A state space, age-structured production model (SSASPM) with application to HMS Atlantic sharpnose shark: computer code

Enric Cortés

SEDAR34-WP-41

20 September 2013

Please cite this document as:

Cortés, E. 2013. A state space, age-structured production model (SSASPM) with application to HMS Atlantic sharpnose shark: computer code. SEDAR34-WP-41. SEDAR, North Charleston, SC. 60 pp.

```
#//////////////////////////////////////////////////////////////////////
#//   INPUT DATA  FILE  FOR   PROGRAM    SSASPM
#//
#//   Important   notes:
#//   -1    Comments    may   be    placed      BEFORE      or    AFTER any
      line  of    data, however     they  MUST  begin
#//   with  a     #     symbol      in    the   first column.
#//   -2    No    comments    of    any   kind  may   appear      on    the
      same  line  as    the   data  (the  #
#//   symbol      will  not   save  you   here)
#//   -3    Blank lines without     a     #     symbol      are   not
      allowed.
#//
#//
#//////////////////////////////////////////////////////////////////////
#####################################################
#     GENERAL     INFORMATION- Atlantic Sharpnose
#####################################################
#     first and   last  year  of    data
      1950  2011
#     number      of    years of    historical  period
22
#     number      of    years to    project
0
#     starting    value for   pup   survival    (allows     model to
      start away  from  mode; enter 0     to    start at    best_guess  in
      .prm  file)
0
#     first and   last  age   of    data
      1     18
#     number      of    seasons     (months)    per   year
12
#     type  of    overall     variance    parameter   (1    =     log
      scale variance,   2     =     observation scale variance,   0=force
      equal weighting)
1
#     pupping     season      (integer    representing      season/month
      of    year  when  spawning    occurs)
6
#     maturity    schedule    (fraction   of    each  age   class that  is
      sexually    mature)
0.185 0.953 0.999 1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0
      1.0   1.0   1.0   1.0   1.0
#     fecundity   schedule    (index      of    per   capita      fecundity
      of    each  age   class)using constant fecundity now
0.501 0.978 1.407 1.714 1.908 2.022 2.087 2.124 2.144 2.154 2.160 2.164
      2.165 2.166 2.167 2.167 2.167 2.167
#####################################################
#     CATCH INFORMATION
#####################################################
#     number      of    catch data  series      (if   there are   no
      series,     there should      be    no    entries     after the   next
      line  below)
5
```

method of setting prehistoric effort (--***--input an integer FOR EACH FLEET--***--)
0 = set equal to effort input values
1 = set equal to constant specified in the parameter file
2 = linearly interpolate from the constant specified in the parameter file for year 1
to the estimate for the first year of the modern period
1 1 2 2 1
pdf of observation error for each series -1 lognormal, -2 normal
1 1 1 1 1
units (1=numbers, 2=weight)
1 1 1 1 1
season (month) when fishing begins for each series
1 1 1 1 1
season (month) when fishing ends for each series
9 9 9 9 9
set of catch variance parameters each series is linked to
1 1 1 1 1
set of q parameters each series is linked to
1 2 3 4 5
set of s parameters each series is linked to
1 4 1 6 3
set of e parameters each series is linked to
1 2 3 4 5
observed catches by set

#Com-BLL	Com-GN	Com-L	Rec	Shrimp	Year
0	0	0	12114	199157	1950
0	0	0	13314	255841	1951
0	0	1	14514	258937	1952
0	0	1	15714	297766	1953
0	0	2	16914	307492	1954
0	0	2	18114	278697	1955
0	0	2	19314	253339	1956
0	0	3	20514	227780	1957
0	0	3	21714	226216	1958
0	0	4	22914	253769	1959
0	0	4	24114	271849	1960
0	0	4	24815	136426	1961
0	0	5	25517	178861	1962
0	0	5	26218	269133	1963
0	0	6	26920	240757	1964
0	0	6	27621	258877	1965
0	0	6	28322	244276	1966
0	0	7	29024	299894	1967
0	0	7	29725	273578	1968
0	0	8	30427	286401	1969
0	0	8	31128	315416	1970
0	0	8	34310	323214	1971
0	0	9	34613	1403939	1972

