Code for AMAK

AMAK ADMB Code

```
///////////////////////////////////////////////////////////////////
// Generic model
// Alaska Fisheries Toolbox Assessment Model
// Version 0.1, May 2003;
//
// Naming Conventions:
//
//  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 surveys
//    nsrv        number of surveys
//  Index values
//    nyrs_srv      Number of years of survey index value (annual)
//    yrs_srv        Years of survey index value (annual)
//    obs_srv        Observed survey index value (annual)
//    obs_se_srv    Observed survey standard errors (annual)
//  Age-comp values
//    nyrs_srv_age  Number of years survey age data available
//    yrs_srv_age   Years of survey age value (annual)
//    oac_srv       Observed age comp from surveys
//    nsmpls_srv    Observed age comp sample sizes from surveys
//
//    eac_srv       Expected age comp from surveys
//
//    sel_srv       selectivity for egg production index
//
//    pred_srv ...
//
//    oac_fsh      Observed age comp from survey
//    obs_srv_size  Observed size comp from survey
//
//    pred_fsh_age    Predicted age comp from fishery data
//    eac_fsh            Expected age comp for fishery data (only years where data available)
//    eac_ ...
//
//    pred_tsrv   Predicted index value for trawl survey
//
//    sel_fsh    selectivity for fishery
//
//////////////////////////////////////////////////////////////////////////////
 // To ADD/FIX:
 //   extend to annual wt and maturity schedules
 //   parameterization of steepness to work the same (wrt prior) for ricker and bholt
```

DATA_SECTION

```
  !!CLASS ofstream mceval("mceval.dat")

  int mcmcmode
  int mcflag

  !! mcmcmode = 0;
  !! mcflag   = 1;
 LOCAL_CALCS
  tmpstring=adprogram_name + adstring(".dat");
  if (argc > 1)
  {
    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]);
      }
    }
    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;
  // !! cntrlfile_name = *(ad_comm::global_datafile) ;
  // !! cout <<cntrlfile_name(1,length(cntrlfile_name)-4)<<endl;exit(0);
  !! ad_comm::change_datafile_name(datafile_name);

  init_int styr
  // !! cout <<styr<<endl<<cntrlfile_name<<endl;exit(1);
  init_int endyr
  init_int rec_age
  init_int oldest_age
  int nages
  !!  nages = oldest_age - rec_age + 1;
  !! cout <<nages<<" "<<oldest_age<<endl;
  int styr_rec
  int styr_sp
  int endyr_sp
  int nyrs
  !!  nyrs = endyr - styr + 1;
  !! styr_rec = (styr - nages) + 1;     // First year of recruitment
  !! styr_sp  = styr_rec - rec_age ;  // First year of spawning biomass
  !! endyr_sp = endyr   - rec_age - 1;// endyr year of (main) spawning biomass

  int junk;
// Fishery specifics
  init_int nfsh                                   //Number of fisheries
  init_matrix catch_bio(1,nfsh,styr,endyr)
  // !! cout << catch_bio<<endl;exit(1);

  init_ivector nyrs_fsh_age(1,nfsh)
  !! cout << "n yrs fish age: "<<nyrs_fsh_age<<endl<<endl;

  init_imatrix yrs_fsh_age(1,nfsh,1,nyrs_fsh_age)
  !! cout << yrs_fsh_age<<endl;// exit(1);

  init_matrix nsmpl_fsh(1,nfsh,1,nyrs_fsh_age)    //Years of survey index value (annual)

  // !! cout << "n smpl fish: "<<nsmpl_fsh<<endl<<endl;cin>>junk;
  init_3darray oac_fsh(1,nfsh,1,nyrs_fsh_age,1,nages)
  // !! cout << "Obs Age Comp fsh: "<<endl<<oac_fsh<<endl<<endl;

  init_3darray wt_fsh(1,nfsh,styr,endyr,1,nages)  //values of Survey proportions at age
  // !! cout <<"fishery weights "<<endl<<wt_fsh<<endl;

  !!CLASS adstring_array fsh_name(1,nfsh);

//  Define surveys
  init_int nsrv                                   //number of surveys
  !! cout <<"Number of surveys "<<nsrv<<endl;
//  Index values
  init_ivector nyrs_srv(1,nsrv)                   //Number of years of survey index value (annual)
  init_imatrix yrs_srv(1,nsrv,1,nyrs_srv)         //Years of survey index value (annual)
  // !! cout <<"yrs srv: "<< yrs_srv<<endl<<endl;cin>>junk;
  init_vector mo_srv(1,nsrv)                      //Month surveys occur
  init_matrix obs_srv(1,nsrv,1,nyrs_srv)          //values of survey index value (annual)
  init_matrix obs_se_srv(1,nsrv,1,nyrs_srv)       //values of survey serrs
  matrix     obs_lse_srv(1,nsrv,1,nyrs_srv) //Survey standard errors (for lognormal)
  !! obs_lse_srv = elem_div(obs_se_srv,obs_srv);
  !! obs_lse_srv = sqrt(log(square(obs_lse_srv) + 1.));
  !!cout <<obs_srv(1)(1,5) << endl;
  // !! cout << "obs srv: "<<obs_srv<<endl<<endl;cin>>junk;
  // !! if (nsrv>0) CLASS adstring_array srv_name(1,nsrv);
  // !! CLASS adstring_array srv_name(1,nsrv);

  init_ivector nyrs_srv_age(1,nsrv)               //Number of years of survey index value (annual)
  init_matrix yrs_srv_age(1,nsrv,1,nyrs_srv_age)  //Years of survey index value (annual)
  // !! cout << "yrs srv age "<<yrs_srv_age<<endl<<endl;cin>>junk;

  !!if (nsrv>0) cout<< " Years of survey age comp: "<<endl<<nyrs_srv_age<<endl<<yrs_srv_age<<endl;

  init_matrix nsmpl_srv(1,nsrv,1,nyrs_srv_age)         //Years of survey index value (annual)
  // !! cout << "n smpl srv: "<<nsmpl_srv<<endl<<endl;cin>>junk;
  init_3darray oac_srv(1,nsrv,1,nyrs_srv_age,1,nages)  //values of Survey proportions at age
  init_3darray  wt_srv(1,nsrv,styr,endyr,1,nages)      //values of Survey proportions at age

  vector age_vector(1,nages);
    !! for (int j=1;j<=nages;j++)
      !!  age_vector(j) = double(j);
  init_vector wt_pop(1,nages)
  init_vector maturity(1,nages)
  !! if (max(maturity)>.9) maturity/=2.;
  !! cout <<endl<<"maturity: "<< maturity<<endl<<endl;// cin>>junk;

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

  int k // Index for fishery or survey
  int i // Index for year
  int j // Index for age
 LOCAL_CALCS
  !! ad_comm::change_datafile_name(cntrlfile_name);
END_CALCS
  !! *(ad_comm::global_datafile) >>  datafile_name;
// Read in "name" of this model...
  !! *(ad_comm::global_datafile) >>  model_name;
  !! cout<<datafile_name<<endl;
  !! cout<<model_name<<endl;
  !! projfile_name = cntrlfile_name(1,length(cntrlfile_name)-4) + ".prj";
  !! cout <<projfile_name<<endl;
  init_int    SrType        // 2 Bholt, 1 Ricker
  init_number steepnessprior
  init_number cvsteepnessprior
  init_int    phase_srec
  !! cout<<" Steep: " <<steepnessprior<<" "<<endl;

  init_number sigmarprior
  number log_sigmarprior
  !! log_sigmarprior = log(sigmarprior);
  !! cout<<" sigmarprior: " <<sigmarprior<<" "<<endl;
  init_number cvsigmarprior
  init_int    phase_sigmar

  init_int    styr_rec_est
  init_int    endyr_rec_est
  int nrecs_est;
  !! nrecs_est = endyr_rec_est-styr_rec_est+1;
  !! nrecs_est = endyr_rec_est-styr_rec_est+1;
  !! cout<<" Rec estimated in styr endyr: " <<styr_rec    <<" "<<endyr        <<" "<<endl;
  !! cout<<" SR Curve fit  in styr endyr: " <<styr_rec_est<<" "<<endyr_rec_est<<" "<<endl;
  !! cout<<"            Model styr endyr: " <<styr        <<" "<<endyr        <<" "<<endl;

  init_number natmortprior
  init_number cvnatmortprior
  init_int    phase_M

  init_vector qprior(1,nsrv)
  vector log_qprior(1,nsrv)
  !! log_qprior = log(qprior);
  !! cout<<" qprior: " <<qprior<<" "<<endl;
  init_vector cvqprior(1,nsrv)
  init_ivector phase_q(1,nsrv)
  init_ivector    q_age_min(1,nsrv)     // Age that q relates to...
  init_ivector    q_age_max(1,nsrv)     // Age that q relates to...
  init_number cv_catchbiomass
  number catchbiomass_pen
  !!catchbiomass_pen= 1./(2*cv_catchbiomass*cv_catchbiomass);
  !! cout<<" cv_catchbiomass: " <<cv_catchbiomass<<" "<<endl;
  !! cout<<" CatchbiomassPen: " <<catchbiomass_pen<<" "<<endl;
  init_number nproj_yrs

  int styr_fut
  int endyr_fut            // LAst year for projections
  int phase_Rzero
  int phase_nosr
  number Steepness_UB
  !! phase_Rzero =  2;
  !! phase_nosr  = -3;
  !! styr_fut    = endyr+1;
  !! endyr_fut   = endyr+nproj_yrs;

  // Selectivity controls
  // read in options for each fishery
  // Loop over fisheries and surveys 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) ;
  int seldecage ;
  !! seldecage = int(nages/2);
  matrix sel_change_in_fsh(1,nfsh,styr,endyr);
  ivector nselages_in_fsh(1,nfsh)

  ivector n_sel_ch_fsh(1,nfsh);
  ivector n_sel_ch_srv(1,nsrv);
  imatrix yrs_sel_ch_tmp(1,nfsh,1,endyr-styr+1);
  imatrix yrs_sel_ch_tsrv(1,nsrv,1,endyr-styr+1);
  !! yrs_sel_ch_tmp.initialize();
  !! yrs_sel_ch_tsrv.initialize();

  ivector   srv_sel_opt(1,nsrv)
  ivector phase_sel_srv(1,nsrv)

  vector   curv_pen_srv(1,nsrv)

  matrix   logsel_slp_in_srv(1,nsrv,1,nyrs)
  matrix   sel_inf_in_srv(1,nsrv,1,nyrs)
  matrix   sel_dslp_in_srv(1,nsrv,1,nyrs)
  matrix   logsel_dslp_in_srv(1,nsrv,1,nyrs)
  matrix   sel_dinf_in_srv(1,nsrv,1,nyrs)
  matrix   sel_slp_in_srv(1,nsrv,1,nyrs)

  vector   logsel_slp_in_srvv(1,nsrv)
  vector   sel_inf_in_srvv(1,nsrv)
  vector   logsel_dslp_in_srvv(1,nsrv)
  vector   sel_dinf_in_srvv(1,nsrv)

  vector seldec_pen_srv(1,nsrv) ;
  matrix sel_change_in_srv(1,nsrv,styr,endyr);
  ivector nselages_in_srv(1,nsrv)

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

