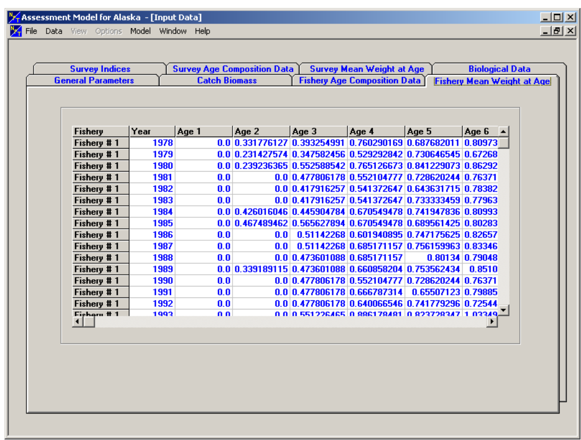


Figure 12 Fishery Mean Weight at Age Tab

3.8 Biological Data

The Biological Data tab allows users to enter the population weight at age, maturity at age, and the month of spawning. Weight at age must be entered in kilograms. Maturity is the percentage of mature adults for the age class. If the population spawns over more than one month then the month of spawning is the mean month of spawning. Figure 13 shows a completed Biological data tab.

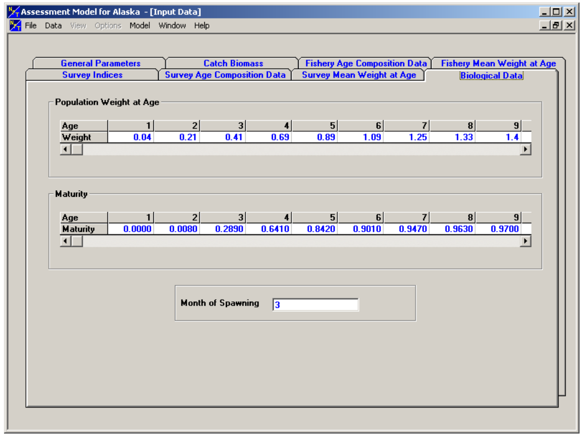


Figure 13 Biological Data Tab

4.0 Estimating Model Parameters

Once all data are entered and priors are specified users may estimate model parameters by selecting "Estimate Model Parameters" on the **Model** dropdown list in the main menu bar. Figure 14 shows the "Estimate Model Parameters" dropdown selected.

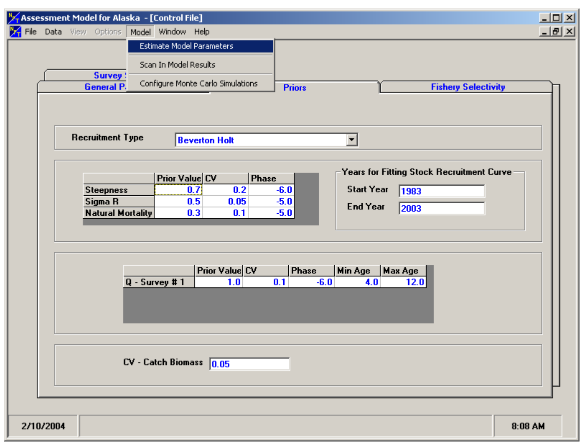


Figure 14 Model dropdown list showing "Estimate Model Properties" selected

The AMAK program will then inform users that the Input files (.dat and .CTR files) will be overwritten. To change the name of the files and avoid overwriting old files users should select cancel. Because AMAK makes it simple to do many models runs and each run produces 11 separate files, it is easy to lose or confuse files. It is advisable to name each model run with an easily remembered name and create a filing system for easy retrieval. All of the various report files will have the same name as the Control file.

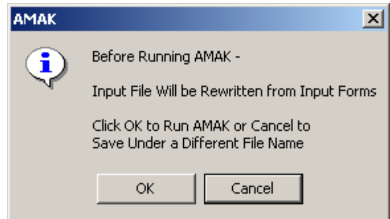


Figure 15 Message Box prior to model run.

Once the model run is started a command prompt window will appear and users can monitor the model run. The command prompt window shows the standard AD-model builder run messages. Once the run is completed the Output Data section of AMAK will be shown. Unsuccessful runs will result in one of two error messages. The "Standard Deviation File is Unavailable" (Fig 16) message means that the run was completed but because of a problem with the configuration some part of the precision estimates for model results could not be estimated. All of the canned reports will be visible, but may not be accurate and may show results from the previous model run. The second message "Report File is Unavailable" (Fig. 16) means that the run was aborted and that none of the reports are available. This can be the result of many different errors. Users should watch the command prompt window during the model run to determine when the error occurred and what caused the error.

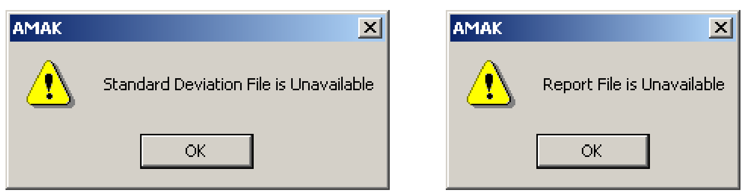


Figure 16 Standard error messages in AMAK

5.0 Results

Results are formatted into 4 report text files and 21 charts. Users can view the 4 report text files by selecting the View menu option on the main menu. Users can view the charts by selecting various tabs in the main window. Table 4 lists the name and description of the report; Table 5 lists the name and description of the available charts.

Table 4. Report text file names and descriptions.

Report Name	Description
Report File	The report file provides users with all of the relevant data from a model run including a detailed description of likelihoods and penalties.
Likelihood Summary	The likelihood summary is a short report providing a detailed report on likelihoods and penalties and the general fit of the model.
Standard Deviation File	The standard deviation file provides a list of estimated parameters with estimates of standard deviation.
Correlation File	Provides estimates of correlation between estimated parameters.

Table 5. Chart tab names and descriptions

Chart Tab Name	**Description**
Numbers at Age	A bar chart of numbers at age by year, users may select which years they wish to display.
Survey Age Composition	A line and point chart of the fit of observed to predicted survey age composition by survey year.
Survey Selectivity	A line chart of survey selectivity by age class, users can choose between views of individual years and surveys.
Total Biomass	A line chart displaying total biomass with upper and lower confidence bounds.
Yield Curve	A two Y axis line chart showing yield to fishing mortality and stock size to fishing mortality curves.
F vs. SSB	A point chart plotting fishing mortality versus spawning stock biomass.
Age Specific F	A point chart plotting fishing mortality by year, users may select which age class to display.
Catch Biomass	A point and line chart comparing observed and predicted catch biomass.
Stock-Recruitment	A point and line chart of the recruitment curve and estimated recruitment.
Recruits	A point chart with estimated recruitment values with confidence bounds plotted by year.
Spawning Biomass Plot	A line chart displaying spawning stock biomass by year.
Survey Indices	A point and line chart comparing observed and predicted survey biomass estimates.
Fishing Mortality	A point chart plotting fishing mortality by year, users may select between average fishing mortality and full selection fishing mortality.
Selectivity – 3D	A 3-D area plot of the selectivity curve of all modeled years, users may choose to view any single fishery or survey.
F Reference Points	A point plot with confidence bounds with Fmsy, F_{50}, F_{40} and F_{35} plotted by fishing mortality (F).
Yield Projections	A line chart of Fmsy, F_{50}, F_{40} and F_{35} plotted by catch and projected year.

Table 5. Chart tab names and descriptions

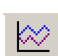
Fishery Proportion at Age	A point and line chart of observed and predicted proportion of catch at age plotted by age class, users choose to view charts between years and fisheries.
Fishery Selectivity	A line chart of fishery selectivity by age class, users can choose between views of individual years and fisheries.
Age Composition Residuals	A 3-D chart of standardized residuals from fits to age composition data plotted by age class and year, users can choose to view any individual survey or fishery.
Reference Point Ratios	A point plot with confidence bounds for reference point ratios of B_{msy}/B_0, $F_{endyear}/F_{msy}$, and $B_{endyear}/B_{msy}$.
SSB Projections	A line plot of spawning stock biomass projections by year by F_0, Fmsy, F_{50}, F_{40} and F_{35} scenarios.

5.1 Chart Toolbar Configuration

All of the charts have the same toolbar. Figure 17 shows the toolbar configuration and Table 5 provides a brief description of each menu item.

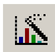
Figure 17 Standard toolbar for all charts in AMAK

Table 5. Toolbar menu items and descriptions.

Menu Item	Description
	Open Chart – allows users to open saved charts.
	Save Chart – allows users to save the current chart in a number of formats.
	Copy to Clipboard – allows users to copy the current chart to the clipboard for use in other documents.
	Gallery – opens a gallery of other chart types from which users can choose. 
	Color – Allows users to select from a variety of colors for the design of a chart. 
	Vertical Grid – Allows users to add vertical gridlines to a chart
	Horizontal Grid – Allows users to add horizontal gridlines to a chart.


Legend Box – Allows users to add or remove a legend to a chart.


Data Editor – Allows users to view and edit data from which a chart is created.


Properties – Allows users to edit various properties of a chart.


3D/2D – Allows users to toggle between 2-dimensional and 3-dimensional chart types.


Rotate – Allows users to rotate 3-dimensional charts on two axes.

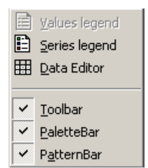

Z-Cluster – Allows users to ….


Zoom – Allows users to zoom in on any part of a chart and re-display only those data selected.


Print Preview – Allows users to preview how a chart will look when printed.

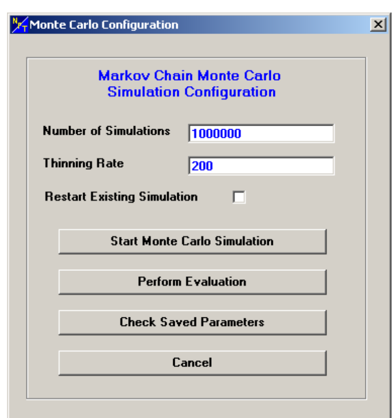

Print – Sends the current chart to a selected printer.


Tools – Allows the user to select from various tools. The Palette and pattern bars are selections for changing the color and patterns of the chart elements. The series legend toggles on and off the chart legend and data editor is the same as that listed above.



7.0 Markov Chain Monte Carlo Simulations

Conduct MCMC analysis of the estimated parameters.



