Age-structured model for Alaska herring stocks

Steve Martell

July 25, 2016

Abstract

1 Introduction

2 Model deconstruction

This section is intended to serve three purposes: 1) to document the model structure given the code in model.tpl, 2) to detail proposed changes to the model code to improve overall numerical stability, and 3) provide statistical approach that is amenable to maximum likelihood and Bayesian methods.

2.1 Model Structure

Table 1 begins with the objective function that is being minimized. There are four components defined in (T1.1), where three of the four components are scaled by user defined coefficients λ. The first component is the commercial age-composition data (QC), the second is the spawnig biomass age-composition (QS), the third is egg deposition data (WQE), and finally the fourth component is a penalty on the recruitment deviations from the underlying Ricker stock-recruitment model (QR).

For the commercial age-composition data, observation errors in the age-proportions are assumed to be normal (T1.2), where the predicted proportion-at-age (T1.3) is a function of the numbers-at-age (T1.4) and selectivity (T1.5). Note that in (T1.4) that the function is not continous. In this case the selectivity curve is rescaled to have a maximum value of 1.0. The `max` operation in the denominator of this function breaks the derivative chain in AD Model Builder and can result in numerical problems during parameter optimization associated with corrupt derivative information.

The same normal error distribution is also assumed for the age-proportions in the spawner catch composition samples (T1.6). In this case, the age proportions aree based on the mature numbers-at-age at the time of spawning, where the fisehries removals are first subtracted from the mature numbers-at-age (T1.7). Note that this further assumes that all removals (i.e.,

fisheries selectivity) only harvests sexually mature fish. This assumption is inconsistent with (T1.4), where a different logistic curve is assumed for fisheries selectivity.

The catch-at-age data is internally derived in the model (T1.9) conditional on the numbers-at-age and the estimated selectivity. The model further assumes the total catch (in short tons) is measured without error. This is also referred to as conditioning the model on catch.

The residual sum of squares for the egg deposition suvey is given by (T1.10). In this case the obsrevation errors are assumed to be lognormal, and each year's observation is weighted by the inverse variance of sampling observation errors (φ_i).

Table 1: Decomposition of the objective function based on the source code provided in `model.tpl`. The objective function f is what AD Model Builder is trying to minimize. Note that $\dot{}$ represents mature state variables (e.g., mature weight-at-age $\dot{w}_j$)

Objective function

$$f = \lambda_C QC + \lambda_S QS + WQE + \lambda_R QR \tag{T1.1}$$

Commercial catch proportion-at-age

$$QC = \sum_i \sum_j (\hat{Q}_{i,j} - Q_{i,j})^2 \qquad \hat{Q}_{i,j} \text{ observed proportions-at-age} \tag{T1.2}$$

$$Q_{i,j} = \frac{V_{i,j}}{\sum_j V_{i,j}} \qquad Q_{i,j} \text{ predicted proportion-at-age} \tag{T1.3}$$

$$V_{i,j} = N_{i,j} \frac{S_{i,j}}{\max(S_{i,j})} \qquad V_{i,j} \text{ vulnerable numbers-at-age} \tag{T1.4}$$

$$S_{i,j} = \frac{1}{1 + \exp(-g_i(j - a_i))} \qquad S_{i,j} \text{ Selectivity in year } i \text{ for age } j \tag{T1.5}$$

Spawn proportion-at-age

$$QS = \sum_i \sum_j (\hat{O}_{i,j} - O_{i,j})^2 \qquad \hat{O}_{i,j} \text{ observed proportion-at-age} \tag{T1.6}$$

$$O_{i,j} = \frac{\dot{N}_{i,j}}{\sum_j \dot{N}_{i,j}} \qquad O_{i,j} \text{ predicted proportion mature-at-age} \tag{T1.7}$$

$$\dot{N}_{i,j} = N_{i,j} M_{i,j} - C_{i,j} \qquad \dot{N}_{ij} \text{ Number-at-age at spawning} \tag{T1.8}$$

$$C_{i,j} = \frac{\hat{c}_i Q_{i,j}}{\sum_j Q_{i,j} w_j} \qquad \hat{c}_i \text{ observed catch, } w_j \text{ weight-at-age} \tag{T1.9}$$

Egg deposition survey

$$WQE = \sum_i \varphi_i \left[\ln(\hat{y}_i) - \ln(y_i)\right]^2 \qquad \hat{y}_i \text{ observed egg deposition, } \varphi_i \text{ weight} \tag{T1.10}$$

$$y_i = 0.5 \sum_j O_{i,j}(\rho_i \dot{w}_{i,j} - \alpha_i) \qquad \rho_i \text{ and } \alpha_i \text{ fecundity-weight regression} \tag{T1.11}$$