0	0	9	34916	1224615	1973
0	0	9	35220	1488981	1974
0	0	10	35523	1007433	1975
0	0	10	35827	1928857	1976
0	0	11	36130	2104965	1977
0	0	11	36434	2746465	1978
0	0	11	36737	3896932	1979
20140	0	12	41970	2261144	1980
20165	0	12	44075	2754240	1981
20202	0	13	34837	2591957	1982
20258	0	13	39881	2557525	1983
20340	0	13	36695	2530402	1984
20463	0	14	22568	2905822	1985
20646	0	14	35633	3402573	1986
20920	663	15	36221	4632197	1987
21328	1326	15	82228	3206838	1988
21937	1989	15	55866	4076237	1989
22845	2652	16	52842	3920057	1990
24200	3315	16	122400	4275462	1991
26222	3978	17	85537	3217356	1992
17791	4641	17	82573	2995442	1993
28788	5305	17	111969	3613709	1994
53212	6310	19	158522	3245293	1995
93206	3090	15	88897	3345772	1996
27196	65059	956	76944	3404777	1997
22017	57737	2128	79455	3976312	1998
21338	60540	4342	80092	4158720	1999
18316	35222	1220	148343	3809182	2000
18376	49853	1301	170093	3197969	2001
25728	45161	953	109597	2969866	2002
48485	21016	2791	113442	2873080	2003
40079	36114	731	100899	2329118	2004
42424	70151	1225	110328	1525548	2005
49001	93272	2243	139702	1764730	2006
20638	122039	3309	143935	1570637	2007
15514	58008	1395	102155	1287044	2008
36266	59639	1342	96923	1715665	2009
22426	39657	5205	156814	1220501	2010
28198	54744	6742	60314	1197353	2011

annual scaling factors

#Com-BLL Com-GN Com-L BLL-Dis Rec Shrimp Year

2	2	2	2	2	1950
2	2	2	2	2	1951
2	2	2	2	2	1952
2	2	2	2	2	1953
2	2	2	2	2	1954
2	2	2	2	2	1955
2	2	2	2	2	1956
2	2	2	2	2	1957
2	2	2	2	2	1958
2	2	2	2	2	1959
2	2	2	2	2	1960
2	2	2	2	2	1961

2	2	2	2	2	1962
2	2	2	2	2	1963
2	2	2	2	2	1964
2	2	2	2	2	1965
2	2	2	2	2	1966
2	2	2	2	2	1967
2	2	2	2	2	1968
2	2	2	2	2	1969
2	2	2	2	2	1970
2	2	2	2	2	1971
2	2	2	2	1	1972
2	2	2	2	1	1973
2	2	2	2	1	1974
2	2	2	2	1	1975
2	2	2	2	1	1976
2	2	2	2	1	1977
2	2	2	2	1	1978
2	2	2	2	1	1979
2	2	2	1	1	1980
2	2	2	1	1	1981
2	2	2	1	1	1982
2	2	2	1	1	1983
2	2	2	1	1	1984
2	2	2	1	1	1985
2	2	2	1	1	1986
2	2	2	1	2	1987
2	2	2	1	1	1988
2	2	2	1	1	1989
2	2	2	1	1	1990
2	2	2	1	1	1991
2	2	2	1	1	1992
2	2	2	1	1	1993
2	2	2	1	1	1994
1	1	1	1	1	1995
1	1	1	1	1	1996
1	1	1	1	1	1997
1	1	1	1	1	1998
1	1	1	1	1	1999
1	1	1	1	1	2000
1	1	1	1	1	2001
1	1	1	1	1	2002
1	1	2	1	1	2003
1	1	1	1	1	2004
1	1	1	1	1	2005
1	1	1	1	1	2006
1	1	1	1	1	2007
1	1	1	1	1	2008
1	1	1	1	1	2009
1	1	1	1	1	2010
1	1	1	1	1	2011

##

INDICES OF ABUNDANCE (e.g., CPUE) If there are no series, there should be no entries between the comment lines.