8.0 ADMB Code

```
/////////////////////////////////////////////////////////////////////
// Generic model
// Alaska Fisheries Toolbox Assessment Model
//  GENERAL:
```

```
//    styr, endyr begining year and ending year of model (catch data available)
//    nages       number of age groups considered
//    nyrs_        number of observations available to specific data set
//
//  DATA SPECIFIC:

//    catch_bio   Observed catch biomass
//    fsh        fishery data
//
//  Define indices
//    nind        number of indices
//  Index values
//    nyrs_ind      Number of years of index value (annual)
//    yrs_ind        Years of index value (annual)
//    obs_ind        Observed index value (annual)
//    obs_se_ind    Observed index standard errors (annual)
//  Age-comp values
//    nyrs_ind_age  Number of years index age data available
//    yrs_ind_age   Years of index age value (annual)
//    oac_ind       Observed age comp from index
//    n_sample_ind_age    Observed age comp sample sizes from index
//
//    eac_ind       Expected age comp from index
//
//    sel_ind       selectivity for egg production index
//
//    pred_ind ...
//
//    oac_fsh      Observed age comp from index
//    obs_ind_size  Observed size comp from index
//
//    pred_fsh_age    Predicted age comp from fishery data
//    eac_fsh            Expected age comp for fishery data (only years where data available)
//    eac_ ...
//
//    pred_tmp_ind   Predicted index value for trawl index
//
//    sel_fsh    selectivity for fishery
//
//     sel_ch indicates time-varying selectivity change
//
//    Add bit for historical F
//
//////////////////////////////////////////////////////////////////////////////
 // To ADD/FIX:
 //   parameterization of steepness to work the same (wrt prior) for ricker and bholt
 //   splines for selectivity
 //   two projection outputs need consolidation
//////////////////////////////////////////////////////////////////////////////


DATA_SECTION
  !!version_info+="Amak.1;July_2012";
  int iseed
  !! iseed=1313;
  int cmp_no // candidate management procedure
  int nnodes_tmp;
  !!CLASS ofstream mceval("mceval.dat")
  !!long int lseed=iseed;
  !!CLASS random_number_generator rng(iseed);

  int oper_mod
  int mcmcmode
  int mcflag

  !! oper_mod = 0;
  !! mcmcmode = 0;
  !! mcflag   = 1;
 LOCAL_CALCS
  write_input_log<<version_info<<endl;
  tmpstring=adprogram_name + adstring(".dat");
```

```
  int on=0;
  if ( (on=option_match(argc,argv,"-ind"))>-1)
  {
    if (on>argc-2 | argv[on+1][0] == '-')
    {
      cerr << "Invalid input data command line option"
         " -- ignored" << endl;
    }
    else
    {
      cntrlfile_name = adstring(argv[on+1]);
    }
  }
  else
  {
      cntrlfile_name =   tmpstring;
  }
  if ( (on=option_match(argc,argv,"-om"))>-1)
  {
    oper_mod  = 1;
    cmp_no = atoi(argv[on+1]);
    cout<<"Got to operating model option "<<oper_mod<<endl;
  }
  if ( (on=option_match(argc,argv,"-mcmc"))>-1)
  {
    mcmcmode = 1;
  }
  global_datafile= new cifstream(cntrlfile_name);
  if (!global_datafile)
  {
  }
  else
  {
    if (!(*global_datafile))
    {
      delete global_datafile;
      global_datafile=NULL;
    }
  }
 END_CALCS
 // Read in "name" of this model...
  !! *(ad_comm::global_datafile) >>  datafile_name; // First line is datafile (not used by this
executable)
  !! *(ad_comm::global_datafile) >>  model_name;
  !! ad_comm::change_datafile_name(datafile_name);
  init_int styr
  init_int endyr
  init_int rec_age
  init_int oldest_age
  int nages
  !!  nages = oldest_age - rec_age + 1;
  int styr_rec
  int styr_sp
  int endyr_sp
  int nyrs
  !! nyrs          = endyr - styr + 1;
  int mc_count;
  !!  mc_count=0;
  !! styr_rec = (styr - nages) + 1;     // First year of recruitment
  !! styr_sp  = styr_rec - rec_age ;  // First year of spawning biomass
  vector yy(styr,endyr);
  !! yy.fill_seqadd(styr,1) ;
  vector aa(1,nages);
  !! aa.fill_seqadd(rec_age,1) ;
  int junk;
// Fishery specifics
  init_int nfsh                                   //Number of fisheries
  imatrix pfshname(1,nfsh,1,2)
  init_adstring fshnameread;
 LOCAL_CALCS
  for(k=1;k<=nfsh;k++)
```

```
  {
    pfshname(k,1)=1;
    pfshname(k,2)=1;
  }    // set whole array to equal 1 in case not enough names are read
  adstring_array CRLF;   // blank to terminate lines
  CRLF+="";
  k=1;
  for(i=1;i<=strlen(fshnameread);i++)
  if(adstring(fshnameread(i))==adstring("%")) {
    pfshname(k,2)=i-1;
    k++;
    pfshname(k,1)=i+1;
  }
  pfshname(nfsh,2)=strlen(fshnameread);
  for(k=1;k<=nfsh;k++)
  {
    fshname += fshnameread(pfshname(k,1),pfshname(k,2))+CRLF(1);
  }
  log_input(datafile_name);
  log_input(model_name);
  log_input(styr);
  log_input(endyr);
  log_input(rec_age);
  log_input(oldest_age);
  log_input(nfsh);
  log_input(fshname);
 END_CALCS
  init_matrix catch_bio_in(1,nfsh,styr,endyr)
  init_matrix catch_bio_sd_in(1,nfsh,styr,endyr)   // Specify catch-estimation precision
  // !! for (i=1;i<=nfsh;i++) catch_bio(i) += .01;
  !! log_input(catch_bio_in);
  !! log_input(catch_bio_sd_in);


//  Define fishery age compositions
  init_ivector nyrs_fsh_age(1,nfsh)
  init_imatrix yrs_fsh_age_in(1,nfsh,1,nyrs_fsh_age)
  init_matrix n_sample_fsh_age_in(1,nfsh,1,nyrs_fsh_age)    //Years of index index value (annual)
  init_3darray oac_fsh_in(1,nfsh,1,nyrs_fsh_age,1,nages)
  init_3darray wt_fsh(1,nfsh,styr,endyr,1,nages)  //values of weights at age

//  Define indices
  init_int nind                                  //number of indices
  !! log_input(nind);
  int nfsh_and_ind
  !! nfsh_and_ind = nfsh+nind;
  imatrix pindname(1,nind,1,2)
  init_adstring indnameread;
 LOCAL_CALCS
  for(int k=1;k<=nind;k++)
  {
    pindname(k,1)=1;
    pindname(k,2)=1;
  }    // set whole array to equal 1 in case not enough names are read
  int k=1;
  for(i=1;i<=strlen(indnameread);i++)
  if(adstring(indnameread(i))==adstring("%")) {
    pindname(k,2)=i-1;
    k++;
    pindname(k,1)=i+1;
  }
  pindname(nind,2)=strlen(indnameread);
  for(k=1;k<=nind;k++)
  {
    indname += indnameread(pindname(k,1),pindname(k,2))+CRLF(1);
  }
  log_input(indname);
 END_CALCS

//  Index values
  init_ivector nyrs_ind(1,nind)                  //Number of years of index value (annual)
```

```
  init_imatrix yrs_ind_in(1,nind,1,nyrs_ind)         //Years of index value (annual)
  init_vector mo_ind(1,nind)                      //Month occur
  init_matrix obs_ind_in(1,nind,1,nyrs_ind)          //values of index value (annual)
  init_matrix obs_se_ind_in(1,nind,1,nyrs_ind)       //values of indices serrs

  vector ind_month_frac(1,nind)
  !! log_input(nyrs_ind);
  !! log_input(yrs_ind_in);
  !! log_input(mo_ind);
  !! ind_month_frac = (mo_ind-1.)/12.;
  !! log_input(obs_ind_in);
  !! log_input(obs_se_ind_in);
  matrix        corr_dev(1,nind,1,nyrs_ind) //Index standard errors (for lognormal)
  matrix        corr_eff(1,nfsh,styr,endyr) //Index standard errors (for lognormal)
  matrix         act_eff(1,nfsh,styr,endyr) //Index standard errors (for lognormal)
  vector              ac(1,nind);

  init_ivector nyrs_ind_age(1,nind)               //Number of years of index value (annual)
  init_matrix yrs_ind_age_in(1,nind,1,nyrs_ind_age)  //Years of index value (annual)
  init_matrix n_sample_ind_age_in(1,nind,1,nyrs_ind_age)         //Years of index value (annual)
  init_3darray oac_ind_in(1,nind,1,nyrs_ind_age,1,nages)  //values of Index proportions at age

  init_3darray  wt_ind(1,nind,styr,endyr,1,nages)      //values of Index proportions at age
  !! log_input(wt_ind);

  vector age_vector(1,nages);
    !! for (j=1;j<=nages;j++)
      !!  age_vector(j) = double(j+rec_age-1);
  init_vector wt_pop(1,nages)
  !! log_input(wt_pop);
  init_vector maturity(1,nages)
  !! log_input(maturity);
  !! if (max(maturity)>.9) maturity /=2.;
  vector wt_mature(1,nages);
  !! wt_mature = elem_prod(wt_pop,maturity) ;

  //Spawning month-----
  init_number spawnmo
  number spmo_frac
  !! spmo_frac = (spawnmo-1)/12.;

  init_matrix age_err(1,nages,1,nages)
  !! log_input(age_err);

  int k // Index for fishery or index
  int i // Index for year
  int j // Index for age
 LOCAL_CALCS
  // Rename data file to the control data section...
  ad_comm::change_datafile_name(cntrlfile_name);
  *(ad_comm::global_datafile) >>  datafile_name;
  *(ad_comm::global_datafile) >>  model_name;
  log_input(cntrlfile_name);
 END_CALCS
  // Matrix of selectivity mappings--row 1 is type (1=fishery, 2=index) and row 2 is index within
that type
  //  e.g., the following for 2 fisheries and 4 indices means that index 3 uses fishery 1
selectivities,
  //         the other fisheries and indices use their own parameterization
  //  1 1 2 2 1 2
  //  1 2 1 2 1 4
  init_imatrix sel_map(1,2,1,nfsh_and_ind)
  // maps fisheries and indices into sequential sel_map for sharing purposes
  !! write_input_log<< "# Map shared selectivity: "<< endl;log_input(sel_map);
  !! log_input(datafile_name);
  !! log_input(model_name);
  !! projfile_name = cntrlfile_name(1,length(cntrlfile_name)-4) + ".prj";
  init_int    SrType        // 2 Bholt, 1 Ricker
  !! log_input(SrType);
  init_int use_age_err      // nonzero value means use...
  !! log_input(use_age_err);
```

```
 init_int retro            // Retro years to peel off (0 means full dataset)
 !! log_input(retro);
 init_number steepnessprior
 init_number cvsteepnessprior
 init_int    phase_srec

 init_number sigmarprior
 number log_sigmarprior
 init_number cvsigmarprior
 init_int    phase_sigmar
 !! log_input(sigmarprior);
 !! log_input(cvsigmarprior);
 !! log_input(phase_sigmar);
 init_int    styr_rec_est
 init_int    endyr_rec_est
 !! log_input(styr_rec_est);
 !! log_input(endyr_rec_est);
 int nrecs_est;
 // Basic M
 init_number natmortprior
 init_number cvnatmortprior
 init_int    phase_M
 !! log_input(natmortprior);
 !! log_input(cvnatmortprior);
 !! log_input(phase_M);

 // age-specific M
 init_int     npars_Mage
 init_ivector ages_M_changes(1,npars_Mage)
 init_vector  Mage_in(1,npars_Mage)
 init_int     phase_Mage
 vector       Mage_offset_in(1,npars_Mage)
 // convert inputs to offsets from prior for initialization purposes
 !! if (npars_Mage>0) Mage_offset_in = log(Mage_in / natmortprior);
 !! log_input(npars_Mage);
 !! log_input(ages_M_changes);
 !! log_input(Mage_in);
 !! log_input(Mage_offset_in);

 // time-varying M
 init_int    phase_rw_M
 init_int npars_rw_M
 init_ivector  yrs_rw_M(1,npars_rw_M);
 init_vector sigma_rw_M(1,npars_rw_M)
LOCAL_CALCS
 log_input(npars_rw_M);
 log_input(yrs_rw_M);
 log_input(phase_rw_M);
 log_input(sigma_rw_M);
END_CALCS

 init_vector qprior(1,nind)
 vector log_qprior(1,nind)
 init_vector cvqprior(1,nind)
 init_ivector phase_q(1,nind)
 !! log_input(qprior);
 !! log_input(cvqprior);
 !! log_input(phase_q);

 init_vector q_power_prior(1,nind)
 vector log_q_power_prior(1,nind)
 init_vector cvq_power_prior(1,nind)
 init_ivector phase_q_power(1,nind)
 // Random walk definition for indices
 init_ivector phase_rw_q(1,nind)
 init_ivector npars_rw_q(1,nind)
 init_imatrix  yrs_rw_q(1,nind,1,npars_rw_q); // Ragged array
 init_matrix sigma_rw_q(1,nind,1,npars_rw_q); // Ragged array
LOCAL_CALCS
 log_input(phase_rw_q);
 log_input(npars_rw_q);
```

```
  log_input(yrs_rw_q);
  log_input(sigma_rw_q);
 END_CALCS

  init_ivector    q_age_min(1,nind)     // Age that q relates to...
  init_ivector    q_age_max(1,nind)     // Age that q relates to...
  !! log_input(q_age_min);
  !! log_input(q_age_max);
  // Need to map to age index range...
  !! for (k=1;k<=nind;k++) {q_age_min(k) =  q_age_min(k) - rec_age + 1; q_age_max(k) = q_age_max(k) -
rec_age + 1;}
  !! log_input(q_age_min);
  !! log_input(q_age_max);

  init_number cv_catchbiomass
  number catchbiomass_pen
  !!catchbiomass_pen= 1./(2*cv_catchbiomass*cv_catchbiomass);
  init_int nproj_yrs

  int styr_fut
  int endyr_fut            // LAst year for projections
  int phase_Rzero
  int phase_nosr
  number Steepness_UB
  !! phase_Rzero =  4;
  !! phase_nosr  = -3;

  // Selectivity controls
  // read in options for each fishery
  // Loop over fisheries and indices to read in data (conditional on sel_options)
  ivector   fsh_sel_opt(1,nfsh)
  ivector phase_sel_fsh(1,nfsh)
  vector   curv_pen_fsh(1,nfsh)
  matrix   sel_slp_in_fsh(1,nfsh,1,nyrs)
  matrix   logsel_slp_in_fsh(1,nfsh,1,nyrs)
  matrix   sel_inf_in_fsh(1,nfsh,1,nyrs)
  vector   logsel_slp_in_fshv(1,nfsh)
  vector   sel_inf_in_fshv(1,nfsh)
  vector   logsel_dslp_in_fshv(1,nfsh)
  vector   sel_dinf_in_fshv(1,nfsh)
  matrix   sel_dslp_in_fsh(1,nfsh,1,nyrs)
  matrix   logsel_dslp_in_fsh(1,nfsh,1,nyrs)
  matrix   sel_dinf_in_fsh(1,nfsh,1,nyrs)

  vector seldec_pen_fsh(1,nfsh) ;
  vector nnodes_fsh(1,nfsh) ;
  int seldecage ;
  !! seldecage = int(nages/2);
  ivector nselages_in_fsh(1,nfsh)

  ivector n_sel_ch_fsh(1,nfsh);
  ivector n_sel_ch_ind(1,nind);
  imatrix yrs_sel_ch_tmp(1,nind,1,endyr-styr+1);
  imatrix yrs_sel_ch_tmp_ind(1,nind,1,endyr-styr+1);
  !! yrs_sel_ch_tmp_ind.initialize();

  ivector   ind_sel_opt(1,nind)
  ivector phase_sel_ind(1,nind)

  vector   curv_pen_ind(1,nind)

  matrix   logsel_slp_in_ind(1,nind,1,nyrs)
  matrix   sel_inf_in_ind(1,nind,1,nyrs)
  matrix   sel_dslp_in_ind(1,nind,1,nyrs)
  matrix   logsel_dslp_in_ind(1,nind,1,nyrs)
  matrix   sel_dinf_in_ind(1,nind,1,nyrs)
  matrix   sel_slp_in_ind(1,nind,1,nyrs)

  vector   logsel_slp_in_indv(1,nind)
  vector   sel_inf_in_indv(1,nind)
  vector   logsel_dslp_in_indv(1,nind)
```

```
 vector   sel_dinf_in_indv(1,nind)


 vector seldec_pen_ind(1,nind) ;
 matrix sel_change_in_ind(1,nind,styr,endyr);
 ivector nselages_in_ind(1,nind)
 matrix sel_change_in_fsh(1,nfsh,styr,endyr);
 imatrix yrs_sel_ch_fsh(1,nfsh,1,endyr-styr);
 matrix sel_sigma_fsh(1,nfsh,1,endyr-styr);
 imatrix yrs_sel_ch_ind(1,nind,1,endyr-styr);
 matrix sel_sigma_ind(1,nind,1,endyr-styr);

 // Phase of estimation
 ivector phase_selcoff_fsh(1,nfsh)
 ivector phase_logist_fsh(1,nfsh)
 ivector phase_dlogist_fsh(1,nfsh)
 ivector phase_sel_spl_fsh(1,nfsh)

 ivector phase_selcoff_ind(1,nind)
 ivector phase_logist_ind(1,nind)
 ivector phase_dlogist_ind(1,nind)
 vector  sel_fsh_tmp(1,nages);
 vector  sel_ind_tmp(1,nages);
 3darray log_selcoffs_fsh_in(1,nfsh,1,nyrs,1,nages)
 3darray log_selcoffs_ind_in(1,nind,1,nyrs,1,nages)
 3darray  log_sel_spl_fsh_in(1,nfsh,1,nyrs,1,nages) // use nages for input to start
 // 3darray log_selcoffs_ind_in(1,nind,1,nyrs,1,nages)

LOCAL_CALCS
 logsel_slp_in_fshv.initialize();
 sel_inf_in_fshv.initialize();
 logsel_dslp_in_fshv.initialize();
 sel_inf_in_fshv.initialize();
 sel_dinf_in_fshv.initialize();

 sel_inf_in_indv.initialize();
 logsel_dslp_in_indv.initialize();
 sel_inf_in_indv.initialize();
 sel_dinf_in_indv.initialize();

 phase_selcoff_ind.initialize();
 phase_logist_ind.initialize();
 phase_dlogist_ind.initialize();
 sel_fsh_tmp.initialize() ;
 sel_ind_tmp.initialize() ;
 log_selcoffs_fsh_in.initialize();
 log_selcoffs_ind_in.initialize();

 // nselages_in_fsh.initialize()   ;
 // nselages_in_ind.initialize()   ;
 nselages_in_fsh = nages-1;
 nselages_in_ind = nages-1;
 sel_change_in_ind.initialize()   ;
 sel_slp_in_fsh.initialize()   ;  // ji
 sel_slp_in_ind.initialize()   ;  // ji
 sel_inf_in_fsh.initialize()   ;  // ji
 sel_inf_in_ind.initialize()   ;  // ji
 logsel_slp_in_fsh.initialize();  // ji
 logsel_slp_in_fshv.initialize();  // ji
 logsel_dslp_in_fsh.initialize(); // ji
 logsel_slp_in_ind.initialize();  // ji
 logsel_slp_in_indv.initialize();  // ji
 logsel_dslp_in_ind.initialize(); // ji
 sel_change_in_fsh.initialize()   ;
 for (k=1;k<=nfsh;k++)
 {
   *(ad_comm::global_datafile) >> fsh_sel_opt(k)  ;
   log_input(fsh_sel_opt(k));
   switch (fsh_sel_opt(k))
   {
     case 1 : // Selectivity coefficients
```

```
      {
        *(ad_comm::global_datafile) >> nselages_in_fsh(k)   ;
        *(ad_comm::global_datafile) >> phase_sel_fsh(k);
        *(ad_comm::global_datafile) >> curv_pen_fsh(k) ;
        *(ad_comm::global_datafile) >> seldec_pen_fsh(k) ;
        seldec_pen_fsh(k) *= seldec_pen_fsh(k) ;
        *(ad_comm::global_datafile) >>  n_sel_ch_fsh(k) ;
        n_sel_ch_fsh(k) +=1;
        yrs_sel_ch_fsh(k,1) = styr; // first year always estimated
        for (int i=2;i<=n_sel_ch_fsh(k);i++)
          *(ad_comm::global_datafile) >>  yrs_sel_ch_fsh(k,i) ;
        for (int i=2;i<=n_sel_ch_fsh(k);i++)
          *(ad_comm::global_datafile) >>  sel_sigma_fsh(k,i) ;
        log_input(nselages_in_fsh(k)) ;
        log_input(phase_sel_fsh(k)) ;
        log_input(curv_pen_fsh(k)) ;
        log_input(seldec_pen_fsh(k)) ;
        log_input(n_sel_ch_fsh(k)) ;
        log_input(yrs_sel_ch_fsh(k)) ;
        log_input(sel_sigma_fsh(k)) ;
        // for (int i=styr;i<=endyr;i++) *(ad_comm::global_datafile) >> sel_change_in_fsh(k,i) ;
        sel_change_in_fsh(k,styr)=1.;
       // Number of selectivity changes is equal to the number of vectors (yr 1 is baseline)
        // This to read in pre-specified selectivity values...
        sel_fsh_tmp.initialize();
        log_selcoffs_fsh_in.initialize();
        for (int j=1;j<=nages;j++)
          *(ad_comm::global_datafile) >> sel_fsh_tmp(j);
        for (int jj=2;jj<=n_sel_ch_fsh(k);jj++)
        {
          // Set the selectivity for the oldest group
          for (int j=nselages_in_fsh(k)+1;j<=nages;j++)
          {
            sel_fsh_tmp(j)  = sel_fsh_tmp(nselages_in_fsh(k));
          }
          // Set tmp to actual initial vectors...
          log_selcoffs_fsh_in(k,jj)(1,nselages_in_fsh(k)) =
log((sel_fsh_tmp(1,nselages_in_fsh(k))+1e-7)/mean(sel_fsh_tmp(1,nselages_in_fsh(k))+1e-7) );
          write_input_log<<"Sel_in_fsh "<< mfexp(log_selcoffs_fsh_in(k,jj))<<endl;
        }
        // exit(1);
        phase_selcoff_fsh(k) = phase_sel_fsh(k);
        phase_logist_fsh(k)  = -1;
        phase_dlogist_fsh(k) = -1;
        phase_sel_spl_fsh(k) = -1;
      }
        break;
      case 2 : // Single logistic
      {
        *(ad_comm::global_datafile) >> phase_sel_fsh(k);
        *(ad_comm::global_datafile) >>  n_sel_ch_fsh(k) ;
        n_sel_ch_fsh(k) +=1;
        yrs_sel_ch_fsh(k,1) = styr;
        for (int i=2;i<=n_sel_ch_fsh(k);i++)
          *(ad_comm::global_datafile) >>  yrs_sel_ch_fsh(k,i) ;
        for (int i=2;i<=n_sel_ch_fsh(k);i++)
          *(ad_comm::global_datafile) >>  sel_sigma_fsh(k,i) ;
        // This to read in pre-specified selectivity values...
        *(ad_comm::global_datafile) >> sel_slp_in_fsh(k,1) ;
        *(ad_comm::global_datafile) >> sel_inf_in_fsh(k,1) ;
        logsel_slp_in_fsh(k,1)   = log(sel_slp_in_fsh(k,1)) ;
        for (int jj=2;jj<=n_sel_ch_fsh(k);jj++)
        {
          sel_inf_in_fsh(k,jj)    =     sel_inf_in_fsh(k,1) ;
          logsel_slp_in_fsh(k,jj) = log(sel_slp_in_fsh(k,1)) ;
        }
        log_input(phase_sel_fsh(k));
        log_input(n_sel_ch_fsh(k));
        log_input(sel_slp_in_fsh(k)(1,n_sel_ch_fsh(k)));
        log_input(sel_inf_in_fsh(k)(1,n_sel_ch_fsh(k)));
        log_input(logsel_slp_in_fsh(k)(1,n_sel_ch_fsh(k)));
```

```
        log_input(yrs_sel_ch_fsh(k)(1,n_sel_ch_fsh(k)));

        phase_selcoff_fsh(k) = -1;
        phase_logist_fsh(k) = phase_sel_fsh(k);
        phase_dlogist_fsh(k) = -1;
        phase_sel_spl_fsh(k) = -1;

        logsel_slp_in_fshv(k) = logsel_slp_in_fsh(k,1);
           sel_inf_in_fshv(k) =    sel_inf_in_fsh(k,1);
        break;
      }
      case 3 : // Double logistic
      {
        write_input_log << "Double logistic abandoned..."<<endl;exit(1);
        break;
      }
      case 4 : // Splines
      {
      }
      break;
      write_input_log << fshname(k)<<" fish sel opt "<<endl<<fsh_sel_opt(k)<<"
"<<endl<<"Sel_change"<<endl<<sel_change_in_fsh(k)<<endl;
    }
  }
  // Indices here..............
  yrs_sel_ch_ind.initialize() ;
  sel_sigma_ind.initialize();
  for(k=1;k<=nind;k++)
  {
    *(ad_comm::global_datafile) >> ind_sel_opt(k)  ;
    write_input_log << endl<<"Survey "<<indname(k)<<endl;
    log_input(ind_sel_opt(k));
    switch (ind_sel_opt(k))
    {
      case 1 : // Selectivity coefficients  indices
      {
        *(ad_comm::global_datafile) >> nselages_in_ind(k)   ;
        *(ad_comm::global_datafile) >> phase_sel_ind(k);
        *(ad_comm::global_datafile) >> curv_pen_ind(k) ;
        *(ad_comm::global_datafile) >> seldec_pen_ind(k) ;
        seldec_pen_ind(k) *= seldec_pen_ind(k);
        *(ad_comm::global_datafile) >>  n_sel_ch_ind(k) ;
        n_sel_ch_ind(k)+=1;
        yrs_sel_ch_ind(k,1) = styr;
        yrs_sel_ch_tmp_ind(k,1) = styr;
        for (int i=2;i<=n_sel_ch_ind(k);i++)
          *(ad_comm::global_datafile) >>  yrs_sel_ch_ind(k,i) ;
        for (int i=2;i<=n_sel_ch_ind(k);i++)
          *(ad_comm::global_datafile) >>  sel_sigma_ind(k,i) ;
        sel_change_in_ind(k,styr)=1.;
       // Number of selectivity changes is equal to the number of vectors (yr 1 is baseline)
        log_input(indname(k));
        log_input(nselages_in_ind(k));
        log_input(phase_sel_ind(k));
        log_input(seldec_pen_ind(k));
        log_input(n_sel_ch_ind(k));
        log_input(sel_change_in_ind(k));
        log_input(n_sel_ch_ind(k));
        log_input(yrs_sel_ch_ind(k)(1,n_sel_ch_ind(k)));
        // This to read in pre-specified selectivity values...
        for (j=1;j<=nages;j++)
          *(ad_comm::global_datafile) >> sel_ind_tmp(j);
        log_input(sel_ind_tmp);
        log_selcoffs_ind_in(k,1)(1,nselages_in_ind(k)) = log((sel_ind_tmp(1,nselages_in_ind(k))+1e-
7)/mean(sel_fsh_tmp(1,nselages_in_ind(k))+1e-7) );
        // set all change selectivity to initial values
        for (int jj=2;jj<=n_sel_ch_ind(k);jj++)
        {
          for (int j=nselages_in_ind(k)+1;j<=nages;j++) // This might be going out of nages=nselages
          {
            sel_ind_tmp(j)  = sel_ind_tmp(nselages_in_ind(k));
```

```
          }
          // Set tmp to actual initial vectors...
          log_selcoffs_ind_in(k,jj)(1,nselages_in_ind(k)) =
log((sel_ind_tmp(1,nselages_in_ind(k))+1e-7)/mean(sel_fsh_tmp(1,nselages_in_ind(k))+1e-7) );
          write_input_log<<"Sel_in_ind "<< mfexp(log_selcoffs_ind_in(k,jj))<<endl;
        }
        phase_selcoff_ind(k) = phase_sel_ind(k);
        phase_logist_ind(k)  = -2;
        phase_dlogist_ind(k)  = -1;
      }
      break;
      case 2 : // Single logistic
      {
        *(ad_comm::global_datafile) >> phase_sel_ind(k);
        *(ad_comm::global_datafile) >>  n_sel_ch_ind(k) ;
        n_sel_ch_ind(k) +=1;
        yrs_sel_ch_ind(k,1) = styr; // first year always estimated
        yrs_sel_ch_tmp_ind(k,1) = styr;
        for (int i=2;i<=n_sel_ch_ind(k);i++)
          *(ad_comm::global_datafile) >>  yrs_sel_ch_ind(k,i) ;
        for (int i=2;i<=n_sel_ch_ind(k);i++)
          *(ad_comm::global_datafile) >>  sel_sigma_ind(k,i) ;
        sel_change_in_ind(k,styr)=1.;

        log_input(indname(k));
        log_input(nselages_in_ind(k));
        log_input(phase_sel_ind(k));
        log_input(sel_change_in_ind(k));
        log_input(n_sel_ch_ind(k));
        log_input(yrs_sel_ch_ind(k)(1,n_sel_ch_ind(k)));
        // This to read in pre-specified selectivity values...
       // Number of selectivity changes is equal to the number of vectors (yr 1 is baseline)
        for (int i=styr+1;i<=endyr;i++) { if(sel_change_in_ind(k,i)>0) { j++; yrs_sel_ch_tmp_ind(k,j)
= i; } }
        // This to read in pre-specified selectivity values...
        *(ad_comm::global_datafile) >> sel_slp_in_ind(k,1) ;
        *(ad_comm::global_datafile) >> sel_inf_in_ind(k,1) ;
        logsel_slp_in_ind(k,1) =   log(sel_slp_in_ind(k,1)) ;
        for (int jj=2;jj<=n_sel_ch_ind(k);jj++)
        {
          sel_inf_in_ind(k,jj)    =     sel_inf_in_ind(k,1) ;
          logsel_slp_in_ind(k,jj) = log(sel_slp_in_ind(k,1)) ;
        }
        log_input(sel_slp_in_ind(k,1));
        log_input(sel_inf_in_ind(k,1));
        log_input(logsel_slp_in_ind(k,1));

        phase_selcoff_ind(k) = -1;
        phase_logist_ind(k) = phase_sel_ind(k);
        phase_dlogist_ind(k)  = -1;

        logsel_slp_in_indv(k) = logsel_slp_in_ind(k,1);
           sel_inf_in_indv(k) =    sel_inf_in_ind(k,1);
        log_input(logsel_slp_in_indv(k));
      }
      break;
      case 3 : // Double logistic
      {
        write_input_log << "Double logistic abandoned..."<<endl;exit(1);
      }
        break;
      case 4 : // spline for indices
      {
      }
      break;
    }
    write_input_log << indname(k)<<" ind sel opt "<<ind_sel_opt(k)<<" "<<sel_change_in_ind(k)<<endl;
  }
  write_input_log<<"Phase indices Sel_Coffs: "<<phase_selcoff_ind<<endl;
 END_CALCS
  init_number test;
```

```
 !! write_input_log<<" Test: "<<test<<endl;
!! if (test!=123456789) {cerr<<"Control file not read in correctly... "<<endl;exit(1);}


 ivector nopt_fsh(1,2) // number of options...
 !! nopt_fsh.initialize();
 !! for (k=1;k<=nfsh;k++) if(fsh_sel_opt(k)==1) nopt_fsh(1)++;else nopt_fsh(2)++;

 // Fishery selectivity description:
 // type 1

 // Number of ages

 !! write_input_log << "# Fshry Selages: " << nselages_in_fsh  <<endl;