```
 ivector phase_dlogist_fsh(1,nfsh)

 ivector phase_selcoff_srv(1,nsrv)
 ivector phase_logist_srv(1,nsrv)
 ivector phase_dlogist_srv(1,nsrv)
 vector  sel_fsh_tmp(1,nages);
 vector  sel_srv_tmp(1,nages);
 3darray log_selcoffs_fsh_in(1,nfsh,1,nyrs,1,nages)
 3darray log_selcoffs_srv_in(1,nsrv,1,nyrs,1,nages)

LOCAL_CALCS
 phase_selcoff_srv.initialize();
 phase_logist_srv.initialize();
 phase_dlogist_srv.initialize();
 sel_fsh_tmp.initialize() ;
 sel_srv_tmp.initialize() ;
 log_selcoffs_fsh_in.initialize();
 log_selcoffs_srv_in.initialize();

 // nselages_in_fsh.initialize()   ;
 // nselages_in_srv.initialize()   ;
 nselages_in_fsh = nages-1;
 nselages_in_srv = nages-1;
 sel_change_in_fsh.initialize()   ;
 sel_change_in_srv.initialize()   ;
 for (int k=1;k<=nfsh;k++)
 {
   *(ad_comm::global_datafile) >> 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) ;
       for (int i=styr;i<=endyr;i++)
         *(ad_comm::global_datafile) >> sel_change_in_fsh(k,i) ;

       sel_change_in_fsh(k,styr)=1.; n_sel_ch_fsh(k)=0.;
       int j=1;
       yrs_sel_ch_tmp(k,j) = styr;
      // 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_fsh(k,i)>0) {
           j++; yrs_sel_ch_tmp(k,j) = i; } }
       n_sel_ch_fsh(k) = j;

       // This to read in pre-specified selectivity values...
       for (int jj=1;jj<=n_sel_ch_fsh(k);jj++)
       {
         for (j=1;j<=nages;j++)
         {
           *(ad_comm::global_datafile) >> sel_fsh_tmp(j);
         }
         // Set the selectivity for the oldest group
         for (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) );
       }

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

       logsel_slp_in_fsh.initialize();
       logsel_dslp_in_fsh.initialize();
       sel_inf_in_fsh.initialize();
       sel_slp_in_fsh.initialize();
     }
       break;
     case 2 : // Single logistic
     {
       *(ad_comm::global_datafile) >> phase_sel_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.; n_sel_ch_fsh(k)=0.;
       int j=1;
       yrs_sel_ch_tmp(k,j) = styr;
      // 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_fsh(k,i)>0) {
           j++; yrs_sel_ch_tmp(k,j) = i; } }
       n_sel_ch_fsh(k) = j;

       // This to read in pre-specified selectivity values...
       for (int jj=1;jj<=n_sel_ch_fsh(k);jj++)
       {
         *(ad_comm::global_datafile) >> sel_slp_in_fsh(k,jj) ;
         *(ad_comm::global_datafile) >> sel_inf_in_fsh(k,jj) ;
       }

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

       logsel_slp_in_fsh(k) = log(sel_slp_in_fsh(k)) ;
       logsel_dslp_in_fsh(k)= 0. ;
       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
     {
       *(ad_comm::global_datafile) >> phase_sel_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.; n_sel_ch_fsh(k)=0.;
       int j=1;
       yrs_sel_ch_tmp(k,j) = styr;
      // 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_fsh(k,i)>0) {
           j++; yrs_sel_ch_tmp(k,j) = i; } }
       n_sel_ch_fsh(k) = j;

       // This to read in pre-specified selectivity values...
       for (int jj=1;jj<=n_sel_ch_fsh(k);jj++)
       {
         *(ad_comm::global_datafile) >> sel_slp_in_fsh(k,jj) ;
```

```
        *(ad_comm::global_datafile) >> sel_inf_in_fsh(k,jj) ;
        *(ad_comm::global_datafile) >> sel_dslp_in_fsh(k,jj) ;
        *(ad_comm::global_datafile) >> sel_dinf_in_fsh(k,jj) ;
      }
      phase_selcoff_fsh(k) = -1;
      phase_logist_fsh(k)  = phase_sel_fsh(k);
      phase_dlogist_fsh(k) = phase_sel_fsh(k)+1;

      logsel_slp_in_fsh(k) = log(sel_slp_in_fsh(k)) ;
      logsel_dslp_in_fsh(k) = log(sel_dslp_in_fsh(k)) ;

      logsel_slp_in_fshv(k) = logsel_slp_in_fsh(k,1);
         sel_inf_in_fshv(k) =    sel_inf_in_fsh(k,1);
      logsel_dslp_in_fshv(k) = logsel_dslp_in_fsh(k,1);
         sel_dinf_in_fshv(k) =    sel_dinf_in_fsh(k,1);
      break;
    }
  }
}
// Surveys here.............
for (int k=1;k<=nsrv;k++)
{
  *(ad_comm::global_datafile) >> srv_sel_opt(k)  ;
  // cout << "Survey Selectivity option: "<<srv_sel_opt(k)  <<endl;
  switch (srv_sel_opt(k))
  {
    case 1 : // Selectivity coefficients
    {
      *(ad_comm::global_datafile) >> nselages_in_srv(k)   ;
      cout << "Survey selages: "<<nselages_in_srv(k)<<endl;
      *(ad_comm::global_datafile) >> phase_sel_srv(k);
      *(ad_comm::global_datafile) >> curv_pen_srv(k) ;
      *(ad_comm::global_datafile) >> seldec_pen_srv(k) ;
      seldec_pen_srv(k) *= seldec_pen_srv(k) ;
      for (int i=styr;i<=endyr;i++)
        *(ad_comm::global_datafile) >> sel_change_in_srv(k,i) ;

      sel_change_in_srv(k,styr)=1.; n_sel_ch_srv(k)=0.; int j=1;
      cout << "srv sel_change_in: "<<sel_change_in_srv(k) << endl;
      yrs_sel_ch_tsrv(k,j) = styr;
     // 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_srv(k,i)>0) {
          j++; yrs_sel_ch_tsrv(k,j) = i; } }
      n_sel_ch_srv(k) = j;