##

number of index data series
15
pdf of observation error for each series -1 lognormal, -2 normal
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
units (1=numbers, 2=weight)
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
season (month) when index begins for each series
4 4 3 1 4 4 6 7 4 8 8 6 10 4 4
season (month) when index ends for each series
10 9 10 12 11 10 9 9 8 12 12 7 11 11 10
option to -1 scale or 0 not to scale index observations
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
set of index variance parameters each series is linked to
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
set of q parameters each series is linked to
6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
set of s parameters each series is linked to
6 1 6 1 5 3 2 1 5 2 2 3 3 2 5
observed indices by series (last column is for year)
#PC-LL PC-GN.a PC-GN.j GNOP BLLOP SEAMAP-SA Texas VA-LL NMFS-LL-SE SC-GN SCDNR SEAMAP-GOM-ES SEAMAP-GOM-EF UNC MML-GN.a MML-GN.j Year (2007)
#PC-LL ATL Coastspan LL GOM Comb BLL BLLOP SEAMAP-SA GOM Comb GN VA-LL NMFS-LL-SE SC-GN SCDNR-LL-RD Early GA-SCDNR-LL-RD Late SEAMAP GOM ES SEAMAP GOM EF UNC-LL GADNR Trawl Year (now)
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1950
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1951
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1952
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1953
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1954
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1955
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1956

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1957
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1958
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1959
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1960
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1961
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1962
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1963
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1964
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1965
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1966
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1967
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1968
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1969
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1970
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1971
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0.203	-1	-1	1972
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0.232	0.010	-1	1973
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1.019	0.004	-1	1974
-1	-1	-1	-1	-1	-1	0.26	-1	-1	-1	-1	-1	0.726	0.009	-1	1975
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0.677	0.004	-1	1976
-1	-1	-1	-1	-1	-1	0.24	-1	-1	-1	-1	-1	0.302	0.007	-1	1977
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0.373	0.016	-1	1978
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0.383	0.012	-1	1979
-1	-1	-1	-1	-1	-1	0.39	-1	-1	-1	-1	-1	0.686	0.010	-1	1980
-1	-1	-1	-1	-1	-1	0.47	-1	-1	-1	-1	-1	0.565	0.007	-1	1981
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0.019	0.479	0.004	-1	1982
-1	-1	-1	-1	-1	-1	0.40	-1	-1	-1	-1	0.452	0.330	0.016	-1	1983

-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0.030
0.273 0.010 -1 1984
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0.100
0.284 0.012 -1 1985
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0.063
0.304 0.014 -1 1986
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0.293
0.559 0.018 -1 1987
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0.274
0.174 0.033 -1 1988
-1 -1 -1 -1 3.114 -1 -1 -1 -1 -1 -1 0.199
0.168 0.012 -1 1989
-1 -1 -1 -1 2.784 -1 0.35 -1 -1 -1 -1 0.067
0.181 0.017 -1 1990
-1 -1 -1 -1 2.968 -1 0.32 -1 -1 -1 -1 0.218
0.122 0.027 -1 1991
-1 -1 -1 -1 2.711 -1 0.42 -1 -1 -1 -1 0.199
0.072 0.054 -1 1992
0.481 -1 -1 -1 2.080 -1 0.27 -1 -1 -1 -1 0.242
0.164 0.031 -1 1993
0.136 -1 -1 14.450 1.468 -1 -1 -1 -1 -1 -1
0.098 0.233 0.027 -1 1994
0.301 -1 -1 92.725 2.935 0.848 0.53 1.03 -1 -1 -1
0.431 0.128 0.049 -1 1995
0.951 -1 -1 80.747 1.693 0.816 0.32 1.37 -1 -1 -1
0.366 0.315 0.022 -1 1996
0.531 -1 -1 181.956 3.695 1.399 0.22 1.23 -1 -1 -1
0.188 0.154 0.031 -1 1997
0.38 -1 -1 245.977 2.530 0.968 0.52 -1 2.366 0.079 -1
0.144 0.139 0.037 -1 1998
1.16 -1 -1 383.974 2.591 1.469 0.60 1.071 -1 0.046 -1
0.201 0.273 0.033 -1 1999
0.445 30.037 -1 445.425 3.660 1.962 0.15 2.490 0.020 0.105 -1
0.217 0.209 0.044 -1 2000
-1 158.545 -1 215.125 3.227 1.595 0.28 3.637 0.303 0.141 -1
0.097 0.092 -1 -1 2001
-1 33.902 -1 184.152 5.152 1.772 0.14 4.626 1.285 0.135 -1
0.253 0.109 0.042 -1 2002
-1 46.325 -1 130.171 5.296 1.529 0.11 5.198 3.990 0.084 -1
0.152 0.149 0.087 3.169 2003
-1 38.637 3.989 126.152 3.684 1.509 0.14 8.477 -1 0.030 -1
0.098 0.139 0.068 2.277 2004
-1 48.276 4.000 149.740 4.587 1.272 0.38 9.053 0.612 0.036 -1
0.124 0.187 0.106 0.892 2005
-1 63.643 3.085 78.149 6.410 2.007 0.37 9.013 1.242 0.078 -1
0.212 0.149 0.059 1.554 2006
-1 28.724 3.040 184.021 6.420 1.763 0.70 3.779 1.193 -1
0.051 0.170 0.140 0.065 1.740 2007
-1 71.656 3.574 317.227 4.451 1.979 0.40 4.891 2.612 -1
0.044 0.295 0.160 0.067 0.832 2008
-1 82.680 3.274 209.265 5.618 2.483 0.82 11.351 1.127 -1
0.055 0.245 0.219 0.040 2.692 2009
-1 119.011 3.661 224.738 4.674 2.785 0.41 7.742 2.602 -1
0.035 0.172 0.132 0.066 1.521 2010