 !! write_input_log << "# Srvy  Selages: " << nselages_in_ind <<endl;



 !! write_input_log << "# Phase for age-spec fishery "<<phase_selcoff_fsh<<endl;
 !! write_input_log << "# Phase for logistic fishery "<<phase_logist_fsh<<endl;
 !! write_input_log << "# Phase for dble logistic fishery "<<phase_dlogist_fsh<<endl;

 !! write_input_log << "# Phase for age-spec indices  "<<phase_selcoff_ind<<endl;
 !! write_input_log << "# Phase for logistic indices  "<<phase_logist_ind<<endl;
 !! write_input_log << "# Phase for dble logistic ind "<<phase_dlogist_ind<<endl;

 !! for (k=1;k<=nfsh;k++) if (phase_selcoff_fsh(k)>0) curv_pen_fsh(k) = 1./
(square(curv_pen_fsh(k))*2);
 !! write_input_log<<"# Curv_pen_fsh: "<<endl<<curv_pen_fsh<<endl;
 !! for (k=1;k<=nind;k++) if (phase_selcoff_ind(k)>0) curv_pen_ind(k) = 1./
(square(curv_pen_ind(k))*2);
 !! write_input_log<<"# Curv_pen_ind: "<<endl<<curv_pen_fsh<<endl;

 int  phase_fmort;
 int  phase_proj;
 ivector   nselages_fsh(1,nfsh);
 matrix xnodes_fsh(1,nfsh,1,nnodes_fsh)
 matrix xages_fsh(1,nfsh,1,nages)

 ivector   nselages_ind(1,nind);
 //Resetting data here for retrospectives//////////////////////////////////////////
LOCAL_CALCS
 for (int k=1;k<=nfsh;k++)
 {
   if ((endyr-retro)<=yrs_sel_ch_fsh(k,n_sel_ch_fsh(k))) n_sel_ch_fsh(k)-- ;
   for (int i=1;i<=retro;i++)
   {
       cout<<"here"<<max(yrs_fsh_age_in(k)(1,nyrs_fsh_age(k)))<<endl;
     if (max(yrs_fsh_age_in(k)(1,nyrs_fsh_age(k)))>=(endyr-retro))
     {
        nyrs_fsh_age(k) -= 1;
     }
   }
 }
 // now for indices
 for (int k=1;k<=nind;k++)
 {
   if ((endyr-retro)<=yrs_sel_ch_ind(k,n_sel_ch_ind(k))) n_sel_ch_ind(k)-- ;
   for (int i=1;i<=retro;i++)
   {
     // index values
     if (max(yrs_ind_in(k)(1,nyrs_ind(k)))>=(endyr-retro))
        nyrs_ind(k) -= 1;
     // Ages (since they can be different than actual index years)
     if (max(yrs_ind_age_in(k)(1,nyrs_ind_age(k)))>=(endyr-retro))
        nyrs_ind_age(k) -= 1;
   }
 }
 endyr_rec_est = endyr_rec_est - retro;
 endyr         = endyr - retro;
 styr_fut      = endyr+1;
```

```
 endyr_fut     = endyr + nproj_yrs;
 endyr_sp      = endyr   - rec_age - 1;// endyr year of (main) spawning biomass
END_CALCS
// now use redimensioned data for retro
 matrix catch_bio(1,nfsh,styr,endyr)         //Catch biomass
 matrix catch_bio_sd(1,nfsh,styr,endyr)      //Catch biomass standard errors
 matrix catch_bio_lsd(1,nfsh,styr,endyr)     //Catch biomass standard errors (for lognormal)
 matrix catch_bio_lva(1,nfsh,styr,endyr)     //Catch biomass variance (for lognormal)
 matrix catch_bioT(styr,endyr,1,nfsh)
 imatrix yrs_fsh_age(1,nfsh,1,nyrs_fsh_age)
 matrix  n_sample_fsh_age(1,nfsh,1,nyrs_fsh_age)    //Years of index index value (annual)
 3darray oac_fsh(1,nfsh,1,nyrs_fsh_age,1,nages)

 imatrix yrs_ind(1,nind,1,nyrs_ind)         //Years of index value (annual)
 matrix obs_ind(1,nind,1,nyrs_ind)          //values of index value (annual)
 matrix obs_se_ind(1,nind,1,nyrs_ind)       //values of indices serrs

 matrix yrs_ind_age(1,nind,1,nyrs_ind_age)  //Years of index value (annual)
 matrix n_sample_ind_age(1,nind,1,nyrs_ind_age)         //Years of index value (annual)
 3darray oac_ind(1,nind,1,nyrs_ind_age,1,nages)  //values of Index proportions at age
 matrix     obs_lse_ind(1,nind,1,nyrs_ind) //Index standard errors (for lognormal)
 matrix     obs_lva_ind(1,nind,1,nyrs_ind) //Index standard errors (for lognormal)
LOCAL_CALCS
 for (int k=1;k<=nfsh;k++)
 {
   catch_bio(k) = catch_bio_in(k)(styr,endyr);
   catch_bio_sd(k) = catch_bio_sd_in(k)(styr,endyr);
   yrs_fsh_age(k) = yrs_fsh_age_in(k)(1,nyrs_fsh_age(k));
   n_sample_fsh_age(k) = n_sample_fsh_age_in(k)(1,nyrs_fsh_age(k));
   for (int i=1;i<=nyrs_fsh_age(k);i++)
     oac_fsh(k,i) = oac_fsh_in(k,i) ;
 }
   catch_bio_lsd = sqrt(log(square(catch_bio_sd) + 1.));
   catch_bio_lva = log(square(catch_bio_sd) + 1.);
   catch_bioT = trans(catch_bio);
 for (int k=1;k<=nind;k++)
 {
   yrs_ind(k)  = yrs_ind_in(k)(1,nyrs_ind(k));
   obs_ind(k)  = obs_ind_in(k)(1,nyrs_ind(k));
   obs_se_ind(k)  = obs_se_ind_in(k)(1,nyrs_ind(k));

   yrs_ind_age(k) = yrs_ind_age_in(k)(1,nyrs_ind_age(k));
   n_sample_ind_age(k) = n_sample_ind_age_in(k)(1,nyrs_ind_age(k));
   for (int i=1;i<=nyrs_ind_age(k);i++)
     oac_ind(k,i) = oac_ind_in(k,i) ;
 }
 log_input(nyrs_fsh_age);
 log_input(yrs_fsh_age);
 log_input(n_sample_fsh_age);
 log_input(oac_fsh);
 log_input(wt_fsh);

 log_input(nyrs_ind_age);
 log_input(yrs_ind_age);
 log_input(n_sample_ind_age);
 log_input(oac_ind);
 obs_lse_ind = elem_div(obs_se_ind,obs_ind);
 obs_lse_ind = sqrt(log(square(obs_lse_ind) + 1.));
 log_input(obs_lse_ind);
 obs_lva_ind = square(obs_lse_ind);
END_CALCS

 /////////////////////////////////////////////////////////////////////////////////
LOCAL_CALCS
 for (k=1; k<=nfsh;k++)
 {
   // xages_fsh increments from 0-1 by number of ages, say
   xages_fsh.initialize();
   log_input(xages_fsh);
   xages_fsh(k).fill_seqadd(0.,1.0/(nages-1));
   log_input(xages_fsh);
```

```
    //  xnodes increments from 0-1 by number of nodes
    xnodes_fsh.initialize();
    xnodes_fsh(k).fill_seqadd(0.,1.0/(nnodes_fsh(k)-1));
    log_input(xnodes_fsh);
    // xages_fsh(k).fill_seqadd(0,1.0/(nselages_in_fsh(k)-1)); //prefer to use nselages but need 3d
version to work
  }
  write_input_log<<"Yrs fsh_sel change: "<<yrs_sel_ch_fsh<<endl;
  // for (k=1; k<=nind;k++) yrs_sel_ch_ind(k) = yrs_sel_ch_tmp_ind(k)(1,n_sel_ch_ind(k));
  write_input_log<<"Yrs ind_sel change: "<<yrs_sel_ch_ind<<endl;
    log_sigmarprior = log(sigmarprior);
    log_input(steepnessprior);
    log_input(sigmarprior);
    nrecs_est = endyr_rec_est-styr_rec_est+1;
    nrecs_est = endyr_rec_est-styr_rec_est+1;
    write_input_log<<"#  SSB estimated in styr endyr: " <<styr_sp    <<" "<<endyr_sp      <<" 
"<<endl;
    write_input_log<<"#  Rec estimated in styr endyr: " <<styr_rec    <<" "<<endyr        <<" 
"<<endl;
    write_input_log<<"#  SR Curve fit  in styr endyr: " <<styr_rec_est<<" "<<endyr_rec_est<<" 
"<<endl;
    write_input_log<<"#             Model styr endyr: " <<styr        <<" "<<endyr        <<" 
"<<endl;
    log_qprior = log(qprior);
    log_input(qprior);
    log_q_power_prior = log(q_power_prior);
    write_input_log<<"# q_power_prior " <<endl<<q_power_prior<<" "<<endl;
    write_input_log<<"# cv_catchbiomass " <<endl<<cv_catchbiomass<<" "<<endl;
    write_input_log<<"# CatchbiomassPen " <<endl<<catchbiomass_pen<<" "<<endl;
    write_input_log<<"# Number of projection years " <<endl<<nproj_yrs<<" "<<endl;// cin>>junk;

 END_CALCS
  number R_guess;

  vector offset_ind(1,nind)
  vector offset_fsh(1,nfsh)
  int do_fmort;
  !! do_fmort=0;
  int Popes;
 LOCAL_CALCS
  Popes=0; // option to do Pope's approximation (not presently flagged outside of code)
  if (Popes)
    phase_fmort = -2;
  else
    phase_fmort = 1;

  phase_proj  =  5;

  Steepness_UB = .9999; // upper bound of steepness
  offset_ind.initialize();
  offset_fsh.initialize();

  double sumtmp;
  for (k=1;k<=nfsh;k++)
    for (i=1;i<=nyrs_fsh_age(k);i++)
    {
      oac_fsh(k,i) /= sum(oac_fsh(k,i)); // Normalize to sum to one
      offset_fsh(k) -= n_sample_fsh_age(k,i)*(oac_fsh(k,i) + 0.001) * log(oac_fsh(k,i) + 0.001 ) ;
    }

  for (k=1;k<=nind;k++)
    for (i=1;i<=nyrs_ind_age(k);i++)
    {
      oac_ind(k,i) /= sum(oac_ind(k,i)); // Normalize to sum to one
      offset_ind(k) -= n_sample_ind_age(k,i)*(oac_ind(k,i) + 0.001) * log(oac_ind(k,i) + 0.001 ) ;
    }
  log_input(offset_fsh);
  log_input(offset_ind);

  if (ad_comm::argc > 1) // Command line argument to profile Fishing mortality rates...
  {
```

```
    int on=0;
    if ( (on=option_match(ad_comm::argc,ad_comm::argv,"-uFmort"))>-1)
      do_fmort=1;
  }

  // Compute an initial Rzero value based on exploitation
   double btmp=0.;
   double ctmp=0.;
   dvector ntmp(1,nages);
   ntmp(1) = 1.;
   for (int a=2;a<=nages;a++)
     ntmp(a) = ntmp(a-1)*exp(-natmortprior-.05);
   btmp = wt_pop * ntmp;
   write_input_log << "Mean Catch"<<endl;
   ctmp = mean(catch_bio);
   write_input_log << ctmp <<endl;
   R_guess = log((ctmp/.02 )/btmp) ;
   write_input_log << "R_guess "<<endl;
   write_input_log << R_guess <<endl;
 END_CALCS


PARAMETER_SECTION
 // Biological Parameters
  init_bounded_number Mest(.02,4.8,phase_M)
  init_bounded_vector Mage_offset(1,npars_Mage,-3,3,phase_Mage)
  vector Mage(1,nages)
  init_bounded_vector  M_rw(1,npars_rw_M,-10,10,phase_rw_M)
  vector natmort(styr,endyr)
  matrix  natage(styr,endyr+1,1,nages)
  matrix N_NoFsh(styr,endyr_fut,1,nages);
  // vector Sp_Biom(styr_sp,endyr)
  vector pred_rec(styr_rec,endyr)
  vector mod_rec(styr_rec,endyr) // As estimated by model
  matrix  M(styr,endyr,1,nages)
  matrix  Z(styr,endyr,1,nages)
  matrix  S(styr,endyr,1,nages)
  number surv

 // Stock rectuitment params
  init_number mean_log_rec(1);
  init_bounded_number steepness(0.21,Steepness_UB,phase_srec)
  init_number log_Rzero(phase_Rzero)
  // OjO
  // init_bounded_vector initage_dev(2,nages,-15,15,4)
  init_bounded_vector rec_dev(styr_rec,endyr,-15,15,2)
  // init_vector rec_dev(styr_rec,endyr,2)
  init_number log_sigmar(phase_sigmar);
  number m_sigmarsq
  number m_sigmar
  number sigmarsq
  number sigmar
  number alpha
  number beta
  number Bzero
  number Rzero
  number phizero
  number avg_rec_dev

 // Fishing mortality parameters
  // init_vector         log_avg_fmort(1,nfsh,phase_fmort)
  // init_bounded_matrix fmort_dev(1,nfsh,styr,endyr,-15,15.,phase_fmort)
  init_bounded_matrix fmort(1,nfsh,styr,endyr,0.00,5.,phase_fmort)
  vector Fmort(styr,endyr);  // Annual total Fmort
  number hrate
  number catch_tmp
  number Fnew

  !! for (k=1;k<=nfsh;k++) nselages_fsh(k)=nselages_in_fsh(k); // Sets all elements of a vector to
one scalar value...
```

```
  !! for (k=1;k<=nind;k++) nselages_ind(k)=nselages_in_ind(k); // Sets all elements of a vector to
one scalar value...

 //  init_3darray log_selcoffs_fsh(1,nfsh,1,n_sel_ch_fsh,1,nselages_fsh,phase_selcoff_fsh)
  init_matrix_vector log_selcoffs_fsh(1,nfsh,1,n_sel_ch_fsh,1,nselages_fsh,phase_selcoff_fsh) // 3rd
dimension out...
  !! if (fsh_sel_opt(1)==4) nnodes_tmp=nnodes_fsh(1);  // NOTE THIS won't work in general
  //init_matrix_vector  log_sel_spl_fsh(1,nfsh,1,n_sel_ch_fsh,1,nnodes_tmp,phase_sel_spl_fsh)
  init_matrix_vector  log_sel_spl_fsh(1,nfsh,1,n_sel_ch_fsh,1,4,phase_sel_spl_fsh)

  !! log_input(nfsh);
  !! log_input(n_sel_ch_fsh);
  !! log_input(nselages_fsh);
  !! log_input(phase_selcoff_fsh);
  init_vector_vector logsel_slope_fsh(1,nfsh,1,n_sel_ch_fsh,phase_logist_fsh)
  matrix                sel_slope_fsh(1,nfsh,1,n_sel_ch_fsh)
  init_vector_vector     sel50_fsh(1,nfsh,1,n_sel_ch_fsh,phase_logist_fsh)
  init_vector_vector logsel_dslope_fsh(1,nfsh,1,n_sel_ch_fsh,phase_dlogist_fsh)
  matrix                sel_dslope_fsh(1,nfsh,1,n_sel_ch_fsh)
  !! int lb_d50=nages/2;
  init_bounded_vector_vector     seld50_fsh(1,nfsh,1,n_sel_ch_fsh,lb_d50,nages,phase_dlogist_fsh)

  // !!exit(1);
  3darray log_sel_fsh(1,nfsh,styr,endyr,1,nages)
  3darray sel_fsh(1,nfsh,styr,endyr,1,nages)
  matrix avgsel_fsh(1,nfsh,1,n_sel_ch_fsh);

  matrix  Ftot(styr,endyr,1,nages)
  3darray F(1,nfsh,styr,endyr,1,nages)
  3darray eac_fsh(1,nfsh,1,nyrs_fsh_age,1,nages)
  matrix  pred_catch(1,nfsh,styr,endyr)
  3darray catage(1,nfsh,styr,endyr,1,nages)
  matrix catage_tot(styr,endyr,1,nages)
  matrix expl_biom(1,nfsh,styr,endyr)

 // Parameters for computing SPR rates
  vector F50(1,nfsh)
  vector F40(1,nfsh)
  vector F35(1,nfsh)

 // Stuff for SPR and yield projections
  number sigmar_fut
  vector f_tmp(1,nfsh)
  number SB0
  number SBF50
  number SBF40
  number SBF35
  vector Fratio(1,nfsh)
  !! Fratio = 1;
  !! Fratio /= sum(Fratio);

  matrix Nspr(1,4,1,nages)

  matrix nage_future(styr_fut,endyr_fut,1,nages)
  init_vector rec_dev_future(styr_fut,endyr_fut,phase_proj);
  vector Sp_Biom_future(styr_fut-rec_age,endyr_fut);
  3darray F_future(1,nfsh,styr_fut,endyr_fut,1,nages);
  matrix Z_future(styr_fut,endyr_fut,1,nages);
  matrix S_future(styr_fut,endyr_fut,1,nages);
  matrix catage_future(styr_fut,endyr_fut,1,nages);
  number avg_rec_dev_future
  vector avg_F_future(1,5)

 // Survey Observation parameters
  init_number_vector log_q_ind(1,nind,phase_q)
  init_number_vector log_q_power_ind(1,nind,phase_q_power)
  init_vector_vector log_rw_q_ind(1,nind,1,npars_rw_q,phase_rw_q)
  init_matrix_vector log_selcoffs_ind(1,nind,1,n_sel_ch_ind,1,nselages_ind,phase_selcoff_ind)

  // init_vector_vector logsel_slope_ind(1,nind,1,n_sel_ch_ind,phase_logist_ind) // Need to make
positive or reparameterize
```

```
  init_vector_vector logsel_slope_ind(1,nind,1,n_sel_ch_ind,phase_logist_ind+1) // Need to make
positive or reparameterize
  init_bounded_vector_vector       sel50_ind(1,nind,1,n_sel_ch_ind,1,20,phase_logist_ind)

  init_vector_vector  logsel_dslope_ind(1,nind,1,n_sel_ch_ind,phase_dlogist_ind) // Need to make
positive or reparameterize
  init_bounded_vector_vector seld50_ind(1,nfsh,1,n_sel_ch_ind,lb_d50,nages,phase_dlogist_ind)

  matrix                sel_slope_ind(1,nind,1,n_sel_ch_ind)
  matrix                sel_dslope_ind(1,nind,1,n_sel_ch_ind)

  3darray log_sel_ind(1,nind,styr,endyr,1,nages)
  3darray sel_ind(1,nind,styr,endyr,1,nages)
  matrix avgsel_ind(1,nind,1,n_sel_ch_ind);

  matrix pred_ind(1,nind,1,nyrs_ind)
  3darray eac_ind(1,nind,1,nyrs_ind_age,1,nages)

 // Likelihood value names
  number sigma
  vector rec_like(1,4)
  vector catch_like(1,nfsh)
  vector age_like_fsh(1,nfsh)
  vector age_like_ind(1,nind)
  matrix sel_like_fsh(1,nfsh,1,4)
  matrix sel_like_ind(1,nind,1,4)
  vector ind_like(1,nind)
  vector fpen(1,6)
  vector post_priors(1,4)
  vector post_priors_indq(1,nind)
  objective_function_value obj_fun
  vector obj_comps(1,12)

  sdreport_number B100
  number F50_est
  number F40_est
  number F35_est
  matrix q_ind(1,nind,1,nyrs_ind)
  vector q_power_ind(1,nind)
  // sdreport_vector q_ind(1,nind)
  sdreport_vector totbiom(styr,endyr+1)
  sdreport_vector totbiom_NoFish(styr,endyr)
  sdreport_vector Sp_Biom(styr_sp,endyr+1)
  sdreport_vector Sp_Biom_NoFish(styr_sp,endyr)
  sdreport_vector Sp_Biom_NoFishRatio(styr,endyr)
  sdreport_number ABCBiom;
  sdreport_vector recruits(styr,endyr+1)
  // vector recruits(styr,endyr+1)
  sdreport_number depletion
  sdreport_number depletion_dyn
  sdreport_number MSY;
  sdreport_number MSYL;
  sdreport_number Fmsy;
  sdreport_number lnFmsy;
  sdreport_number Fcur_Fmsy;
  sdreport_number Rmsy;
  sdreport_number Bmsy;
  sdreport_number Bcur_Bmsy;
  sdreport_vector pred_ind_nextyr(1,nind);
  sdreport_number OFL;
  // NOTE TO DAVE: Need to have a phase switch for sdreport variables(
  matrix catch_future(1,4,styr_fut,endyr_fut); // Note, don't project for F=0 (it will bomb)
  sdreport_matrix SSB_fut(1,5,styr_fut,endyr_fut)
  !! write_input_log <<"logRzero "<<log_Rzero<<endl;
  !! write_input_log <<"logmeanrec "<<mean_log_rec<<endl;
  !! write_input_log<< "exp(log_sigmarprior "<<exp(log_sigmarprior)<<endl;
 // Initialize coefficients (if needed)
 LOCAL_CALCS
  for (k=1;k<=nfsh;k++)
  {
    write_input_log<<"Fish sel phase: "<<phase_selcoff_fsh(k)<<" "<<fshname(k)<<endl;
```

```
    switch (fsh_sel_opt(k))
    {
      case 1 : // Selectivity coefficients
      {
        if(phase_selcoff_fsh(k)<0)
        {
          write_input_log<<"Initial fixing fishery sel to"<<endl<<n_sel_ch_fsh(k)<<endl;
          for (int jj=1;jj<=n_sel_ch_fsh(k);jj++)
          {
            log_selcoffs_fsh(k,jj)(1,nselages_in_fsh(k)) =
log_selcoffs_fsh_in(k,jj)(1,nselages_in_fsh(k));
            write_input_log <<"Init coef:"<<endl<<exp(log_selcoffs_fsh(k,jj)(1,nselages_in_fsh(k)))
<<endl;
          }
        }
      }
        break;
      case 2 : // Single logistic
      {
        if(phase_logist_fsh(k)<0)
        {
          logsel_slope_fsh(k,1) = logsel_slp_in_fsh(k,1)  ;
          write_input_log<<"Fixing fishery sel to"<<endl<<n_sel_ch_fsh(k)<<endl;
          for (int jj=1;jj<=n_sel_ch_fsh(k);jj++)
          {
            logsel_slope_fsh(k,jj) = logsel_slp_in_fsh(k,jj)  ;
            sel50_fsh(k,jj)        =    sel_inf_in_fsh(k,jj)  ;
          }
        }
      }
      case 3 : // Double logistic
      {
        if(phase_dlogist_fsh(k)<0)
        {
          write_input_log<<"Fixing fishery sel to"<<endl<<n_sel_ch_fsh(k)<<endl;
          for (int jj=1;jj<=n_sel_ch_fsh(k);jj++)
          {
            logsel_slope_fsh(k,jj) = logsel_slp_in_fsh(k,jj)  ;
            sel50_fsh(k,jj)        =    sel_inf_in_fsh(k,jj)  ;
          }
        }
      }
      case 4 : // Selectivity spline initialize
      /* {
        if(phase_sel_spl_fsh(k)<0)
        {
          write_input_log<<"Initial fishery spline to"<<endl<<n_sel_ch_fsh(k)<<endl;
          for (int jj=1;jj<=n_sel_ch_fsh(k);jj++)
          {
            log_sel_spl_fsh(k,jj)(1,nnodes_tmp) = log_sel_spl_fsh_in(k,jj)(1,nnodes_tmp);
            // write_input_log <<"Init coef:"<<endl<<exp(log_sel_spl_fsh(k,jj)(1,nselages_in_fsh(k)))
<<endl;
          }
          log_input(log_sel_spl_fsh);
        }
       }*/
     break;
    }
  }
  for (k=1;k<=nind;k++)
  {
    write_input_log<<"Srvy sel phase: "<<phase_selcoff_ind(k)<<endl;
    if(phase_selcoff_ind(k)<0)
    {
      write_input_log<<"Fixing "<<indname(k)<<" indices sel to"<<endl<<n_sel_ch_ind(k)<<endl;
      for (int jj=1;jj<=n_sel_ch_ind(k);jj++)
      {
        log_selcoffs_ind(k,jj)(1,nselages_in_ind(k)) =
log_selcoffs_ind_in(k,jj)(1,nselages_in_ind(k));
        // write_input_log <<"Init coef:"<<endl<<exp(log_selcoffs_ind(k,jj)(1,nselages_in_ind(k)))
<<endl;
```

```
      }
    }
    if(phase_logist_ind(k)<0)
    {
      write_input_log<<"Fixing index sel to"<<endl<<n_sel_ch_ind(k)<<endl;
      for (int jj=1;jj<=n_sel_ch_ind(k);jj++)
      {
        logsel_slope_ind(k,jj) = logsel_slp_in_ind(k,jj)  ;
        // logsel_slope_ind(k,jj)    = 0.   ;
        sel50_ind(k,jj)           = sel_inf_in_ind(k,jj)  ;
      }
    }
  }
  log_input( logsel_slp_in_indv);
  write_input_log <<"Leaving parameter init secton"<<endl;
 END_CALCS


PRELIMINARY_CALCS_SECTION
  Mage_offset = Mage_offset_in;
  M(styr) = Mest;
  int jj=1;
  for (j=1;j<=nages;j++)
  {
    if (j==ages_M_changes(jj))
    {
      M(styr,j) = M(styr,1)*mfexp(Mage_offset(jj));
      jj++;
      if (npars_Mage < jj) jj=npars_Mage;
    }
    else
      if(j>1)
        M(styr,j) = M(styr,j-1);
  }
  for (i=styr+1;i<=endyr;i++)
    M(i) = M(i-1);


INITIALIZATION_SECTION
  Mest natmortprior;
  steepness steepnessprior
  log_sigmar log_sigmarprior;
  log_Rzero R_guess;
  mean_log_rec R_guess;
  // log_avg_fmort -2.065
  log_q_ind log_qprior;
  log_q_power_ind log_q_power_prior;

  sel50_fsh sel_inf_in_fshv

  logsel_dslope_fsh logsel_dslp_in_fshv ;
  seld50_fsh sel_dinf_in_fshv

  logsel_slope_ind logsel_slp_in_indv ;
  sel50_ind sel_inf_in_indv ;

  logsel_dslope_ind logsel_dslp_in_indv ;
  seld50_ind sel_dinf_in_indv ;

 //+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==

PROCEDURE_SECTION
  fpen.initialize();
  for (k=1;k<=nind;k++)
  {
    q_ind(k) = mfexp(log_q_ind(k) );
    q_power_ind(k) = mfexp(log_q_power_ind(k) );
  }

  // Main model calcs---------------------
```

```
  Get_Selectivity();
  Get_Mortality();
  Get_Bzero();
  Get_Numbers_at_Age();
  Get_Survey_Predictions();
  Get_Fishery_Predictions();
  // Objective function calcs------------
  evaluate_the_objective_function();

  // Output calcs-------------------------
  if (sd_phase())
  {
    compute_spr_rates();
    Calc_Dependent_Vars();
    if (mcmcmode)
    {
      // Calc_Dependent_Vars();
      mcflag   = 0;
      mcmcmode = 0;
    }
    else
    {
      // if (mcflag) Calc_Dependent_Vars();
    }
  }
  // Other calcs-------------------------
  if (mceval_phase())
  {
    if (oper_mod)
      Oper_Model();
    else
    {
      compute_spr_rates();
      write_mceval();
    }
  }
  if (do_fmort) Profile_F();
 //+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==


FUNCTION write_mceval
  if (mcmcmode != 3)
    write_mceval_hdr();
  mcmcmode = 3;
  mceval<< model_name         << " "  ;
  mceval<< obj_fun            << " "  ;
  // mceval<< rec_dev_future << " "  ;
  // mceval<<endl;
  get_msy();
  Future_projections();
  Calc_Dependent_Vars();
 /*
  mceval<<
  q_ind(1,1)  << " "<<
  M          << " "<<
  steepness << " "<<
  depletion << " "<<
  MSY        << " "<<
  MSYL       << " "<<
  Fmsy       << " "<<
  Fcur_Fmsy << " "<<
  Bcur_Bmsy << " "<<
  Bmsy       << " "<<
  ABCBiom    << " "<<
  F35        << " "<<
  F40        << " "<<
  F50        << " "<<
  SSB_fut(1,endyr_fut) << " "<<
  SSB_fut(2,endyr_fut) << " "<<
  SSB_fut(3,endyr_fut) << " "<<
  SSB_fut(4,endyr_fut) << " "<<
```

```
  SSB_fut(5,endyr_fut) << " "<<
  catch_future(1,styr_fut)    << " "<<
  catch_future(2,styr_fut)    << " "<<
  catch_future(3,styr_fut)    << " "<<
  catch_future(4,styr_fut)    << " "<<  endl;
 */
  for (int k=1;k<=nind;k++) mceval<< q_ind(k,1)  << " " ;
  mceval    <<
  M(endyr)  << " "<<
  steepness << " "<<
  depletion << " "<<
  MSY       << " "<<
  MSYL      << " "<<
  Fmsy      << " "<<
  Fcur_Fmsy << " "<<
  Bcur_Bmsy << " "<<
  Bmsy      << " "<<
  ABCBiom   << " "<<
  SSB_fut(1,endyr+1)        << " "<<
  B100                      << " "<<
  SSB_fut(1,endyr+1)/B100   << " "<<
  SSB_fut(1,endyr+2)/B100   << " "<<
  SSB_fut(1,endyr+3)/B100   << " "<<
  SSB_fut(1,endyr+4)/B100   << " "<<
  SSB_fut(1,endyr+5)/B100   << " "<<
  SSB_fut(1,endyr_fut)/B100 << " "<<
  catch_future(1,endyr+1)<< " "<<
  catch_future(1,endyr+2)<< " "<<
  catch_future(1,endyr+3)<< " "<<
  catch_future(1,endyr+4)<< " "<<
  catch_future(1,endyr+5)<< " "<<
  F35       << " "<<
  F40       << " "<<
  F50       << " "<<
  SSB_fut(1,endyr_fut) << " "<<
  SSB_fut(2,endyr_fut) << " "<<
  SSB_fut(3,endyr_fut) << " "<<
  SSB_fut(4,endyr_fut) << " "<<
  SSB_fut(5,endyr_fut) << " "<<
  catch_future(1,styr_fut)    << " "<<
  catch_future(2,styr_fut)    << " "<<
  catch_future(3,styr_fut)    << " "<<
  catch_future(4,styr_fut)    << " "<<  endl;


FUNCTION Get_Selectivity
  // Calculate the logistic selectivity (Only if being used...)
  for (k=1;k<=nfsh;k++)
  {
    switch (fsh_sel_opt(k))
    {
      case 1 : // Selectivity coefficients
      //---Calculate the fishery selectivity from the sel_coffs (Only if being used...)
      {
        int isel_ch_tmp = 1 ;
        dvar_vector sel_coffs_tmp(1,nselages_fsh(k));
        for (i=styr;i<=endyr;i++)
        {
          if (i==yrs_sel_ch_fsh(k,isel_ch_tmp))
          {
            sel_coffs_tmp.initialize();
            sel_coffs_tmp = log_selcoffs_fsh(k,isel_ch_tmp);
            avgsel_fsh(k,isel_ch_tmp)              = log(mean(mfexp(sel_coffs_tmp)));
            // Increment if there is still space to do so...
            if (isel_ch_tmp<n_sel_ch_fsh(k))
              isel_ch_tmp++;
          }
         // Need to flag for changing selectivity....XXX
          log_sel_fsh(k,i)(1,nselages_fsh(k))        = sel_coffs_tmp;
          log_sel_fsh(k,i)(nselages_fsh(k),nages)    = log_sel_fsh(k,i,nselages_fsh(k));
          log_sel_fsh(k,i)                                -= log(mean(mfexp(log_sel_fsh(k,i) )));
```

```
      }
    }
    break;
    case 2 : // Single logistic
    {
      sel_slope_fsh(k) = mfexp(logsel_slope_fsh(k));
      int isel_ch_tmp = 1 ;
      dvariable sel_slope_tmp = sel_slope_fsh(k,isel_ch_tmp);
      dvariable sel50_tmp     = sel50_fsh(k,isel_ch_tmp);
      for (i=styr;i<=endyr;i++)
      {
        if (i==yrs_sel_ch_fsh(k,isel_ch_tmp))
        {
          sel_slope_tmp = sel_slope_fsh(k,isel_ch_tmp);
          sel50_tmp     =     sel50_fsh(k,isel_ch_tmp);
          if (isel_ch_tmp<n_sel_ch_fsh(k))
            isel_ch_tmp++;
        }
        log_sel_fsh(k,i)(1,nselages_fsh(k))     = -1.*log( 1.0 + mfexp(-1.*sel_slope_tmp *
                                              ( age_vector(1,nselages_fsh(k)) - sel50_tmp) ));
        log_sel_fsh(k,i)(nselages_fsh(k),nages) = log_sel_fsh(k,i,nselages_fsh(k));
      }
  }
  break;
  case 3 : // Double logistic
  {
    sel_slope_fsh(k)  = mfexp(logsel_slope_fsh(k));
    sel_dslope_fsh(k) = mfexp(logsel_dslope_fsh(k));
    int isel_ch_tmp = 1 ;
    dvariable sel_slope_tmp = sel_slope_fsh(k,isel_ch_tmp);
    dvariable sel50_tmp     = sel50_fsh(k,isel_ch_tmp);
    dvariable sel_dslope_tmp = sel_dslope_fsh(k,isel_ch_tmp);
    dvariable seld50_tmp     = seld50_fsh(k,isel_ch_tmp);
    for (i=styr;i<=endyr;i++)
    {
      if (i==yrs_sel_ch_fsh(k,isel_ch_tmp))
      {
        sel_slope_tmp  = sel_slope_fsh(k,isel_ch_tmp);
        sel50_tmp      =     sel50_fsh(k,isel_ch_tmp);
        sel_dslope_tmp = sel_dslope_fsh(k,isel_ch_tmp);
        seld50_tmp     =     seld50_fsh(k,isel_ch_tmp);
        if (isel_ch_tmp<n_sel_ch_fsh(k))
          isel_ch_tmp++;
      }
      log_sel_fsh(k,i)(1,nselages_fsh(k))     =
                   -log( 1.0 + mfexp(-1.*sel_slope_tmp *
                   ( age_vector(1,nselages_fsh(k)) - sel50_tmp) ))+