      // This to read in pre-specified selectivity values...
      for (int jj=1;jj<=n_sel_ch_srv(k);jj++) {
        for (j=1;j<=nages;j++) {
          *(ad_comm::global_datafile) >> sel_srv_tmp(j);  }
        for (j=nselages_in_srv(k)+1;j<=nages;j++) {
          sel_srv_tmp(j)  = sel_srv_tmp(nselages_in_srv(k));  }
        log_selcoffs_srv_in(k,jj)(1,nselages_in_srv(k)) = log(sel_srv_tmp(1,nselages_in_srv(k))+
                         1e-7/mean(sel_srv_tmp(1,nselages_in_srv(k))+1e-7) );
      }

      phase_selcoff_srv(k) = phase_sel_srv(k);
      phase_logist_srv(k)  = -1;
      logsel_slp_in_srv.initialize() ;
      sel_inf_in_srv.initialize() ;
      sel_slp_in_srv.initialize() ;

      phase_dlogist_srv(k)  = -1;
      logsel_dslp_in_srv.initialize();
      sel_dinf_in_srv.initialize();
    }
    break;

    case 2 : // Single logistic
    {
      *(ad_comm::global_datafile) >> phase_sel_srv(k);
      // cout<<"PHase sel srv "<<phase_sel_srv(k)<<endl;

      for (int i=styr;i<=endyr;i++) {
        *(ad_comm::global_datafile) >> sel_change_in_srv(k,i) ; }

      sel_change_in_srv(k,styr)=1.; n_sel_ch_srv(k)=0.; int j=1;
      cout << "srv sel_change_in: "<<sel_change_in_srv(k) << endl;
      yrs_sel_ch_tsrv(k,j) = styr;
     // 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_srv(k,i)>0) {
            j++; yrs_sel_ch_tsrv(k,j) = i; } }
      n_sel_ch_srv(k) = j;
      cout<<"Year sel change 1 "<<yrs_sel_ch_tsrv(1,n_sel_ch_srv(k))<<endl;// exit(1);

      // This to read in pre-specified selectivity values...
      for (int jj=1;jj<=n_sel_ch_srv(k);jj++) {
        *(ad_comm::global_datafile) >> sel_slp_in_srv(k,jj) ;
        *(ad_comm::global_datafile) >> sel_inf_in_srv(k,jj) ;
      }

      cout<<k<<" "<< phase_sel_srv(k)<<endl;
      cout<<k<<" "<< sel_slp_in_srv(k)(1,n_sel_ch_srv(k)) <<endl;// exit(1);
      phase_selcoff_srv(k) = -1;
      phase_logist_srv(k) = phase_sel_srv(k);
      logsel_slp_in_srv(k) = log(sel_slp_in_srv(k)) ;
      phase_dlogist_srv(k)  = -1;
      logsel_dslp_in_srv.initialize();
      sel_dinf_in_srv.initialize();

      logsel_slp_in_srvv(k) = logsel_slp_in_srv(k,1);
         sel_inf_in_srvv(k) =    sel_inf_in_srv(k,1);

    }
    break;
    case 3 : // Double logistic
    {
      *(ad_comm::global_datafile) >> phase_sel_srv(k);
      for (int i=styr;i<=endyr;i++) {
        *(ad_comm::global_datafile) >> sel_change_in_srv(k,i) ; }
      sel_change_in_srv(k,styr)=1.; n_sel_ch_srv(k)=0.; int j=1;
      cout << "srv sel_change_in: "<<sel_change_in_srv(k) << endl;
      yrs_sel_ch_tsrv(k,j) = styr;
     // 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_srv(k,i)>0) {
          j++; yrs_sel_ch_tsrv(k,j) = i; } }
      n_sel_ch_srv(k) = j;

      // This to read in pre-specified selectivity values...
      for (int jj=1;jj<=n_sel_ch_srv(k);jj++) {
        *(ad_comm::global_datafile) >> sel_slp_in_srv(k,jj) ;
        *(ad_comm::global_datafile) >> sel_inf_in_srv(k,jj) ;
        *(ad_comm::global_datafile) >> sel_dslp_in_srv(k,jj) ;
        *(ad_comm::global_datafile) >> sel_dinf_in_srv(k,jj) ;
      }
```

```
        phase_selcoff_srv(k) = -1;
        phase_logist_srv(k)  = phase_sel_srv(k);
        phase_dlogist_srv(k) = phase_sel_srv(k)+1;
        logsel_slp_in_srv(k) = log(sel_slp_in_srv(k)) ;
        logsel_dslp_in_srv(k) = log(sel_dslp_in_srv(k)) ;
        logsel_slp_in_srvv(k) = logsel_slp_in_srv(k,1);
           sel_inf_in_srvv(k) =    sel_inf_in_srv(k,1);
        logsel_dslp_in_srvv(k) = logsel_dslp_in_srv(k,1);
           sel_dinf_in_srvv(k) =    sel_dinf_in_srv(k,1);
      }
        break;
    }
    cout << k<<" srv sel opt "<<srv_sel_opt(k)<<" "<<sel_change_in_srv(k)<<endl;
  }
  cout<<"Phase survey Sel_Coffs: "<<phase_selcoff_srv<<endl;
 END_CALCS
  ivector nopt_fsh(1,2) // number of options...
  !! nopt_fsh.initialize();
  !! for (int 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

  !! cout << "Fshry Selages: " << nselages_in_fsh  <<endl;
  !! cout << "Srvy  Selages: " << nselages_in_srv <<endl;



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

  !! cout << "Phase for age-spec survey  "<<phase_selcoff_srv<<endl;
  !! cout << "Phase for logistic survey  "<<phase_logist_srv<<endl;
  !! cout << "Phase for dble logistic srvy "<<phase_dlogist_srv<<endl;

  !! for (int k=1;k<=nfsh;k++) if (phase_selcoff_fsh(k)>0) curv_pen_fsh(k) = 1./
(square(curv_pen_fsh(k))*2);
  !! cout<<"Curv_pen_fsh: "<<curv_pen_fsh<<endl;
  !! for (int k=1;k<=nsrv;k++) if (phase_selcoff_srv(k)>0) curv_pen_srv(k) = 1./
(square(curv_pen_srv(k))*2);

  // !! cout<< curv_pen_fsh<<endl<<curv_pen_srv<<endl;exit(1);