Penalized recruitment deviations

$$QR = \sum_i^{(I-k)} \left[\ln\left(N_{i+k,k}\right) - \ln\left(f(\dot{N}_{i,j})\right)\right]^2 \qquad N_{i+k,k} \text{ number of age } k \text{ recruits} \tag{T1.12}$$

$$f(\dot{N}_{i,j}) = a\dot{B}_i \exp(-b\dot{B}_i) \qquad \text{Ricker stock-recruitment} \tag{T1.13}$$

$$\dot{B}_i = \sum_j \dot{N}_{i,j} \dot{w}_j \qquad \dot{B}_i \text{ mature biomass at spawning} \tag{T1.14}$$

3 Technical description of the proposed model changes

3.1 Input Data

The best resourcee for looking at the input data is the html file that describes the input data. There are 7 major sections to the data file.

1. Model dimensions.
2. Fecundity regression coefficients.
3. Total Annual Catch.
4. Empirical Weight-at-age (spawn and commercial).
5. Age-composition (spawn and commercial).
6. Egg deposition data.
7. Mile milt days.

These are the same data inputs that were used in the ASA model; however, there have a been a number of siginificant changes to the input data file. The most siginificant change is the addition of the log standar error for each catch, egg deposition, and spawn survey observation.

3.2 Control file

There are also siginficant changes to the control file. In fact, its a completely different control file than what was used in the ASA model. Again, the best resource for looking at the specific contents of the control file, is the control file itself.

To highlight some of the major changes, the control file now consists of a design matrix for controling the leading model parameters; specificially, the bounds and phases in which these parameters are estimated. There is a block for time-varying maturity, a block for time-varying naural mortality rate deviations. A block for selectivity, where the user can choose among alternative parametric and non-parametric selectivity curves. Lastly, there is a vector of other miscellaneous model controls for, *inter alia*, re-scaling catch, conditioning the model on catch or effort.

3.3 Age-schedule information

Age-schedule information is part date input: weight-at-age, or maturity-at-age via regression coefficients. The other part of age-schedule information is estimated based on fitting the model to data: selectivity-at-age, maturity-at-age.

3.3.1 Maturity-at-age

For the maturity-at-age, the HAM assumes that age-specific maturity follows a logistic relationship, where the estimated paraemeters define the age-at-50% maturity and the standard deviation, for each unique block period (the block periods are user defined).

The source code for the TPL file is as follows:

```
FUNCTION void initializeMaturitySchedules()
  int iyr = mod_syr;
  int jyr;
  mat.initialize();
  for(int h = 1; h <= nMatBlocks; h++) {
    dvariable mat_a = mat_params(h,1);
    dvariable mat_b = mat_params(h,2);

    jyr = h != nMatBlocks ? nMatBlockYear(h) : nMatBlockYear(h)-retro_yrs;
    // fill maturity array using logistic function
    do{
      mat(iyr++) = plogis(age,mat_a,mat_b);
    } while(iyr <= jyr);
  }
```

where `plogis` is a built in ADMB function for the logistic:

$$f(x) = (1 + \exp(-(x - \mu)/\sigma))^{-1}$$

where μ and σ are the location and scale parameters that are estimated.

3.3.2 Natural mortality

Natural mortality is both age- and time-specific. At the time of writing, there is no code that allows for age-dependent natural mortality, but this option is easily added as a feature to the HAM.

The source code for the TPL file is as follows:

```
FUNCTION void calcNaturalMortality()

  int iyr = mod_syr;
  int jyr;
  Mij.initialize();

  for(int h = 1; h <= nMortBlocks; h++){
    dvariable mi = mfexp(log_natural_mortality + log_m_devs(h));

    jyr = h != nMortBlocks ? nMortBlockYear(h) : nMortBlockYear(h)-retro_yrs;
    // fill mortality array by block
```

```
    do{
      Mij(iyr++) = mi;
    } while(iyr <= jyr);
  }
  //COUT(Mij)
```

where the Matrix $M_{i,j}$ is the instananeous natural mortality rate for year i and age j. At this point the code just fills each row of the matrix with the same annual natural mortality rate (i.e., age-independent). This is where you would want to modify the code to allow for age-dependent natural mortality rates.

