```
calculate_biomass_and_predicted_catch();
// calculate the objective function
calculate_the_objective_function();
```

There are three user-defined functions called at the beginning of the `PROCEDURE_SECTION` The code to define the `FUNCTIONS` comes next. To define a function whose name is `name` the template directive `FUNCTION name` is used. Notice that no parentheses `()` are used in the definition of the function, but to call the function the statement takes the form `name();`

1.20 A fisheries catch-at-age model

This section describes a simple catch-at age model. The data input to this model include estimates of the numbers at age caught by the fishery each year and estimates the fishing effort each year. This example introduces AD Model Builder's ability to automatically calculate profile likelihoods for carrying out Bayesian inference. To cause the profile likelihood calculations to be carried out use the -lprof command line argument.

Let i index fishing years $1 \le i \le n$ and j index age classes with $1 \le j \le r$. The instantaneous fishing mortality rate is assumed to have the form $F_{ij} = qE_i s_j \exp(\delta_i)$ where q is called the catchability, E_i is the observed fishing effort, s_j is an age-dependent effect termed the selectivity, and the δ_i are deviations from the expected relationship between the observed fishing effort and the resulting fishing mortality. The δ_i are assumed to be normally distributed with mean 0. The instantaneous natural mortality rate M is assumed to be independent of year and age class. It is not estimated in this version of the model. The instantaneous total mortality rate is given by $Z_{ij} = F_{ij} + M$. The survival rate is given by $S_{ij} = \exp(-Z_{ij})$. The number of age class j fish in the population in year i is denoted by N_{ij}. The relationship $N_{i+1,j+1} = N_{ij}S_{ij}$ is assumed to hold. Note that using this relationship if one knows S_{ij} then all the N_{ij} can be calculated from knowledge of the initial population in year 1, $N_{11}, N_{12}, \ldots, N_{1r}$ and knowledge of the recruitment in each year $N_{21}, N_{31}, \ldots N_{n1}$.

The purpose of the model is to estimate quantities of interest to managers such as the population size and exploitation rates and to make projections about the population. In particular we can get an estimate of the numbers of fish in the population in year $n+1$ for age classes 2 or greater from the relationship $N_{n+1,j+1} = N_{nj}S_{nj}$. If we have estimates m_j for the mean weight at age j, then the projected biomass level B_{n+1} of age class 2+ fish for year $n+1$ can be computed from the relationship $B_{n+1} = \sum_{j=2}^{r} m_j N_{n+1,j}$.

Besides getting a point estimate for quantities of interest like B_{n+1} we also want to get an idea of how well determined the estimate is. AD Model Builder has completely automated the process of deriving good confidence limits for these parameters in a Bayesian context. One simply needs to declare the parameter to be of type `likeprof_number`. The results are given in the section on Bayesian inference.

The code for the catch-at-age model is:

```
DATA_SECTION
  // the number of years of data
  init_int nyrs
  // the number of age classes in the population
  init_int nages
  // the catch-at-age data
  init_matrix obs_catch_at_age(1,nyrs,1,nages)
  //estimates of fishing effort
  init_vector effort(1,nyrs)
  // natural mortality rate
  init_number M
  // need to have relative weight at age to calculate biomass of 2+
  vector relwt(2,nages)
INITIALIZATION_SECTION
  log_q -1
  log_P 5
PARAMETER_SECTION
  init_number log_q(1)   // log of the catchability
  init_number log_P(1)  // overall population scaling parameter
  init_bounded_dev_vector log_sel_coff(1,nages-1,-15.,15.,2)
  init_bounded_dev_vector log_relpop(1,nyrs+nages-1,-15.,15.,2)
  init_bounded_dev_vector effort_devs(1,nyrs,-5.,5.,3)
  vector log_sel(1,nages)
  vector log_initpop(1,nyrs+nages-1);
  matrix F(1,nyrs,1,nages)   // the instantaneous fishing mortality
  matrix Z(1,nyrs,1,nages)   // the instantaneous total mortality
  matrix S(1,nyrs,1,nages)   // the survival rate
  matrix N(1,nyrs,1,nages)   // the predicted numbers at age
  matrix C(1,nyrs,1,nages)   // the predicted catch at age
  objective_function_value f
  sdreport_number avg_F
  sdreport_vector predicted_N(2,nages)
  sdreport_vector ratio_N(2,nages)
  likeprof_number pred_B
PRELIMINARY_CALCS_SECTION
  // this is just to invent some relative average
  // weight numbers
  relwt.fill_seqadd(1.,1.);
  relwt=pow(relwt,.5);
  relwt/=max(relwt);
PROCEDURE_SECTION
  // example of using FUNCTION to structure the procedure section
  get_mortality_and_survival_rates();

  get_numbers_at_age();

  get_catch_at_age();

  evaluate_the_objective_function();

FUNCTION get_mortality_and_survival_rates
  // calculate the selectivity from the sel_coffs
  for (int j=1;j<nages;j++)
  {
    log_sel(j)=log_sel_coff(j);
  }
  // the selectivity is the same for the last two age classes
  log_sel(nages)=log_sel_coff(nages-1);
```

```
  // This is the same as F(i,j)=exp(log_q)*effort(i)*exp(log_sel(j));
  F=outer_prod(mfexp(log_q)*effort,mfexp(log_sel));
  if (active(effort_devs))
  {
    for (int i=1;i<=nyrs;i++)
    {
      F(i)=F(i)*exp(effort_devs(i));
    }
  }
  // get the total mortality
  Z=F+M;
  // get the survival rate
  S=mfexp(-1.0*Z);

FUNCTION get_numbers_at_age
  log_initpop=log_relpop+log_P;
  for (int i=1;i<=nyrs;i++)
  {
    N(i,1)=mfexp(log_initpop(i));
  }
  for (int j=2;j<=nages;j++)
  {
    N(1,j)=mfexp(log_initpop(nyrs+j-1));
  }
  for (i=1;i<nyrs;i++)
  {
    for (j=1;j<nages;j++)
    {
      N(i+1,j+1)=N(i,j)*S(i,j);
    }
  }
  // calculated predicted numbers at age for next year
  for (j=1;j<nages;j++)
  {
    predicted_N(j+1)=N(nyrs,j)*S(nyrs,j);
    ratio_N(j+1)=predicted_N(j+1)/N(1,j+1);
  }
  // calculate predicted biomass for profile
  // likelihood report
  pred_B=predicted_N *relwt;

FUNCTION get_catch_at_age
  C=elem_prod(elem_div(F,Z),elem_prod(1.-S,N));

FUNCTION evaluate_the_objective_function
  // penalty functions to ``regularize '' the solution
  f+=.01*norm2(log_relpop);
  avg_F=sum(F)/double(size_count(F));
  if (last_phase())
  {
    // a very small penalty on the average fishing mortality
    f+= .001*square(log(avg_F/.2));
  }
  else
  {
    // use a large penalty during the initial phases to keep the
    // fishing mortality high
    f+= 1000.*square(log(avg_F/.2));
```

```
  }
  // errors in variables type objective function with errors in
  // the catch at age and errors in the effort fishing mortality
  // relationship
  if (active(effort_devs)
  {
    // only include the effort_devs in the objective function if
    // they are active parameters
    f+=0.5*double(size_count(C)+size_count(effort_devs))
      * log( sum(elem_div(square(C-obs_catch_at_age),.01+C))
      + 0.1*norm2(effort_devs));
  }
  else
  {
    // objective function without the effort_devs
    f+=0.5*double(size_count(C))
      * log( sum(elem_div(square(C-obs_catch_at_age),.01+C)));
  }
REPORT_SECTION
  report << "Estimated numbers of fish " << endl;
  report << N << endl;
  report << "Estimated numbers in catch " << endl;
  report << C << endl;
  report << "Observed numbers in catch " << endl;
  report << obs_catch_at_age << endl;
  report << "Estimated fishing mortality " << endl;
  report << F << endl;
```