-1 89.741 3.091 133.191 4.110 2.577 0.51 4.877 1.430 -1
0.046 0.110 0.131 0.035 1.865 2011
annual scaling factors for observation variance (use this option to scale up the variance for observations based on very little data)
#PC-LL PC-GN.a PC-GN.j GNOP BLLOP SEAMAP-SA Texas VA-LL NMFS-LL-SE SC-GN SCDNR SEAMAP-GOM-ES SEAMAP-GOM-EF UNC MML-GN.a MML-GN.j Year (2007)
#PC-LL ATL Coastspan LL GOM Comb BLL BLLOP SEAMAP-SA GOM Comb GN VA-LL NMFS-LL-SE SC-GN SCDNR-LL-RD Early GA-SCDNR-LL-RD Late SEAMAP GOM ES SEAMAP GOM EF UNC-LL GADNR Trawl Year (now)
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1950
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1951
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1952
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1953
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1954
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1955
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1956
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1957
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1958
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1959
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1960
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1961
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1962
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1963
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1964
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1965
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1966
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1967
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1968
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1969
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1970

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1971
1	1	1	1	1	1	1	1	1	1	1	1	0.482	1	1	1972
1	1	1	1	1	1	1	1	1	1	1	1	0.289	0.769	1	1973
1	1	1	1	1	1	1	1	1	1	1	1	0.238	1.166	1	1974
1	1	1	1	1	1	2.77	1	1	1	1	1	0.252	0.652	1	1975
1	1	1	1	1	1	1	1	1	1	1	1	0.215	0.830	1	1976
1	1	1	1	1	1	2.150	1	1	1	1	1	0.260	0.669	1	1977
1	1	1	1	1	1	1	1	1	1	1	1	0.290	0.470	1	1978
1	1	1	1	1	1	1	1	1	1	1	1	0.274	0.461	1	1979
1	1	1	1	1	1	0.83	1	1	1	1	1	0.262	0.427	1	1980
1	1	1	1	1	1	0.380	1	1	1	1	1	0.259	0.504	1	1981
1	1	1	1	1	1	1	1	1	1	1	0.751	0.271	0.502	1	1982
1	1	1	1	1	1	1.350	1	1	1	1	0.367	0.303	0.357	1	1983
1	1	1	1	1	1	1	1	1	1	1	1.030	0.332	0.362	1	1984
1	1	1	1	1	1	1	1	1	1	1	0.543	0.331	0.394	1	1985
1	1	1	1	1	1	1	1	1	1	1	0.753	0.316	0.600	1	1986
1	1	1	1	1	1	1	1	1	1	1	0.311	0.423	0.411	1	1987
1	1	1	1	1	1	1	1	1	1	1	0.301	0.324	0.347	1	1988
1	1	1	1	0.334	1	1	1	1	1	1	0.338	0.419	0.501	1	1989
1	1	1	1	0.328	1	0.760	1	1	1	1	0.392	0.341	0.355	1	1990
1	1	1	1	0.306	1	0.910	1	1	1	1	0.304	0.356	0.358	1	1991
1	1	1	1	0.319	1	0.560	1	1	1	1	0.268	0.284	0.319	1	1992
0.516	1	1	1	0.349	1	0.73	1	1	1	1	0.324	0.338	0.588	1	1993
0.882	1	1	0.567	0.389	1	1	1	1	1	1	0.418	0.344	0.406	1	1994
0.52	1	1	0.468	0.275	0.67	0.48	0.330	1	1	1	0.238	0.328	0.389	1	1995
0.098	1	1	0.466	0.374	0.42	0.540	0.372	1	1	1	0.243	0.327	0.303	1	1996
0.196	1	1	0.473	0.286	0.35	0.71	0.274	1	1	1	0.302	0.384	0.356	1	1997

```
0.413 1     1     0.448 0.318 0.53  0.4   1     0.795 0.210 1     0.276
      0.452 0.261 1     1998
0.111 1     1     0.449 0.313 0.4   0.460 0.292 1     0.224 1     0.291
      0.435 0.304 1     1999
0.337 0.34  1     0.467 0.291 0.35  0.700 0.231 1.697 0.201 1     0.234
      0.319 0.273 1     2000
1     0.34  1     0.461 0.246 0.35  0.56  0.218 0.730 0.177 1     0.490
      0.364 1     1     2001
1     0.57  1     0.454 0.223 0.34  0.740 0.145 0.492 0.241 1     0.238