3.3.3 Selectivity

Currently only the logistic selectivity option is implemented. But the source code is structured such that alternative parametric and non-parametric functions can be easily added to the source code using a swtich statement.

The source code for the TPL file is as follows:

```
FUNCTION void calcSelectivity()
  /**
    - Loop over each of the selectivity block/pattern
    - Determine which selectivity type is being used.
    - get parameters from log_slx_pars
    - calculate the age-dependent selectivity pattern
    - fill selectivty array for that block.
    - selectivity is scaled to have a mean = 1 across all ages.
  */
  dvariable p1,p2;
  dvar_vector slx(sage,nage);
  log_slx.initialize();



  for(int h = 1; h <= nSlxBlks; h++){

    switch(nSelType(h)){
      case 1: //logistic
        p1  = mfexp(log_slx_pars(h,1,1));
        p2  = mfexp(log_slx_pars(h,1,2));
        slx = plogis(age,p1,p2) + TINY;
      break;
    }

    int jyr = h != nSlxBlks ? nslx_nyr(h):nslx_nyr(h)-retro_yrs;
```

```
    for(int i = nslx_syr(h); i <= jyr; i++){
      log_slx(i) = log(slx) - log(mean(slx));
    }
  }
  Sij.sub(mod_syr,mod_nyr) = mfexp(log_slx);
```

The matrix $S_{i,j}$ is the relative selectivity for age j in year i. Additional functions for computing the vector `slx` can be new cases (e.g. case 2: // coefficients).

There is a very important feature in the selectivity is parameterized in this model. The reason for this particular parameterization is to ensure the objective function remains continuous and differentiable. In each year, the vector of age-specific selectivity coefficients is scaled to have a mean value of 1.0. This is accomplished, in log-space, but subtracting the mean from the vector of age-specific selectivities. This avoids having to use the max function; the use of the max function can lead to a discontinuity in the objective function which result in non-convergence. The tradeoff for this numerical stability is that the interpretation of Fishing mortality rates changes. The estimator is calculating the average-age-specific fishing mortality rate. The fully-selected fishing mortality rate is more commonly used metric, and this is easily accommodated post-estimation by re-scaling the vector of fishing mortality rates by the maximum age-specific selectivity in each year.

If the previous paragraph didn't make sense, go read the R-code.

3.4 Fishing mortality

If the model is conditioned on effort, then a routine that calculates the age-specific fishing mortality rate is invoked. In this routine, a vector of fishing mortality rate parametes, in log-space, is estimated, and the age-specific fishing mortality rate is the product of the fishing rate and age-specific selectivity. The source code for this routine is as follows:

```
FUNCTION void calcFishingMortalitiy()
  /**
    - Calculate Fishing mortality, and then Zij = Mij + Fij
    */

  for(int i = mod_syr; i <= mod_nyr; i++) {
    Fij(i) = exp(log_ft_pars(i)) * Sij(i);
  }
```

If the model is conditioned on catch (NB this is the method the ASA model uses), then there is a resulting difference equation which has the potential to result in negative numbers-at-age, which results in negative Infinity in log-space. To guard against this, many excel users would simply use a Max function to ensure that a positive number is returned. This is another occurance where the objective function is disconituous and subject to non-convergence issues. AD Model Builder has a function `posfun` that can be used to ensure the objective function remains continuous and differentiable.

3.5 Population dynamics

Estimated parameters for the population dynamics model include the initial abundance of ages 3-9+ for the intial year, abundance of age-3 recruits each year, and the natural mortality rate. In the original parameterizeation of the model, these initial recruitments and the vector of initial numbers-at-age result in creating $(N + A - 1)$ scaling parameters. To reduce the potential confounding with other global scaling parmaters, updates to the model code include estimation of two recruitment scaling parameters, and two vectors of deviates that represent deviations from the mean. This modification reduces the potential for parameter confounding among the many paraemters that affect global scaling (i.e., catchabiltiy coefficients, natural mortlaity rates).

Table 2: Notation and equations for population dynamics model.