                   log( 1. - 1/(1.0 + mfexp(-sel_dslope_tmp *
                   ( age_vector(1,nselages_fsh(k)) - seld50_tmp))) );

      log_sel_fsh(k,i)(nselages_fsh(k),nages) =
                   log_sel_fsh(k,i,nselages_fsh(k));

      log_sel_fsh(k,i) -= max(log_sel_fsh(k,i));
    }
  }
  break;
  //---Calculate the fishery selectivity from the sel_spl from nodes...
  case 4 : // Splines
  {
     int isel_ch_tmp = 1 ;
     dvar_matrix tempsel(1,n_sel_ch_fsh(k),1,nselages_fsh(k));
     dvector agetmp(1,nselages_fsh(k));
     agetmp.fill_seqadd(0.,1.0/(nselages_fsh(k)-1));
     // This needs to be dimensioned by the number of changes and the number of coefficients
     vcubic_spline_function_array a(1,n_sel_ch_fsh(k),xnodes_fsh(k),log_sel_spl_fsh(k));
     tempsel = a(agetmp);
     int j=1;
     // cout <<tempsel<<endl;exit(1);
     log_sel_fsh(k,styr)(1,nselages_fsh(k)) = tempsel(j);;
     avgsel_fsh(k,j)  = mean(log_sel_spl_fsh(k,j)); //log(mean(mfexp(log_sel_fsh(k,styr))));
```

```
      log_sel_fsh(k,styr)(nselages_fsh(k),nages) = log_sel_fsh(k,styr,nselages_fsh(k))  ;
      log_sel_fsh(k,styr)-= log(mean(mfexp(log_sel_fsh(k,styr) )));
     }
     break;
    } // End of switch for fishery selectivity type
  } // End of fishery loop
  // Survey specific---
  for (k=1;k<=nind;k++)
  {
    switch (ind_sel_opt(k))
    {
      case 1 : // Selectivity coefficients
      //---Calculate the fishery selectivity from the sel_coffs (Only if being used...)
      {
        int isel_ch_tmp = 1 ;
        dvar_vector sel_coffs_tmp(1,nselages_ind(k));
        for (i=styr;i<=endyr;i++)
        {
          if (i==yrs_sel_ch_ind(k,isel_ch_tmp))
          {
            sel_coffs_tmp.initialize();
            sel_coffs_tmp = log_selcoffs_ind(k,isel_ch_tmp);
            avgsel_ind(k,isel_ch_tmp)               = log(mean(mfexp(sel_coffs_tmp)));
            if (isel_ch_tmp<n_sel_ch_ind(k))
              isel_ch_tmp++;
          }
          log_sel_ind(k,i)(1,nselages_ind(k))        = sel_coffs_tmp;
          log_sel_ind(k,i)(nselages_ind(k),nages)    = log_sel_ind(k,i,nselages_ind(k));
          log_sel_ind(k,i)                                -=
log(mean(mfexp(log_sel_ind(k,i)(q_age_min(k),q_age_max(k)))));
        }
      }

        break;
      case 2 : // Asymptotic logistic
        {
          sel_slope_ind(k) = mfexp(logsel_slope_ind(k));
          int isel_ch_tmp = 1 ;
          dvariable sel_slope_tmp = sel_slope_ind(k,isel_ch_tmp);
          dvariable sel50_tmp     = sel50_ind(k,isel_ch_tmp);
          for (i=styr;i<=endyr;i++)
          {
            if (i==yrs_sel_ch_ind(k,isel_ch_tmp))
            {
              sel_slope_tmp = sel_slope_ind(k,isel_ch_tmp);
              sel50_tmp     =     sel50_ind(k,isel_ch_tmp);
              if (isel_ch_tmp<n_sel_ch_ind(k))
                isel_ch_tmp++;
            }
            log_sel_ind(k,i) = - log( 1.0 + mfexp(-sel_slope_tmp * ( age_vector - sel50_tmp) ));
            // log_sel_ind(k,i)                                -=
log(mean(mfexp(log_sel_ind(k,i)(q_age_min(k),q_age_max(k)))));
          }
        }
        break;
      case 3 : // Double logistic
        {
          sel_slope_ind(k)  = mfexp(logsel_slope_ind(k));
          sel_dslope_ind(k) = mfexp(logsel_dslope_ind(k));
          int isel_ch_tmp = 1 ;
          dvariable sel_slope_tmp = sel_slope_ind(k,isel_ch_tmp);
          dvariable sel50_tmp     = sel50_ind(k,isel_ch_tmp);
          dvariable sel_dslope_tmp = sel_dslope_ind(k,isel_ch_tmp);
          dvariable seld50_tmp     = seld50_ind(k,isel_ch_tmp);
          for (i=styr;i<=endyr;i++)
          {
            if (i==yrs_sel_ch_ind(k,isel_ch_tmp))
            {
              sel_slope_tmp  = sel_slope_ind(k,isel_ch_tmp);
              sel50_tmp      =     sel50_ind(k,isel_ch_tmp);
              sel_dslope_tmp = sel_dslope_ind(k,isel_ch_tmp);
```

```
            seld50_tmp     =     seld50_ind(k,isel_ch_tmp);
            if (isel_ch_tmp<n_sel_ch_ind(k))
              isel_ch_tmp++;
          }
          log_sel_ind(k,i)(1,nselages_ind(k))     =
                         -log( 1.0 + mfexp(-1.*sel_slope_tmp *
                         ( age_vector(1,nselages_ind(k)) - sel50_tmp) ))+
                         log( 1. - 1/(1.0 + mfexp(-sel_dslope_tmp *
                         ( age_vector(1,nselages_ind(k)) - seld50_tmp))) );

          log_sel_ind(k,i)(nselages_ind(k),nages) =
                         log_sel_ind(k,i,nselages_ind(k));

          log_sel_ind(k,i) -= max(log_sel_ind(k,i));
          log_sel_ind(k,i)                                 -=
log(mean(mfexp(log_sel_ind(k,i)(q_age_min(k),q_age_max(k)))));
        }
      }
    break;
  }// end of swtiches for indices selectivity
} // End of indices loop

// Map selectivities across fisheries and indices as needed.
for (k=1;k<=nfsh;k++)
  if (sel_map(2,k)!=k)  // If 2nd row shows a different fishery then use that fishery
    log_sel_fsh(k) = log_sel_fsh(sel_map(2,k));

for (k=1+nfsh;k<=nfsh_and_ind;k++)
  if (sel_map(1,k)!=2)
    log_sel_ind(k-nfsh) = log_sel_fsh(sel_map(2,k));
  else if (sel_map(2,k)!=(k-nfsh))
    log_sel_ind(k-nfsh) = log_sel_ind(sel_map(2,k));

sel_fsh = mfexp(log_sel_fsh);
sel_ind = mfexp(log_sel_ind);


FUNCTION Get_NatMortality
  surv    = mfexp(-Mest);
  natmort = Mest;
  // Age varying part
  if (npars_Mage>0 & (active(Mest) || active(Mage_offset)))
  {
    M(styr) = Mest;
    int jj=1;
    for (j=1;j<=nages;j++)
    {
      if (j==ages_M_changes(jj))
      {
        M(styr,j) = M(styr,1)*mfexp(Mage_offset(jj));
        jj++;
        if (npars_Mage < jj) jj=npars_Mage;
      }
      else
        if(j>1)
          M(styr,j) = M(styr,j-1);
    }
  }

  // Time varying part
  if (npars_rw_M>0 & active(M_rw))
  {
    int ii=1;
    for (i=styr+1;i<=endyr;i++)
    {
      if (i==yrs_rw_M(ii))
      {
        M(i) = M(i-1)*mfexp(M_rw(ii));
        ii++;
        if (npars_rw_M < ii) ii=npars_rw_M;
      }
```

```
      else
        M(i) = M(i-1);
    }
  }
  else
    for (i=styr+1;i<=endyr;i++)
      M(i) = M(i-1);

FUNCTION Get_Mortality2
  Get_NatMortality();
  Z       = M;
  for (k=1;k<=nfsh;k++)
  {
    F(k)   = elem_div(catage(k),natage);
    Z     += F(k);
  }
  S = mfexp(-1.*Z);

FUNCTION Get_Mortality
  Get_NatMortality();
  Z = M;
  if (!Popes)
  {
    Fmort.initialize();
    for (k=1;k<=nfsh;k++)
    {
      Fmort +=  fmort(k);
      for (i=styr;i<=endyr;i++)
      {
        F(k,i)   =  fmort(k,i) * sel_fsh(k,i) ;
        Z(i)    += F(k,i);
      }
    }
    S  = mfexp(-1.*Z);
  }



FUNCTION Get_Numbers_at_Age
  // natage(styr,1) = mfexp(mean_log_rec + rec_dev(styr));
  // Recruitment in subsequent years
  for (i=styr+1;i<=endyr;i++)
    natage(i,1)=mfexp(mean_log_rec+rec_dev(i));

  mod_rec(styr)  = natage(styr,1);

  for (i=styr;i<=endyr;i++)
  {
    if (Popes)
    {
      dvariable  t1=mfexp(-natmort(i)*0.5);
      dvariable  t2=mfexp(-natmort(i));
      Catch_at_Age(i);
      // Pope's approximation //   Next year N     =   This year x NatSurvivl - catch
      natage(i+1)(2,nages) = ++(natage(i)(1,nages-1)*t2 - catage_tot(i)(1,nages-1)*t1);
      Ftot(i)(1,nages-1) = log(natage(i)(1,nages-1)) - --log(natage(i+1)(2,nages)) - natmort(i);
      natage(i+1,nages)   += natage(i,nages)*t2 - catage_tot(i,nages)*t1;
      // Approximation to "F" continuous form for computing within-year sp biomass
      Ftot(i,nages)      = log(natage(i,nages-1)+natage(i,nages)) -log(natage(i+1,nages)) -
natmort(i);
      // write_input_log <<i<<" "<<Ftot(i)(nages-4,nages)<<endl; // cout <<i<<" "<<natage(i)<<endl;
// cout <<i<<" "<<natage(i+1)<<endl;
      dvariable ctmp=sum(catage_tot(i));
      for (k=1;k<=nfsh;k++)
      {
        F(k,i)  = Ftot(i) * sum(catage(k,i))/ctmp;
      }
      Z(i)    = Ftot(i)+natmort(i);
      S(i)    = mfexp(-Z(i));
    }
    else // Baranov
```

```
    {
      // get_Fs( i ); //ojo, add switch here for different catch equation XX
      // if (i!=endyr)
      // {
        natage(i+1)(2,nages) = ++elem_prod(natage(i)(1,nages-1),S(i)(1,nages-1));
        natage(i+1,nages)   +=natage(i,nages)*S(i,nages);
      // }
    }
    Catch_at_Age(i);
    Sp_Biom(i)  = elem_prod(natage(i),pow(S(i),spmo_frac)) * wt_mature;
    if (i<endyr) mod_rec(i+1)  = natage(i+1,1);
  }


FUNCTION Get_Survey_Predictions
  // Survey computations------------------
  dvariable sum_tmp;
  sum_tmp.initialize();
  int iyr;
  for (k=1;k<=nind;k++)
  {
    // Set rest of q's in time series equal to the random walk for current (avoids tricky tails...)
    for (i=1;i<=npars_rw_q(k);i++)
    {
      iyr = yrs_rw_q(k,i);
      q_ind(k,iyr)  = q_ind(k,iyr-1)*mfexp(log_rw_q_ind(k,i));
      for (int ii=yrs_rw_q(k,i);ii<=nyrs_ind(k);ii++)
        q_ind(k,ii)  = q_ind(k,iyr);
    }

    for (i=1;i<=nyrs_ind(k);i++)
    {
      iyr=yrs_ind(k,i);
      pred_ind(k,i) = q_ind(k,i) * pow(elem_prod(natage(iyr),pow(S(iyr),ind_month_frac(k))) *
                                     elem_prod(sel_ind(k,iyr) , wt_ind(k,iyr)),q_power_ind(k));
    }
    for (i=1;i<=nyrs_ind_age(k);i++)
    {
      iyr = int(yrs_ind_age(k,i));
      dvar_vector tmp_n   =
elem_prod(pow(S(iyr),ind_month_frac(k)),elem_prod(sel_ind(k,iyr),natage(iyr)));
      sum_tmp             = sum(tmp_n);
      if (use_age_err)
        eac_ind(k,i)      = age_err * tmp_n/sum_tmp;
      else
        eac_ind(k,i)      = tmp_n/sum_tmp;
    }
    iyr=yrs_ind(k,nyrs_ind(k));

    dvar_vector natagetmp = elem_prod(S(endyr),natage(endyr));
    natagetmp(2,nages) = ++natagetmp(1,nages-1);
    natagetmp(1)       = SRecruit(Sp_Biom(endyr+1-rec_age));
    natagetmp(nages)  += natage(endyr,nages)*S(endyr,nages);
    // Assume same survival in 1st part of next year as same as first part of current
    pred_ind_nextyr(k) = q_ind(k,nyrs_ind(k)) *
pow(elem_prod(natagetmp,pow(S(endyr),ind_month_frac(k))) *
                                     elem_prod(sel_ind(k,endyr) , wt_ind(k,endyr)),q_power_ind(k));
  }


FUNCTION Get_Fishery_Predictions
  for (k=1;k<=nfsh;k++)
  {
    for (i=1; i<=nyrs_fsh_age(k); i++)
      if (use_age_err)
        eac_fsh(k,i) = age_err * catage(k,yrs_fsh_age(k,i))/sum(catage(k,yrs_fsh_age(k,i)));
      else
        eac_fsh(k,i) = catage(k,yrs_fsh_age(k,i))/sum(catage(k,yrs_fsh_age(k,i)));
  }
```

```
FUNCTION Calc_Dependent_Vars
  // cout<<"In DepVar stage 1"<<endl;
  get_msy();
  if (phase_proj>0) Future_projections();

  N_NoFsh.initialize();
  N_NoFsh(styr) = natage(styr);
  for (i=styr_sp;i<=styr;i++)
    Sp_Biom_NoFish(i) = Sp_Biom(i);
  for (i=styr;i<=endyr;i++)
  {
    recruits(i)  = natage(i,1);
    if (i>styr)
    {
      N_NoFsh(i,1)        = recruits(i);
      N_NoFsh(i,1)       *= SRecruit(Sp_Biom_NoFish(i-rec_age)) / SRecruit(Sp_Biom(i-rec_age));
      N_NoFsh(i)(2,nages) = ++N_NoFsh(i-1)(1,nages-1)*exp(-natmort(i-1));
      N_NoFsh(i,nages)   += N_NoFsh(i-1,nages)*exp(-natmort(i-1));
    }
    totbiom_NoFish(i) = N_NoFsh(i)*wt_pop;
    totbiom(i)        = natage(i)*wt_pop;
    Sp_Biom_NoFish(i) = (N_NoFsh(i)*pow(exp(-natmort(i)),spmo_frac) * wt_mature);
    Sp_Biom_NoFishRatio(i) = Sp_Biom(i) / Sp_Biom_NoFish(i) ;
    // cout <<spmo_frac<<endl;exit(1);
    depletion         = totbiom(endyr)/totbiom(styr);
    depletion_dyn     = totbiom(endyr)/totbiom_NoFish(endyr);
  }
  B100 = phizero * mean(recruits(styr_rec_est, endyr_rec_est));
  dvar_vector Ntmp(1,nages);
  Ntmp(2,nages) = ++elem_prod(natage(endyr)(1,nages-1),S(endyr)(1,nages-1));
  Ntmp(nages)  += natage(endyr,nages)*S(endyr,nages);
  Ntmp(1)       = SRecruit(Sp_Biom(endyr+1-rec_age));
  ABCBiom       = Ntmp*wt_pop;
  recruits(endyr+1) = Ntmp(1);
  totbiom(endyr+1)  = ABCBiom;
  // Now do OFL for next year...
  dvar_vector ntmp(1,nages);
  dvar_matrix seltmp(1,nfsh,1,nages);
  dvar_matrix Fatmp(1,nfsh,1,nages);
  dvar_vector Ztmp(1,nages);
  seltmp.initialize();
  Fatmp.initialize();
  Ztmp.initialize();
  ntmp.initialize();
  for (k=1;k<=nfsh;k++)
    seltmp(k) = (sel_fsh(k,endyr));
  Ztmp = (natmort(styr));
  for (k=1;k<=nfsh;k++)
  {
    Fatmp(k) = (Fratio(k) * Fmsy * seltmp(k));
    Ztmp    += Fatmp(k);
  }
  dvar_vector survmsy = exp(-Ztmp);
  ntmp(1)       = (SRecruit(Sp_Biom(endyr+1-rec_age)));
  ntmp(2,nages) = ( ++elem_prod(natage(endyr)(1,nages-1),S(endyr)(1,nages-1)));
  ntmp(nages)  += ( natage(endyr,nages)*S(endyr,nages));
  dvar_vector ctmp(1,nages);
  ctmp.initialize();
  OFL=0.;
  for (k=1;k<=nfsh;k++)
  {
      for ( j=1 ; j <= nages; j++ )
        ctmp(j)      = ntmp(j) * Fatmp(k,j) * (1. - survmsy(j)) / Ztmp(j);
      OFL  += wt_fsh(k,endyr) * ctmp;
  }


FUNCTION void Catch_at_Age(const int& i)
  dvariable vbio=0.;
  dvariable pentmp;
  dvar_vector Nmid(1,nages);
```

```
  dvar_vector Ctmp(1,nages);
  catage_tot(i).initialize();
  if (Popes)
  {
    Nmid = natage(i)*mfexp(-natmort(i)/2 );
  }
  for (k=1;k<=nfsh;k++)
  {
    if (Popes)
    {
      pentmp=0.;
      Ctmp = elem_prod(Nmid,sel_fsh(k,i));
      vbio = Ctmp*wt_fsh(k,i);
      //Kludge to go here...
      // dvariable SK = posfun( (.98*vbio - catch_bio(k,i))/vbio , 0.1 , pentmp );
      dvariable SK = posfun( (vbio - catch_bio(k,i))/vbio , 0.1 , pentmp );
      catch_tmp     = vbio - SK*vbio;
      hrate         = catch_tmp / vbio;
      fpen(4) += pentmp;
      Ctmp *= hrate;
      if (hrate>1) {cout << catch_tmp<<" "<<vbio<<endl;exit(1);}
      catage_tot(i) += Ctmp;
      catage(k,i)     = Ctmp;
      if (last_phase())
        pred_catch(k,i) = Ctmp*wt_fsh(k,i);
    }
    else
    {
      catage(k,i) = elem_prod(elem_div(F(k,i),Z(i)),elem_prod(1.-S(i),natage(i)));
      pred_catch(k,i) = catage(k,i)*wt_fsh(k,i);
    }
  }
  //+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==

FUNCTION evaluate_the_objective_function
  // if (active(fmort_dev))
  if (active(fmort))
  {
    Cat_Like();
    Fmort_Pen();
  }
  Rec_Like();
  if (active(rec_dev))
    Age_Like();
  Srv_Like();
  Sel_Like();
  Compute_priors();
  if (active(log_Rzero)) // OjO
    obj_fun += .5 * square(log_Rzero-mean_log_rec); // A slight penalty to keep Rzero in reality...

  obj_comps.initialize();
  obj_comps(1) = sum(catch_like);
  obj_comps(2) = sum(age_like_fsh);
  obj_comps(3) = sum(sel_like_fsh);
  obj_comps(4) = sum(ind_like);
  obj_comps(5) = sum(age_like_ind);
  obj_comps(6) = sum(sel_like_ind);
  obj_comps(7) = sum(rec_like);
  obj_comps(8) = sum(fpen);
  obj_comps(9) = sum(post_priors_indq);
  obj_comps(10)= sum(post_priors);
  obj_fun     += sum(obj_comps);


FUNCTION Cat_Like
  // Eases into the catch-biomass likelihoods.  If too far off to start, full constraint to fit can
be too aggressive
  catch_like.initialize();
  dvariable catch_pen;
  switch (current_phase())
```

```
  {
    case 1:
      catch_pen = .1;
      break;
    case 2:
      catch_pen = .5;
      break;
    case 3:
      catch_pen = .8;
      break;
    case 4:
      catch_pen = 1.0;
      break;
    case 5:
      catch_pen = 1;
      break;
    default:
      catch_pen = 1;
      break;
  }
  if (current_phase()>3)
  {
    for (k=1;k<=nfsh;k++)
      for (i=styr;i<=endyr;i++)
         catch_like(k) += .5*square(log(catch_bio(k,i)+.0001) - log(pred_catch(k,i)+.0001)
)/catch_bio_lva(k,i);
  }
  else
  {
    for (k=1;k<=nfsh;k++)
      catch_like(k) += catchbiomass_pen * norm2(log(catch_bio(k)
                       +.000001) - log(pred_catch(k) +.000001));
  }

  catch_like *= catch_pen;


FUNCTION Rec_Like
  rec_like.initialize();
  if (active(rec_dev))
  {
    sigmar     =  mfexp(log_sigmar);
    sigmarsq   =  square(sigmar);
    if (current_phase()>2)
    {
      if (last_phase())
        pred_rec = SRecruit(Sp_Biom(styr_rec-rec_age,endyr-rec_age).shift(styr_rec)(styr_rec,endyr));
      else
        pred_rec = .1+SRecruit(Sp_Biom(styr_rec-rec_age,endyr-
rec_age).shift(styr_rec)(styr_rec,endyr));

      dvariable SSQRec;
      SSQRec.initialize();
      dvar_vector chi(styr_rec_est,endyr_rec_est);
      chi = log(mod_rec(styr_rec_est,endyr_rec_est)) - log(pred_rec(styr_rec_est,endyr_rec_est));
      SSQRec   = norm2( chi ) ;
      m_sigmarsq =  SSQRec/nrecs_est;
      m_sigmar   =  sqrt(m_sigmarsq);

      if (current_phase()>4||last_phase())
        rec_like(1) = (SSQRec+ m_sigmarsq/2.)/(2*sigmarsq) + nrecs_est*log_sigmar;
      else
        rec_like(1) = .1*(SSQRec+ m_sigmarsq/2.)/(2*sigmarsq) + nrecs_est*log_sigmar;
    }

    if (last_phase())
    {
      // Variance term for the parts not estimated by sr curve
      rec_like(4) += .5*norm2( rec_dev(styr_rec,styr_rec_est) )/sigmarsq + (styr_rec_est-
styr_rec)*log(sigmar) ;
```

```
      if ( endyr > endyr_rec_est)
        rec_like(4) += .5*norm2( rec_dev(endyr_rec_est,endyr  ) )/sigmarsq + (endyr-
endyr_rec_est)*log(sigmar) ;
    }
    else // JNI comment next line
       rec_like(2) += norm2( rec_dev(styr_rec_est,endyr) ) ;

    rec_like(2) += norm2( rec_dev(styr_rec_est,endyr) ) ;

    if (active(rec_dev_future))
    {
      // Future recruitment variability (based on past)
      sigmar_fut   = sigmar ;
      rec_like(3) += norm2(rec_dev_future)/(2*square(sigmar_fut))+
size_count(rec_dev_future)*log(sigmar_fut);
    }
  }


FUNCTION Compute_priors
  post_priors.initialize();
  post_priors_indq.initialize();
  for (k=1;k<=nind;k++)
  {
    if (active(log_q_ind(k)))
      post_priors_indq(k) += square(log(q_ind(k,1)/qprior(k)))/(2.*cvqprior(k)*cvqprior(k));

    if (active(log_q_power_ind(k)))
      post_priors_indq(k) +=
square(log(q_power_ind(k)/q_power_prior(k)))/(2.*cvq_power_prior(k)*cvq_power_prior(k));

    if (active(log_rw_q_ind(k)))
      for (int i=1;i<=npars_rw_q(k);i++)
      {
        post_priors_indq(k) += square(log_rw_q_ind(k,i))/
(2.*sigma_rw_q(k,yrs_rw_q(k,i))*sigma_rw_q(k,yrs_rw_q(k,i))) ;
        // cout<<"Hear "<<sigma_rw_q(k,yrs_rw_q(k,i))<<endl;
      }
     //  -q_power_prior(k))/(2*cvq_power_prior(k)*cvq_power_prior(k));
  }

  if (active(Mest))
    post_priors(1) += square(log(Mest/natmortprior))/(2.*cvnatmortprior*cvnatmortprior);

  if (active(Mage_offset))
    post_priors(1) += norm2(Mage_offset)/(2.*cvnatmortprior*cvnatmortprior);

  if (active(M_rw))
    for (int i=1;i<=npars_rw_M;i++)
      post_priors(1) +=  square(M_rw(i))/ (2.*sigma_rw_M(i)*sigma_rw_M(i)) ;

  if (active(steepness))
    post_priors(2) += square(log(steepness/steepnessprior))/(2*cvsteepnessprior*cvsteepnessprior);

  if (active(log_sigmar))
    post_priors(3) += square(log(sigmar/sigmarprior))/(2*cvsigmarprior*cvsigmarprior);


FUNCTION Fmort_Pen
  // Phases less than 3, penalize High F's---------------------------------
  if (current_phase()<3)
    fpen(1) += 1.* norm2(F - .2);
  else
    fpen(1) += 0.0001*norm2(F - .2);

  // for (k=1;k<=nfsh;k++)  fpen(2) += 20.*square(mean(fmort_dev(k)) ); // this is just a normalizing
constraint (fmort_devs sum to zero) }
```

```
FUNCTION Sel_Like
  sel_like_fsh.initialize();
  sel_like_ind.initialize();
  for (k=1;k<=nfsh;k++)
  {
    if (active(logsel_slope_fsh(k)))
    {
      for (i=2;i<=n_sel_ch_fsh(k);i++)
      {
          int iyr = yrs_sel_ch_fsh(k,i) ;
          dvariable var_tmp = square(sel_sigma_fsh(k,i));
          sel_like_fsh(k,2)    += .5*norm2( log_sel_fsh(k,iyr-1) - log_sel_fsh(k,iyr) ) / var_tmp ;
      }
    }

    if (active(log_selcoffs_fsh(k)))
    {
      for (i=1;i<=n_sel_ch_fsh(k);i++)
      {
        int iyr = yrs_sel_ch_fsh(k,i) ;
        sel_like_fsh(k,1) += curv_pen_fsh(k)*norm2(first_difference(
first_difference(log_sel_fsh(k,iyr))));
        if (i>1)
        {
          // This part is the penalty on the change itself--------------
          dvariable var_tmp = square(sel_sigma_fsh(k,i));
          sel_like_fsh(k,2)    += .5*norm2( log_sel_fsh(k,iyr-1) - log_sel_fsh(k,iyr) ) / var_tmp ;
        }
        int nagestmp = nselages_fsh(k);
        for (j=seldecage;j<=nagestmp;j++)
        {
          dvariable difftmp = log_sel_fsh(k,iyr,j-1)-log_sel_fsh(k,iyr,j) ;
          if (difftmp > 0.)
            sel_like_fsh(k,3)    += .5*square( difftmp ) / seldec_pen_fsh(k);
        }
        obj_fun            += 20 * square(avgsel_fsh(k,i)); // To normalize selectivities
      }
    }
  }
  for (k=1;k<=nind;k++)
  {
    if (active(logsel_slope_ind(k)))
    {
      for (i=2;i<=n_sel_ch_ind(k);i++)
      {
          int iyr = yrs_sel_ch_ind(k,i) ;
          dvariable var_tmp = square(sel_sigma_ind(k,i));
          sel_like_ind(k,2)    += .5*norm2( log_sel_ind(k,iyr-1) - log_sel_ind(k,iyr) ) / var_tmp ;
      }
    }
    if (active(log_selcoffs_ind(k)))
    {
      int nagestmp = nselages_ind(k);
      for (i=1;i<=n_sel_ch_ind(k);i++)
      {
        int iyr = yrs_sel_ch_ind(k,i) ;
        sel_like_ind(k,1) += curv_pen_ind(k)*norm2(first_difference(
first_difference(log_sel_ind(k,iyr))));
        // This part is the penalty on the change itself--------------
        if (i>1)
        {
          dvariable var_tmp = square(sel_sigma_ind(k,i));
          sel_like_ind(k,2)    += .5*norm2( log_sel_ind(k,iyr-1) - log_sel_ind(k,iyr) ) / var_tmp ;
        }
        for (j=seldecage;j<=nagestmp;j++)
        {
          dvariable difftmp = log_sel_ind(k,iyr,j-1)-log_sel_ind(k,iyr,j) ;
          if (difftmp > 0.)
            sel_like_ind(k,3)    += .5*square( difftmp ) / seldec_pen_ind(k);
        }
        obj_fun            += 20. * square(avgsel_ind(k,i));  // To normalize selectivities
```

```
      }
    }
  }


FUNCTION Srv_Like
  // Fit to indices (log-Normal) -------------------------------------------
  ind_like.initialize();
  int iyr;
  for (k=1;k<=nind;k++)
    for (i=1;i<=nyrs_ind(k);i++)
    {
      iyr = int(yrs_ind(k,i));
      ind_like(k) += square(log(obs_ind(k,i)) - log(pred_ind(k,i)) ) /
                                   (2.*obs_lse_ind(k,i)*obs_lse_ind(k,i));
    }
  /* normal distribution option to add someday...
    for (i=1;i<=nyrs_ind(k);i++)
      ind_like(k) += square(obs_ind(k,i) - pred_ind(k,yrs_ind(k,i)) ) /
                                   (2.*obs_se_ind(k,i)*obs_se_ind(k,i));
  */


FUNCTION Age_Like
  age_like_fsh.initialize();
  for (k=1;k<=nfsh;k++)
    for (int i=1;i<=nyrs_fsh_age(k);i++)
      age_like_fsh(k) -= n_sample_fsh_age(k,i)*(oac_fsh(k,i) + 0.001) * log(eac_fsh(k,i) + 0.001 ) ;
  age_like_fsh -= offset_fsh;

  age_like_ind.initialize();
  for (k=1;k<=nind;k++)
    for (int i=1;i<=nyrs_ind_age(k);i++)
      age_like_ind(k) -= n_sample_ind_age(k,i)*(oac_ind(k,i) + 0.001) * log(eac_ind(k,i) + 0.001 ) ;
  age_like_ind -= offset_ind;

FUNCTION Oper_Model
 // Initialize things used here only
  mc_count++;
  get_msy();
  Write_SimDatafile();
  Write_Datafile();
  dmatrix new_ind(1,nind,1,nyrs_ind);
  new_ind.initialize();

  int nsims;
  ifstream sim_in("nsims.dat");
  sim_in >> nsims; sim_in.close();

  dvector ran_ind_vect(1,nind);
  ofstream SaveOM("Om_Out.dat",ios::app);
  double C_tmp;
  dvariable Fnow;
  // system("cls"); cout<<"Number of replicates: "<<endl;
  // Initialize recruitment in first year
  for (i=styr_fut-rec_age;i<styr_fut;i++)
    Sp_Biom_future(i) = Sp_Biom(i);
  nage_future(styr_fut)(2,nages)              = ++elem_prod(natage(endyr)(1,nages-
1),S(endyr)(1,nages-1));
  nage_future(styr_fut,nages)                += natage(endyr,nages)*S(endyr,nages);

  // assume survival same as in last year...
  Sp_Biom_future(styr_fut) = elem_prod(nage_future(styr_fut),pow(S(endyr),spmo_frac)) * wt_mature;
  for (int isim=1;isim<=nsims;isim++)
  {
    cout<<isim<<" "<<cmp_no<<" "<<mc_count<<" "<<endl;
    // Copy file to get mean for Mgt Strategies
    system("init_stuff.bat");
    for (i=styr_fut;i<=endyr_fut;i++)
    {
```

```
      // Some unit normals...for generating data
      ran_ind_vect.fill_randn(rng);
      cout<<ran_ind_vect<<endl;
      // Create new indices observations
      // for (k = 1 ; k<= nind ; k++) new_ind(k) =
mfexp(ran_ind_vect(k)*.2)*value(nage_future(i)*q_ind(k,nyrs_ind(k))*sel_ind(k,endyr)); // use value
function since converts to a double
      // new_ind(1) = mfexp(ran_ind_vect(1)*0.2)*value(sum(nage_future(i)*q_ind(1,nyrs_ind(1))));
      if(styr_fut==i)
        new_ind(1) = mfexp(ran_ind_vect(1)*0.2)*value(wt_ind(1,endyr)*(natage(i-1)));
      else
        new_ind(1) = mfexp(ran_ind_vect(1)*0.2)*value(wt_ind(1,endyr)*(nage_future(i-1)));
      // now for Selecting which MP to use
      // Append new indices observation to datafile
      ifstream tacin("ctac.dat");
      int nobstmp;
      tacin >> nobstmp ;
      dvector t_tmp(1,nobstmp);
      tacin >> t_tmp;
      tacin.close();
      ofstream octac("ctac.dat");
      octac<<nobstmp+1<<endl;
      octac<<t_tmp<<endl;
      octac<<new_ind(1)<<endl;
      octac.close();
      system("ComputeTAC.bat " + (itoa(cmp_no,10))); // commandline function to get TAC
(catchnext.dat)
     // Now read in TAC (actual catch)
     ifstream CatchNext("CatchNext.dat");
     CatchNext >> C_tmp;
     CatchNext.close();
     //if (cmp_no==5) C_tmp=value((natmort(styr))*mean(t_tmp(nobstmp-2,nobstmp)));
     //if (cmp_no==6) C_tmp=value((natmort(styr))*.75*mean(t_tmp(nobstmp-2,nobstmp)));