      0.351 0.295 1     2002
1     0.27  1     0.451 0.252 0.36  1.170 0.166 0.296 0.189 1     0.338
      0.305 0.283 0.162 2003
1     0.27  0.211 0.461 0.256 0.37  0.760 0.151 1     0.369 1     0.372
      0.402 0.304 0.233 2004
1     0.27  0.203 0.458 0.289 0.46  0.770 0.247 0.608 0.407 1     0.372
      0.418 0.237 0.350 2005
1     0.19  0.140 0.460 0.240 0.38  0.370 0.162 0.525 0.256 1     0.292
      0.389 0.197 0.144 2006
1     0.28  0.118 0.581 0.202 0.33  0.350 0.270 0.438 1     0.107 0.301
      0.363 0.262 0.170 2007
1     0.19  0.131 0.502 0.226 0.33  0.350 0.200 0.372 1     0.108 0.243
      0.282 0.298 0.195 2008
1     0.17  0.147 0.476 0.206 0.31  0.440 0.137 0.708 1     0.117 0.222
      0.305 0.350 0.134 2009
1     0.13  0.122 0.439 0.233 0.3   0.550 0.172 0.462 1     0.125 0.270
      0.306 0.324 0.151 2010
1     0.14  0.146 0.448 0.226 0.32  0.500 0.128 0.422 1     0.121 0.308
      0.350 0.237 0.137 2011
####################################################
#     EFFORT      OBSERVATIONS      If    there are   no    series,
      there should      be    no    entries     between     the   comment
      lines.
####################################################
#     number      of    effort      data  series
0
####################################################
#     AGE   COMPOSITION OBSERVATIONS      If    there are   no    series,
      there should      be    no    entries     between     the   comment
      lines.
####################################################
#     number      of    age-composition   series      (If   there are   no
      series,there      should      be    no    more  entries     in    this
      section)
0

#########################################################################
# PARAMETER INPUT FILE
######################################################################
#=======================================================================
# Total number of process parameters (must match number of entries in
'Specifications 1' section)
#=====================================================================
82
#======================================================================
```

```
# Number of sets of each class of parameters (must be at least 1)
#========================================================================
# q (catchability)
# |     Effort
# |     |     Vulnerability (selectivity)
# |     |     |     catch observation variance scalar
# |     |     |     |     index variance scalar
# |     |     |     |     |     effort variance scalar
# |     |     |     |     |     |
#------------------------------------------------------------------------
 20     5     6     1     1     1
#======================================================================
# Specifications 1: process parameters and observation error parameters
#========================================================================
#  class (nature) of parameter (1=constant, 2-4 = polynomial of degree x,
5=knife edge, 6=logistic, 7=gamma, 8=Chapman-Richards, 10=Bev-Holt,
15=double logistic, 22=age-specific vector )
#  |       best estimate (or central tendency of prior)
#  |       |                lower bound    upper bound
#  |       |                |              |              phase to
estimate (-1 = don't estimate)
#  |       |                |              |              |     prior
density (1= lognormal, 2=normal, 3=uniform)
#  |       |                |              |              |     |
prior variance
#  |       |                |              |              |     |
|
#-----------------------------------------------------------------------