Model parameters			
$\theta = \{\ln(M), \ln(\bar{R}), \ln(\ddot{R}), \ln(\alpha), \ln(\beta), \vec{\delta}\}$			(T2.1)
Initial States ($i = 1980$)			
$\iota_j = \begin{cases} \exp(-M * (j - \min(j))) & \text{where } 3 \leq j \leq 7 \\ \dfrac{\exp(-M * (j - \min(j)))}{1 - \exp(-M)} & \text{where } j = 8 \end{cases}$		survivorship	(T2.2)
$N_{i,j} = \ddot{R} \exp(\delta_{i-j}) \iota_j$	$i = 1980, \forall j$	initial numbers-at-age	(T2.3)
$N_{i,j} = \bar{R} \exp(\delta_i)$	$\forall i, j = 3$	age-3 recruits	(T2.4)
Dynamic States ($i > 1980$)			
$Q_{i,j} = \dfrac{N_{i,j} S_{i,j}}{\sum_j N_{i,j} S_{i,j}}$		vulnerable proportions	(T2.5)
$\bar{w}_i = \sum_j w_j Q_{i,j}$		average weight of catch	(T2.6)
$C_{i,j} = \dfrac{\hat{c}_i Q_{i,j}}{\bar{w}_i}$	where $\hat{c}_i$ is the observed catch (mt)	$C_{i,j}$ catch-at-age	(T2.7)
$\acute{N}_{i,j} = N_{i,j} \exp(-F_{i,j})$			(T2.8)
$N_{i+1,j+1} = \acute{N}_{i,j} \exp(-M_{i,j})$			(T2.9)
Spawning stock biomass			
$B_i = \sum_j (N_{ij} - C_{ij}) \omega_{ij} w_j$		Spawning stock biomass	(T2.10)
Stock-recruitment			

3.5.1 Initial state variables.

In this routine, the objective is to set the initial values for the numbers-at-age matrix $N_{i,j}$. Specifically, the row dimensions of the matrix are from the start year to the terminal year + 1, the column dimensions are the ages. This routine first calculates the survivorship to age j based on natural mortality rates, then initializes the first row of $N_{i,j}$ matrix using the average initial recruitment and deviations around that average recruitment multiplied by the survivorship at age j. The source code also includes the taylor series for the plus group:

```
FUNCTION void initializeStateVariables()
  /**
    - Set initial values for numbers-at-age matrix in first year
      and sage recruits for all years.
    */
  Nij.initialize();

  // initialize first row of numbers-at-age matrix
  // lx is a vector of survivorship (probability of surviving to age j)
  dvar_vector lx(sage,nage);

  for(int j = sage; j <= nage; j++){

    lx(j) = exp(-Mij(mod_syr,j)*(j-sage));
    if( j==nage ) lx(j) /= (1.0-exp(-Mij(mod_syr,j)));


    if( j > sage ){
      Nij(mod_syr)(j) = mfexp(log_rinit + log_rinit_devs(j)) * lx(j);
    }
  }



  // iniitialize first column of numbers-at-age matrix
  for(int i = mod_syr; i <= mod_nyr + 1; i++){
    Nij(i,sage) = mfexp(log_rbar + log_rbar_devs(i));
  }
  //COUT(lx);
  //COUT(Nij);
```

The survivorship calculation `lx` correspondds to (T2.2) in Table 2, and the numbers-at-age in the initial year and age-3 recruits corresponds to (T2.3) and (T2.4), respectively.