     if (cmp_no==5)
     {
       C_tmp = min(C_tmp*1.1,value(natmort(styr)*(t_tmp(nobstmp))));
       ofstream cnext("CatchNext.dat");
       cnext <<C_tmp<<endl;
       cnext.close();
     }
     if (cmp_no==6)
     {
       C_tmp = min(C_tmp*1.1,value(natmort(styr)*.75*(t_tmp(nobstmp))));
       ofstream cnext("CatchNext.dat");
       cnext <<C_tmp<<endl;
       cnext.close();
     }

     Fnow = SolveF2(endyr,nage_future(i), C_tmp);

      F_future(1,i) = sel_fsh(1,endyr) * Fnow;
      //Z_future(i)   = F_future(1,i) + max(natmort);
      Z_future(i)   = F_future(1,i) + mean(natmort);
      S_future(i)   = mfexp(-Z_future(i));
      nage_future(i,1)  = SRecruit( Sp_Biom_future(i-rec_age) ) * mfexp(rec_dev_future(i)) ;
      Sp_Biom_future(i) = wt_mature * elem_prod(nage_future(i),pow(S_future(i),spmo_frac)) ;
      // Now graduate for the next year....
      if (i<endyr_fut)
      {
        nage_future(i+1)(2,nages) = ++elem_prod(nage_future(i)(1,nages-1),S_future(i)(1,nages-1));
        nage_future(i+1,nages)   += nage_future(i,nages)*S_future(i,nages);
      }
      catage_future(i) = 0.;
      for (k = 1 ; k<= nfsh ; k++)
        catage_future(i) += elem_prod(nage_future(i) , elem_prod(F_future(k,i) , elem_div( ( 1.-
S_future(i) ) , Z_future(i))));

      SaveOM << model_name       <<
        " "  << cmp_no           <<
        " "  << mc_count         <<
        " "  << isim             <<
```

```
        " "  << i                <<
        " "  << Fnow             <<
        " "  << Fnow/Fmsy        <<
        " "  << Sp_Biom_future(i-rec_age)                       <<
        " "  << nage_future(i)                                  <<
        " "  << catage_future(i)*wt_fsh(1,endyr)                <<
        " "  << mean(natmort)                                   <<
        " "  << t_tmp(nobstmp)                                  <<
      endl;
    }
  }
  // if (mc_count>5) exit(1);
  SaveOM.close();
  if (!mceval_phase())
    exit(1);


FUNCTION void get_future_Fs(const int& i,const int& iscenario)
    // get F's
    switch (iscenario)
    {
      case 1:
        f_tmp = F50;
        // cout <<"F35 "<<i<<" "<<f_tmp<<endl;
        break;
      case 2:
        // f_tmp = SolveF2(endyr,nage_future(i), 1.0  * sum(catch_bioT(endyr)) );
        for (int k=1;k<=nfsh;k++) f_tmp(k) = Fratio(k)*Fmsy; // mean(F(k,endyr));
        break;
      case 3:
        // f_tmp = SolveF2(endyr,nage_future(i), 0.5 * sum(catch_bioT(endyr)) );
        for (int k=1;k<=nfsh;k++) f_tmp(k) = .5*mean(F(k,endyr));
        break;
      case 4:
        f_tmp = SolveF2(endyr,nage_future(i), 0.25 * sum(catch_bioT(endyr)) );
        // New control rule
        //if (Sp_Biom_NoFishRatio(i) < 0.4)
          //f_tmp = Fmsy * Sp_Biom_NoFishRatio(i)/.40;
        //else
          //f_tmp = F40;
        break;
      case 5:
        f_tmp = 0.0;
        break;
    }
    Z_future(i) = natmort(endyr);
    for (k=1;k<=nfsh;k++)
    {
      F_future(k,i) = sel_fsh(k,endyr) * f_tmp(k);
      Z_future(i)  += F_future(k,i);
    }
    S_future(i) = mfexp(-Z_future(i));

FUNCTION Future_projections
  // Need to check on treatment of Fratio--whether it should be included or not
  SSB_fut.initialize();
  catch_future.initialize();
  /* for (i=styr_fut-rec_age;i<styr_fut;i++)
    Sp_Biom_future(i) = wt_mature * elem_prod(natage(i),pow(S(i),spmo_frac)) ;
  for (i=endyr+1;i<=endyr_fut;i++)
  {
    nage_future(styr_fut)(2,nages) = ++elem_prod(natage(endyr)(1,nages-1),S(endyr)(1,nages-1));
    nage_future(styr_fut,nages)   += natage(endyr,nages)*S(endyr,nages);
    nage_future(i,1)  = SRecruit( Sp_Biom_future(i-rec_age) ) * mfexp(rec_dev_future(i)) ;
    N_NoFsh(i,1)        = nage_future(i,1);
    // Adjustment for no-fishing recruits (ratio of R_nofish/R_fish)
    N_NoFsh(i,1)       *= SRecruit(Sp_Biom_NoFish(i-rec_age)) / SRecruit(Sp_Biom_future(i-rec_age));
    N_NoFsh(i)(2,nages) = ++N_NoFsh(i-1)(1,nages-1)*exp(-mean(natmort));
    N_NoFsh(i,nages)    +=   N_NoFsh(i-1,nages)*exp(-mean(natmort));
    Sp_Biom_NoFish(i)    = (N_NoFsh(i)*pow(exp(-mean(natmort)),spmo_frac) * wt_mature);
    // Sp_Biom_NoFishRatio(i)  = Sp_Biom_future(i) / Sp_Biom_NoFish(i) ;
```

```
  }
  */
  for (int iscen=1;iscen<=5;iscen++)
  {
   // Future Sp_Biom set equal to estimated Sp_Biom w/ right lag
    // Sp_Biom_future(styr_fut-rec_age,styr_fut-1) = Sp_Biom(endyr-rec_age+1,endyr);
    for (i=styr_fut-rec_age;i<styr_fut;i++)
      Sp_Biom_future(i) = wt_mature * elem_prod(natage(i),pow(S(i),spmo_frac)) ;

    // cout<<Sp_Biom(endyr-10,endyr)<<endl<<Sp_Biom_future<<endl;exit(1);
    nage_future(styr_fut)(2,nages) = ++elem_prod(natage(endyr)(1,nages-1),S(endyr)(1,nages-1));
    nage_future(styr_fut,nages)   += natage(endyr,nages)*S(endyr,nages);
    Sp_Biom_future(styr_fut)       = wt_mature * elem_prod(nage_future(i),pow(S_future(i),spmo_frac))
;
    // Future Recruitment (and Sp_Biom)
    for (i=styr_fut;i<endyr_fut;i++)
    {
      nage_future(i,1)  = SRecruit( Sp_Biom_future(i-rec_age) ) * mfexp(rec_dev_future(i)) ;
      get_future_Fs(i,iscen);
      // Now graduate for the next year....
      nage_future(i+1)(2,nages) = ++elem_prod(nage_future(i)(1,nages-1),S_future(i)(1,nages-1));
      nage_future(i+1,nages)   += nage_future(i,nages)*S_future(i,nages);
      Sp_Biom_future(i) = wt_mature * elem_prod(nage_future(i),pow(S_future(i),spmo_frac)) ;
    }
    nage_future(endyr_fut,1)  = SRecruit( Sp_Biom_future(endyr_fut-rec_age) ) *
mfexp(rec_dev_future(endyr_fut)) ;
    get_future_Fs(endyr_fut,iscen);
    Sp_Biom_future(endyr_fut)  = wt_mature *
elem_prod(nage_future(endyr_fut),pow(S_future(endyr_fut),spmo_frac)) ;
    // cout<<mean(natmort)<<endl;
    // Now get catch at future ages
    for (i=styr_fut; i<=endyr_fut; i++)
    {
      catage_future(i) = 0.;
      for (k = 1 ; k<= nfsh ; k++)
      {
        catage_future(i) += elem_prod(nage_future(i) , elem_prod(F_future(k,i) ,
                             elem_div( ( 1.- S_future(i) ) , Z_future(i))));
        if (iscen!=5)
          catch_future(iscen,i)   += catage_future(i)*wt_fsh(k,endyr);
      }
      SSB_fut(iscen,i) = Sp_Biom_future(i);
      // cout<<iscen<<" "<<i<<" "<< SSB_fut(iscen,i) <<" "<<nage_future(i)(1,4)<<endl;
    }
  }   //End of loop over F's
  Sp_Biom(endyr+1) = Sp_Biom_future(endyr+1);


FUNCTION get_msy
 /*Function calculates used in calculating MSY and MSYL for a designated component of the
  population, given values for stock recruitment and selectivity...
  Fmsy is the trial value of MSY example of the use of "funnel" to reduce the amount of storage for
derivative calculations */

  dvariable sumF=0.;
  for (k=1;k<=nfsh;k++)
    sumF += sum(F(k,endyr));
  for (k=1;k<=nfsh;k++)
    Fratio(k) = sum(F(k,endyr)) / sumF;

  dvariable Stmp;
  dvariable Rtmp;
  double df=1.e-05;
  dvariable F1;
  F1.initialize();
  F1 = (0.8*natmortprior);
  dvariable F2;
  dvariable F3;
  dvariable yld1;
  dvariable yld2;
  dvariable yld3;
```

```
  dvariable dyld;
  dvariable dyldp;
  int breakout=0;
  // Newton Raphson stuff to go here
  for (int ii=1;ii<=8;ii++)
  {
    if (mceval_phase()&&(F1>5||F1<0.01))
    {
      ii=8;
      if (F1>5) F1=5.0;
      else      F1=0.001;
      breakout    = 1;
    }
    F2     = F1 + df*.5;
    F3     = F2 - df;
    // yld1    = yield(Fratio,F1, Stmp,Rtmp); // yld2   = yield(Fratio,F2,Stmp,Rtmp); // yld3   =
yield(Fratio,F3,Stmp,Rtmp);
    yld1    = yield(Fratio,F1);
    yld2    = yield(Fratio,F2);
    yld3    = yield(Fratio,F3);
    dyld    = (yld2 - yld3)/df;                          // First derivative (to find the root of
this)
    dyldp  = (yld2 + yld3 - 2.*yld1)/(.25*df*df);       // Second derivative (for Newton Raphson)
    if (breakout==0)
    {
      F1    -= dyld/dyldp;
    }
    else
    {
      if (F1>5)
        cout<<"Fmsy v. high "<< endl;// yld1<<" "<< yld2<<" "<< yld3<<" "<< F1<<" "<< F2<<" "<< F3<<"
"<< endl;
      else
        cout<<"Fmsy v. low "<< endl;// yld1<<" "<< yld2<<" "<< yld3<<" "<< F1<<" "<< F2<<" "<< F3<<"
"<< endl;
    }
  }
  {
    dvar_vector ttt(1,5);
    ttt      = yld(Fratio,F1);
    Fmsy     = F1;
    Rtmp     = ttt(3);
    MSY      = ttt(2);
    Bmsy     = ttt(1);
    MSYL     = ttt(1)/Bzero;
    lnFmsy   = log(MSY/ttt(5)); // Exploitation fraction relative to total biomass
    Bcur_Bmsy= Sp_Biom(endyr)/Bmsy;

    dvariable FFtmp;
    FFtmp.initialize();
    for (k=1;k<=nfsh;k++)
      FFtmp += mean(F(k,endyr));
    Fcur_Fmsy= FFtmp/Fmsy;
    Rmsy     = Rtmp;
  }


FUNCTION void get_msy(int iyr)
 /*Function calculates used in calculating MSY and MSYL for a designated component of the
  population, given values for stock recruitment and selectivity...
  Fmsy is the trial value of MSY example of the use of "funnel" to reduce the amount of storage for
derivative calculations */

  dvariable sumF=0.;
  for (k=1;k<=nfsh;k++)
    sumF += sum(F(k,iyr));
  for (k=1;k<=nfsh;k++)
    Fratio(k) = sum(F(k,iyr)) / sumF;

  dvariable Stmp;
  dvariable Rtmp;
```

```
  double df=1.e-05;
  dvariable F1;
  F1.initialize();
  F1 = (0.8*natmortprior);
  dvariable F2;
  dvariable F3;
  dvariable yld1;
  dvariable yld2;
  dvariable yld3;
  dvariable dyld;
  dvariable dyldp;
  int breakout=0;
  // Newton Raphson stuff to go here
  for (int ii=1;ii<=8;ii++)
  {
    if (mceval_phase()&&(F1>5||F1<0.01))
    {
      ii=8;
      if (F1>5) F1=5.0;
      else      F1=0.001;
      breakout    = 1;
    }
    F2     = F1 + df*.5;
    F3     = F2 - df;
    // yld1   = yield(Fratio,F1, Stmp,Rtmp); // yld2   = yield(Fratio,F2,Stmp,Rtmp); // yld3   =
yield(Fratio,F3,Stmp,Rtmp);
    yld1   = yield(Fratio,F1,iyr);
    yld2   = yield(Fratio,F2,iyr);
    yld3   = yield(Fratio,F3,iyr);
    dyld   = (yld2 - yld3)/df;                          // First derivative (to find the root of
this)
    dyldp  = (yld2 + yld3 - 2.*yld1)/(.25*df*df);   // Second derivative (for Newton Raphson)
    if (breakout==0)
    {
      F1    -= dyld/dyldp;
    }
    else
    {
      if (F1>5)
        cout<<"Fmsy v. high "<< endl;// yld1<<" "<< yld2<<" "<< yld3<<" "<< F1<<" "<< F2<<" "<< F3<<"
"<< endl;
      else
        cout<<"Fmsy v. low "<< endl;// yld1<<" "<< yld2<<" "<< yld3<<" "<< F1<<" "<< F2<<" "<< F3<<"
"<< endl;
    }
  }
  {
    dvar_vector ttt(1,5);
    ttt      = yld(Fratio,F1,iyr);
    Fmsy     = F1;
    Rtmp     = ttt(3);
    MSY      = ttt(2);
    Bmsy     = ttt(1);
    MSYL     = ttt(1)/Bzero;
    lnFmsy   = log(MSY/ttt(5)); // Exploitation fraction relative to total biomass
    Bcur_Bmsy= Sp_Biom(iyr)/Bmsy;

    dvariable FFtmp;
    FFtmp.initialize();
    for (k=1;k<=nfsh;k++)
      FFtmp += mean(F(k,iyr));
    Fcur_Fmsy= FFtmp/Fmsy;
    Rmsy     = Rtmp;
  }


FUNCTION dvar_vector yld(const dvar_vector& Fratio, const dvariable& Ftmp,int iyr)
  RETURN_ARRAYS_INCREMENT();
  /*dvariable utmp=1.-mfexp(-(Ftmp)); dvariable Ntmp; dvariable Btmp; dvariable yield; dvariable
survtmp=exp(-1.*natmort); dvar_vector seltmp=sel_fsh(endyr); Ntmp = 1.; Btmp = Ntmp*wt(1)*seltmp(1);
Stmp = .5*Ntmp*wt(1)*maturity(1); yield= 0.; for ( j=1 ; j < nages ; j++ ) { Ntmp  *= (1.-
```

```
utmp*seltmp(j))*survtmp; Btmp  += Ntmp*wt(j+1)*seltmp(j+1); Stmp  += .5 * Ntmp
*wt(j+1)*maturity(j+1); } //Max Age - 1 yr yield   += utmp * Btmp; Ntmp   /= (1-survtmp*(1.-
utmp*seltmp(nages))); Btmp   += Ntmp*wt(nages)*seltmp(nages); Stmp   += 0.5 *wt(nages)* Ntmp
*maturity(nages); yield  += utmp * Btmp; //cout<<yield<<" "<<Stmp<<" "<<Btmp<<" ";*/
  dvar_vector msy_stuff(1,5);
  dvariable phi;
  dvar_vector Ntmp(1,nages);
  dvar_vector Ctmp(1,nages);
  msy_stuff.initialize();

  dvar_matrix seltmp(1,nfsh,1,nages);
  for (k=1;k<=nfsh;k++)
   seltmp(k) = sel_fsh(k,iyr); // NOTE uses last-year of fishery selectivity for projections.

  dvar_matrix Fatmp(1,nfsh,1,nages);
  dvar_vector Ztmp(1,nages);

  Ztmp = natmort(iyr);
  for (k=1;k<=nfsh;k++)
  {
    Fatmp(k) = Fratio(k) * Ftmp * seltmp(k);
    Ztmp    += Fatmp(k);
  }
  dvar_vector survtmp = mfexp(-Ztmp);

  Ntmp(1) = 1.;
  for ( j=1 ; j < nages; j++ )
    Ntmp(j+1)  =   Ntmp(j) * survtmp(j); // Begin numbers in the next year/age class
  Ntmp(nages)  /= (1.- survtmp(nages));

  for (k=1;k<=nfsh;k++)
  {
    Ctmp.initialize();
    for ( j=1 ; j <= nages; j++ )
      Ctmp(j)      = Ntmp(j) * Fatmp(k,j) * (1. - survtmp(j)) / Ztmp(j);

    msy_stuff(2)  += wt_fsh(k,iyr) * Ctmp;
  }
  phi    = elem_prod( Ntmp , pow(survtmp,spmo_frac ) ) * wt_mature;
  // Req    = Requil(phi) * exp(sigmarsq/2);
  msy_stuff(5)  = Ntmp * wt_pop;
  msy_stuff(4)  = phi/phizero ;       // SPR
  msy_stuff(3)  = Requil(phi) ;       // Eq Recruitment
  msy_stuff(5) *= msy_stuff(3);       // BmsyTot
  msy_stuff(2) *= msy_stuff(3);       // MSY
  msy_stuff(1)  = phi*(msy_stuff(3)); // Bmsy
  RETURN_ARRAYS_DECREMENT();
  return msy_stuff;

 //+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+

FUNCTION dvar_vector yld(const dvar_vector& Fratio, const dvariable& Ftmp)
  RETURN_ARRAYS_INCREMENT();
  /*dvariable utmp=1.-mfexp(-(Ftmp)); dvariable Ntmp; dvariable Btmp; dvariable yield; dvariable
survtmp=exp(-1.*natmort); dvar_vector seltmp=sel_fsh(endyr); Ntmp = 1.; Btmp = Ntmp*wt(1)*seltmp(1);
Stmp = .5*Ntmp*wt(1)*maturity(1); yield= 0.; for ( j=1 ; j < nages ; j++ ) { Ntmp  *= (1.-
utmp*seltmp(j))*survtmp; Btmp  += Ntmp*wt(j+1)*seltmp(j+1); Stmp  += .5 * Ntmp
*wt(j+1)*maturity(j+1); } //Max Age - 1 yr yield   += utmp * Btmp; Ntmp   /= (1-survtmp*(1.-
utmp*seltmp(nages))); Btmp   += Ntmp*wt(nages)*seltmp(nages); Stmp   += 0.5 *wt(nages)* Ntmp
*maturity(nages); yield  += utmp * Btmp; //cout<<yield<<" "<<Stmp<<" "<<Btmp<<" ";*/
  dvar_vector msy_stuff(1,5);
  dvariable phi;
  dvar_vector Ntmp(1,nages);
  dvar_vector Ctmp(1,nages);
  msy_stuff.initialize();

  dvar_matrix seltmp(1,nfsh,1,nages);
  for (k=1;k<=nfsh;k++)
   seltmp(k) = sel_fsh(k,endyr); // NOTE uses last-year of fishery selectivity for projections.

  dvar_matrix Fatmp(1,nfsh,1,nages);
```

```
  dvar_vector Ztmp(1,nages);

  Ztmp = natmort(styr);
  for (k=1;k<=nfsh;k++)
  {
    Fatmp(k) = Fratio(k) * Ftmp * seltmp(k);
    Ztmp    += Fatmp(k);
  }
  dvar_vector survtmp = mfexp(-Ztmp);

  Ntmp(1) = 1.;
  for ( j=1 ; j < nages; j++ )
    Ntmp(j+1)  =   Ntmp(j) * survtmp(j); // Begin numbers in the next year/age class
  Ntmp(nages)  /= (1.- survtmp(nages));

  for (k=1;k<=nfsh;k++)
  {
    Ctmp.initialize();
    for ( j=1 ; j <= nages; j++ )
      Ctmp(j)      = Ntmp(j) * Fatmp(k,j) * (1. - survtmp(j)) / Ztmp(j);

    msy_stuff(2)  += wt_fsh(k,endyr) * Ctmp;
  }
  phi    = elem_prod( Ntmp , pow(survtmp,spmo_frac ) ) * wt_mature;
  // Req    = Requil(phi) * exp(sigmarsq/2);
  msy_stuff(5)  = Ntmp * wt_pop;
  msy_stuff(4)  = phi/phizero ;       // SPR
  msy_stuff(3)  = Requil(phi) ;       // Eq Recruitment
  msy_stuff(5) *= msy_stuff(3);       // BmsyTot
  msy_stuff(2) *= msy_stuff(3);       // MSY
  msy_stuff(1)  = phi*(msy_stuff(3)); // Bmsy
  RETURN_ARRAYS_DECREMENT();
  return msy_stuff;


FUNCTION dvariable yield(const dvar_vector& Fratio, const dvariable& Ftmp,int iyr)
  RETURN_ARRAYS_INCREMENT();
  /*dvariable utmp=1.-mfexp(-(Ftmp)); dvariable Ntmp; dvariable Btmp; dvariable yield; dvariable
survtmp=exp(-1.*natmort); dvar_vector seltmp=sel_fsh(endyr); Ntmp = 1.; Btmp = Ntmp*wt(1)*seltmp(1);
Stmp = .5*Ntmp*wt(1)*maturity(1); yield= 0.; for ( j=1 ; j < nages ; j++ ) { Ntmp  *= (1.-
utmp*seltmp(j))*survtmp; Btmp  += Ntmp*wt(j+1)*seltmp(j+1); Stmp  += .5 * Ntmp
*wt(j+1)*maturity(j+1); } //Max Age - 1 yr yield   += utmp * Btmp; Ntmp    /= (1-survtmp*(1.-
utmp*seltmp(nages))); Btmp    += Ntmp*wt(nages)*seltmp(nages); Stmp    += 0.5 *wt(nages)* Ntmp
*maturity(nages); yield   += utmp * Btmp; //cout<<yield<<" "<<Stmp<<" "<<Btmp<<" ";*/
  dvariable phi;
  dvariable Req;
  dvar_vector Ntmp(1,nages);
  dvar_vector Ctmp(1,nages);
  dvariable   yield;
  yield.initialize();

  dvar_matrix seltmp(1,nfsh,1,nages);
  for (k=1;k<=nfsh;k++)
   seltmp(k) = sel_fsh(k,iyr); // NOTE uses last-year of fishery selectivity for projections.

  dvar_matrix Fatmp(1,nfsh,1,nages);
  dvar_vector Ztmp(1,nages);

  Ztmp = natmort(iyr);
  for (k=1;k<=nfsh;k++)
  {
    Fatmp(k) = Fratio(k) * Ftmp * seltmp(k);
    Ztmp    += Fatmp(k);
  }
  dvar_vector survtmp = mfexp(-Ztmp);
  // cout<<Ftmp<<" ";

  Ntmp(1) = 1.;
  for ( j=1 ; j < nages; j++ )
    Ntmp(j+1)  =   Ntmp(j) * survtmp(j); // Begin numbers in the next year/age class
  Ntmp(nages)  /= (1.- survtmp(nages));
```

```
  for (k=1;k<=nfsh;k++)
  {
    Ctmp.initialize();
    for ( j=1 ; j <= nages; j++ )
      Ctmp(j)      = Ntmp(j) * Fatmp(k,j) * (1. - survtmp(j)) / Ztmp(j);

    yield  += wt_fsh(k,iyr) * Ctmp;
  }
  phi    = elem_prod( Ntmp , pow(survtmp,spmo_frac ) )* wt_mature;
  // Req    = Requil(phi) * mfexp(sigmarsq/2);
  Req    = Requil(phi) ;
  yield *= Req;

  RETURN_ARRAYS_DECREMENT();
  return yield;


FUNCTION dvariable yield(const dvar_vector& Fratio, const dvariable& Ftmp)
  RETURN_ARRAYS_INCREMENT();
  /*dvariable utmp=1.-mfexp(-(Ftmp)); dvariable Ntmp; dvariable Btmp; dvariable yield; dvariable
survtmp=exp(-1.*natmort); dvar_vector seltmp=sel_fsh(endyr); Ntmp = 1.; Btmp = Ntmp*wt(1)*seltmp(1);
Stmp = .5*Ntmp*wt(1)*maturity(1); yield= 0.; for ( j=1 ; j < nages ; j++ ) { Ntmp  *= (1.-
utmp*seltmp(j))*survtmp; Btmp  += Ntmp*wt(j+1)*seltmp(j+1); Stmp  += .5 * Ntmp
*wt(j+1)*maturity(j+1); } //Max Age - 1 yr yield   += utmp * Btmp; Ntmp   /= (1-survtmp*(1.-
utmp*seltmp(nages))); Btmp    += Ntmp*wt(nages)*seltmp(nages); Stmp    += 0.5 *wt(nages)* Ntmp
*maturity(nages); yield   += utmp * Btmp; //cout<<yield<<" "<<Stmp<<" "<<Btmp<<" ";*/
  dvariable phi;
  dvariable Req;
  dvar_vector Ntmp(1,nages);
  dvar_vector Ctmp(1,nages);
  dvariable   yield;
  yield.initialize();

  dvar_matrix seltmp(1,nfsh,1,nages);
  for (k=1;k<=nfsh;k++)
   seltmp(k) = sel_fsh(k,endyr); // NOTE uses last-year of fishery selectivity for projections.

  dvar_matrix Fatmp(1,nfsh,1,nages);
  dvar_vector Ztmp(1,nages);

  Ztmp = natmort(styr);
  for (k=1;k<=nfsh;k++)
  {
    Fatmp(k) = Fratio(k) * Ftmp * seltmp(k);
    Ztmp    += Fatmp(k);
  }
  dvar_vector survtmp = mfexp(-Ztmp);
  // cout<<Ftmp<<" ";

  Ntmp(1) = 1.;
  for ( j=1 ; j < nages; j++ )
    Ntmp(j+1)  =   Ntmp(j) * survtmp(j); // Begin numbers in the next year/age class
  Ntmp(nages)  /= (1.- survtmp(nages));

  for (k=1;k<=nfsh;k++)
  {
    Ctmp.initialize();
    for ( j=1 ; j <= nages; j++ )
      Ctmp(j)      = Ntmp(j) * Fatmp(k,j) * (1. - survtmp(j)) / Ztmp(j);

    yield  += wt_fsh(k,endyr) * Ctmp;
  }
  phi    = elem_prod( Ntmp , pow(survtmp,spmo_frac ) )* wt_mature;
  // Req    = Requil(phi) * mfexp(sigmarsq/2);
  Req    = Requil(phi) ;
  yield *= Req;

  RETURN_ARRAYS_DECREMENT();
  return yield;
```

```
FUNCTION dvariable yield(const dvar_vector& Fratio, dvariable& Ftmp, dvariable& Stmp,dvariable& Req)
  RETURN_ARRAYS_INCREMENT();
  dvariable phi;
  dvar_vector Ntmp(1,nages);
  dvar_vector Ctmp(1,nages);
  dvariable   yield   = 0.;

  dvar_matrix seltmp(1,nfsh,1,nages);
  for  (k=1;k<=nfsh;k++)
   seltmp(k) = sel_fsh(k,endyr); // NOTE uses last-year of fishery selectivity for projections.

  dvar_matrix Fatmp(1,nfsh,1,nages);
  dvar_vector Ztmp(1,nages);

  Ztmp = natmort(styr);
  for  (k=1;k<=nfsh;k++)
  {
    Fatmp(k) = Fratio(k) * Ftmp * seltmp(k);
    Ztmp    += Fatmp(k);
  }
  dvar_vector survtmp = mfexp(-Ztmp);

  Ntmp(1) = 1.;
  for ( j=1 ; j < nages; j++ )
    Ntmp(j+1)  =   Ntmp(j) * survtmp(j); // Begin numbers in the next year/age class
  Ntmp(nages)  /= (1.- survtmp(nages));
  for  (k=1;k<=nfsh;k++)
  {
    Ctmp.initialize();
    for ( j=1 ; j <= nages; j++ )
      Ctmp(j)      = Ntmp(j) * Fatmp(k,j) * (1. - survtmp(j)) / Ztmp(j);
    yield  += wt_fsh(k,endyr) * Ctmp;
  }
  phi    = elem_prod( Ntmp , pow(survtmp,spmo_frac ) )* wt_mature;
  // Req    = Requil(phi) * exp(sigmarsq/2);
  Req    = Requil(phi) ;
  yield *= Req;
  Stmp   = phi*Req;

  RETURN_ARRAYS_DECREMENT();
  return yield;


FUNCTION Profile_F
  cout << "Doing a profile over F...."<<endl;
  ofstream prof_F("Fprof.yld");
 /* NOTE THis will need to be conditional on SrType too Function calculates
  used in calculating MSY and MSYL for a designated component of the
  population, given values for stock recruitment and selectivity...
  Fmsy is the trial value of MSY example of the use of "funnel" to
  reduce the amount of storage for derivative calculations */
 dvariable sumF=0.;
  for  (k=1;k<=nfsh;k++)
    sumF += sum(F(k,endyr));
  for  (k=1;k<=nfsh;k++)
    Fratio(k) = sum(F(k,endyr)) / sumF;
  dvariable Stmp;
  dvariable Rtmp;
  double df=1.e-7;
  dvariable F1=.05;
  dvariable F2;
  dvariable F3;
  dvariable yld1;
  dvariable yld2;
  dvariable yld3;
  dvariable dyld;
  dvariable dyldp;
  prof_F <<"Profile of stock, yield, and recruitment over F"<<endl;
  prof_F << model_name<<" "<<datafile_name<<endl;
  prof_F <<endl<<endl<<"F  Stock  Yld  Recruit SPR"<<endl;
  prof_F <<0.0<<" "<< Bzero <<" "<<0.0<<" "<<Rzero<< " 1.00"<<endl;
```

```
  dvar_vector ttt(1,5);
  for (int ii=1;ii<=500;ii++)
  {
    F1    = double(ii)/500;
    yld1  = yield(Fratio,F1,Stmp,Rtmp);
    ttt   = yld(Fratio,F1);
    prof_F <<F1<<" "<< ttt << endl;
  }


FUNCTION dvar_vector SRecruit(const dvar_vector& Stmp)
  RETURN_ARRAYS_INCREMENT();
  dvar_vector RecTmp(Stmp.indexmin(),Stmp.indexmax());
  switch (SrType)
  {
    case 1:
      RecTmp = elem_prod((Stmp / phizero) , mfexp( alpha * ( 1. - Stmp / Bzero ))) ; //Ricker form
from Dorn
      break;
    case 2:
      RecTmp = elem_prod(Stmp , 1. / ( alpha + beta * Stmp));        //Beverton-Holt form
      break;
    case 3:
      RecTmp = mfexp(mean_log_rec);                    //Avg recruitment
      break;
    case 4:
      RecTmp = elem_prod(Stmp , mfexp( alpha  - Stmp * beta)) ; //Old Ricker form
      break;
  }
  RETURN_ARRAYS_DECREMENT();
  return RecTmp;


FUNCTION dvariable SRecruit(const double& Stmp)
  RETURN_ARRAYS_INCREMENT();
  dvariable RecTmp;
  switch (SrType)
  {
    case 1:
      RecTmp = (Stmp / phizero) * mfexp( alpha * ( 1. - Stmp / Bzero )) ; //Ricker form from Dorn
      break;
    case 2:
      RecTmp = Stmp / ( alpha + beta * Stmp);        //Beverton-Holt form
      break;
    case 3:
      RecTmp = mfexp(mean_log_rec);                    //Avg recruitment
      break;
    case 4:
      RecTmp = Stmp * mfexp( alpha  - Stmp * beta) ; //old Ricker form
      break;
  }
  RETURN_ARRAYS_DECREMENT();
  return RecTmp;


FUNCTION dvariable SRecruit(_CONST dvariable& Stmp)
  RETURN_ARRAYS_INCREMENT();
  dvariable RecTmp;
  switch (SrType)
  {
    case 1:
      RecTmp = (Stmp / phizero) * mfexp( alpha * ( 1. - Stmp / Bzero )) ; //Ricker form from Dorn
      break;
    case 2:
      RecTmp = Stmp / ( alpha + beta * Stmp);        //Beverton-Holt form
      break;
    case 3:
      RecTmp = mfexp(mean_log_rec );                    //Avg recruitment
      break;
    case 4:
```

```
    RecTmp = Stmp * mfexp( alpha  - Stmp * beta) ; //old Ricker form
    break;
  }
  RETURN_ARRAYS_DECREMENT();
  return RecTmp;

 //=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

FUNCTION Get_Bzero
  Bzero.initialize();
  Rzero    =  mfexp(log_Rzero);

  dvariable survtmp = mfexp(-natmort(styr));
  surv = survtmp;

  dvar_matrix natagetmp(styr_rec,styr,1,nages);
  natagetmp.initialize();

  natagetmp(styr_rec,1) = Rzero;
  for (j=2; j<=nages; j++)
    natagetmp(styr_rec,j) = natagetmp(styr_rec,j-1) * survtmp;
  natagetmp(styr_rec,nages) /= (1.-survtmp);

  Bzero = wt_mature * pow(survtmp,spmo_frac)*natagetmp(styr_rec) ;
  phizero = Bzero/Rzero;

  switch (SrType)
  {
    case 1:
      alpha = log(-4.*steepness/(steepness-1.));
      break;
    case 2:
    {
      alpha  =  Bzero * (1. - (steepness - 0.2) / (0.8*steepness) ) / Rzero;
      beta   = (5. * steepness - 1.) / (4. * steepness * Rzero);
    }
    break;
    case 4:
    {
      beta  = log(5.*steepness)/(0.8*Bzero) ;
      alpha = log(Rzero/Bzero)+beta*Bzero;
    }
      break;
  }
  Sp_Biom.initialize();
  Sp_Biom(styr_sp,styr_rec-1) = Bzero;
  for (i=styr_rec;i<styr;i++)
  {
    Sp_Biom(i) = natagetmp(i)*pow(surv,spmo_frac) * wt_mature;
    // natagetmp(i,1)          = mfexp(rec_dev(i) + log_Rzero); // OjO numbers a function of mean not
SR curve...
    natagetmp(i,1)          = mfexp(rec_dev(i) + mean_log_rec);
    natagetmp(i+1)(2,nages) = ++(natagetmp(i)(1,nages-1)*mfexp(-natmort(styr) ));
    natagetmp(i+1,nages)   += natagetmp(i,nages)*mfexp(-natmort(styr));
  }
  natagetmp(styr,1)   = mfexp(rec_dev(styr) + mean_log_rec);
  mod_rec(styr_rec,styr) = column(natagetmp,1);
  natage(styr)  = natagetmp(styr); // OjO
  Sp_Biom(styr) = natagetmp(styr)*pow(surv,spmo_frac) * wt_mature;
  // cout <<natagetmp<<endl;exit(1);


FUNCTION dvariable Requil(dvariable& phi)
  RETURN_ARRAYS_INCREMENT();
  dvariable RecTmp;
  switch (SrType)
  {
    case 1:
      RecTmp =  Bzero * (alpha + log(phi) - log(phizero) ) / (alpha*phi);
      break;
```

```
    case 2:
      RecTmp =  (phi-alpha)/(beta*phi);
      break;
    case 3:
      RecTmp =  mfexp(mean_log_rec);
      break;
    case 4:
      RecTmp =  (log(phi)+alpha) / (beta*phi); //RecTmp =  (log(phi)/alpha + 1.)*beta/phi;
      break;
  }
  // Req    = Requil(phi) * exp(sigmarsq/2);
  // return RecTmp* exp(sigmarsq/2);
  RETURN_ARRAYS_DECREMENT();
  return RecTmp;


FUNCTION write_mceval_hdr
  /*
    for (k=1;k<=nind;k++)
      mceval<< " q_ind_"<< k<< " ";
    mceval<<"M steepness depletion MSY MSYL Fmsy Fcur_Fmsy Bcur_Bmsy Bmsy totbiom_"<<endyr<<" "<<
    " F35          "<<
    " F40          "<<
    " F50          "<<
    " fut_SPB_Fmsy_"<< endyr_fut<<" "<<
    " fut_SPB_F50%_"<< endyr_fut<<" "<<
    " fut_SPB_F40%_"<< endyr_fut<<" "<<
    " fut_SPB_F35%_"<< endyr_fut<<" "<<
    " fut_SPB_F0_"  << endyr_fut<<" "<<
    " fut_catch_Fmsy_"<<styr_fut<<" "<<
    " fut_catch_F50%_"<<styr_fut<<" "<<
    " fut_catch_F40%_"<<styr_fut<<" "<<
    " fut_catch_F35%_"<<styr_fut<<" "<<  endl;
    mceval<<"M steepness depletion MSY MSYL Fmsy Fcur_Fmsy Bcur_Bmsy Bmsy totbiom_"<<endyr<<" "<<
    " F35          "<<
    " F40          "<<
    " F50          "<<
    " fut_SPB_Fmsy_"<< endyr_fut<<" "<<
    " fut_SPB_F50%_"<< endyr_fut<<" "<<
    " fut_SPB_F40%_"<< endyr_fut<<" "<<
    " fut_SPB_F35%_"<< endyr_fut<<" "<<
    " fut_SPB_F0_"  << endyr_fut<<" "<<
    " fut_catch_Fmsy_"<<styr_fut<<" "<<
    " fut_catch_F50%_"<<styr_fut<<" "<<
    " fut_catch_F40%_"<<styr_fut<<" "<<
    " fut_catch_F35%_"<<styr_fut<<" "<<  endl; */
    for (k=1;k<=nind;k++)
      mceval<< " q_"<< indname(k)<< " ";
    for (j=1;j<=nages;j++)
      mceval<< " age_"<< j<< " ";
    mceval<<"M steepness depletion MSY MSYL Fmsy Fcur_Fmsy Bcur_Bmsy Bmsy totbiom_"<<endyr<<" "<<
    " SSB_yr_"<<endyr+1 <<" "<<
    " B100         "<<
    " B_"<<endyr+1   <<"_over_B100 " <<
    " B_"<<endyr+2   <<"_over_B100 " <<
    " B_"<<endyr+3   <<"_over_B100 " <<
    " B_"<<endyr+4   <<"_over_B100 " <<
    " B_"<<endyr+5   <<"_over_B100 " <<
    " B_"<<endyr_fut <<"_over_B100 " <<
    " fut_catch_"<<endyr+1<<" "<<
    " fut_catch_"<<endyr+2<<" "<<
    " fut_catch_"<<endyr+3<<" "<<
    " fut_catch_"<<endyr+4<<" "<<
    " fut_catch_"<<endyr+5<<" "<<
    " F35          "<<
    " F40          "<<
    " F50          "<<
    " fut_SPB_Fmsy_"<< endyr_fut<<" "<<
    " fut_SPB_F50%_"<< endyr_fut<<" "<<
    " fut_SPB_F40%_"<< endyr_fut<<" "<<
    " fut_SPB_F35%_"<< endyr_fut<<" "<<
```

```
    " fut_SPB_F0_"  << endyr_fut<<" "<<
    " fut_catch_Fmsy_"<<styr_fut<<" "<<
    " fut_catch_F50%_"<<styr_fut<<" "<<
    " fut_catch_F40%_"<<styr_fut<<" "<<
    " fut_catch_F35%_"<<styr_fut<<" "<<  endl;


//+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+

REPORT_SECTION
  if (last_phase())
  {
    int nvar1=initial_params::nvarcalc(); // get the number of active parameters
    int ndvar=stddev_params::num_stddev_calc();
    int offset=1;
    dvector param_values(1,nvar1+ndvar);
    initial_params::copy_all_values(param_values,offset);
    for (int i=0;i<initial_params::num_initial_params;i++)
    {
      // cout << "# " << initial_params::varsptr[i]->label() << "\n" << endl;
      if (withinbound(0,(initial_params::varsptr[i])->phase_start, initial_params::current_phase))
      {
        int sc = (initial_params::varsptr[i])->size_count();
        if (sc>0)
        {
          // write_input_log << "# " << initial_params::varsptr[i]->label() <<
endl<<param_values(i)<<"\n" << endl;
        }
      }
    }

    for (k=1;k<=nind;k++)
    {
      // cout<<indname(k)<<endl;
      // cout<<get_AC(k)<<endl<<endl;
    }
    if (!Popes)
      for (k=1;k<=nfsh;k++)
        Ftot += F(k);
    log_param(Mest);
    log_param(mean_log_rec);
    log_param(steepness);
    log_param(log_Rzero);
    log_param(rec_dev);
    log_param(log_sigmar);
    log_param(fmort);
    // log_param(log_selcoffs_fsh);
    // log_param(log_sel_spl_fsh);
    // log_param(logsel_slope_fsh);
    // log_param(sel50_fsh);
    // log_param(logsel_dslope_fsh);
    // log_param(seld50_fsh);
    log_param(rec_dev_future);
    // log_param(log_q_ind);
    // log_param(log_q_power_ind);
    // log_param(log_selcoffs_ind);
    // log_param(logsel_slope_ind);
    // log_param(logsel_dslope_ind);
    // log_param(sel50_ind);
    // log_param(seld50_ind);
  }

  if (oper_mod)
    Oper_Model();

  cout <<"==============================================================="<<endl;
  if(last_phase())
    cout<<"||  ++++++ Completed phase "<<current_phase()<<" In last phase now +++++"<<
endl<<"||"<<endl<<"||  "<<cntrlfile_name <<endl;
  else
```

```
    cout<<"||  ++++++ Completed phase "<<current_phase()<<" ++++++++++++++++"<< endl<<"||"<<endl<<"||
"<<cntrlfile_name <<endl;
  cout<<"||"<<endl<<"||"<<endl;
  cout <<"_______________________________________________________________"<<endl;
    adstring comma = adstring(",");
    report << model_name<<" "<< endl<< endl;
    report << "Estimated annual F's " << endl;
    Fmort.initialize();
    for (k=1;k<=nfsh;k++)
      for (i=styr;i<=endyr;i++)
        Fmort(i) += mean(F(k,i));
    report << Fmort<<endl;
    report << "Total mortality (Z)"<<endl;
    report << Z<<endl;
    report << "Estimated numbers of fish " << endl;
    for (i=styr;i<=endyr;i++)
      report <<"       Year: "<< i << " "<< natage(i) << endl;
    report << endl<< "Estimated F mortality " << endl;
    for (k=1;k<=nfsh;k++)
    {
      report << "Fishery "<< k <<" : "<< endl ;
      for (i=styr;i<=endyr;i++)
        report << "        Year: "<<i<<" "<<F(k,i)<<  " "<< endl;
    }

    report << endl<< "Observed survey values " << endl;
    for (k=1;k<=nind;k++)
    {
      int ii=1;
      report <<endl<< "Yr_Obs_Pred_Survey "<< k <<" : "<< endl ;
      for (int iyr=styr;iyr<=endyr;iyr++)
      {
        dvariable pred_tmp ;
        if (ii<=yrs_ind(k).indexmax())
        {
        pred_tmp = q_ind(k,ii) * pow(elem_prod(natage(iyr),pow(S(iyr),ind_month_frac(k))) *
                        elem_prod(sel_ind(k,iyr) , wt_ind(k,iyr)),q_power_ind(k));
          if (yrs_ind(k,ii)==iyr)
          {
            report << iyr<< " "<<
                     obs_ind(k,ii) << " "<< pred_tmp <<endl;
            ii++;
          }
          else
            report << iyr<< " -1 "<< " "<< pred_tmp   <<endl;
        }
      }
    }

    report << endl<< "Survey_Q:  "<<q_ind << endl;

    report << endl<< "Observed Prop " << endl;
    for (k=1;k<=nfsh;k++)
    {
      report << "ObsFishery "<< k <<" : "<< endl ;
      for (i=1;i<=nyrs_fsh_age(k);i++)
        report << yrs_fsh_age(k,i)<< " "<< oac_fsh(k,i) << endl;
    }
    report << endl<< "Predicted prop  " << endl;
    for (k=1;k<=nfsh;k++)
    {
      report << "PredFishery "<< k <<" : "<< endl;
      for (i=1;i<=nyrs_fsh_age(k);i++)
        report << yrs_fsh_age(k,i)<< " "<< eac_fsh(k,i) << endl;
    }
    report << endl<< "Observed prop Survey" << endl;
    for (k=1;k<=nind;k++)
    {
      report << "ObsSurvey "<<k<<" : "<<  endl;
      for (i=1;i<=nyrs_ind_age(k);i++)
        report << yrs_ind_age(k,i)<< " "<< oac_ind(k,i) << endl;
```

```
    }
    report << endl<< "Predicted prop Survey" << endl;
    for (k=1;k<=nind;k++)
    {
      report << "PredSurvey "<<k<<" : "<<  endl;
      for (i=1;i<=nyrs_ind_age(k);i++)
        report << yrs_ind_age(k,i)<< " "<< eac_ind(k,i) << endl;
    }
    report << endl<< "Observed catch biomass " << endl;
    report << catch_bio << endl;
    report << "predicted catch biomass " << endl;
    report << pred_catch << endl;

    report << endl<< "Estimated annual fishing mortality " << endl;
    for (k=1;k<=nfsh;k++)
      report << " Average_F_Fshry_"<<k<< " Full_selection_F_Fshry_"<<k;

    report << endl;
    for (i=styr;i<=endyr;i++)
    {
      report<< i<< " ";
      for (k=1;k<=nfsh;k++)
        report<< mean(F(k,i)) <<" "<< mean(F(k,i))*max(sel_fsh(k,i)) << " ";

      report<< endl;
    }
    report << endl<< "Selectivity" << endl;
    for (k=1;k<=nfsh;k++)
      for (i=styr;i<=endyr;i++)
        report << "Fishery "<< k <<"  "<< i<<" "<<sel_fsh(k,i) << endl;
    for (k=1;k<=nind;k++)
      for (i=styr;i<=endyr;i++)
        report << "Survey  "<< k <<"  "<< i<<" "<<sel_ind(k,i) << endl;

    report << endl<< "Stock Recruitment stuff "<< endl;
    for (i=styr_rec;i<=endyr;i++)
      if (active(log_Rzero))
        report << i<< " "<<Sp_Biom(i-rec_age)<< " "<< SRecruit(Sp_Biom(i-rec_age))<< " "<<
mod_rec(i)<<endl;
      else
        report << i<< " "<<Sp_Biom(i-rec_age)<< " "<< " 999" << " "<< mod_rec(i)<<endl;

    report << endl<< "Curve to plot "<< endl;
    report <<"stock Recruitment"<<endl;
    report <<"0 0 "<<endl;
    dvariable stock;
    for (i=1;i<=30;i++)
    {
      stock = double (i) * Bzero /25.;
      if (active(log_Rzero))
        report << stock <<" "<< SRecruit(stock)<<endl;
      else
        report << stock <<" 99 "<<endl;
    }

    report   << endl<<"Likelihood Components" <<endl;
    report   << "---------------------------------------- " <<endl;
    report   << "  catch_like  age_like_fsh sel_like_fsh ind_like age_like_ind sel_like_ind rec_like
fpen post_priors_indq post_priors residual total"<<endl;
    report   << " "<<obj_comps<<endl;

    obj_comps(11)= obj_fun - sum(obj_comps) ; // Residual
    obj_comps(12)= obj_fun ;                  // Total
    report   <<"  catch_like       "<<setw(10)<<obj_comps(1) <<endl
             <<"  age_like_fsh     "<<setw(10)<<obj_comps(2) <<endl
             <<"  sel_like_fsh     "<<setw(10)<<obj_comps(3) <<endl
             <<"  ind_like        "<<setw(10)<<obj_comps(4) <<endl
             <<"  age_like_ind     "<<setw(10)<<obj_comps(5) <<endl
             <<"  sel_like_ind     "<<setw(10)<<obj_comps(6) <<endl
             <<"  rec_like         "<<setw(10)<<obj_comps(7) <<endl
             <<"  fpen             "<<setw(10)<<obj_comps(8) <<endl
```

```
            <<"  post_priors_indq "<<setw(10)<<obj_comps(9) <<endl
            <<"  post_priors      "<<setw(10)<<obj_comps(10)<<endl
            <<"  residual         "<<setw(10)<<obj_comps(11)<<endl
            <<"  total            "<<setw(10)<<obj_comps(12)<<endl;

    report   << endl;
    report   << "Fit to Catch Biomass "<<endl;
    report   << "-------------------------" <<endl;
    for (k=1;k<=nfsh;k++)
      report << "  Catch_like_Fshry_#"<< k <<"  "<< catch_like(k) <<endl;
    report   << endl;

    report << "Age likelihoods for fisheries :"<<endl;
    report   << "-------------------------" <<endl;
    for (k=1;k<=nfsh;k++)
      report << "  Age_like_Fshry_#"<< k <<"  "<< age_like_fsh(k) <<endl;
    report   << endl;

    report   << "Selectivity penalties for fisheries :"<<endl;
    report   << "-------------------------" <<endl;
    report   << "  Fishery Curvature_Age Change_Time Dome_Shaped"<<endl;
    for (k=1;k<=nfsh;k++)
      report << "  Sel_Fshry_#"<< k <<"  "<< sel_like_fsh(k,1) <<" "<<sel_like_fsh(k,2)<<"
"<<sel_like_fsh(k,3)<< endl;
    report   << endl;

    report   << "survey Likelihood(s) " <<endl;
    report   << "-------------------------" <<endl;
    for (k=1;k<=nind;k++)
      report << "  Survey_Index_#"<< k <<"  " << ind_like(k)<<endl;
    report   << endl;

    report << setw(10)<< setfixed() << setprecision(5) <<endl;
    report   << "Age likelihoods for surveys :"<<endl;
    report   << "-------------------------" <<endl;
    for (k=1;k<=nind;k++)
      report << "  Age_Survey_#"<< k <<"  " << age_like_ind(k)<<endl;
    report   << endl;

    report   << "Selectivity penalties for surveys :"<<endl;
    report   << "-------------------------" <<endl;
    report   << "  Survey Curvature_Age Change_Time Dome_Shaped"<<endl;
    for (k=1;k<=nind;k++)
      report << "  Sel_Survey_#"<< k <<"  "<< sel_like_ind(k,1) <<" "<<sel_like_ind(k,2)<<"
"<<sel_like_ind(k,3)<< endl;
    report   << endl;

    report << setw(10)<< setfixed() << setprecision(5) <<endl;
    report   << "Recruitment penalties: " <<rec_like<<endl;
    report   << "-------------------------" <<endl;
    report   << "  (sigmar)            " <<sigmar<<endl;
    report   << "  S-R_Curve           " <<rec_like(1)<< endl;
    report   << "  Regularity          " <<rec_like(2)<< endl;
    report   << "  Future_Recruits     " <<rec_like(3)<< endl;
    report   << endl;

    report   << "F penalties:          " <<endl;
    report   << "-------------------------" <<endl;
    report   << "  Avg_F               " <<fpen(1) <<endl;
    report   << "  Effort_Variability  " <<fpen(2) <<endl;
    report   << endl;

    report   << "Contribution of Priors:"<<endl;
    report   << "-------------------------" <<endl;
    report   << "Source                ";
    report   <<           " Posterior";
    report   <<           " Param_Val";
    report   <<           " Prior_Val";
    report   <<           "  CV_Prior"<<endl;
  // (*ad_printf)("f = %lf\n",value(f));
    for (k=1;k<=nind;k++)
```

```
    {
      report << "Q_Survey_#"<< k <<"           "
             << setw(10)<<post_priors_indq(k)
             << setw(10)<< q_ind(k)
             << setw(10)<< qprior(k)
             << setw(10)<< cvqprior(k)<<endl;

      report << "Q_power_Survey_#"<< k <<"           "
             << setw(10)<<post_priors_indq(k)
             << setw(10)<< q_power_ind(k)
             << setw(10)<< q_power_prior(k)
             << setw(10)<< cvq_power_prior(k)<<endl;
    }

    // writerep(post_priors(1),repstring);
    // cout <<repstring<<endl;
    report   << "Natural_Mortality     "
             << setw(10)<< post_priors(1)
             << setw(10)<< M
             << setw(10)<< natmortprior
             << setw(10)<< cvnatmortprior <<endl;
    report   << "Steepness             "
             << setw(10)<< post_priors(2)
             << setw(10)<< steepness
             << setw(10)<< steepnessprior
             << setw(10)<< cvsteepnessprior <<endl;
    report   << "SigmaR                "
             << setw(10)<< post_priors(3)
             << setw(10)<< sigmar
             << setw(10)<< sigmarprior
             << setw(10)<< cvsigmarprior <<endl;
    report   << endl;
    report<<"Num_parameters_Estimated "<<initial_params::nvarcalc()<<endl;

  report <<cntrlfile_name<<endl;
  report <<datafile_name<<endl;
  report <<model_name<<endl;
  if (SrType==2)
    report<< "Beverton-Holt" <<endl;
  else
    report<< "Ricker" <<endl;
  report<<"Steepnessprior,_CV,_phase: " <<steepnessprior<<" "<<
    cvsteepnessprior<<" "<<
    phase_srec<<" "<< endl;

  report<<"sigmarprior,_CV,_phase: " <<sigmarprior<<" "<<  cvsigmarprior <<" "<<phase_sigmar<<endl;

  report<<"Rec_estimated_in_styr_endyr: " <<styr_rec    <<" "<<endyr        <<" "<<endl;
  report<<"SR_Curve_fit__in_styr_endyr: " <<styr_rec_est<<" "<<endyr_rec_est<<" "<<endl;
  report<<"Model_styr_endyr:            " <<styr        <<" "<<endyr        <<" "<<endl;

  report<<"M_prior,_CV,_phase "<< natmortprior<< " "<< cvnatmortprior<<" "<<phase_M<<endl;
  report<<"qprior,_CV,_phase " <<qprior<<" "<<cvqprior<<" "<< phase_q<<endl;
  report<<"q_power_prior,_CV,_phase " <<q_power_prior<<" "<<cvq_power_prior<<" "<<
phase_q_power<<endl;

  report<<"cv_catchbiomass: " <<cv_catchbiomass<<" "<<endl;
  report<<"Projection_years "<< nproj_yrs<<endl;
  for (k=1;k<=nfsh;k++)
    report << "Fsh_sel_opt_fish: "<<k<<" "<<fsh_sel_opt(k)<<" "<<sel_change_in_fsh(k)<<endl;
  for (k=1;k<=nind;k++)
    report<<"Survey_Sel_Opt_Survey: " <<k<<" "<<(ind_sel_opt(k))<<endl;

  report <<"Phase_survey_Sel_Coffs: "<<phase_selcoff_ind<<endl;
  report <<"Fshry_Selages: " << nselages_in_fsh  <<endl;
  report <<"Survy_Selages: " << nselages_in_ind <<endl;
  report << "Phase_for_age-spec_fishery "<<phase_selcoff_fsh<<endl;
  report << "Phase_for_logistic_fishery "<<phase_logist_fsh<<endl;
  report << "Phase_for_dble_logistic_fishery "<<phase_dlogist_fsh<<endl;
  report << "Phase_for_age-spec_survey  "<<phase_selcoff_ind<<endl;
  report << "Phase_for_logistic_survey  "<<phase_logist_ind<<endl;
```

```
  report << "Phase_for_dble_logistic_indy "<<phase_dlogist_ind<<endl;

  for (k=1; k<=nfsh;k++)
  {
    report <<"Number_of_select_changes_fishery: "<<k<<" "<<n_sel_ch_fsh(k)<<endl;
    report<<"Yrs_fsh_sel_change: "<<yrs_sel_ch_fsh(k)<<endl;
    report << "sel_change_in: "<<sel_change_in_fsh(k) << endl;
  }
  for (k=1; k<=nind;k++)
  {
    report <<"Number_of_select_changes_survey: "<<k<<" "<<n_sel_ch_ind(k)<<endl;
    report<<"Yrs_ind_sel_change: "<<yrs_sel_ch_ind(k)<<endl;
    report << "sel_change_in: "<<sel_change_in_ind(k) << endl;
  }


FUNCTION write_msy_out
  ofstream msyout("msyout.dat");
  msyout << " # Natural Mortality       " <<endl;
  for (j=1;j<=nages;j++)
    msyout <<M <<" ";
  msyout <<endl;
  msyout << spawnmo<< "  # Spawnmo                   " <<endl;
  msyout <<"# Wt spawn"<<endl<< wt_pop<< endl;
  msyout <<"# Wt fish"<<endl;
  for (k=1;k<=nfsh;k++)
    msyout <<wt_fsh(k,endyr)<< " ";
  msyout <<endl;
  msyout <<"# Maturity"<<endl<< maturity<< endl;
  msyout <<"# selectivity"<<endl;
  for (k=1;k<=nfsh;k++)
    msyout<< sel_fsh(k,endyr) <<" ";
  msyout<< endl;
  msyout<<"Srec_Option "<<SrType<< endl;
  msyout<<"Alpha "<<alpha<< endl;
  msyout<<"beta "<<beta<< endl;
  msyout<<"steepness "<<steepness<< endl;
  msyout<<"Bzero "<<Bzero<< endl;
  msyout<<"Rzero "<<Rzero<< endl;


FUNCTION write_projout
// Function to write out data file for projection model....
  ofstream projout( projfile_name );

  projout <<"# "<<model_name <<" "<< projfile_name<<endl;
  projout <<"123  # seed"<<endl;
  // Flag to tell if this is a SSL species
  projout << "1 # Flag to tell if this is a SSL forage species                 "<<endl;
  projout << "0 # Flag to Dorn's version of a constant buffer                  "<<endl;
  // Flag to solve for F in first year or not 0==don't solve
  projout<< " 1 # Flag to solve for F in first year or not 0==don't solve"<<endl;
  // Flag to use 2nd-year catch/TAC
  projout<< "0 # Flag to use 2nd-year catch/TAC"<<endl;
  projout << nfsh<<"   # Number of fisheries"<<endl;
  projout <<"14   # Number of projection years"<<endl;
  projout <<"1000 # Number of simulations"<<endl;
  projout <<endyr<< " # Begin year of projection" <<endl;
  projout <<nages<< " # Number of ages" <<endl;
  for (j=1;j<=nages;j++)
    projout <<M <<" ";
  projout << " # Natural Mortality       " <<endl;
  double sumtmp;
  sumtmp = 0.;
  for (k=1;k<=nfsh;k++)
    sumtmp += catch_bio(k,endyr);
  projout << sumtmp<< " # TAC in current year (assumed catch) " <<endl;
  projout << sumtmp<< " # TAC in current year+1 (assumed catch) " <<endl;
  for (k=1;k<=nfsh;k++)
    projout <<  F(k,endyr)/mean((F(k,endyr)))<<" "<<endl;
```

```
  //  + fmort_dev(k,endyr)) /Fmort(endyr)<<" ";

  projout << "   # Fratio                  " <<endl;
  dvariable sumF=0.;
  for (k=1;k<=nfsh;k++)
  {
    Fratio(k) = sum(F(k,endyr)) ;
    sumF += Fratio(k) ;
  }
  Fratio /= sumF;
  projout << Fratio         <<endl;
  projout <<"  # average f" <<endl;
  projout << " 1  # author f                  " <<endl;
  projout << spawnmo<< "  # Spawnmo                   " <<endl;
  projout <<"# Wt spawn"<<endl<< wt_pop<< endl;
  projout <<"# Wt fish"<<endl;
  for (k=1;k<=nfsh;k++)
    projout <<wt_fsh(k,endyr)<< " ";
  projout <<endl;
  projout <<"# Maturity"<<endl<< maturity<< endl;
  projout <<"# selectivity"<<endl;
  for (k=1;k<=nfsh;k++)
    projout<< sel_fsh(k,endyr) <<" "<<endl;
  projout<< endl;
  projout <<"# natage"<<endl<< natage(endyr) << endl;
  if (styr<(1977-rec_age-1))
  {
    projout <<"#_N_recruitment_years (not including last 1 estimates)"<<endl<<endyr-(1977+rec_age+1)
<< endl;
    projout <<"#_Recruitment_start_at_1977_yearclass=1978_for_age_1_recruits"<<yy(1977+rec_age,endyr-
1)<<endl<<mod_rec(1977+rec_age,endyr-1)<< endl;
  }


FUNCTION write_proj
 ofstream newproj("proj.dat");
// Function to write out data file for new Ianelli 2005 projection model....
 newproj <<"#Species name here:"<<endl;
 newproj <<model_name+"_"+datafile_name<<endl;
 newproj <<"#SSL Species?"<<endl;
 newproj <<"1"<<endl;
 newproj <<"#Constant buffer of Dorn?"<<endl;
 newproj <<"0"<<endl;
 newproj <<"#Number of fisheries?"<<endl;
 newproj <<"1"<<endl;
 newproj <<"#Number of sexes?"<<endl;
 newproj <<"1"<<endl;
 newproj <<"#5year_Average_F(endyr-4,endyr_as_estimated_by_ADmodel)"<<endl;
 // Need to correct for maxf standardization

 dvector seltmp(1,nages);
 double sumF = 0. ;
 seltmp.initialize();
 for (k=1;k<=nfsh;k++)
 {
    Fratio(k) = sum(F(k,endyr)) ;
    sumF += value(Fratio(k)) ;
 }
 Fratio /= sumF;
 // compute a 5-year recent average fishery-aggregated selectivity for output to projection model
 for (k=1;k<=nfsh;k++)
   for (j=1;j<=nages;j++)
     seltmp(j) += value(Fratio(k))*(value(sel_fsh(k,endyr,j))
                 +value(sel_fsh(k,endyr-1,j))
                 +value(sel_fsh(k,endyr-2,j))
                 +value(sel_fsh(k,endyr-3,j))
                 +value(sel_fsh(k,endyr-4,j))
                 )/5.;

 newproj << mean(Fmort(endyr-4,endyr))<<endl;
 newproj <<"#_Author_F_as_fraction_F_40%"<<endl;
```

```
 newproj <<"1"<<endl;
 newproj <<"#ABC SPR" <<endl;
 newproj <<"0.4"<<endl;
 newproj <<"#MSY SPR" <<endl;
 newproj <<"0.35"<<endl;
 newproj <<"#_Spawn_month"<<endl;
 newproj << spmo_frac*12+1<<endl;
 newproj <<"#_Number_of_ages"<<endl;
 newproj <<nages<<endl;
 newproj <<"#_F_ratio(must_sum_to_one_only_one_fishery)"<<endl;
 newproj <<"1"<<endl;
 newproj <<"#_Natural_Mortality" << aa << endl;
   for (j=1;j<=nages;j++) newproj <<natmort(endyr)<<" "; newproj<<endl;
 newproj
<<"#_Maturity_divided_by_2(projection_program_uses_to_get_female_spawning_biomass_if_divide_by_2"<<aa
<<endl<<2.*maturity<< endl;
 newproj <<"#_Wt_at_age_spawners"<<aa<<endl<<wt_pop<< endl;
 newproj <<"#_Wt_at_age_fishery" <<aa<<endl<<wt_fsh(1,endyr) << endl;
 newproj <<"#" <<endl;

 newproj <<"#_Selectivity_fishery_scaled_to_max_at_one_3_yr_avg "<<aa<<endl;
 seltmp = value(sel_fsh(1,endyr)) +value(sel_fsh(1,endyr-1))  +value(sel_fsh(1,endyr-2));
 newproj << seltmp/max(seltmp)<<endl;
 newproj <<"#_Numbers_at_age_end_year"<<aa<<endl<<natage(endyr)<< endl;
  if (styr<=1977)
  {
   newproj <<"#_N_recruitment_years (not including last estimate)"<<endl<<endyr-(1977+rec_age) <<
endl;
   newproj <<"#_Recruitment_start_at_1977_yearclass=1978_for_age_1_recruits"<<yy(1977+rec_age,endyr-
1)
         <<endl<<mod_rec(1977+rec_age,endyr-1)<< endl;
  }

 newproj <<"#_Spawning biomass "<<endl<<Sp_Biom(styr-rec_age,endyr-rec_age)/1000<< endl;
 newproj.close();


RUNTIME_SECTION
  convergence_criteria 1.e-1,1.e-01,1.e-03,1e-5,1e-5
  maximum_function_evaluations 100,100,200,300,2500


TOP_OF_MAIN_SECTION
  gradient_structure::set_MAX_NVAR_OFFSET(4500);
  gradient_structure::set_NUM_DEPENDENT_VARIABLES(4500);
  gradient_structure::set_GRADSTACK_BUFFER_SIZE(1000000);
  gradient_structure::set_CMPDIF_BUFFER_SIZE(10000000);
  arrmblsize=500000000;


FINAL_SECTION
  // Calc_Dependent_Vars();
  write_proj();
  write_projout();
  // write_msy_out();
  Profile_F();
  Write_R();


GLOBALS_SECTION
  #include <admodel.h>
        #undef write_SIS_rep
        #define write_SIS_rep(object) SIS_rep << #object "\n" << object << endl;
        #undef truth
        #define truth(object) trudat << #object "\n" << object << endl;
        #undef REPORT
        #define REPORT(object) REPORT << #object "\n" << object << endl;
        #undef R_Report
        #define R_Report(object) R_report << "$"#object "\n" << object << endl;
        /**
```

```
        \def log_input(object)
       Prints name and value of \a object on ADMB report %ofstream file.
       */
       #undef log_input
       #define log_input(object) write_input_log << "# " #object "\n" << object << endl;
       #undef log_param
  // #define log_param(object) for(int i=0;i<initial_params::num_initial_params;i++)
{if(withinbound(0,(initial_params::varsptr[i])->phase_start, initial_params::current_phase)) { int
sc= (initial_params::varsptr[i])->size_count(); if (sc>0) { write_input_log << "# " <<
initial_params::varsptr[i] ->label() << "\n" << object<<endl; } }}
  //
       #define log_param(object) if (active(object)) write_input_log << "# " #object "\n" << object
<< endl;
  ofstream write_input_log("input.log");
  ofstream SIS_rep("SIS_out.rep");

 // void get_sel_changes(int& k);
  adstring_array fshname;
  adstring_array indname;
  adstring truname;
  adstring simname;
  adstring model_name;
  adstring projfile_name;
  adstring datafile_name;
  adstring cntrlfile_name;
  adstring tmpstring;
  adstring repstring;
  adstring version_info;

 typedef vcubic_spline_function * pvcubic_spline_function;
 class vcubic_spline_function_array
 {
 public:
   int indexmin(void) {return index_min;}
   int indexmax(void) {return index_max;}
   vcubic_spline_function_array(int,int,const dvector & x,
           const dvar_matrix& _y);
   ~vcubic_spline_function_array();
   vcubic_spline_function & operator () (int i);
   vcubic_spline_function_array(const dvector & x);
   // so that this will fail define it if you need it.
   vcubic_spline_function_array(const vcubic_spline_function_array&);