# Natural mortality rate
22    0.232 0.0E+00    1.0E+00    -1    0    -0.2
22    0.232 0.0E+00    1.0E+00    -1    0    -0.2
22    0.232 0.0E+00    1.0E+00    -1    0    -0.2
22    0.232 0.0E+00    1.0E+00    -1    0    -0.2
22    0.232 0.0E+00    1.0E+00    -1    0    -0.2
22    0.232 0.0E+00    1.0E+00    -1    0    -0.2
22    0.232 0.0E+00    1.0E+00    -1    0    -0.2
22    0.232 0.0E+00    1.0E+00    -1    0    -0.2
22    0.232 0.0E+00    1.0E+00    -1    0    -0.2
22    0.232 0.0E+00    1.0E+00    -1    0    -0.2
22    0.232 0.0E+00    1.0E+00    -1    0    -0.2
22    0.232 0.0E+00    1.0E+00    -1    0    -0.2
22    0.232 0.0E+00    1.0E+00    -1    0    -0.2
22    0.232 0.0E+00    1.0E+00    -1    0    -0.2
22    0.232 0.0E+00    1.0E+00    -1    0    -0.2
22    0.232 0.0E+00    1.0E+00    -1    0    -0.2
22    0.232 0.0E+00    1.0E+00    -1    0    -0.2
22    0.232 0.0E+00    1.0E+00    -1    0    -0.2
# Recruitment (10=Beverton/Holt, 11=Ricker)
# 10       5.76E+07        0.1000E+04      1.0000E+10     2     0
-0.7000E+00
 10       6.15E+07        0.1000E+04      1.0000E+10     2     3      -
0.3000E+00
 10       0.760E+00       0.2000E+00      0.9900E+00     3     1      -
0.3000E+00
```

```
# Growth (type 8 = von Bertalanfy/Richards, Linf, K, t0, m, a, b
(weight=al^b)
8     80.2  1.00E-04    1.00E+12    -1    0     1.00E+00
8     0.61  0.00E+00    1.00E+12    -1    0     1.00E+00
8     -0.84 -3            1.00E+12 -1    0     1.00E+00
8     1         0.00E+00     1.00E+12    -1    0     1.00E+00
8     5.56E-06    -1.00E+00   1.00E+12    -1    0     1.00E+00
8     3.074 0.00E+00    1.00E+12    -1    0     1.00E+00
#von bert unit conversion factors (scalar, constant) for L-W
(new_Length_unit = scalar*old_Length_unit + constant)
#    (--***-- FIX First ENTRIES TO 1.0, Second to 0.0 IF NO UNIT
CONVERSION IS NEEDED)
8     1.00E+00    0.00E+00    1.00E+02    -1.00E+00   0     -1.00E+00
8     0.00E+00    -1.00E+00   1.00E+02    -1.00E+00   0     -1.00E+00
# catchability (FOR CATCH SERIES)
  1       1      1.1000E-06       1.1000E+01    -1    0       0.1000E+01

  1       1      1.1000E-06       1.1000E+01    -1    0       0.1000E+01

  1       1      1.1000E-06       1.1000E+01    -1    0       0.1000E+01

  1       1      1.1000E-06       1.1000E+01    -1    0       0.1000E+01

  1       1      1.1000E-06       1.1000E+01    -1    0       0.1000E+01
# catchability (FOR INDEX SERIES)
  1       5.6990E-06       1.1000E-08       0.1000E-04     2     0
0.1000E+01
  1       3.4360E-07       1.1000E-08       0.1000E-04     1     0
0.1000E+01
  1       5.6990E-07       1.1000E-08       0.1000E-04     2     0
0.1000E+01
  1       3.4360E-02       1.1000E-05       0.1000E-01     1     0
0.1000E+01
  1       5.6990E-05       1.1000E-08       0.1000E-03     2     0
0.1000E+01
  1       3.4360E-06       1.1000E-08       0.1000E-04     1     0
0.1000E+01
  1       5.6990E-07       1.1000E-09       0.1000E-05     2     0
0.1000E+01
  # 1       3.4360E-07       1.1000E-08       0.1000E-04     1     0
0.1000E+01
  1       3.4360E-06       1.1000E-08       0.1000E-04     1     0
0.1000E+01
  1       5.6990E-05       1.1000E-08       0.1000E-03     2     0
0.1000E+01
  1       3.4360E-06       1.1000E-08       0.1000E-04     1     0
0.1000E+01
  # 1       5.6990E-06       1.1000E-08       0.1000E-04     2     0
0.1000E+01
  1       5.6990E-08       1.1000E-10       0.1000E-04     2     0
0.1000E+01
  1       3.4360E-06       1.1000E-08       0.1000E-04     1     0
0.1000E+01
```

```
  1       5.6990E-06       1.1000E-08       0.1000E-04     2     0