3.5.2 Update state variables

In this routine, the numbers-at-age are propagated in time, where the-age specific survival rate is partitioned into two periods: a fishing period, and a period of natural mortality. The ASA model currently in use for Sitka sound herring assumes a pulse fishery. At the start of each time step, the model first calculates the predicted catch-at-age in numbers in year i. This is done by first converting the catch weight to catch in numbers by dividing by the predicted average of the catch. This corresponds to the `wbar` variable in the following code chunk (note the dependencey on predicted proportions-at-age):

```
FUNCTION void updateStateVariables()
  /**
  - Update the numbers-at-age conditional on the catch-at-age.
  - Assume a pulse fishery.
  - step 1 calculate a vector of vulnerable-numbers-at-age
  - step 2 calculate vulnerable proportions-at-age.
  - step 3 calc average weight of catch (wbar) conditional on Qij.
  - step 4 calc catch-at-age | catch in biomass Cij = Ct/wbar * Qij.
  - step 5 condition on Ft or else condition on observed catch.
  - step 6 update numbers-at-age (using a very dangerous difference eqn.)
  */

  Qij.initialize();
  Cij.initialize();
  Pij.initialize();
  dvariable wbar;   // average weight of the catch.
  dvar_vector vj(sage,nage);
  //dvar_vector pj(sage,nage);
  dvar_vector sj(sage,nage);


  for(int i = mod_syr; i <= mod_nyr; i++){


    // step 1.
    vj = elem_prod(Nij(i),Sij(i));

    // step 2.
    Qij(i) = vj / sum(vj);

    // ADF&G's approach.
    if( !dMiscCont(2) ) {
      // step 3.
      dvector wa = data_cm_waa(i)(sage,nage);
```

```
      wbar = wa * Qij(i);

      // step 4.
      Cij(i) = data_catch(i,2) / wbar * Qij(i);
      // should use posfun here
      Pij(i) = posfun(Nij(i) - Cij(i),0.01,fpen);

      // step 6. update numbers at age
      sj = mfexp(-Mij(i));
      Nij(i+1)(sage+1,nage) =+ elem_prod(Pij(i)(sage,nage-1),
                                         sj(sage,nage-1));
      Nij(i+1)(nage) += Pij(i,nage) * sj(nage);
    }
    // step 5.
    // Condition on Ft
    else {
      // add flexibility here for Popes approx, or different seasons
      Pij(i) = elem_prod( Nij(i), exp(-Fij(i)) );
      Cij(i) = elem_prod( Nij(i), 1.-exp(-Fij(i)));

      // the following assumes fishery is at the start of the year.
      dvar_vector zi = Mij(i) + Fij(i);
      Nij(i+1)(sage+1,nage) = ++ elem_prod(Nij(i)(sage,nage-1),
                                           mfexp(-zi(sage,nage-1)));
      Nij(i+1)(nage) += Nij(i,nage) * mfexp(-zi(nage));
    }
  }

  // cross check... Looks good.
  // COUT(Cij(mod_syr) * data_cm_waa(mod_syr)(sage,nage));
```

Once the catch-at-age data is calculated, the pulse fishery proceeds by subtracting the C_{ij} from the $N_{i,j}$. The last two steps correspond to step 4 in the annotated code chunk. The last steps the survive the remaining numbers-at-age using the natural mortality rate in year i. Then finally, update the numbers-at-age j to $j+1$ in year i to year $i+1$, including the plus group for the terminal age-group.

3.5.3 Stock-recruitment & Spawning stock biomass

The spawning stock biomass (after roe-fishery removal) is the product of the remaining numbers-at-age, the maturity-at-age, and the weight-at-age from spawn samples. The equation is defined in (T2.10) in Table 2. Note that the lag between spawning biomass and age-3 recruits is taken into account.

The stock recruitment relationship assumes that recruitment follows a Ricker type. The form of the Ricker model is as follows

$$R = s_o B_i \exp(-\beta B_i + \epsilon_i)$$

where the parameter s_o is the slope at the origin (or maximum number of recruits per spawning unit), and β defines slope of $\ln(R_i/B_i)$ versus the independent variable B_i. These two parameters are derived from the leading parameters the unfished recruitment R_0 and the steepness of the stock recruitment relationship as defined by (Mace and Doonan, 1988; **?**).

The source code for the stock-recruitment relationship is well annotated and describes some of the derivations:

```
FUNCTION void calcSpawningStockRecruitment()
  /**
  The functional form of the stock recruitment model follows that of a
  Ricker model, where R = so * SSB * exp(-beta * SSB).  The two parameters
  so and beta where previously estimated as free parameters in the old
  herring model.  Herein this fucntion I derive so and beta from the
  leading parameters Ro and reck; Ro is the unfished sage recruits, and reck
  is the recruitment compensation parameter, or the relative improvement in
  juvenile survival rates as the spawning stock SSB tends to 0. Simply a
  multiple of the replacement line Ro/Bo.

  At issue here is time varying maturity and time-varying natural mortality.
  When either of these two variables are assumed to change over time, then
  the underlying stock recruitment relationship will also change. This
  results in a non-stationary distribution.  For the purposes of this
  assessment model, I use the average mortality and maturity schedules to
  derive the spawning boimass per recruit, which is ultimately used in
  deriving the parameters for the stock recruitment relationship.
    */

  /*
    Spoke to Sherri about this. Agreed to change the equation to prevent
  */
  for(int i = mod_syr; i <= mod_nyr; i++){
    //Oij(i) = elem_prod(mat(i),Nij(i));
    //ssb(i) = (Oij(i) - Cij(i)) * data_sp_waa(i)(sage,nage);

    Oij(i) = elem_prod(mat(i),Nij(i)-Cij(i));
    ssb(i) = Oij(i) * data_sp_waa(i)(sage,nage);
  }


  // average natural mortality
```

```
dvar_vector mbar(sage,nage);
int n = Mij.rowmax() - Mij.rowmin() + 1;
mbar  = colsum(Mij)/n;

// average maturity
dvar_vector mat_bar(sage,nage);
mat_bar = colsum(mat)/n;


// unfished spawning biomass per recruit
dvar_vector lx(sage,nage);
lx(sage) = 1.0;
for(int j = sage + 1; j <= nage; j++){
  lx(j) = lx(j-1) * mfexp(-mbar(j-1));
  if(j == nage){
    lx(j) /= 1.0 - mfexp(-mbar(j));
  }
}
dvariable phie = lx * elem_prod(avg_sp_waa,mat_bar);


// Ricker stock-recruitment function
// so = reck/phiE; where reck > 1.0
// beta = log(reck)/(ro * phiE)
dvariable ro   = mfexp(log_ro);
dvariable reck = mfexp(log_reck);
dvariable so   = reck/phie;
dvariable beta = log(reck) / (ro * phie);

spawners = ssb(mod_syr,mod_nyr-sage+1).shift(rec_syr);
recruits = elem_prod( so*spawners , mfexp(-beta*spawners) );
resd_rec = log(column(Nij,sage)(rec_syr,mod_nyr+1)+TINY)
           - log(recruits+TINY);
```