   dvar_matrix operator( ) (const dvector & v);

 private:
   vcubic_spline_function ** ptr;
   int index_min;
   int index_max;
 };

  vcubic_spline_function & vcubic_spline_function_array::operator () (int i)
  {
    if (i<indexmin() || i> indexmax())
    {
      cerr << "index out of range in function"
           " vcubic_spline_function & operator () (int i)"
       << endl;
      ad_exit(1);
    }
    return *(ptr[i]);
  }

  dvar_matrix vcubic_spline_function_array::operator( ) (const dvector & v)
  {
    int mmin=indexmin();
    int mmax=indexmax();
    dvar_matrix tmp(mmin,mmax,v.indexmin(),v.indexmax());
    for (int i=mmin;i<=mmax;i++)
    {
      tmp(i)=(*this)(i)(v);
    }
```

```
    return tmp;
  }

  vcubic_spline_function_array::vcubic_spline_function_array(int mmin,int mmax,
  const dvector & x, const dvar_matrix& y)
  {
    index_min=mmin;
    index_max=mmax;
    int n=mmax-mmin+1;
    ptr = new pvcubic_spline_function[n];
    ptr-=mmin;
    for (int i=mmin;i<=mmax;i++)
    {
      ptr[i]= new  vcubic_spline_function(x,y(i));
    }
  }
  vcubic_spline_function_array::~vcubic_spline_function_array()
  {
    int mmin=indexmin();
    int mmax=indexmax();

    for (int i=mmin;i<=mmax;i++)
    {
      delete ptr[i];
    }
    ptr+=mmin;
    delete ptr;
    ptr=0;
  }


FUNCTION dvariable get_spr_rates(double spr_percent)
  RETURN_ARRAYS_INCREMENT();
  dvar_matrix sel_tmp(1,nages,1,nfsh);
  sel_tmp.initialize();
  for (k=1;k<=nfsh;k++)
    for (j=1;j<=nages;j++)
      sel_tmp(j,k) = sel_fsh(k,endyr,j); // NOTE uses last-year of fishery selectivity for
projections.
  dvariable sumF=0.;
  for (k=1;k<=nfsh;k++)
  {
    Fratio(k) = sum(F(k,endyr)) ;
    sumF += Fratio(k) ;
  }
  Fratio /= sumF;
  double df=1.e-3;
  dvariable F1 ;
  F1.initialize();
  F1 = .8*natmortprior;
  dvariable F2;
  dvariable F3;
  dvariable yld1;
  dvariable yld2;
  dvariable yld3;
  dvariable dyld;
  dvariable dyldp;
  // Newton Raphson stuff to go here
  for (int ii=1;ii<=6;ii++)
  {
    F2     = F1 + df;
    F3     = F1 - df;
    yld1   = -1000*square(log(spr_percent/spr_ratio(F1, sel_tmp,styr)));
    yld2   = -1000*square(log(spr_percent/spr_ratio(F2, sel_tmp,styr)));
    yld3   = -1000*square(log(spr_percent/spr_ratio(F3, sel_tmp,styr)));
    dyld   = (yld2 - yld3)/(2*df);                          // First derivative (to find the root of
this)
    dyldp  = (yld3-(2*yld1)+yld2)/(df*df);  // Newton-Raphson approximation for second derivitive
    F1    -= dyld/dyldp;
  }
  RETURN_ARRAYS_DECREMENT();
```

```
  return(F1);


FUNCTION dvariable spr_ratio(dvariable trial_F,dvar_matrix sel_tmp,int iyr)
  dvariable SBtmp;
  dvar_vector Ntmp(1,nages);
  dvar_vector srvtmp(1,nages);
  SBtmp.initialize();
  Ntmp.initialize();
  srvtmp.initialize();
  dvar_matrix Ftmp(1,nages,1,nfsh); // note that this is in reverse order of usual indexing (age,
fshery)
  Ftmp = sel_tmp;
  for (j=1;j<=nages;j++)
  {
    Ftmp(j) = elem_prod(Ftmp(j), trial_F * Fratio);
    srvtmp(j)  = mfexp(-sum(Ftmp(j)) - natmort(iyr));
  }
  Ntmp(1)=1.;
  j=1;
  SBtmp  += Ntmp(j)*wt_mature(j)*pow(srvtmp(j),spmo_frac);
  for (j=2;j<nages;j++)
  {
    Ntmp(j) = Ntmp(j-1)*srvtmp(j-1);
    SBtmp  += Ntmp(j)*wt_mature(j)*pow(srvtmp(j),spmo_frac);
  }
  Ntmp(nages)=Ntmp(nages-1)*srvtmp(nages-1)/(1.-srvtmp(nages));
  SBtmp  += Ntmp(nages)*wt_mature(nages)*pow(srvtmp(nages),spmo_frac);
  return(SBtmp/phizero);


FUNCTION dvariable spr_unfished(int i)
  dvariable Ntmp;
  dvariable SBtmp;
  SBtmp.initialize();
  Ntmp = 1.;
  for (j=1;j<nages;j++)
  {
    SBtmp += Ntmp*wt_mature(j)*exp(-spmo_frac * natmort(i));
    Ntmp  *= mfexp( -natmort(i));
  }
  Ntmp    /= (1.-exp(-natmort(i)));
  SBtmp += Ntmp*wt_mature(j)*exp(-spmo_frac * natmort(i));
  return(SBtmp);


FUNCTION compute_spr_rates
  //Compute SPR Rates and add them to the likelihood for Females
  dvariable sumF=0.;
  for (k=1;k<=nfsh;k++)
  {
    Fratio(k) = sum(F(k,endyr)) ;
    sumF += Fratio(k) ;
  }
  Fratio /= sumF;

  F35_est = get_spr_rates(.35);
  F50_est = get_spr_rates(.50);
  F40_est = get_spr_rates(.40);

  for (k=1;k<=nfsh;k++)
  {
    F50(k) = F50_est * (Fratio(k));
    F40(k) = F40_est * (Fratio(k));
    F35(k) = F35_est * (Fratio(k));
  }


  /* FUNCTION get_agematrix
  for (i=1;i<=nages;i++)
```

```
    {
     sd_age(i)=cv_age(i)*len_age(i);
     var_age(i)=sd_age(i)*sd_age(i);
     for (int j=1;j<=nlenbins;j++)
       {
        diff = len(j)-len_age(i);
        trans(i,j)=2/sd_age(i)*exp(-diff*diff/(2*var_age(i)));
       }
      trans(i)=trans(i)/sum(trans(i));
    }
    */



FUNCTION void writerep(dvariable& tmp,adstring& tmpstring)
  cout <<tmpstring<<endl<<endl;
  tmpstring = printf("3.5%f",value(tmp));


FUNCTION dvariable SolveF2(const int& iyr, const dvar_vector& N_tmp, const double&  TACin)
  RETURN_ARRAYS_INCREMENT();
  dvariable dd = 10.;
  dvariable cc;
  dvar_matrix Fratsel(1,nfsh,1,nages);
  dvar_vector M_tmp(1,nages) ;
  dvar_vector Z_tmp(1,nages) ;
  dvar_vector S_tmp(1,nages) ;
  dvar_vector Ftottmp(1,nages);
  dvariable btmp =  N_tmp * elem_prod(sel_fsh(1,iyr),wt_pop);
  dvariable ftmp;
  M_tmp = M(iyr);
  ftmp = TACin/btmp;
    for (k=1;k<=nfsh;k++)
      Fratsel(k) = Fratio(k)*sel_fsh(k,iyr);
    for (int ii=1;ii<=5;ii++)
    {
      Ftottmp.initialize();
      for (k=1;k<=nfsh;k++)
        Ftottmp += ftmp*Fratsel(k);

      Z_tmp = Ftottmp  + M_tmp;
      S_tmp = mfexp( -Z_tmp );
      cc = 0.0;
      for (k=1;k<=nfsh;k++)
        cc += wt_fsh(k,endyr) * elem_prod(elem_div(ftmp*Fratsel(k),  Z_tmp),elem_prod(1.-
S_tmp,N_tmp)); // Catch equation (vectors)

      dd = cc / TACin - 1.;
      if (dd<0.) dd *= -1.;
      ftmp += (TACin-cc) / btmp;
    }
  RETURN_ARRAYS_DECREMENT();
  return(ftmp);


FUNCTION dvar_vector SolveF2(const int& iyr, const dvector&  Catch)
  // Returns vector of F's (given year) by fleet
  // Requires: N and fleet specific wts & selectivities at age, catch
  // iterate to get Z's right
  RETURN_ARRAYS_INCREMENT();
  dvariable dd = 10.;
  dvariable cc;
  dvar_matrix  seltmp(1,nfsh,1,nages);
  dvar_matrix  wt_tmp(1,nfsh,1,nages);
  dvar_matrix Fratsel(1,nfsh,1,nages);
  dvar_vector N_tmp = natage(iyr);
  dvar_vector M_tmp(1,nages) ;
  dvar_vector Z_tmp(1,nages) ;
  dvar_vector S_tmp(1,nages) ;
  dvar_vector Ftottmp(1,nages);
```

```
  dvar_vector Frat(1,nfsh);
  dvar_vector btmp(1,nfsh);
  dvar_vector ftmp(1,nfsh);
  dvar_vector hrate(1,nfsh);
  btmp.initialize();
  M_tmp = M(iyr);
  // Initial guess for Fratio
  for (k=1;k<=nfsh;k++)
  {
    seltmp(k)= sel_fsh(k,iyr); // Selectivity
    wt_tmp(k)= wt_fsh(k,iyr); //
    btmp(k)  =  N_tmp * elem_prod(seltmp(k),wt_tmp(k));
    hrate(k) = Catch(k)/btmp(k);
    Frat(k)  = Catch(k)/sum(Catch);
    Fratsel(k) = Frat(k)*seltmp(k);
    ftmp(k) = 1.1*(1.- posfun(1.-hrate(k),.10,fpen(4)));
  }
  // Initial fleet-specific F
  // iterate to balance effect of multiple fisheries..........
  for (int kk=1;kk<=nfsh;kk++)
  {
    for (k=1;k<=nfsh;k++)
    {
      if (hrate(k) <.9999)
      {
        for (int ii=1;ii<=8;ii++)
        {
          Ftottmp.initialize();
          Ftottmp   = ftmp*Fratsel;
          Z_tmp     = Ftottmp  + M_tmp;
          S_tmp     = mfexp( -Z_tmp );
          cc        = wt_tmp(k) * elem_prod(elem_div(ftmp(k)*Fratsel(k),  Z_tmp),elem_prod(1.-
S_tmp,N_tmp)); // Catch equation (vectors)
          ftmp(k)  += ( Catch(k)-cc ) / btmp(k);
        }
        Frat(k)    = ftmp(k)/sum(ftmp);
        Fratsel(k) = Frat(k)*seltmp(k);
      }
    }
  }
  RETURN_ARRAYS_DECREMENT();
  return(ftmp);


FUNCTION Write_SimDatafile
  {
  int nsims;
  // get the number of simulated datasets to create...
  ifstream sim_in("nsims.dat"); sim_in >> nsims; sim_in.close();
  char buffer [33];
  ofstream SimDB("simout.dat",ios::app);
  ofstream TruDB("truout.dat",ios::app);
  // compute the autocorrelation term for residuals of fit to indices...
  // for (k=1;k<=nind;k++) ac(k) = get_AC(k);
  int nyrs_fsh_age_sim=endyr-styr;
  int nyrs_ind_sim    = 1+endyr-styr;
  int nyrs_ind_age_sim= 1+endyr-styr;
  ivector yrs_fsh_age_sim(1,nyrs_fsh_age_sim);
  ivector yrs_ind_sim(1,nyrs_ind_sim);
  ivector yrs_ind_age_sim(1,nyrs_ind_sim);
  yrs_fsh_age_sim.fill_seqadd(1977,1);
  yrs_ind_sim.fill_seqadd(1977,1);
  yrs_ind_age_sim.fill_seqadd(1977,1);
  ivector n_sample_fsh_age_sim(1,nyrs_fsh_age_sim);
  ivector n_sample_ind_age_sim(1,nyrs_ind_age_sim);
  dvector new_ind_sim(1,nyrs_ind_sim);
  dvector sim_rec_devs(styr_rec,endyr);
  dvector sim_Sp_Biom(styr_rec,endyr);
  dmatrix sim_natage(styr_rec,endyr,1,nages);
  dmatrix catagetmp(styr,endyr,1,nages);
  dvector sim_catchbio(styr,endyr);
```

```
  double survtmp = value(mfexp(-natmort(styr)));
  for (k=1;k<=nfsh;k++) Ftot += F(k);

  for (int isim=1;isim<=nsims;isim++)
  {
    new_ind_sim.initialize();
    sim_natage.initialize();
    // Start w/ simulated population
    // Simulate using new recruit series (same F's)
    // fill vector with unit normal RVs
    sim_rec_devs.fill_randn(rng);
    sim_rec_devs *= value(sigmar);
    sim_natage(styr_rec,1) = value(Rzero)*exp(sim_rec_devs(styr_rec));
    for (j=2; j<=nages; j++)
      sim_natage(styr_rec,j) = sim_natage(styr_rec,j-1) * survtmp;
    sim_natage(styr_rec,nages) /= (1.-survtmp);

    // Simulate population w/ process errors in recruits
    for (i=styr_rec;i<=endyr;i++)
    {
      sim_Sp_Biom(i) = sim_natage(i)*pow(survtmp,spmo_frac) * wt_mature;
      if (i>styr_rec+rec_age)
        sim_natage(i,1)          = value(SRecruit(sim_Sp_Biom(i-rec_age)))*mfexp(sim_rec_devs(i));
      else
        sim_natage(i,1)          = value(SRecruit(sim_Sp_Biom(i)))*mfexp(sim_rec_devs(i));

      if (i>=styr)
      {
        // apply estimated survival rates
        sim_Sp_Biom(i)          = value( elem_prod(sim_natage(i),pow(S(i),spmo_frac)) * wt_mature);
        catagetmp(i)            = value( elem_prod(elem_div(Ftot(i),Z(i)),elem_prod(1.-
S(i),sim_natage(i))));
        sim_catchbio(i)         = catagetmp(i)*wt_fsh(1,i);
        if (i<endyr)
        {
          sim_natage(i+1)(2,nages) = value( ++elem_prod(sim_natage(i)(1,nages-1),S(i)(1,nages-1)));
          sim_natage(i+1,nages)   += value( sim_natage(i,nages)*S(i,nages));
        }
      }
      else
      {
        if (i<endyr)
        {
          sim_natage(i+1)(2,nages) = ++(sim_natage(i)(1,nages-1) * survtmp);
          sim_natage(i+1,nages)   += sim_natage(i,nages)*survtmp;
        }
      }
    }

    //===============================================
    //Now write from simulated population
    //
    // Create the name of the simulated dataset
    simname = "sim_"+ adstring(itoa(isim,buffer,10)) + ".dat";
    truname = "tru_"+ adstring(itoa(isim,buffer,10)) + ".dat";
    ofstream trudat(truname);
    truth(Rzero);
    truth(Fmsy);
    truth(MSY);
    dvector ntmp(1,nages);
    dmatrix seltmp(1,nfsh,1,nages);
    dmatrix Fatmp(1,nfsh,1,nages);
    dvector Ztmp(1,nages);
    seltmp.initialize();
    Fatmp.initialize();
    Ztmp.initialize();
    ntmp.initialize();
    for (k=1;k<=nfsh;k++)
     seltmp(k) = value(sel_fsh(k,endyr));
    Ztmp = value(natmort(styr));
    for (k=1;k<=nfsh;k++)
```

```
    {
      Fatmp(k) = value(Fratio(k) * Fmsy * seltmp(k));
      Ztmp    += Fatmp(k);
    }
    dvector survmsy = exp(-Ztmp);
    ntmp(1) = value(Rmsy);
    for (j=2;j<=nages;j++)
      ntmp(j) = ntmp(j-1)*survmsy(j-1);
    ntmp(nages) /= (1-survmsy(nages));
    // dvariable phi    = elem_prod( ntmp , pow(survmsy,spmo_frac ) )* wt_mature;
    truth(Rmsy);
    truth(seltmp);
    double SurvBmsy;
    double q_ind_sim=value(mean(q_ind(1)));
    SurvBmsy = value(elem_prod(wt_ind(1,endyr),elem_prod(pow(survmsy,ind_month_frac(1)), ntmp)) *
q_ind_sim*sel_ind(1,endyr));
    truth(ntmp);
    double Cmsy   = value(yield(Fratio,  Fmsy));
    truth(Cmsy);
    // Now do OFL for next year...
    ntmp(1)       = value(SRecruit(sim_Sp_Biom(endyr+1-rec_age)));
    ntmp(2,nages) = value( ++elem_prod(sim_natage(endyr)(1,nages-1),S(endyr)(1,nages-1)));
    ntmp(nages)  += value( sim_natage(endyr,nages)*S(endyr,nages));
    dvector ctmp(1,nages);
    ctmp.initialize();
    OFL=0.;
    for (k=1;k<=nfsh;k++)
    {
      for ( j=1 ; j <= nages; j++ )
        ctmp(j)      = ntmp(j) * Fatmp(k,j) * (1. - survmsy(j)) / Ztmp(j);
      OFL  += wt_fsh(k,endyr) * ctmp;
    }
    double NextSurv = value(elem_prod(wt_ind(1,endyr),elem_prod(pow(survmsy,ind_month_frac(1)),
ntmp)) *
                        q_ind_sim*sel_ind(1,endyr));
    double NextSSB  = elem_prod(ntmp, pow(survmsy,spmo_frac)) * wt_mature;
    // Catch at following year for Fmsy
    truth(OFL);
    truth(SurvBmsy);
    truth(steepness);
    truth(natmort);
    truth(sim_natage);
    truth(sim_Sp_Biom);
    // Open the simulated dataset for writing
    ofstream simdat(simname);
    simdat << "# first year" <<endl;
    simdat << styr <<endl;
    simdat << "# Last  year" <<endl;
    simdat << endyr <<endl;
    simdat << "# age recruit" <<endl;
    simdat << rec_age <<endl;
    simdat << "# oldest age" <<endl;
    simdat << oldest_age <<endl;
    simdat << "# Number of fisheries " <<endl;
    simdat << nfsh <<endl;
    simdat << fshnameread <<endl;
    simdat << "# Catch biomass by fishery " <<endl;
    for (k=1;k<=nfsh;k++)
    {
      simdat << "# " <<fshname(k) <<" " << k <<endl;
      simdat << sim_catchbio <<endl;
    }
    simdat << "# Catch biomass uncertainty by fishery (std errors)" <<endl;
    for (k=1;k<=nfsh;k++)
    {
      simdat << "# " <<fshname(k) <<" " << k <<endl;
      simdat << catch_bio_sd(k) <<endl;
    }
    simdat << "# number of years for fishery age data " <<endl;
    for (k=1;k<=nfsh;k++)
    {
```

```
  simdat << "# " <<fshname(k)<< " " << k <<endl;
  simdat << nyrs_fsh_age_sim <<endl;
}
simdat << "# years for fishery age data " <<endl;
for (k=1;k<=nfsh;k++)
{
  simdat << "# " <<fshname(k)<< " " << k <<endl;
  simdat << yrs_fsh_age_sim  <<endl;
}
simdat << "# sample sizes for fishery age data " <<endl;
for (k=1;k<=nfsh;k++)
{
  n_sample_fsh_age_sim = mean(n_sample_fsh_age(k));
  simdat << "# " <<fshname(k)<< " " << k <<endl;
  simdat << n_sample_fsh_age_sim         <<endl;
}
simdat << "# Observed age compositions for fishery" <<endl;
for (k=1;k<=nfsh;k++)
{
  dvector p(1,nages);
  double Ctmp; // total catch
  dvector freq(1,nages);
  simdat << "# " << fshname(k) <<endl;
  for (i=1;i<=nyrs_fsh_age_sim;i++)
  {
    int iyr = yrs_fsh_age_sim(i);
    // Add noise here
    freq.initialize();
    ivector bin(1,n_sample_fsh_age_sim(i));
    p  = catagetmp(iyr);
    p /= sum(p);
    bin.fill_multinomial(rng,p); // fill a vector v
    for (int j=1;j<=n_sample_fsh_age_sim(i);j++)
      freq(bin(j))++;
    // Apply ageing error to samples.............
    // p = age_err *freq/sum(freq);
    p = freq/sum(freq);
    // cout << p  <<endl;
    simdat << p  <<endl;
    // Compute total catch given this sample size for catch-age
    Ctmp = sim_catchbio(iyr) / (p*wt_fsh(k,iyr));
    // Simulated catage = proportion sampled
    // sim_catage(k,i) = p * Ctmp;
  }
}
simdat << "# Annual wt-at-age for fishery" <<endl;
for (k=1;k<=nfsh;k++)
{
  simdat << "# " <<fshname(k)<< " " << (k) <<endl;
  // Add noise here
  simdat << wt_fsh(k)  <<endl;
}
simdat << "# number of indices" <<endl;
simdat << nind <<endl;
simdat << indnameread <<endl;
simdat << "# Number of years of index values (annual)" <<endl;
for (k=1;k<=nind;k++)
{
  simdat << "# " << indname(k) <<endl;
  simdat << nyrs_ind_sim  <<endl;
}
simdat << "# Years of index values (annual)" <<endl;
for (k=1;k<=nind;k++)
{
  simdat << "# " << indname(k) <<endl;
  simdat << yrs_ind_sim <<endl;
}
simdat << "# Month that index occurs "<<endl;
for (k=1;k<=nind;k++)
{
  simdat << "# " << indname(k) <<endl;
```

```
      simdat << mo_ind(k) <<endl;
    }
    simdat << "# values for indices (annual)"<<endl;
    // note assumes only one index...
    double ind_sigma;
    dvector ind_devs(1,nyrs_ind_sim);
    for (k=1;k<=nind;k++)
    {
      ind_sigma = mean(obs_lse_ind(k)) ;
      ind_sigma = 0.10 ;
      simdat << "# " <<indname(k)<< " " << k <<endl;
      // Add noise here
      // fill vector with unit normal RVs
      ind_devs.fill_randn(rng);
      ind_devs *= ind_sigma ;
      for (i=1;i<=nyrs_ind_sim;i++)
      {
        int iyr=yrs_ind_sim(i);
        //uncorrelated...corr_dev(k,i) = ac(k) * corr_dev(k,i-1) + sqrt(1.-square(ac(k))) *
corr_dev(k,i);
        new_ind_sim(i) = mfexp(ind_devs(i) - ind_sigma/2.) *
value(elem_prod(wt_ind(k,iyr),elem_prod(pow(S(iyr),ind_month_frac(k)),
                        sim_natage(iyr))) * q_ind_sim*sel_ind(k,iyr));
      }
      simdat << new_ind_sim     <<endl;
      dvector ExactSurvey = elem_div(new_ind_sim,exp(ind_devs-ind_sigma/2.));
      truth(ExactSurvey);
    }
    simdat << "# standard errors for indices (by year) " <<endl;
    for (k=1;k<=nind;k++)
    {
      simdat << "# " <<indname(k)<< " " << k <<endl;
      // simdat << new_ind_sim*mean(elem_div(obs_se_ind(k),obs_ind(k)))  <<endl;
      simdat << new_ind_sim*ind_sigma  <<endl;
    }
    simdat << "# Number of years of age data available for index" <<endl;
    for (k=1;k<=nind;k++)
    {
      simdat << "# " <<indname(k)<< " " << k <<endl;
      simdat << nyrs_ind_age_sim <<endl;
    }
    simdat << "# Years of index values (annual)" <<endl;
    for (k=1;k<=nind;k++)
    {
      simdat << "# " <<indname(k)<< endl;
      simdat << yrs_ind_age_sim <<endl;
    }
    simdat << "# Sample sizes for age data from indices" <<endl;
    for (k=1;k<=nind;k++)
    {
      n_sample_ind_age_sim = mean(n_sample_ind_age(k));
      simdat << "# " <<indname(k)<< endl;
      simdat << n_sample_ind_age_sim <<endl;
    }
    simdat << "# values of proportions at age in index" <<endl;
    for (k=1;k<=nind;k++)
    {
      simdat << "# " <<indname(k)<< endl;
      dvector p(1,nages);
      dvector freq(1,nages);
      for (i=1;i<=nyrs_ind_age_sim;i++)
      {
        int iyr = yrs_ind_age_sim(i);
        // Add noise here
        freq.initialize();
        ivector bin(1,n_sample_ind_age_sim(i));
        // p = age_err * value(elem_prod( elem_prod(pow(S(iyr),ind_month_frac(k)),
sim_natage(iyr))*q_ind_sim , sel_ind(k,iyr)));
        p = value(elem_prod( elem_prod(pow(S(iyr),ind_month_frac(k)), sim_natage(iyr))*q_ind_sim ,
sel_ind(k,iyr)));
        p /= sum(p);
```

```
    // fill vector with multinomial samples
    bin.fill_multinomial(rng,p); // fill a vector v
    for (int j=1;j<=n_sample_ind_age_sim(i);j++)
      freq(bin(j))++;
    simdat << "# " <<indname(k)<< " year: "<< iyr<< endl;
    simdat << freq/sum(freq) <<endl;
  }
}
simdat << "# Mean wts at age for indices" <<endl;
for (k=1;k<=nind;k++)
{
  simdat << "# " <<indname(k)<< endl;
  // Could add noise here
  simdat <<  wt_ind(k)  <<endl;
}

simdat << "# Population mean wt at age" <<endl;
simdat << wt_pop <<endl;

simdat << "# Population maturity at age" <<endl;
simdat << maturity  <<endl;

simdat << "# Peak spawning month" <<endl;
simdat << spawnmo <<endl;

simdat << "# ageing error " <<endl;
simdat << age_err <<endl;

simdat <<endl<<endl<<"Additional output"<<endl;
simdat << "# Fishery_Effort " <<endl;
for (k=1;k<=nfsh;k++)
{
  dvector ran_fsh_vect(styr,endyr);
  // fill vector with unit normal RVs
  ran_fsh_vect.fill_randn(rng);
  // Sigma on effort is ~15% white noise (add red noise later)
  ran_fsh_vect *= 0.15;
  dvector avail_biom(styr,endyr);
  for (i=styr;i<=endyr;i++)
  {
    avail_biom(i) = wt_fsh(k,i)*value(elem_prod(sim_natage(i),sel_fsh(k,i)));
  }
  act_eff(k) = elem_prod(exp(ran_fsh_vect), (elem_div(catch_bio(k), avail_biom)) );
  // Normalize effort
  act_eff(k) /= mean(act_eff(k));
  for (i=styr;i<=endyr;i++)
    simdat<<fshname(k)<<" "<<i<<" "<<act_eff(k,i) <<endl;
}
simdat << "# Fishery catch-at-age " <<endl;
for (k=1;k<=nfsh;k++)
{
  simdat << "# " <<fshname(k)<< " " << k <<endl;
  simdat << "Fishery Year "<<age_vector << endl;
  for (i=1;i<=nyrs_fsh_age(k);i++)
    simdat<<fshname(k)<<" "<<yrs_fsh_age(k,i)<<" "<<catagetmp(yrs_fsh_age(k,i)) <<endl;
}
// Write simple file by simulation
dvector ExactSurvey = elem_div(new_ind_sim,exp(ind_devs-ind_sigma/2.));
for (i=styr;i<=endyr;i++)
{
  SimDB<<model_name<<" "<<isim<<" "<< i<<" "<<
    sim_catchbio(i)       <<" "<<
    new_ind_sim(i-styr+1) <<" "<<
    new_ind_sim(i-styr+1)*ind_sigma  <<endl;
  TruDB<<model_name<<" " <<isim<<" "<< i<<" "<<
    sim_catchbio(i)      <<" "<<
    sim_natage(i,1)      <<" "<<
    sim_Sp_Biom(i)       <<" "<<
    ExactSurvey(i-styr+1)<<" "<<
    steepness            <<" "<<
    Bmsy                 <<" "<<
```

```
        MSYL                 <<" "<<
        MSY                  <<" "<<
        SurvBmsy             <<" "<<
        endl;
    }
    TruDB<<model_name<<" "<<isim<<" "<< endyr+1<<" "<<
        OFL                  <<" "<<
        SRecruit(sim_Sp_Biom(endyr+1-rec_age))<<" "<<
        sim_Sp_Biom(endyr)   <<" "<<
        NextSurv             <<" "<<
        steepness            <<" "<<
        Bmsy                 <<" "<<
        MSYL                 <<" "<<
        MSY                  <<" "<<
        SurvBmsy             <<" "<<
        endl;

    trudat.close();
  }
  SimDB.close();
  TruDB.close();
  exit(1);
  // End of simulating datasets..................
  }


FUNCTION Write_Datafile
  dmatrix new_ind(1,nind,1,nyrs_ind);
  new_ind.initialize();
  int nsims;
  // get the number of simulated datasets to create...
  ifstream sim_in("nsims.dat"); sim_in >> nsims; sim_in.close();
  char buffer [33];
  // compute the autocorrelation term for residuals of fit to indices...
  for (k=1;k<=nind;k++)
    ac(k) = get_AC(k);
  for (int isim=1;isim<=nsims;isim++)
  {
    // Create the name of the simulated dataset
    simname = "sim_"+ adstring(itoa(isim,buffer,10)) + ".dat";
    // Open the simulated dataset for writing
    ofstream simdat(simname);
    simdat << "# first year" <<endl;
    simdat << styr <<endl;
    simdat << "# Last  year" <<endl;
    simdat << endyr <<endl;
    simdat << "# age recruit" <<endl;
    simdat << rec_age <<endl;
    simdat << "# oldest age" <<endl;
    simdat << oldest_age <<endl;
    simdat << "# Number of fisheries " <<endl;
    simdat << nfsh <<endl;
    simdat << fshnameread <<endl;
    simdat << "# Catch biomass by fishery " <<endl;
    for (k=1;k<=nfsh;k++)
    {
      simdat << "# " <<fshname(k) <<" " << k <<endl;
      simdat << catch_bio(k) <<endl;
    }
    simdat << "# Catch biomass uncertainty by fishery (std errors)" <<endl;
    for (k=1;k<=nfsh;k++)
    {
      simdat << "# " <<fshname(k) <<" " << k <<endl;
      simdat << catch_bio_sd(k) <<endl;
    }
    simdat << "# number of years for fishery age data " <<endl;
    for (k=1;k<=nfsh;k++)
    {
      simdat << "# " <<fshname(k)<< " " << k <<endl;
      simdat << nyrs_fsh_age(k) <<endl;
    }
```

```
simdat << "# years for fishery age data " <<endl;
for (k=1;k<=nfsh;k++)
{
  simdat << "# " <<fshname(k)<< " " << k <<endl;
  simdat << yrs_fsh_age(k)  <<endl;
}
simdat << "# sample sizes for fishery age data " <<endl;
for (k=1;k<=nfsh;k++)
{
  simdat << "# " <<fshname(k)<< " " << k <<endl;
  simdat << n_sample_fsh_age(k)  <<endl;
}
simdat << "# Observed age compositions for fishery" <<endl;
for (k=1;k<=nfsh;k++)
{
  dvector p(1,nages);
  double Ctmp; // total catch
  dvector freq(1,nages);
  simdat << "# " << fshname(k) <<endl;
  for (i=1;i<=nyrs_fsh_age(k);i++)
  {
    int iyr = yrs_fsh_age(k,i);
    // Add noise here
    freq.initialize();
    ivector bin(1,n_sample_fsh_age(k,i));
    p  = value(catage(k,iyr));
    p /= sum(p);
    bin.fill_multinomial(rng,p); // fill a vector v
    for (int j=1;j<=n_sample_fsh_age(k,i);j++)
      freq(bin(j))++;
    // Apply ageing error to samples..............
    p = age_err *freq/sum(freq);
    // cout << p  <<endl;
    simdat << p  <<endl;
    // Compute total catch given this sample size
    Ctmp = catch_bio(k,iyr) / (p*wt_fsh(k,iyr));
    // Simulated catage = proportion sampled
    catage(k,i) = p * Ctmp;
  }
}
simdat << "# Annual wt-at-age for fishery" <<endl;
for (k=1;k<=nfsh;k++)
{
  simdat << "# " <<fshname(k)<< " " << (k) <<endl;
  // Add noise here
  simdat << wt_fsh(k)  <<endl;
}
simdat << "# number of indices" <<endl;
simdat << nind <<endl;
simdat << indnameread <<endl;
simdat << "# Number of years of index values (annual)" <<endl;
for (k=1;k<=nind;k++)
{
  simdat << "# " << indname(k) <<endl;
  simdat << nyrs_ind(k)  <<endl;
}
simdat << "# Years of index values (annual)" <<endl;
for (k=1;k<=nind;k++)
{
  simdat << "# " << indname(k) <<endl;
  simdat << yrs_ind(k)  <<endl;
}
simdat << "# Month that index occurs "<<endl;
for (k=1;k<=nind;k++)
{
  simdat << "# " << indname(k) <<endl;
  simdat << mo_ind(k) <<endl;
}
simdat << "# values for indices (annual)"<<endl;
for (k=1;k<=nind;k++)
{
```

```
      simdat << "# " <<indname(k)<< " " << k <<endl;
      // Add noise here
      dvector ran_ind_vect(1,nyrs_ind(k));
      // fill vector with unit normal RVs
      ran_ind_vect.fill_randn(rng);
      // do first year uncorrelated