 // Read matrix of selectivity changes by fishery and year...
 // Actual values will be ~CV of change allowed
  // init_matrix sel_change_in(1,nfsh,styr,endyr);
  // !! cout << sel_change_in << endl;exit(1);

  ivector n_sel_ch_fsh(1,nfsh);
  imatrix yrs_sel_ch_tmp(1,nfsh,1,endyr-styr+1);
  int  phase_fmort;
  int  phase_proj;
 LOCAL_CALCS
  for (int k=1; k<=nfsh;k++)
  {
    sel_change_in_fsh(k,styr)=1.;
    n_sel_ch_fsh(k)=0.;
    int j=1;
    cout << "sel_change_in: "<<sel_change_in_fsh(k) << endl;
    yrs_sel_ch_tmp(k,j) = styr;
   // 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_fsh(k,i)>0)
      {
        j++;
        yrs_sel_ch_tmp(k,j) = i;
      }
    }
    n_sel_ch_fsh(k) = j;
    cout <<"Number of select changes: fishery "<<k<<": "<<n_sel_ch_fsh(k)<<endl;
  }
 END_CALCS
  ivector n_sel_ch_srv(1,nsrv);
  imatrix yrs_sel_ch_tsrv(1,nsrv,1,endyr-styr+1);
 LOCAL_CALCS
  for (int k=1; k<=nsrv;k++)
  {
    sel_change_in_srv(k,styr)=1.;
    n_sel_ch_srv(k)=0.;
    int j=1;
    cout << "sel_change_in: "<<sel_change_in_srv(k) << endl;
    yrs_sel_ch_tsrv(k,j) = styr;
   // 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_srv(k,i)>0)
      {
        j++;
        yrs_sel_ch_tsrv(k,j) = i;
      }
    }
    n_sel_ch_srv(k) = j;
    cout <<"Number of select changes: survey "<<k<<": "<<n_sel_ch_srv(k)<<endl;
  }
 END_CALCS
  imatrix yrs_sel_ch_fsh(1,nfsh,1,n_sel_ch_fsh);
  imatrix   nselages_fsh(1,nfsh,1,n_sel_ch_fsh);

  imatrix yrs_sel_ch_srv(1,nsrv,1,n_sel_ch_srv);
  imatrix   nselages_srv(1,nsrv,1,n_sel_ch_srv);
 LOCAL_CALCS
  for (int k=1; k<=nfsh;k++)
    yrs_sel_ch_fsh(k) = yrs_sel_ch_tmp(k)(1,n_sel_ch_fsh(k));
  cout<<"Yrs fsh_sel change: "<<yrs_sel_ch_fsh<<endl;
  for (int k=1; k<=nsrv;k++)
    yrs_sel_ch_srv(k) = yrs_sel_ch_tsrv(k)(1,n_sel_ch_srv(k));
  cout<<"Yrs srv_sel change: "<<yrs_sel_ch_srv<<endl;
 END_CALCS
  number R_guess;

  vector offset_srv(1,nsrv)
  vector offset_fsh(1,nfsh)
  int do_fmort;
  !! do_fmort=0;
 LOCAL_CALCS
  phase_fmort =  2;
  phase_proj  =  7;
  Steepness_UB = .999;

  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) -= nsmpl_fsh(k,i)*(oac_fsh(k,i) + 0.001) * log(oac_fsh(k,i) + 0.001 ) ;
    }

  for (k=1;k<=nsrv;k++)
    for (i=1;i<=nyrs_srv_age(k);i++)
    {
      oac_srv(k,i) /= sum(oac_srv(k,i)); // Normalize to sum to one
      offset_srv(k) -= nsmpl_srv(k,i)*(oac_srv(k,i) + 0.001) * log(oac_srv(k,i) + 0.001 ) ;
    }
  cout<<offset_srv<<endl;
  cout<<offset_fsh<<endl;

  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;
   cout << "Mean Catch"<<endl;
   ctmp = mean(catch_bio);
   cout << ctmp <<endl;
   R_guess = log((ctmp/.05 )/btmp) ;
   cout << "R_guess "<<endl;
   cout << R_guess <<endl;
END_CALCS
```

PARAMETER_SECTION

```
 // Biological Parameters
  init_bounded_number M(.02,.8,phase_M)
  matrix  natage(styr,endyr,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  Z(styr,endyr,1,nages)
  matrix  S(styr,endyr,1,nages)
  number  surv
  number  natmort

 // 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)
  init_bounded_vector rec_dev(styr_rec,endyr,-15,15,2)
  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,1)
  init_bounded_matrix fmort_dev(1,nfsh,styr,endyr,-15,15,phase_fmort)
  vector Fmort(styr,endyr);  // Annual total Fmort
  number hrate
  number Kobs_tot_catch
  number Fnew

  !! for (k=1;k<=nfsh;k++) nselages_fsh(k)=nselages_in_fsh(k);
  !! for (k=1;k<=nsrv;k++) nselages_srv(k)=nselages_in_srv(k);

  init_matrix_vector log_selcoffs_fsh(1,nfsh,1,n_sel_ch_fsh,1,nselages_fsh,phase_selcoff_fsh)
// 3rd dimension out...

  !! cout << " Number of selectivity changes: "<<n_sel_ch_fsh<<endl;
  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)
  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);

  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 expl_biom(1,nfsh,styr,endyr)

 // Parameters for computing SPR rates
  sdreport_number F50_est
  sdreport_number F40_est
  sdreport_number F35_est
  vector F50(1,nfsh)
  vector F40(1,nfsh)
  vector F35(1,nfsh)

 // Stuff for SPR and yield projections
  number sigmar_fut
  vector ftmp(1,nfsh)
  number SB0
  number SBF50
  number SBF40
  number SBF35
  vector Fratio(1,nfsh)

  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_bounded_vector q_srv(1,nsrv,.01,2,-6)
  init_number_vector log_q_srv(1,nsrv,phase_q)
  // init_matrix log_selcoffs_srv(1,nsrv,1,nselages_srv,phase_selcoff_srv)
  !! cout <<nsrv<<endl<<n_sel_ch_srv<<endl<<nselages_srv<<endl<<phase_selcoff_srv<<endl;
  init_matrix_vector log_selcoffs_srv(1,nsrv,1,n_sel_ch_srv,1,nselages_srv,phase_selcoff_srv)

// Need to make positive or reparameterize
  init_vector_vector logsel_slope_srv(1,nsrv,1,n_sel_ch_srv,phase_logist_srv)
  init_vector_vector logsel_dslope_srv(1,nsrv,1,n_sel_ch_srv,phase_dlogist_srv)
  matrix                sel_slope_srv(1,nsrv,1,n_sel_ch_srv)
  matrix                sel_dslope_srv(1,nsrv,1,n_sel_ch_srv)
  init_vector_vector     sel50_srv(1,nsrv,1,n_sel_ch_srv,phase_logist_srv)
  init_bounded_vector_vector     seld50_srv(1,nfsh,1,n_sel_ch_srv,lb_d50,nages,phase_dlogist_srv)

  3darray log_sel_srv(1,nsrv,styr,endyr,1,nages)
  3darray sel_srv(1,nsrv,styr,endyr,1,nages)
  matrix avgsel_srv(1,nsrv,1,n_sel_ch_srv);

  matrix pred_srv(1,nsrv,styr,endyr)
  3darray eac_srv(1,nsrv,1,nyrs_srv_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_srv(1,nsrv)
  matrix sel_like_fsh(1,nfsh,1,4)
  matrix sel_like_srv(1,nsrv,1,4)
  vector surv_like(1,nsrv)
  vector fpen(1,6)
  vector post_priors(1,4)
  vector post_priors_srvq(1,nsrv)
  objective_function_value obj_fun
  vector obj_comps(1,12)
  vector q_srv(1,nsrv)
  // sdreport_vector q_srv(1,nsrv)
  sdreport_vector totbiom(styr,endyr)
  sdreport_vector recruits(styr,endyr)
  sdreport_number depletion
  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_matrix catch_future(1,4,styr_fut,endyr_fut);
  sdreport_matrix future_biomass(1,5,styr_fut,endyr_fut)
  !! cout <<"logRzero "<<log_Rzero<<endl;
  !! cout <<"logmeanrec "<<mean_log_rec<<endl;
  !! cout<<  "sel_slp_in: "<< logsel_slp_in_srv <<endl;
  !! cout<<  "sel_inf_in: "<< sel_inf_in_srv    <<endl;
  !! cout<< "exp(log_sigmarprior "<<exp(log_sigmarprior)<<endl;
```

INITIALIZATION_SECTION