There is an issue with regards to calculating reference points in cases where there is non-stationarity (i.e., any time varying parameters such as natural mortality, maturity etc.). At what period should be used to define the average weight-at-age for spawning herring? What period should be used for calculating the average maturity? All of these are subjective decisions, and based on my experience will have little impact on the overall fit to the data, but could have major implications for harvest policy changes.

3.6 Observation models

3.6.1 Age composition

The predicted age-composition is based on the vulnerable proportions at age $Q_{i,j}$ in (T2.5). The residual difference is used to compute the negative log likelihood in the objective function. The code for the age composition residual calculation is as follows:

```
FUNCTION void calcAgeCompResiduals()
  /**
  - Commercial catch-age comp residuals
  - Spawning survey catch-age comp residuals.
  */

  resd_cm_comp.initialize();
  resd_sp_comp.initialize();
  for(int i = mod_syr; i <= mod_nyr; i++){

    // commercial age-comp prediction
    pred_cm_comp(i) = Qij(i);
    if( data_cm_comp(i,sage) >= 0 ){
      resd_cm_comp(i) = data_cm_comp(i)(sage,nage) - pred_cm_comp(i);
    }

    // spawning age-comp prediction
    pred_sp_comp(i) = (Oij(i)+TINY) / sum(Oij(i)+TINY);
    if( data_sp_comp(i,sage) >= 0 ){
      resd_sp_comp(i) = data_sp_comp(i)(sage,nage) - pred_sp_comp(i);
    }
  }

  //COUT(resd_sp_comp);
```

Note that both the residual difference between the commercial and spawning samples are calculated in this routine. If there are missing data for a given year (denoted by a -9.0 in the data file), then the residual difference is set to 0 for that year and there is no contribution to the negative log likelihood.

3.6.2 Egg deposition

The observation model for the egg deposition data treats these observations as estimates of absolute abundance. Therefore, there is no associated scaling parameter that is estimated. The observation errors in the egg deposition data are assumed to be lognormal. The predicted age-deposition data is based on the female (assuming a 50:50 sex ratio) mature numbers-at-age multiplied by the fecundity at age. The annotated source code is as follows:

```
FUNCTION void calcEggSurveyResiduals()
  /**
  - Observed egg data is in trillions of eggs
  - Predicted eggs is the mature female numbers-at-age multiplied
    by the fecundity-at-age, which comes from a regession of
    fecundity = slope * obs_sp_waa - intercept
  - Note Eij is the Fecundity-at-age j in year i.
  -
  */
  resd_egg_dep.initialize();
  for(int i = mod_syr; i <= mod_nyr; i++){
    pred_egg_dep(i) = (0.5 * Oij(i)) * Eij(i);
    if(data_egg_dep(i,2) > 0){
      resd_egg_dep(i) = log(data_egg_dep(i,2)) - log(pred_egg_dep(i));
    }
  }
```