      i=1;
      int iyr=yrs_ind(k,i);
      corr_dev(k)  = ran_ind_vect;
      new_ind(k,i) = mfexp(corr_dev(k,i) * obs_lse_ind(k,i) ) *
                     value(elem_prod(wt_ind(k,iyr),elem_prod(pow(S(iyr),ind_month_frac(k)),
natage(iyr)))*
                     q_ind(k,i)*sel_ind(k,iyr));
      // do next years correlated with previous
      for (i=2;i<=nyrs_ind(k);i++)
      {
        iyr=yrs_ind(k,i);
        corr_dev(k,i) = ac(k) * corr_dev(k,i-1) + sqrt(1.-square(ac(k))) * corr_dev(k,i);
        new_ind(k,i) = mfexp(corr_dev(k,i) * obs_lse_ind(k,i) ) *
                         value(elem_prod(wt_ind(k,iyr),elem_prod(pow(S(iyr),ind_month_frac(k)),
                         natage(iyr))) * q_ind(k,i)*sel_ind(k,iyr));
      }
      simdat << new_ind(k)      <<endl;
    }
    simdat << "# standard errors for indices (by year) " <<endl;
    for (k=1;k<=nind;k++)
    {
      simdat << "# " <<indname(k)<< " " << k <<endl;
      simdat << obs_se_ind(k)  <<endl;
    }
    simdat << "# Number of years of age data available for index" <<endl;
    for (k=1;k<=nind;k++)
    {
      simdat << "# " <<indname(k)<< " " << k <<endl;
      simdat << nyrs_ind_age(k)  <<endl;
    }
    simdat << "# Years of index values (annual)" <<endl;
    for (k=1;k<=nind;k++)
    {
      simdat << "# " <<indname(k)<< endl;
      simdat << yrs_ind_age(k)  <<endl;
    }
    simdat << "# Sample sizes for age data from indices" <<endl;
    for (k=1;k<=nind;k++)
    {
      simdat << "# " <<indname(k)<< endl;
      simdat << n_sample_ind_age(k)  <<endl;
    }
    simdat << "# values of proportions at age in index" <<endl;
    for (k=1;k<=nind;k++)
    {
      simdat << "# " <<indname(k)<< endl;
      dvector p(1,nages);
      dvector freq(1,nages);
      for (i=1;i<=nyrs_ind_age(k);i++)
      {
        int iyr = yrs_ind_age(k,i);
        // Add noise here
        freq.initialize();
        ivector bin(1,n_sample_ind_age(k,i));
        p = age_err * value(elem_prod( elem_prod(pow(S(iyr),ind_month_frac(k)),
natage(iyr))*q_ind(k,i) , sel_ind(k,iyr)));
        p /= sum(p);
        // fill vector with multinomial samples
        bin.fill_multinomial(rng,p); // fill a vector v
        for (int j=1;j<=n_sample_ind_age(k,i);j++)
          freq(bin(j))++;
        simdat << "# " <<indname(k)<< " year: "<< iyr<< endl;
        simdat << freq/sum(freq) <<endl;
      }
    }
```

```
    simdat << "# Mean wts at age for indices" <<endl;
    for (k=1;k<=nind;k++)
    {
      simdat << "# " <<indname(k)<< endl;
      // Could add noise here
      simdat <<  wt_ind(k)  <<endl;
    }

    simdat << "# Population mean wt at age" <<endl;
    simdat << wt_pop <<endl;

    simdat << "# Population maturity at age" <<endl;
    simdat << maturity  <<endl;

    simdat << "# Peak spawning month" <<endl;
    simdat << spawnmo <<endl;

    simdat << "# ageing error " <<endl;
    simdat << age_err <<endl;

    simdat <<endl<<endl<<"Additional output"<<endl;
    simdat << "# Fishery_Effort " <<endl;
    for (k=1;k<=nfsh;k++)
    {
      dvector ran_fsh_vect(styr,endyr);
      // fill vector with unit normal RVs
      ran_fsh_vect.fill_randn(rng);
      // Sigma on effort is ~15% white noise (add red noise later)
      ran_fsh_vect *= 0.15;
      dvector avail_biom(styr,endyr);
      for (i=styr;i<=endyr;i++)
      {
        avail_biom(i) = wt_fsh(k,i)*value(elem_prod(natage(i),sel_fsh(k,i)));
      }
      act_eff(k) = elem_prod(exp(ran_fsh_vect), (elem_div(catch_bio(k), avail_biom)) );
      // Normalize effort
      act_eff(k) /= mean(act_eff(k));
      for (i=styr;i<=endyr;i++)
        simdat<<fshname(k)<<" "<<i<<" "<<act_eff(k,i) <<endl;
    }
    simdat << "# Fishery catch-at-age " <<endl;
    for (k=1;k<=nfsh;k++)
    {
      simdat << "# " <<fshname(k)<< " " << k <<endl;
      simdat << "Fishery Year "<<age_vector << endl;
      for (i=1;i<=nyrs_fsh_age(k);i++)
        simdat<<fshname(k)<<" "<<yrs_fsh_age(k,i)<<" "<<catage(k,i) <<endl;
    }
  }
  exit(1);
  // End of simulating datasets..................



FUNCTION Write_R
  ofstream R_report("For_R.rep");
  R_report<<"$M"<<endl;
  R_report<<M<<endl;
  R_report<<"$SurvNextYr"<<endl; R_report<< pred_ind_nextyr <<endl;
  R_report<<"$Yr"<<endl; for (i=styr;i<=endyr;i++) R_report<<i<<" "; R_report<<endl;

  R_report<<"$TotF"<<endl << Ftot<<endl;

  R_report<<"$TotBiom_NoFish"<<endl; for (i=styr;i<=endyr;i++)
  {
    double
lb=value(totbiom_NoFish(i)/exp(2.*sqrt(log(1+square(totbiom_NoFish.sd(i))/square(totbiom_NoFish(i))))
));
    double
ub=value(totbiom_NoFish(i)*exp(2.*sqrt(log(1+square(totbiom_NoFish.sd(i))/square(totbiom_NoFish(i))))
));
```

```
    R_report<<i<<" "<<totbiom_NoFish(i)<<" "<<totbiom_NoFish.sd(i)<<" "<<lb<<" "<<ub<<endl;
  }
  R_report<<"$SSB_NoFishR"<<endl; for (i=styr+1;i<=endyr;i++)
  {
    double
lb=value(Sp_Biom_NoFishRatio(i)/exp(2.*sqrt(log(1+square(Sp_Biom_NoFishRatio.sd(i))/square(Sp_Biom_No
FishRatio(i))))));
    double
ub=value(Sp_Biom_NoFishRatio(i)*exp(2.*sqrt(log(1+square(Sp_Biom_NoFishRatio.sd(i))/square(Sp_Biom_No
FishRatio(i))))));
    R_report<<i<<" "<<Sp_Biom_NoFishRatio(i)<<" "<< Sp_Biom_NoFishRatio.sd(i)<<" "<<lb<<"
"<<ub<<endl;
  }

  R_report<<"$TotBiom"<<endl;
  for (i=styr;i<=endyr;i++)
  {
    double lb=value(totbiom(i)/exp(2.*sqrt(log(1+square(totbiom.sd(i))/square(totbiom(i))))));
    double ub=value(totbiom(i)*exp(2.*sqrt(log(1+square(totbiom.sd(i))/square(totbiom(i))))));
    R_report<<i<<" "<<totbiom(i)<<" "<<totbiom.sd(i)<<" "<<lb<<" "<<ub<<endl;
  }

  for (k=1;k<=5;k++){
    R_report<<"$SSB_fut_"<<k<<endl;
    for (i=styr_fut;i<=endyr_fut;i++)
    {
      double
lb=value(SSB_fut(k,i)/exp(2.*sqrt(log(1+square(SSB_fut.sd(k,i))/square(SSB_fut(k,i))))));
      double
ub=value(SSB_fut(k,i)*exp(2.*sqrt(log(1+square(SSB_fut.sd(k,i))/square(SSB_fut(k,i))))));
      R_report<<i<<" "<<SSB_fut(k,i)<<" "<<SSB_fut.sd(k,i)<<" "<<lb<<" "<<ub<<endl;
    }
  }
  double ctmp;
  for (k=1;k<=5;k++){
    R_report<<"$Catch_fut_"<<k<<endl;
    for (i=styr_fut;i<=endyr_fut;i++)
    {
      if (k==5) ctmp=0.;else ctmp=value(catch_future(k,i));
      R_report<<i<<" "<<ctmp<<endl;
    }
  }

  R_report<<"$SSB"<<endl; for (i=styr_sp;i<=endyr+1;i++)
  {
    double lb=value(Sp_Biom(i)/exp(2.*sqrt(log(1+square(Sp_Biom.sd(i))/square(Sp_Biom(i))))));
    double ub=value(Sp_Biom(i)*exp(2.*sqrt(log(1+square(Sp_Biom.sd(i))/square(Sp_Biom(i))))));
    R_report<<i<<" "<<Sp_Biom(i)<<" "<<Sp_Biom.sd(i)<<" "<<lb<<" "<<ub<<endl;
  }

  R_report<<"$R"<<endl; for (i=styr;i<=endyr;i++)
  {
    double lb=value(recruits(i)/exp(2.*sqrt(log(1+square(recruits.sd(i))/square(recruits(i))))));
    double ub=value(recruits(i)*exp(2.*sqrt(log(1+square(recruits.sd(i))/square(recruits(i))))));
    R_report<<i<<" "<<recruits(i)<<" "<<recruits.sd(i)<<" "<<lb<<" "<<ub<<endl;
  }
    R_report << "$N"<<endl;
    for (i=styr;i<=endyr;i++)
      R_report <<   i << " "<< natage(i) << endl;
      R_report   << endl;

    for (k=1;k<=nfsh;k++)
    {
      R_report << "$F_age_"<< (k) <<""<< endl ;
      for (i=styr;i<=endyr;i++)
        R_report <<i<<" "<<F(k,i)<<" "<< endl;
        R_report   << endl;
    }

    R_report <<endl<< "$Fshry_names"<< endl;
    for (k=1;k<=nfsh;k++)
```

```
      R_report << fshname(k) << endl ;

    R_report <<endl<< "$Index_names"<< endl;
    for (k=1;k<=nind;k++)
      R_report << indname(k) << endl ;

    for (k=1;k<=nind;k++)
    {
      int ii=1;
      R_report <<endl<< "$Obs_Survey_"<< k <<""<< endl ;
      for (i=styr;i<=endyr;i++)
      {
        if (ii<=yrs_ind(k).indexmax())
        {
          if (yrs_ind(k,ii)==i)
          {
            R_report << i<< " "<< obs_ind(k,ii) << " "<< pred_ind(k,ii) <<" "<< obs_se_ind(k,ii)
<<endl; //values of survey index value (annual)
            ii++;
          }
          // else
            // R_report << i<< " -1 "<< " "<< pred_ind(k,i)<<" -1 "<<endl;
        }
        // else
          // R_report << i<< " -1 "<< " "<< pred_ind(k,i)<<" -1 "<<endl;
      }
      R_report   << endl;
      R_report << endl<< "$Index_Q_"<<k<<endl;
      R_report<< q_ind(k) << endl;
    }
    R_report   << endl;
    for (k=1;k<=nfsh;k++)
    {
      if (nyrs_fsh_age(k)>0)
      {
        R_report << "$pobs_fsh_"<< (k) <<""<< endl;
        for (i=1;i<=nyrs_fsh_age(k);i++)
          R_report << yrs_fsh_age(k,i)<< " "<< oac_fsh(k,i) << endl;
        R_report   << endl;
      }
    }
    for (k=1;k<=nfsh;k++)
    {
      if (nyrs_fsh_age(k)>0)
      {
        R_report << "$phat_fsh_"<< (k) <<""<< endl;
        for (i=1;i<=nyrs_fsh_age(k);i++)
          R_report << yrs_fsh_age(k,i)<< " "<< eac_fsh(k,i) << endl;
          R_report   << endl;
      }
    }
    for (k=1;k<=nind;k++)
    {
      if (nyrs_ind_age(k)>0)
      {
        R_report << "$pobs_ind_"<<(k)<<""<<  endl;
        for (i=1;i<=nyrs_ind_age(k);i++)
          R_report << yrs_ind_age(k,i)<< " "<< oac_ind(k,i) << endl;
          R_report   << endl;
      }
    }
    for (k=1;k<=nind;k++)
    {
      if (nyrs_ind_age(k)>0)
      {
        R_report << "$phat_ind_"<<(k)<<""<<  endl;
        for (i=1;i<=nyrs_ind_age(k);i++)
          R_report << yrs_ind_age(k,i)<< " "<< eac_ind(k,i) << endl;
          R_report   << endl;
      }
    }
```

```
    for (k=1;k<=nfsh;k++)
    {
      R_report << endl<< "$Obs_catch_"<<(k) << endl;
      R_report << catch_bio(k) << endl;
      R_report   << endl;
      R_report << "$Pred_catch_" <<(k) << endl;
      R_report << pred_catch(k) << endl;
      R_report   << endl;
    }

    for (k=1;k<=nfsh;k++)
    {
      R_report << "$F_fsh_"<<(k)<<" "<<endl;
      for (i=styr;i<=endyr;i++)
      {
        R_report<< i<< " ";
        R_report<< mean(F(k,i)) <<" "<< mean(F(k,i))*max(sel_fsh(k,i)) << " ";
        R_report<< endl;
      }
    }

    for (k=1;k<=nfsh;k++)
    {
      R_report << endl<< "$sel_fsh_"<<(k)<<"" << endl;
      for (i=styr;i<=endyr;i++)
        R_report << k <<"  "<< i<<" "<<sel_fsh(k,i) << endl;
      R_report   << endl;
    }

    for (k=1;k<=nind;k++)
    {
      R_report << endl<< "$sel_ind_"<<(k)<<"" << endl;
      for (i=styr;i<=endyr;i++)
        R_report << k <<"  "<< i<<" "<<sel_ind(k,i) << endl;
        R_report << endl;

    }
    R_report << endl<< "$Stock_Rec"<< endl;
    for (i=styr_rec;i<=endyr;i++)
      if (active(log_Rzero))
        R_report << i<< " "<<Sp_Biom(i-rec_age)<< " "<< SRecruit(Sp_Biom(i-rec_age))<< " "<<
mod_rec(i)<<endl;
      else
        R_report << i<< " "<<Sp_Biom(i-rec_age)<< " "<< " 999" << " "<< mod_rec(i)<<endl;

        R_report   << endl;

    R_report <<"$stock_Rec_Curve"<<endl;
    R_report <<"0 0"<<endl;
    dvariable stock;
    for (i=1;i<=30;i++)
    {
      stock = double (i) * Bzero /25.;
      if (active(log_Rzero))
        R_report << stock <<" "<< SRecruit(stock)<<endl;
      else
        R_report << stock <<" 99 "<<endl;
    }
    R_report   << endl;

    R_report   << endl<<"$Like_Comp" <<endl;
    obj_comps(11)= obj_fun - sum(obj_comps) ; // Residual
    obj_comps(12)= obj_fun ;

    R_report   <<obj_comps<<endl;
    R_report   << endl;
    R_report   << endl<<"$Like_Comp_names" <<endl;
    R_report   <<"catch_like     "<<endl
             <<"age_like_fsh     "<<endl
             <<"sel_like_fsh     "<<endl
             <<"ind_like       "<<endl
```

```
         <<"age_like_ind     "<<endl
         <<"sel_like_ind     "<<endl
         <<"rec_like         "<<endl
         <<"fpen             "<<endl
         <<"post_priors_indq "<<endl
         <<"post_priors      "<<endl
         <<"residual         "<<endl
         <<"total            "<<endl;
for (k=1;k<=nfsh;k++)
{
  R_report << "$Sel_Fshry_"<< (k) <<""<<endl;
  R_report << sel_like_fsh(k) << endl;
}
R_report   << endl;

for (k=1;k<=nind;k++)
{
  R_report << "$Survey_Index_"<< (k) <<"" <<endl;
  R_report<< ind_like(k)<<endl;
}
R_report   << endl;

R_report << setw(10)<< setfixed() << setprecision(5) <<endl;
for (k=1;k<=nind;k++)
{
  R_report << "$Age_Survey_"<< (k) <<"" <<endl;
  R_report << age_like_ind(k)<<endl;
}
R_report   << endl;

for (k=1;k<=nind;k++)
{
  R_report << "$Sel_Survey_"<< (k) <<""<<endl;
  R_report<< sel_like_ind(k,1) <<" "<<sel_like_ind(k,2)<<" "<<sel_like_ind(k,3)<< endl;
}
R_report   << endl;

R_report << setw(10)<< setfixed() << setprecision(5) <<endl;
R_report   << "$Rec_Pen" <<endl<<sigmar<<"  "<<rec_like<<endl;
R_report   << endl;
R_Report(m_sigmar);
R_Report(sigmar);

R_report   << "$F_Pen" <<endl;
R_report<<fpen(1)<<"  "<<fpen(2)<<endl;
R_report   << endl;
for (k=1;k<=nind;k++)
{
  R_report << "$Q_Survey_"<< (k) <<""<<endl
         << " "<<post_priors_indq(k)
         << " "<< q_ind(k,1)
         << " "<< qprior(k)
         << " "<< cvqprior(k)<<endl;
  R_report << "$Q_power_Survey_"<< (k) <<""<<endl
         << " "<<post_priors_indq(k)
         << " "<< q_power_ind(k)
         << " "<< q_power_prior(k)
         << " "<< cvq_power_prior(k)<<endl;
}
         R_report   << endl;
R_report << "$Mest"<<endl;
R_report << " "<< post_priors(1)
         << " "<< Mest
         << " "<< natmortprior
         << " "<< cvnatmortprior <<endl;
R_report   << endl;
R_report << "$Steep"<<endl;
R_report << " "<< post_priors(2)
         << " "<< steepness
         << " "<< steepnessprior
         << " "<< cvsteepnessprior <<endl;
```

```
  R_report   << endl;
  R_report << "$Sigmar"<<endl;
  R_report << " "<< post_priors(3)
           << " "<< sigmar
           << " "<< sigmarprior
           << " "<< cvsigmarprior <<endl;
  R_report   << endl;
  R_report<<"$Num_parameters_Est"<<endl;
  R_report<<initial_params::nvarcalc()<<endl;
  R_report   << endl;

R_report<<"$Steep_Prior" <<endl;
R_report<<steepnessprior<<" "<<
  cvsteepnessprior<<" "<<
  phase_srec<<" "<< endl;
  R_report   << endl;

R_report<<"$sigmarPrior " <<endl;
R_report<<sigmarprior<<" "<<  cvsigmarprior <<" "<<phase_sigmar<<endl;
R_report   << endl;

R_report<<"$Rec_estimated_in_styr_endyr " <<endl;
R_report<<styr_rec    <<" "<<endyr        <<" "<<endl;
R_report   << endl;
R_report<<"$SR_Curve_fit__in_styr_endyr " <<endl;
R_report<<styr_rec_est<<" "<<endyr_rec_est<<" "<<endl;
R_report   << endl;
R_report<<"$Model_styr_endyr" <<endl;
R_report<<styr        <<" "<<endyr        <<" "<<endl;
R_report   << endl;

R_report<<"$M_prior "<<endl;
R_report<< natmortprior<< " "<< cvnatmortprior<<" "<<phase_M<<endl;
R_report   << endl;
R_report<<"$qprior " <<endl;
R_report<< qprior<<" "<<cvqprior<<" "<< phase_q<<endl;
R_report<<"$q_power_prior " <<endl;
R_report<< q_power_prior<<" "<<cvq_power_prior<<" "<< phase_q_power<<endl;
R_report   << endl;

R_report<<"$cv_catchbiomass " <<endl;
R_report<<cv_catchbiomass<<" "<<endl;
R_report   << endl;
R_report<<"$Projection_years"<<endl;
R_report<< nproj_yrs<<endl;
R_report   << endl;

R_report << "$Fsh_sel_opt_fish "<<endl;
for (k=1;k<=nfsh;k++)
  R_report<<k<<" "<<fsh_sel_opt(k)<<" "<<sel_change_in_fsh(k)<<endl;
  R_report   << endl;
 R_report<<"$Survey_Sel_Opt_Survey " <<endl;
for (k=1;k<=nind;k++)
R_report<<k<<" "<<(ind_sel_opt(k))<<endl;
R_report   << endl;

R_report <<"$Phase_survey_Sel_Coffs "<<endl;
R_report <<phase_selcoff_ind<<endl;
R_report   << endl;
R_report <<"$Fshry_Selages " << endl;
R_report << nselages_in_fsh  <<endl;
R_report   << endl;
R_report <<"$Survy_Selages " <<endl;
R_report <<nselages_in_ind <<endl;
R_report   << endl;

R_report << "$Phase_for_age_spec_fishery"<<endl;
R_report <<phase_selcoff_fsh<<endl;
R_report   << endl;
R_report << "$Phase_for_logistic_fishery"<<endl;
R_report <<phase_logist_fsh<<endl;
```

```
R_report   << endl;
R_report << "$Phase_for_dble_logistic_fishery "<<endl;
R_report <<phase_dlogist_fsh<<endl;
R_report   << endl;

R_report << "$Phase_for_age_spec_survey  "<<endl;
R_report <<phase_selcoff_ind<<endl;
R_report   << endl;
R_report << "$Phase_for_logistic_survey  "<<endl;
R_report <<phase_logist_ind<<endl;
R_report   << endl;
R_report << "$Phase_for_dble_logistic_indy "<<endl;
R_report <<phase_dlogist_ind<<endl;
R_report   << endl;

for (k=1;k<=nfsh;k++)
{
  if (nyrs_fsh_age(k)>0)
  {
    R_report <<"$EffN_Fsh_"<<(k)<<""<<endl;
    for (i=1;i<=nyrs_fsh_age(k);i++)
    {
      double sda_tmp = Sd_age(oac_fsh(k,i));
      R_report << yrs_fsh_age(k,i);
      R_report << " "<<Eff_N(oac_fsh(k,i),eac_fsh(k,i)) ;
      R_report << " "<<Eff_N2(oac_fsh(k,i),eac_fsh(k,i));
      R_report << " "<<mn_age(oac_fsh(k,i));
      R_report << " "<<mn_age(eac_fsh(k,i));
      R_report << " "<<sda_tmp;
      R_report << " "<<mn_age(oac_fsh(k,i)) - sda_tmp *2. / sqrt(n_sample_fsh_age(k,i));
      R_report << " "<<mn_age(oac_fsh(k,i)) + sda_tmp *2. / sqrt(n_sample_fsh_age(k,i));
      R_report <<endl;
    }
  }
}

for (k=1;k<=nfsh;k++)
{
  R_report <<"$C_fsh_" <<(k)<<"" << endl;
  for (i=styr;i<=endyr;i++)
    R_report <<i<<" "<<catage(k,i)<< endl;
}

R_report <<"$wt_a_pop" << endl<< wt_pop  <<endl;
R_report <<"$mature_a" << endl<< maturity<<endl;
for (k=1;k<=nfsh;k++)
{
  R_report <<"$wt_fsh_"<<(k)<<""<<endl;
  for (i=styr;i<=endyr;i++)
    R_report <<i<<" "<<wt_fsh(k,i)<< endl;
}

for (k=1;k<=nind;k++)
{
  R_report <<"$wt_ind_"<<(k)<<""<<endl;
  for (i=styr;i<=endyr;i++)
    R_report <<i<<" "<<wt_ind(k,i)<< endl;
}
for (k=1;k<=nind;k++)
{
  if (nyrs_ind_age(k)>0)
  {
    R_report <<"$EffN_Survey_"<<(k)<<""<<endl;
    for (i=1;i<=nyrs_ind_age(k);i++)
    {
      double sda_tmp = Sd_age(oac_ind(k,i));
      R_report << yrs_ind_age(k,i)
               << " "<<Eff_N(oac_ind(k,i),eac_ind(k,i))
               << " "<<Eff_N2(oac_ind(k,i),eac_ind(k,i))
               << " "<<mn_age(oac_ind(k,i))
               << " "<<mn_age(eac_ind(k,i))
```

```
                << " "<<sda_tmp
                << " "<<mn_age(oac_ind(k,i)) - sda_tmp *2. / sqrt(n_sample_ind_age(k,i))
                << " "<<mn_age(oac_ind(k,i)) + sda_tmp *2. / sqrt(n_sample_ind_age(k,i))
                <<endl;
    }
  }
}
R_report<<"$msy_mt"<<endl;
dvar_matrix sel_tmp(1,nages,1,nfsh);
dvariable sumF;
sel_tmp.initialize();
for (i=styr;i<=endyr;i++)
{
  sumF=0.;
  for (k=1;k<=nfsh;k++)
  {
    Fratio(k) = sum(F(k,i)) ;
    sumF += Fratio(k) ;
  }
  Fratio /= sumF;
  sumF /= nages;
  for (k=1;k<=nfsh;k++)
    for (j=1;j<=nages;j++)
      sel_tmp(j,k) = sel_fsh(k,i,j);
  get_msy(i);
  // important for time-varying natural mortality...
  dvariable spr_mt_ft = spr_ratio(sumF,sel_tmp,i)  ;
  dvariable spr_mt_f0 = spr_ratio(0.,sel_tmp,i)  ;
  R_report<< i<<
          " "<< spr_mt_ft                  <<
          " "<< spr_mt_f0                  <<
          " "<< (1.-spr_mt_f0)/(1-spr_mt_ft)<<
          " "<< Fcur_Fmsy                  <<
          " "<< Fmsy                       <<
          " "<< sumF                       <<
          " "<< spr_ratio(Fmsy,sel_tmp,i)   <<
          " "<< MSY                        <<
          " "<< Bmsy                       <<
          " "<< MSYL                       <<
          " "<< Bcur_Bmsy                  <<
          endl ;
}

R_report<<"$msy_m0"<<endl;
sel_tmp.initialize();
// NOTE Danger here
dvar_vector mtmp = natmort;
natmort = natmort(styr);
for (i=styr;i<=endyr;i++)
{
  sumF=0.;
  for (k=1;k<=nfsh;k++)
  {
    Fratio(k) = sum(F(k,i)) ;
    sumF += Fratio(k) ;
  }
  Fratio /= sumF;
  for (k=1;k<=nfsh;k++)
    for (j=1;j<=nages;j++)
      sel_tmp(j,k) = sel_fsh(k,i,j);
  get_msy(i);
  sumF /= nages;
  // important for time-varying natural mortality...
  dvariable spr_mt_ft = spr_ratio(sumF,sel_tmp,i)  ;
  dvariable spr_mt_f0 = spr_ratio(0.,sel_tmp,i)  ;
  R_report<< i<<
          " "<< spr_mt_ft                  <<
          " "<< spr_mt_f0                  <<
          " "<< (1.-spr_mt_f0)/(1-spr_mt_ft)<<
          " "<< Fcur_Fmsy                  <<
          " "<< Fmsy                       <<
```

```
            " "<< sumF                        <<
            " "<< spr_ratio(Fmsy,sel_tmp,i)   <<
            " "<< MSY                         <<
            " "<< Bmsy                        <<
            " "<< MSYL                        <<
            " "<< Bcur_Bmsy                   <<
            endl ;
  }

  natmort = mtmp;
  R_Report(F40_est);
  R_Report(F35_est);

  R_report.close();



FUNCTION double mn_age(_CONST dvector& pobs)
  // int lb1 = pobs.indexmin();
  // int ub1 = pobs.indexmax();
  // dvector av = age_vector(lb1,ub1)  ;
  // double mobs = value(pobs.shift(rec_age)*age_vector);
  double mobs = (pobs*age_vector);
  return mobs;


FUNCTION double mn_age(_CONST dvar_vector& pobs)
  // int lb1 = pobs.indexmin();
  // int ub1 = pobs.indexmax();
  // dvector av = age_vector(lb1,ub1)  ;
  // double mobs = value(pobs.shift(rec_age)*age_vector);
  double mobs = value(pobs*age_vector);
  return mobs;


FUNCTION double Sd_age(_CONST dvector& pobs)
  // double mobs = (pobs.shift(rec_age)*age_vector);
  // double stmp = (sqrt(elem_prod(age_vector,age_vector)*pobs.shift(rec_age) - mobs*mobs));
  double mobs = (pobs*age_vector);
  double stmp = sqrt((elem_prod(age_vector,age_vector)*pobs) - mobs*mobs);
  return stmp;


FUNCTION double Eff_N_adj(_CONST double, _CONST dvar_vector& pobs, _CONST dvar_vector& phat)
  int lb1 = pobs.indexmin();
  int ub1 = pobs.indexmax();
  dvector av = age_vector(lb1,ub1)  ;
  double mobs = value(pobs*av);
  double mhat = value(phat*av );
  double rtmp = mobs-mhat;
  double stmp = value(sqrt(elem_prod(av,av)*pobs - mobs*mobs));
  return square(stmp)/square(rtmp);

FUNCTION double Eff_N2(_CONST dvector& pobs, _CONST dvar_vector& phat)
  int lb1 = pobs.indexmin();
  int ub1 = pobs.indexmax();
  dvector av = age_vector(lb1,ub1)  ;
  double mobs =       (pobs*av);
  double mhat = value(phat*av );
  double rtmp = mobs-mhat;
  double stmp = (sqrt(elem_prod(av,av)*pobs - mobs*mobs));
  return square(stmp)/square(rtmp);

FUNCTION double Eff_N(_CONST dvector& pobs, _CONST dvar_vector& phat)
  dvar_vector rtmp = elem_div((pobs-phat),sqrt(elem_prod(phat,(1-phat))));
  double vtmp;
  vtmp = value(norm2(rtmp)/size_count(rtmp));
  return 1./vtmp;

FUNCTION double get_AC(_CONST int& indind)
```

```
  // Functions to compute autocorrelation in residuals
  int i1,i2,iyr;
  i1 = 1;
  i2 = nyrs_ind(indind);
  double actmp;
  dvector res(1,i2);
  for (i=1;i<=i2;i++)
  {
    iyr = int(yrs_ind(indind,i));
    cout<<iyr<<" "<<obs_ind(indind,i)<<" " <<pred_ind(indind,iyr)<<endl;
    res(i) = log(obs_ind(indind,i)) - value(log(pred_ind(indind,iyr)));
  }
  double m1 = (mean(res(i1,i2-1)));
  double m2 = (mean(res(i1+1,i2)));
  actmp = mean( elem_prod( ++res(i1,i2-1) - m1, res(i1+1,i2) - m2)) /
          (sqrt(mean( square(res(i1,i2-1) - m1 )))  * sqrt(mean(square(res(i1+1,i2) - m2 ))) );
  return(actmp);


FUNCTION Write_SIS
  SIS_rep << " # constants ########################### \n"
          << " # species  \n"

          << " # region     (AI AK BOG BSAI EBS GOA)  \n"

          << " # assess_year  \n"

          << " # split_sex (True or false) (1 or 0) (if true, FEMALE, Male, else combined)  \n"

          << " # number of fisheries  \n "

          << " # list of fisheries (ALL TWL LGL POT FIX FOR DOM ...) separated w/ %  \n "

          << " # mulitiplier for recruitment and N at age     (1,1000,1000000)  \n"

          << " # mulitiplier for biomass mt, catch mt, and surveybiomass mt    (1,1000,1000000)  \n"

          << " # recruitment age used by model  \n"

          << " # age+ used for biomass estimate  \n"

          << " # number of surveys  \n "

          << " # list of surveys (longline, trawl, acoustic) separated w/ %  \n"

          << "#YEARS -list all years used in model (starting w/ first year of catch)  \n"

          << "#AGES -list ages used in model  \n"

          << "#RECRUITMENT -Number of recruits by model year (see multiplier above)  \n"

          << "#SPAWNBIOMASS -Spawning biomass by model year (see mt multiplier above)  \n"

          << "#TOTALBIOMASS -Total biomass by year (see mt multiplier above and age+ above)  \n"

          << "#TOTFSHRYMORT -Fishing mortality rate by year  \n"

          << "#TOTALCATCH -Total catch by year (see mt multiplier above)  \n"

          << "#FISHERYMORT -Fishing mortality rates by year (a line for each fishery) only if
multiple fisheries  \n"

          << "#FISHERYCATCH -Catches by year (a line for each fishery) only if multiple fisheries
\n"

          << "#MATURITY -Maturity ratio by age  \n"

          << "#SPAWNWT -Average Spawning weight (in kg) by age  \n"

          << "#NATMORT -Natural mortality rate by age (a line for each sex)  \n"
```

```
<< "#N_AT_AGE -N at age by age (see number multiplier above)(a line for each sex)  \n"

<< "#FSHRY_WT_KG_SEX1 -Fishery weight at age (in kg)(a line for each fishery)  \n"

<< "#FSHRY_WT_KG_SEX2 -Fishery weight at age (in kg)(a line for each fishery)  \n"

<< "#SELECTIVITY_SEX1 -Fishery selectivity (a line for each fishery)  \n"

<< "#SELECTIVITY_SEX2 -Fishery selectivity (a line for each fishery)  \n"

<< "#SURVEYYEARS - list the survey years (a line for each survey)  \n"

<< "#SURVEYBIOMASS -Survey biomass by survey year (see mt multiplier above)(a line for each survey)  \n"

<< endl;
```