```
  M natmortprior;
  steepness steepnessprior
  log_sigmar log_sigmarprior;
  log_Rzero R_guess;
  mean_log_rec R_guess;
  // log_Rzero 5.94898123049
  // mean_log_rec 5.56119660751
  log_avg_fmort -8.;

  log_q_srv log_qprior;

  logsel_slope_fsh logsel_slp_in_fsh ;
  sel50_fsh sel_inf_in_fsh

  logsel_dslope_fsh logsel_dslp_in_fsh ;
  seld50_fsh sel_dinf_in_fsh

  logsel_slope_srv logsel_slp_in_srv ;
  sel50_srv sel_inf_in_srv ;

  logsel_dslope_srv logsel_dslp_in_srv ;
  seld50_srv sel_dinf_in_srv ;
```

PROCEDURE_SECTION

```
  for (k=1;k<=nsrv;k++) q_srv(k) = mfexp(log_q_srv(k) );

  Get_Selectivity();
  Get_Mortality();
  // if (current_phase()>=phase_Rzero)
    Get_Bzero();
  Get_Numbers_at_Age();
  Get_Survey_Predictions();
  Catch_at_Age();

  if (sd_phase())
  {
    compute_spr_rates();
    if (mcmcmode)
    {
      Calc_Dependent_Vars();
      mcflag   = 0;
      mcmcmode = 0;
    }
    else
    if (mcflag)
    {
      Calc_Dependent_Vars();
    }
  }

  evaluate_the_objective_function();
  if (mceval_phase())
  {
    compute_spr_rates();
    write_mceval();
  }

  if (do_fmort)
    Profile_F();
```

// Functions

FUNCTION write_mceval

```
  if (mcmcmode != 3)
    write_mceval_hdr();
  mcmcmode = 3;
  get_msy();
  Future_projections();
  Calc_Dependent_Vars();
  mceval<<
  q_srv     << " "<<
  M         << " "<<
  steepness << " "<<
  depletion << " "<<
  MSY       << " "<<
  MSYL      << " "<<
  Fmsy      << " "<<
  Fcur_Fmsy << " "<<
  Bcur_Bmsy << " "<<
  Bmsy      << " "<<
  totbiom(endyr)              << " "<<
  F35       << " "<<
  F40       << " "<<
  F50       << " "<<
  future_biomass(1,endyr_fut) << " "<<
  future_biomass(2,endyr_fut) << " "<<
  future_biomass(3,endyr_fut) << " "<<
  future_biomass(4,endyr_fut) << " "<<
  future_biomass(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...)
        if (active(log_selcoffs_fsh(k)))
        {
          int isel_ch_tmp = 1 ;

          dvar_vector sel_coffs_tmp(1,nselages_fsh(k,isel_ch_tmp));
          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,isel_ch_tmp))         = sel_coffs_tmp;
            log_sel_fsh(k,i)(nselages_fsh(k,isel_ch_tmp),nages)     =
                                  log_sel_fsh(k,i,nselages_fsh(k,isel_ch_tmp));
            log_sel_fsh(k,i)                     -= log(mean(mfexp(log_sel_fsh(k,i) )));
          }
        }
      }
      break;
      case 2 : // Single logistic
      {
        // if (active(logsel_slope_fsh(k))) // Need====
        {
          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,isel_ch_tmp))      =
                    -log( 1.0 + mfexp(-sel_slope_tmp *
                    ( age_vector(1,nselages_fsh(k,isel_ch_tmp)) - sel50_tmp) ));
            log_sel_fsh(k,i)(nselages_fsh(k,isel_ch_tmp),nages) =
                    log_sel_fsh(k,i,nselages_fsh(k,isel_ch_tmp));
          }
        }
      }
      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,isel_ch_tmp))      =
                         -log( 1.0 + mfexp(-1.*sel_slope_tmp *
                         ( age_vector(1,nselages_fsh(k,isel_ch_tmp)) - sel50_tmp) ))+
                         log( 1. - 1/(1.0 + mfexp(-sel_dslope_tmp *
                         ( age_vector(1,nselages_fsh(k,isel_ch_tmp)) - seld50_tmp))) );

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

            log_sel_fsh(k,i) -= max(log_sel_fsh(k,i));
```

```
         }
      }
    }
    break;
  }
}

// Survey specific---
for (k=1;k<=nsrv;k++)
{
  switch (srv_sel_opt(k))
  {
    case 1 : // Selectivity coefficients
    //---Calculate the fishery selectivity from the sel_coffs (Only if being used...)
    if (active(log_selcoffs_srv(k)))
    {
      int isel_ch_tmp = 1 ;

      dvar_vector sel_coffs_tmp(1,nselages_srv(k,isel_ch_tmp));
      for (i=styr;i<=endyr;i++)
      {
        if (i==yrs_sel_ch_srv(k,isel_ch_tmp))
        {
          sel_coffs_tmp.initialize();
          sel_coffs_tmp = log_selcoffs_srv(k,isel_ch_tmp);
          avgsel_srv(k,isel_ch_tmp)              =
                  log(mean(mfexp(sel_coffs_tmp(q_age_min(k),q_age_max(k)))));
          // Increment if there is still space to do so...
          if (isel_ch_tmp<n_sel_ch_srv(k))
            isel_ch_tmp++;
        }
       // Need to flag for changing selectivity....XXX
        log_sel_srv(k,i)(1,nselages_srv(k,isel_ch_tmp))        = sel_coffs_tmp;
        log_sel_srv(k,i)(nselages_srv(k,isel_ch_tmp),nages)    =
                  log_sel_srv(k,i,nselages_srv(k,isel_ch_tmp));
        log_sel_srv(k,i)                                  -=
                  log(mean(mfexp(log_sel_srv(k,i)(q_age_min(k),q_age_max(k)))));
      }
    }
      break;
    case 2 : // Asymptotic logistic
      // if (active(logsel_slope_srv(k))) // Need====
      {
        sel_slope_srv(k) = mfexp(logsel_slope_srv(k));
        int isel_ch_tmp = 1 ;
        dvariable sel_slope_tmp = sel_slope_srv(k,isel_ch_tmp);
        dvariable sel50_tmp     = sel50_srv(k,isel_ch_tmp);
        for (i=styr;i<=endyr;i++)
        {
          if (i==yrs_sel_ch_srv(k,isel_ch_tmp))
          {
            sel_slope_tmp = sel_slope_srv(k,isel_ch_tmp);
            sel50_tmp     =     sel50_srv(k,isel_ch_tmp);
            if (isel_ch_tmp<n_sel_ch_srv(k))
              isel_ch_tmp++;
          }
          log_sel_srv(k,i)(1,nselages_srv(k,isel_ch_tmp))     =
                  -1.*log( 1.0 + mfexp(-1.*sel_slope_tmp *
                  ( age_vector(1,nselages_srv(k,isel_ch_tmp)) - sel50_tmp) ));
          log_sel_srv(k,i)(nselages_srv(k,isel_ch_tmp),nages) =
                  log_sel_srv(k,i,nselages_srv(k,isel_ch_tmp));
        }
      }
```

```
        break;
      case 3 : // Double logistic
        // if (active(logsel_slope_srv(k))) // Need====
        {
          sel_slope_srv(k)  = mfexp(logsel_slope_srv(k));
          sel_dslope_srv(k) = mfexp(logsel_dslope_srv(k));
          int isel_ch_tmp = 1 ;
          dvariable sel_slope_tmp = sel_slope_srv(k,isel_ch_tmp);
          dvariable sel50_tmp     = sel50_srv(k,isel_ch_tmp);
          dvariable sel_dslope_tmp = sel_dslope_srv(k,isel_ch_tmp);
          dvariable seld50_tmp     = seld50_srv(k,isel_ch_tmp);
          for (i=styr;i<=endyr;i++)
          {
            if (i==yrs_sel_ch_srv(k,isel_ch_tmp))
            {
              sel_slope_tmp  = sel_slope_srv(k,isel_ch_tmp);
              sel50_tmp      =     sel50_srv(k,isel_ch_tmp);
              sel_dslope_tmp = sel_dslope_srv(k,isel_ch_tmp);
              seld50_tmp     =     seld50_srv(k,isel_ch_tmp);
              if (isel_ch_tmp<n_sel_ch_srv(k))
                isel_ch_tmp++;
            }
            log_sel_srv(k,i)(1,nselages_srv(k,isel_ch_tmp))     =
                         -log( 1.0 + mfexp(-1.*sel_slope_tmp *
                         ( age_vector(1,nselages_srv(k,isel_ch_tmp)) - sel50_tmp) ))+
                         log( 1. - 1/(1.0 + mfexp(-sel_dslope_tmp *
                         ( age_vector(1,nselages_srv(k,isel_ch_tmp)) - seld50_tmp))) );

            log_sel_srv(k,i)(nselages_srv(k,isel_ch_tmp),nages) =
                         log_sel_srv(k,i,nselages_srv(k,isel_ch_tmp));

            log_sel_srv(k,i) -= max(log_sel_srv(k,i));
          }
        }
      break;
    }
  }
  sel_fsh = mfexp(log_sel_fsh);
  sel_srv = mfexp(log_sel_srv);
```

FUNCTION Get_Mortality