3.6.3 Aerial surveys

Indicies from aerial surveys, or mile milt days, are treated as relative abundance data. The underlying assumption in this observation model is that the observation errors are log normal, and that the index is proportional to the spawning stock biomass. Note that the code does not estimate the coefficient, rather the conditional maximum likelihood estimate of the scaling coefficent is used (see Walters and Ludwig, 1994, for a full explanation). The annotated code follows:

```
FUNCTION void calcMiledaySurveyResiduals()
  /**
  - Assumed index from aerial survey is a relative abundance
  index. The slope of the regression ln(SSB) = q * ln(MileMilt) + 0
  is computed on the fly in case of missing data.
  - See Walters and Ludwig 1994 for more details.
  */
  resd_mileday.initialize();
  pred_mileday.initialize();
  int n = 1;
  dvar_vector zt(mod_syr,mod_nyr); zt.initialize();
  dvariable zbar = 0;
  for(int i = mod_syr; i <= mod_nyr; i++){
    if(data_mileday(i,2) > 0){
      zt(i) = log(data_mileday(i,2)) - log(ssb(i));
      zbar  = zt(i)/n + zbar *(n-1)/n;
      n++ ;
```

```
    }
  }
  pred_mileday = ssb * exp(zbar);
  resd_mileday = zt - zbar;
  // COUT(resd_mileday);
```

3.6.4 Predicted catch

In the case where the model is conditioned on effort and fitted to the catch time serie data, the predicted catch is the sum of products betwen the catch-at-age in numbers and the observed weight-at-age in the commercial fishery. We further assume observation errors are lognormal. The source code follows:

```
FUNCTION void calcCatchResiduals()
  /**
  - Catch residuals assuming a lognormal error structure.
  */
  pred_catch.initialize();
  resd_catch.initialize();
  for(int i = mod_syr; i <= mod_nyr; i++) {
    if(data_catch(i,2) > 0) {
      pred_catch(i) = Cij(i) * data_cm_waa(i)(sage,nage);
      resd_catch(i) = log(data_catch(i,2)) - log(pred_catch(i));
    }
  }
```

3.7 Objective function

The objective function is organized into two sections, the first contains the negative loglikelihoods for the data given the model parameters. The second are a series of penalties, in the case of maximum likelihood estimation, prior density functions in the case of a Bayesian estimation.

3.7.1 Negative loglikelihoods

The negative loglikelihoods There are five 6 negative loglikelihoods functions in the objective function that correspond to the 5 different data elements and the residual process errors associated with a stock-recruitment relationship. Table 3 summarises the available options currently implemented.

The source code for the negative loglikelihoods follows:

```
  // 1. Negative loglikelihoods
  dvar_vector nll(1,7);
  nll.initialize();
```

Table 3: Data and types of likelihoods implemented

Data	normal	log-normal	multivariate-logistic	multinomial
Commercial Age-comps			X	
Spawn Survey Age-comps			X	
Egg deposition		X		
Aerial survey		X		
Catch		X		
Stock-Recruitment		X		

```
// Mulitvariate logistic likelihood for composition data.
double sp_tau2;
double minp = 0.00;
dmatrix d_sp_comp = trans(trans(data_sp_comp).sub(sage,nage)).sub(mod_syr,n
nll(1) = dmvlogistic(d_sp_comp,pred_sp_comp,resd_sp_comp,sp_tau2,minp);

// Mulitvariate logistic likelihood for composition data.
double cm_tau2;
dmatrix d_cm_comp  = trans(trans(data_cm_comp).sub(sage,nage)).sub(mod_syr,
nll(2) = dmvlogistic(d_cm_comp,pred_cm_comp,resd_cm_comp,cm_tau2,minp);

// Negative loglikelihood for egg deposition data
dvector std_egg_dep = TINY + column(data_egg_dep,3)(mod_syr,mod_nyr);
nll(3) = dnorm(resd_egg_dep,std_egg_dep);

// Negative loglikelihood for milt mile day
dvector std_mileday = TINY + column(data_mileday,3)(mod_syr,mod_nyr);
nll(4) = dnorm(resd_mileday,std_mileday);

// Negative loglikelihood for stock-recruitment data
dvector std_rec(rec_syr,mod_nyr+1);
std_rec = 0.6;
nll(5) = dnorm(resd_rec,std_rec);

// Negative loglikelihood for catch data
dvector std_catch = column(data_catch,3)(mod_syr,mod_nyr);
nll(6) = dnorm(resd_catch,std_catch);
```

For the composition data, the multivariate logistic likelihood is currently implemented, as this is a self-scaling likelihood. A good reference for this particular likelihood and how its implemented in AD Model Builder can be found in Schnute and Richards (1995). More recent work on the weighting of composition data is available in Francis (2011). The function