This model employs several instances of the `init_bounded_dev_vector` type. This type consists of a vector of numbers which sum to 0, that is they are deviations from a common mean, and are bounded. For example the quantities `log_relpop` are used to parameterize the logarithm of the variations in year class strength of the fish population. Putting bounds on the magnitude of the deviations helps to improve the stability of the model. The bounds are from -15.0 to 15.0 which gives the estimates of relative year class strength a dynamic range of exp(30.0).

The `FUNCTION` keyword has been employed a number of times in the `PARAMETER_SECTION` to help structure the code. A function is defined simply by using the `FUNCTION` keyword followed by the name of the function.

```
FUNCTION get_mortality_and_survival_rates
```

Don't include the parentheses or semicolon here. To use the function simply write its name in the procedure section.

```
get_mortality_and_survival_rates();
```

You must include the parentheses and the semicolon here.

The `REPORT_SECTION` shows how to generate a report for an AD Model Builder program. The default report generating machinery utilizes the C++ stream formalism. You don't need to know much about streams to make a report, but a few comments are in order. The stream formalism associates stream object with a file. In this case the stream object associated with

the AD Model Builder report file is `report`. To write an object `xxx` into the report file you insert the line

```
report << xxx;
```

into the `REPORT_SECTION`. If you want to skip to a new line after writing the object you can include the stream manipulator `endl` as in

```
report << "Estimated numbers of fish " << endl;
```

Notice that the stream operations know about common C objects such as strings, so that it is a simple matter to put comments or labels into the report file.

1.21 Bayesian inference and the profile likelihood

AD Model Builder enables one to quickly build models with large numbers of parameters – this is especially useful for employing Bayesian analysis. Traditionally however it has been difficult to interpret the results of analysis using such models. In a Bayesian context the results are represented by the posterior probability distribution for the model parameters. To get exact results from the posterior distribution it is necessary to evaluate integrals over large dimensional spaces and this can be computationally intractable. AD Model Builder provides an approximations to these integrals in the form of the profile likelihood. The profile likelihood can be used to estimates for extreme values (such as estimating a value β so that for a parameter b the probability that $b < \beta \approx 0.10$ or the probability that $b > \beta \approx 0.10$) for any model parameter. To use this facility simply declare the parameter of interest to be of type `likeprof_number` in the `PARAMETER_SECTION` and assign the correct value to the parameter in the `PROCEDURE_SECTION`.

The code for the catch at age model estimates the profile likelihood for the projected biomass of age class 2+ fish. (Age class 2+ has been used to avoid the extra problem of dealing with the uncertainty of the recruitment of age class 1 fish). As a typical application of the method, the user of the model can estimate the probability that the biomass of fish for next year will be larger or smaller than a certain value. Estimates like these are obviously of great interest to managers of natural resources.

The profile likelihood report for a variable is in a file with the same name as the variable (truncated to eight letters, if necessary, with the suffix `.PLT` appended). For this example the report is in the file `PRED_B.PLT`. Part of the file is shown here.

```
pred_B:
Profile likelihood
 -1411.23 1.1604e-09
 -1250.5 1.71005e-09
 -1154.06 2.22411e-09
  ...................      // skip some here
  ...................
 278.258 2.79633e-05
```