```
  surv    = mfexp(-1.0* M);
  natmort = M;
  Z       = M;
  Fmort.initialize();
  for (k=1;k<=nfsh;k++)
  {
    Fmort += mfexp(log_avg_fmort(k) + fmort_dev(k));
    for (i=styr;i<=endyr;i++)
      F(k,i)    = mfexp(log_avg_fmort(k) + fmort_dev(k,i)) * sel_fsh(k,i) ;
    Z     += F(k);
  }
  S  = mfexp(-1*Z);
```

FUNCTION void get_Fs(int i)

```
  Fmort(i) = 0.;
  if (phase_fmort < 0)
  {
    for (k=1;k<=nfsh;k++)
    {
    // Exploitable biomass from each fishery
      expl_biom(k,i) = natage(i)*elem_prod(sel_fsh(k,i),wt_fsh(k,i));
      // this "kludges" the total catch in case it exceeds the population
      if(catch_bio(k,i) > 0.)
      {
        dvariable SK   = posfun( (expl_biom(k,i)-catch_bio(k,i))/expl_biom(k,i) , 0.1 , fpen(4) );
        Kobs_tot_catch = expl_biom(k,i)-SK*expl_biom(k,i);
        hrate          = Kobs_tot_catch / expl_biom(k,i);
        // do newton raphson to get the F
        do_Newton_Raphson_for_mortality( hrate );
        F(k,i)    = Fnew * sel_fsh(k,i);
        Fmort(i) += Fnew;
      }
      Z(i) += F(k,i);
    }
    S(i) = mfexp(-Z(i));
  }
  else
  {
    for (k=1;k<=nfsh;k++)
    {
      Fmort(i) += mfexp(log_avg_fmort(k) + fmort_dev(k,i));
      F(k,i)    = Fmort(i) * sel_fsh(k,i) ;
    }
    Z(i) += Fmort(i);
    S(i)  = mfexp(-Z(i));
  }

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++)
  {
    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);
    Sp_Biom(i)   = elem_prod(natage(i),pow(S(i),spmo_frac)) * elem_prod(wt_pop,maturity);
    // Need to add rec_age part
    mod_rec(i+1)  = natage(i+1,1);
  }
Sp_Biom(endyr)  = elem_prod(natage(endyr),pow(S(endyr),spmo_frac)) * elem_prod(wt_pop,maturity);
FUNCTION Get_Survey_Predictions
  // Survey computations------------------
  dvariable sum_tmp;
  sum_tmp.initialize();
  int yy;
  for (k=1;k<=nsrv;k++)
  {
    for (i=styr;i<=endyr;i++)
    {
      pred_srv(k,i) = q_srv(k) * natage(i) * elem_prod(sel_srv(k,i) , wt_srv(k,i));
    }
    for (i=1;i<=nyrs_srv_age(k);i++)
    {
      yy = yrs_srv_age(k,i);
      dvar_vector tmp_n =elem_prod(sel_srv(k,yy),natage(yy));
      sum_tmp           = sum(tmp_n);
      eac_srv(k,i)      = tmp_n/sum_tmp;
    }
  }
```

```
FUNCTION Calc_Dependent_Vars
  get_msy();

  if (phase_proj>0) Future_projections();
  for (i=styr;i<=endyr;i++)
  {
    recruits(i)=natage(i,1);
    totbiom(i)=natage(i)*wt_pop;
    depletion=totbiom(endyr)/totbiom(styr);
  }


FUNCTION Catch_at_Age
  for (k=1;k<=nfsh;k++)
  {
    catage(k) = elem_prod(elem_div(F(k),Z),elem_prod(1.-S,natage));
    dvar_matrix Ctmp = catage(k); // Copy 3darray to matrix for efficiency...
    for (i=styr; i<=endyr; i++)
      pred_catch(k,i) = Ctmp(i)*wt_fsh(k,i);
    for (i=1; i<=nyrs_fsh_age(k); i++)
      eac_fsh(k,i)=Ctmp(yrs_fsh_age(k,i))/sum(Ctmp(yrs_fsh_age(k,i)));
  }
FUNCTION evaluate_the_objective_function
  Cat_Like();
  Rec_Like();
  Age_Like();
  Srv_Like();
  Sel_Like();
  Fmort_Pen();
  Compute_priors();

  if (active(log_Rzero))
    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(surv_like);
  obj_comps(5) = sum(age_like_srv);
  obj_comps(6) = sum(sel_like_srv);
  obj_comps(7) = sum(rec_like);
  obj_comps(8) = sum(fpen);
  obj_comps(9) = sum(post_priors_srvq);
  obj_comps(10)= sum(post_priors);
  obj_fun     += sum(obj_comps);
FUNCTION Cat_Like
  catch_like.initialize();
  if (current_phase()<2)
  {
    for (k=1;k<=nfsh;k++)
      catch_like(k) = .01 * catchbiomass_pen * norm2(log(catch_bio(k) +.000001)-log(pred_catch(k)
+.000001));
      // cout<< "1st C_Like "<<catch_like<< endl;
  }
  else
  {
    for (k=1;k<=nfsh;k++)
      catch_like(k) =       catchbiomass_pen * norm2(log(catch_bio(k) +.000001)-log(pred_catch(k)
+.000001));
      // cout<< "2nd C_Like "<<catch_like<< endl;
  }
```

```
FUNCTION Rec_Like
  rec_like.initialize();
  if (active(rec_dev))
  {
    sigmar     =  mfexp(log_sigmar);
    sigmarsq   =  square(sigmar);
    dvariable SSQRec;
    SSQRec.initialize();
    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));
      }
      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_sigmar   =  sqrt( SSQRec  / nrecs_est);
      m_sigmarsq =  m_sigmar * m_sigmar   ;

      if (current_phase()>4||last_phase())
        rec_like(1) = (SSQRec+ sigmarsq/2.)/(2*sigmarsq) + nrecs_est*log_sigmar;
      else
        rec_like(1) = .1*(SSQRec+ 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
    {
       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_srvq.initialize();
  for (k=1;k<=nsrv;k++)
    if (active(log_q_srv(k)))
      post_priors_srvq(k) += square(q_srv(k)-qprior(k))/(2*cvqprior(k)*cvqprior(k));
  if (active(M))
    post_priors(1) += square(M-natmortprior)/(2*cvnatmortprior*cvnatmortprior);