`dmvlogistic` requires 5 arguments: the observed and predicted composition matrixes, a matrix for returning the residuals, a variable for the conditional MLE of the variance of the observation errors, and a threshold value called `minp`. The multivariate logistic likelihood does not accommodate 0 observations for age proportions. Therefore there are two options for dealing with 0s: 1) add a small constant to all observed and predicted observations and re-normalize, or 2) pool the adjacent cohorts if the observed proportion is less than some minimum observed proportion. The first option is widely used in programs such as stock synthesis, and is akin to manufacturing data. If a particular cohort is relatively weak, and only partially selected, the sample size required to observed just one individual of a certain age in a given year could be infinitely large. Instead of adding a constant, option 2 pools the data such that the likelihood of ages, for example, 3-4 are evaluated jointly, rather than the likelihood of age-3 plus the likelihood of age-4. This pooling of the data and predictions does not require the addition of a constant that could potentially bias the result. A good practice that I've found is to just set `minp=0`, and then conduct sensitivity tests using where proportions less than 1% or 2% etc are pooled into the adjacent cohort.

In the case of the likelihoods for the egg deposition index, aerial surveys, and catch data, the function `dnorm` is used, where the arguments are the vector of residuals, and a vector of standard-deviations for each observation. Note that you can effectively turn individual years of data off by setting the `log.se` value to a large number (e.g., 5.0).

3.7.2 Penalties

Currently the code for the penalties is as follows:

```
// 2. Penalized logliklelihoods


dvar_vector penll(1,4);
penll.initialize();

// | Terminal phase of estimation.
dvariable log_fbar = mean(log_ft_pars);
if( last_phase() ){
  penll(1) = dnorm(log_rinit_devs,0.0,5.0);
  penll(2) = dnorm(log_rbar_devs,0.0,5.0);
  penll(3) = dnorm(log_fbar,0.2,2.0);
} else {
  penll(1) = dnorm(log_rinit_devs,0.0,1.0);
  penll(2) = dnorm(log_rbar_devs,0.0,1.0);
  penll(3) = dnorm(log_fbar,0.2,0.07);
}
```

The penalties are implemented in a phased approach, where in the initial phases of parameter estimation, the penalties initially have small standard deviations. This increases

the overall efficiency of the non-linear search routine to help resolve the overall scaling. Then in the terminal phase, these penalties are relaxed (increased variance) such that they provide little or no influence on the gradient structure. A similar strategy is also used with the recruitment deviation parameters.

References

Francis, R. (2011). Data weighting in statistical fisheries stock assessment models. *Canadian Journal of Fisheries and Aquatic Sciences*, 68(6):1124–1138.

Mace, P. M. and Doonan, I. (1988). *A generalised bioeconomic simulation model for fish population dynamics*. MAFFish, NZ Ministry of Agriculture and Fisheries.

Schnute, J. and Richards, L. (1995). The influence of error on population estimates from catch-age models. *Canadian Journal of Fisheries and Aquatic Sciences*, 52(10):2063–2077.

Walters, C. and Ludwig, D. (1994). Calculation of Bayes posterior probability distributions for key population parameters. *Canadian Journal of Fisheries and Aquatic Sciences*, 51(3):713–722.

Table 4: Mathematical notation, symbols and descriptions.

Symbol	Description
Index	
g	group
h	sex
i	year
j	time step (years)
k	gear or fleet
l	index for length class
m	index for maturity state
o	index for shell condition.
Leading Model Parameters	
M	Instantaneous natural mortality rate
$\bar{R}$	Average recruitment
$\ddot{R}$	Initial recruitment
α_r	Mode of size-at-recruitment
β_r	Shape parameter for size-at-recruitment
R_0	Unfished average recruitment
κ	Recruitment compensation ratio
Size schedule information	
$w_{h,l}$	Mean weight-at-length l
$m_{h,l}$	Average proportion mature-at-length l
Per recruit incidence functions	
ϕ_B	Spawning biomass per recruit
ϕ_{Q_k}	Yield per recruit for fishery k
ϕ_{Y_k}	Retained catch per recruit for fishery k
ϕ_{D_k}	Discarded catch per recruit for fishery k
Selectivity parameters	
$a_{h,k,l}$	Length at 50% selectivity in length interval l
$\sigma_{s_{h,k}}$	Standard deviation in length-at-selectivity
$r_{h,k,l}$	Length at 50% retention
$\sigma_{y_{h,k}}$	Standard deviation in length-at-retention
$\xi_{h,k}$	Discard mortality rate for gear k and sex h