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

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

FUNCTION Fmort_Pen
  fpen.initialize();
  // Phases less than 3, penalize High F's---------------------------------
  if (current_phase()<3)
  {
    fpen(1) += 10.* norm2(Fmort - .2);
  }
  else
  {
    fpen(1) +=.001*norm2(Fmort - .2);
  }

  for (k=1;k<=nfsh;k++)
  {
    fpen(2) += 20.*square(mean(fmort_dev(k)) );
  }

FUNCTION Sel_Like
  sel_like_fsh.initialize();
  sel_like_srv.initialize();
  for (k=1;k<=nfsh;k++)
  {
    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 ))));
        // This part is the penalty on the change itself--------------
        if (i>1)
        {
          dvariable var_tmp = square(sel_change_in_fsh(k,iyr ));
          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,1);
        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<=nsrv;k++)
  {
    if (active(log_selcoffs_srv(k)))
    {
      for (i=1;i<=n_sel_ch_srv(k);i++)
      {
        int iyr = yrs_sel_ch_srv(k,i) ;
        sel_like_srv(k,1) += curv_pen_srv(k)*norm2(first_difference(
                                                 first_difference(log_sel_srv(k,iyr))));
        // This part is the penalty on the change itself--------------
        if (i>1)
        {
          dvariable var_tmp = square(sel_change_in_srv(k,iyr ));
          sel_like_srv(k,2)    += .5*norm2( log_sel_srv(k,iyr-1) - log_sel_srv(k,iyr) )
                                   / var_tmp ;
        }
        int nagestmp = nselages_srv(k,1);
        for (j=seldecage;j<=nagestmp;j++)
        {
          dvariable difftmp = log_sel_srv(k,iyr,j-1)-log_sel_srv(k,iyr,j) ;
          if (difftmp > 0.)
            sel_like_srv(k,3)    += .5*square( difftmp ) / seldec_pen_srv(k);
        }
        obj_fun            += 20. * square(avgsel_srv(k,i));  // To normalize selectivities
      }
    }
  }
```

FUNCTION Srv_Like

```
  // Fit to indices (Normal) -------------------------------------------
  surv_like.initialize();
  for (k=1;k<=nsrv;k++)
    for (i=1;i<=nyrs_srv(k);i++)
      surv_like(k) += square(obs_srv(k,i) - pred_srv(k,yrs_srv(k,i)) ) /
                                   (2.*obs_se_srv(k,i)*obs_se_srv(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) -= nsmpl_fsh(k,i)*(oac_fsh(k,i) + 0.001) * log(eac_fsh(k,i) + 0.001 ) ;
  age_like_fsh-=offset_fsh;

  age_like_srv.initialize();
  for (k=1;k<=nsrv;k++)
    for (int i=1;i<=nyrs_srv_age(k);i++)
      age_like_srv(k) -= nsmpl_srv(k,i)*(oac_srv(k,i) + 0.001) * log(eac_srv(k,i) + 0.001 ) ;
  age_like_srv-=offset_srv;
```

FUNCTION dvariable get_spr_rates(double spr_percent)

```
  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)));
    yld2   = -1000*square(log(spr_percent/spr_ratio(F2, sel_tmp)));
    yld3   = -1000*square(log(spr_percent/spr_ratio(F3, sel_tmp)));
    // cout<<F1<<" "<<F2<<" "<<F3<<" "<<yld1<<" "<<yld2<<endl;
    // cout <<spr_percent<<" "<<spr_ratio(F1,sel_tmp)<<endl;
    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(F1);
FUNCTION dvariable spr_ratio(dvariable trial_F,dvar_matrix sel_tmp)
  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);
  Ftmp = sel_tmp;
  for (j=1;j<=nages;j++)
  {
    Ftmp(j) = elem_prod(Ftmp(j), trial_F * Fratio);
    srvtmp(j)  = exp(-sum(Ftmp(j)) - natmort);
  }
  Ntmp(1)=1.;
  j=1;
  SBtmp  += Ntmp(j)*maturity(j)*wt_pop(j)*pow(srvtmp(j),spmo_frac);
  for (j=2;j<nages;j++)
  {
    Ntmp(j) = Ntmp(j-1)*srvtmp(j-1);
    SBtmp  += Ntmp(j)*maturity(j)*wt_pop(j)*pow(srvtmp(j),spmo_frac);
  }
  Ntmp(nages)=Ntmp(nages-1)*srvtmp(nages-1)/(1.-srvtmp(nages));

  SBtmp  += Ntmp(nages)*maturity(nages)*wt_pop(nages)*pow(srvtmp(nages),spmo_frac);
  return(SBtmp/phizero);

FUNCTION dvariable spr_unfished()
  dvariable Ntmp;
  dvariable SBtmp;
  SBtmp.initialize();
  Ntmp = 1.;
  for (j=1;j<nages;j++)
  {
    SBtmp += Ntmp*maturity(j)*wt_pop(j)*exp(-spmo_frac * natmort);
    Ntmp  *= exp( -natmort);
  }
  Ntmp    /= (1.-exp(-natmort));
  SBtmp += Ntmp*maturity(j)*wt_pop(j)*exp(-spmo_frac * natmort);

  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 Future_projections
  // Need to check on treatment of Fratio--whether it should be included or not
  future_biomass.initialize();
  catch_future.initialize();
  for (int l=1;l<=5;l++)
  {
    // get F's
    switch (l)
    {
      case 1:
        ftmp = F50;
        break;
      case 2:
        ftmp = F40;
        break;
      case 3:
        ftmp = F35;
        break;
      case 4:
        ftmp = (Fmsy ); //, Fratio);
        break;
      case 5:
        ftmp = 0.0;
        break;
    }
    // Get future F's
    Z_future = natmort;
    for (i=endyr+1;i<=endyr_fut;i++)
    {
      for (k=1;k<=nfsh;k++)
      {
        F_future(k,i) = sel_fsh(k,endyr) * ftmp(k);
        Z_future(i)  += F_future(k,i);
      }
      S_future(i) = exp(-Z_future(i));
    }
 // 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);

    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);

    // 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)) ;
        Sp_Biom_future(i) = elem_prod(wt_pop ,maturity) *
                            elem_prod(nage_future(i),pow(S_future(i),spmo_frac)) ;
       // 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);
      }
      nage_future(endyr_fut,1)  = SRecruit( Sp_Biom_future(endyr_fut-rec_age) ) *
                                 mfexp(rec_dev_future(endyr_fut)) ;
      Sp_Biom_future(endyr_fut)  = elem_prod(wt_pop ,maturity) *
                                 elem_prod(nage_future(endyr_fut),
                                 pow(S_future(endyr_fut),spmo_frac)) ;
      // 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 (l!=5)
            catch_future(l,i)   += catage_future(i)*wt_fsh(k,endyr);
        }
        future_biomass(l,i) = Sp_Biom_future(i);
      }
    }   //End of loop over F's
```

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 = (1.1*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);
    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;
      else
        cout<<"Fmsy v. low "<< endl;
    }
  }
  {
    dvar_vector ttt(1,4);
    ttt      = yld(Fratio,F1);
    lnFmsy   = log(F1);
    Fmsy     = F1;
    MSY      = ttt(2);
    Bmsy     = ttt(1);
    MSYL     = ttt(1)/Bzero;
    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 dvar_vector yld(const dvar_vector& Fratio, const dvariable& Ftmp)
  RETURN_ARRAYS_INCREMENT();
  dvar_vector msy_stuff(1,4);
  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;
  for (k=1;k<=nfsh;k++)
  {
    Fatmp(k) = Fratio(k) * Ftmp * seltmp(k);
    Ztmp    += Fatmp(k);
  }
  dvar_vector survtmp = exp(-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( elem_prod( Ntmp , pow(survtmp,spmo_frac ) ), maturity ) * wt_pop;
  msy_stuff(4)  = phi/phizero ;       // SPR
  msy_stuff(3)  = Requil(phi) ;       // Eq Recruitment
  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)
  RETURN_ARRAYS_INCREMENT();
  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;
  for (k=1;k<=nfsh;k++)
  {
    Fatmp(k) = Fratio(k) * Ftmp * seltmp(k);
    Ztmp    += Fatmp(k);
  }
  dvar_vector survtmp = exp(-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( elem_prod( Ntmp , pow(survtmp,spmo_frac ) ), maturity ) * wt_pop;
  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;
  for (k=1;k<=nfsh;k++)
  {
    Fatmp(k) = Fratio(k) * Ftmp * seltmp(k);
    Ztmp    += Fatmp(k);
  }
  dvar_vector survtmp = exp(-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( elem_prod( Ntmp , pow(survtmp,spmo_frac ) ), maturity ) * wt_pop;
  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(1,4);
  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());
      // dvariable R_alpha;
      // dvariable R_beta;
  switch (SrType)
  {
    case 1:
       //Ricker form from Dorn
      RecTmp = elem_prod((Stmp / phizero) , mfexp( alpha * ( 1. - Stmp / Bzero ))) ;
      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;
      // dvariable R_alpha;
      // dvariable R_beta;
  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 = exp(-natmort);

  dvar_matrix natagetmp(styr_rec,styr,1,nages);
  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 = elem_prod(wt_pop , maturity) * 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:
    //R = S * EXP(alpha - beta * S))
      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) * elem_prod(wt_pop,maturity);

    // natagetmp(i,1)          = mfexp(rec_dev(i) + log_Rzero); // OjO
    natagetmp(i,1)          = mfexp(rec_dev(i) + mean_log_rec);
    natagetmp(i+1)(2,nages) = ++(natagetmp(i)(1,nages-1)*mfexp(-natmort ));
    natagetmp(i+1,nages)    += natagetmp(i,nages)*mfexp(-natmort);
  }
  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) * elem_prod(wt_pop,maturity);

FUNCTION dvariable Requil(dvariable& phi)
  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 RecTmp;

FUNCTION write_mceval_hdr
    for (k=1;k<=nsrv;k++)
      mceval<< " q_srv_"<< 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;
```

REPORT_SECTION

```
  // system("cls");
  cout <<"================================================================"<<endl;
  cout<<"||"<<endl<<"||  Amak version 1.0 2003"<<endl<<"||"<<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;
  //   cout<<"Fishery Selectivity coefficients: "<<endl<< log_selcoffs_fsh<<endl;
    adstring comma = adstring(",");
    report << model_name<<" "<< endl<< endl;
    // cout<<"Debugging new spr calcs"<<endl;
    //cout<< "F35: "<<F35<<" "<<get_spr_rates(.35)<<endl;;
    // cout<< "F40: "<<F40<<" "<<get_spr_rates(.4)<<endl;;
    // cout<< "F50: "<<F50<<" "<<get_spr_rates(.5)<<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<=nsrv;k++)
    {
      int ii=1;
      report <<endl<< "Yr_Obs_Pred_Survey "<< k <<" : "<< endl ;
      for (i=styr;i<=endyr;i++)
      {
        if (ii<=yrs_srv(k).indexmax())
        {
          if (yrs_srv(k,ii)==i)
          {
            report << i<< " "<< obs_srv(k,ii) << " "<< pred_srv(k,i) <<endl;
            ii++;
          }
          else
            report << i<< " -1 "<< " "<< pred_srv(k,i)<<endl;
        }
        else
          report << i<< " -1 "<< " "<< pred_srv(k,i)<<endl;
      }
    }

    report << endl<< "Survey_Q:  "<<q_srv << 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<=nsrv;k++)
    {
      report << "ObsSurvey "<<k<<" : "<<  endl;
      for (i=1;i<=nyrs_srv_age(k);i++)
        report << yrs_srv_age(k,i)<< " "<< oac_srv(k,i) << endl;
    }
    report << endl<< "Predicted prop Survey" << endl;
    for (k=1;k<=nsrv;k++)
    {
      report << "PredSurvey "<<k<<" : "<<  endl;
      for (i=1;i<=nyrs_srv_age(k);i++)
        report << yrs_srv_age(k,i)<< " "<< eac_srv(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 << mfexp(log_avg_fmort+fmort_dev) << 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<=nsrv;k++)
      for (i=styr;i<=endyr;i++)
        report << "Survey  "<< k <<"  "<< i<<" "<<sel_srv(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 surv_like age_like_srv sel_like_srv
rec_like fpen post_priors_srvq 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
             <<"  surv_like        "<<setw(10)<<obj_comps(4) <<endl
             <<"  age_like_srv     "<<setw(10)<<obj_comps(5) <<endl
             <<"  sel_like_srv     "<<setw(10)<<obj_comps(6) <<endl
             <<"  rec_like         "<<setw(10)<<obj_comps(7) <<endl
             <<"  fpen             "<<setw(10)<<obj_comps(8) <<endl
             <<"  post_priors_srvq "<<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<=nsrv;k++)
      report << "  Survey_Index_#"<< k <<"  " << surv_like(k)<<endl;
    report   << endl;

    report << setw(10)<< setfixed() << setprecision(5) <<endl;
    report   << "Age likelihoods for surveys :"<<endl;
    report   << "-------------------------" <<endl;
    for (k=1;k<=nsrv;k++)
      report << "  Age_Survey_#"<< k <<"  " << age_like_srv(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<=nsrv;k++)
      report << "  Sel_Survey_#"<< k <<"  "<< sel_like_srv(k,1) <<" "<<sel_like_srv(k,2)<<"
"<<sel_like_srv(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));
  // printf("loaded simdll.dll successfully\n");
    for (k=1;k<=nsrv;k++)
      report << "Q_Survey_#"<< k <<"           "
             << setw(10)<<post_priors_srvq(k)
             << setw(10)<< q_srv(k)
             << setw(10)<< qprior(k)
             << setw(10)<< cvqprior(k)<<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<<"cv_catchbiomass: " <<cv_catchbiomass<<" "<<endl;
  report<<"Projection_years "<< nproj_yrs<<endl;
  for (int k=1;k<=nfsh;k++)
    report << "Fsh_sel_opt_fish: "<<k<<" "<<fsh_sel_opt(k)<<" "<<sel_change_in_fsh(k)<<endl;
  for (int k=1;k<=nsrv;k++)
    report<<"Survey_Sel_Opt_Survey: " <<k<<" "<<(srv_sel_opt(k))<<endl;

  report <<"Phase_survey_Sel_Coffs: "<<phase_selcoff_srv<<endl;
  report <<"Fshry_Selages: " << nselages_in_fsh  <<endl;
  report <<"Survy_Selages: " << nselages_in_srv <<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_srv<<endl;
  report << "Phase_for_logistic_survey  "<<phase_logist_srv<<endl;
  report << "Phase_for_dble_logistic_srvy "<<phase_dlogist_srv<<endl;

  for (int 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 (int k=1; k<=nsrv;k++)
  {
    report <<"Number_of_select_changes_survey: "<<k<<" "<<n_sel_ch_srv(k)<<endl;
    report<<"Yrs_srv_sel_change: "<<yrs_sel_ch_srv(k)<<endl;
    report << "sel_change_in: "<<sel_change_in_srv(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 <<  exp(log_avg_fmort(k) + fmort_dev(k,endyr)) /Fmort(endyr)<<" ";

  projout << "   # Fratio                  " <<endl;

  projout << mean(Fmort(endyr-4,endyr))<<"  # 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) <<" ";
  projout<< endl;
  projout <<"# natage"<<endl<< natage(endyr) << endl;
  projout <<"# Nrec"<<endl<< endyr-1978<< endl;
  projout <<"# rec"<<endl<< mod_rec(1978,endyr) << endl;
```

RUNTIME_SECTION

```
  convergence_criteria 1.e-1,1.e-2,1.e-2,1.e-2,1.e-4,1.e-7
  maximum_function_evaluations 20, 20, 40, 1000
```

TOP_OF_MAIN_SECTION

```
  gradient_structure::set_MAX_NVAR_OFFSET(1000);
  gradient_structure::set_NUM_DEPENDENT_VARIABLES(1000);
  gradient_structure::set_GRADSTACK_BUFFER_SIZE(1000000);
  gradient_structure::set_CMPDIF_BUFFER_SIZE(10000000);
  arrmblsize=500000000;
```

FINAL_SECTION

```
  write_projout();
  write_msy_out();
  Profile_F();
```

GLOBALS_SECTION

```
  #include <admodel.h>
  adstring model_name;
  adstring projfile_name;
  adstring datafile_name;
  adstring cntrlfile_name;
  adstring tmpstring;
  adstring repstring;
```

```
FUNCTION void do_Newton_Raphson_for_mortality(dvariable hrate)
  dvariable Fold ;
  Fold = hrate;
  for (int ii=1;ii<=4;ii++)
  {
      dvariable ZZ = Fold + natmort;
      dvariable XX = exp(-ZZ);
      dvariable AA = Fold * (1. - XX);
      dvariable BB = ZZ;
      dvariable CC = 1. + (Fold - 1) * XX;
      dvariable dd = 1.;
      dvariable FX = AA / BB - hrate;
      dvariable FPX = (BB * CC - AA * dd) / (BB * BB);
      Fnew = Fold - FX / FPX;
      Fold = Fnew;
  }
```