0.1000E+01
  # 1       3.4360E-07       1.1000E-08       0.1000E-04     1     0
0.1000E+01
  1       3.4360E-05       1.1000E-11       0.1000E-04     1     0
0.1000E+01
  1       5.6990E-07       1.1000E-08       0.1000E-04     2     0
0.1000E+01
# effort for "prehistoric" period when data is sparse (1950-1971)
  1       0.00           -1.0            0.9900E+00     -1     0       -
0.3000E+00
  1       0.00           -1.0             0.2000E+00     -1    0       -
0.3000E+00
  1       0.00           -1.0            0.0100E+00     -1     0       -
0.3000E+00
# 1       0.00           -1.0            0.2000E+00     -1     0       -
0.3000E+00 BLL disc no longer used
  1       0.006            0.0            0.2000E+00     1     0       -
0.3000E+00
  1       0.06            0.0            0.1000E+00      1     0       -
0.200E+00
# effort for period with useful data (1972-2005)
  1       0.05           0.0            0.4000E+00      2     0       -
0.3000E+00
  1       0.12           0.0            0.4000E+00      2     0       -
0.3000E+00
  1       0.08           0.0            0.4000E+00      3     0       -
0.3000E+00
# 1       0.05           0.0            0.4000E+00      2     0       -
0.3000E+00 BLL disc no longer used
  1       0.01           0.0            0.4000E+00      1     0       -
0.3000E+00
  1       0.085           0.0           0.4000E+00      1     0       -
0.3000E+00
# vulnerability (selectivity)
#---S1 Longline age 3
  6       1.323            0.0000E-10       0.2000E+02    -1     0
0.1000E+01
  6       0.598             0.0000E+00       0.400E+02     -4     2
0.6250E-01
#---S2 Longline age 4
  6       2.954             0.0000E-10       0.2000E+02    -1     0
0.1000E+01
  6       0.323            -0.5000E+03       0.4000E+03     -4     2
0.6250E-01
#---S3 Gillnet age 1
  15       1.00           0.0000E-10       0.2000E+02    -1     0
0.1000E+01
  15       12             0.0000E+00       0.4000E+02    -4     2
0.6250E-01
  15       1.5              0.0000E-10       0.2000E+02    -1     0
0.1000E+01
  15       1.0            0.0000E+00       0.4000E+02    -4     2
0.6250E-01
```

```
  15       0.31           0.0000E-10       0.2000E+02    -1     0
0.1000E+01
#---S4 Gillnet age 4
  15       3.0       0.0000E-10       0.2000E+02    -1     0
0.1000E+01
  15       0.5       0.0000E+00       0.4000E+02    -4     2
0.6250E-01
  15       6.0       0.0000E-10       0.2000E+02    -1     0
0.1000E+01
  15       0.8       0.0000E+00       0.4000E+02    -4     2
0.6250E-01
  15       0.81      0.0000E-10       0.2000E+02    -1     0
0.1000E+01
#---S5 Gillnet age 3
  15       2.00      0.0000E-10       0.2000E+02    -1     0
0.1000E+01
  15       0.1       0.0000E+00       0.4000E+02    -4     2
0.6250E-01
  15       5.0       0.0000E-10       0.2000E+02    -1     0
0.1000E+01
  15       0.5       0.0000E+00       0.4000E+02    -4     2
0.6250E-01
  15       0.98      0.0000E-10       0.2000E+02    -1     0
0.1000E+01
#---S6 Longline age 1
  6      0.50             0.0000E-10       0.2000E+02    -1     0
0.1000E+01
  6      0.25            -0.5000E+03       0.4000E+03     -4     2
0.6250E-01
# catch observation error variance scalar
  1      1.0000E+00       0.1000E+00       0.5000E+01    -4     0
0.1000E+01
# 1      1.0000E+00       0.1000E+00       0.5000E+01    4     0
0.1000E+01
# 1      1.0000E+00       0.1000E+00       0.5000E+01    4     0
0.1000E+01
# 1      1.0000E+00       0.1000E+00       0.5000E+01    4     0
0.1000E+01
# 1      3.0000E+00       0.1000E+00       0.5000E+01    4     0
0.1000E+01
# index observation error variance scalar
  1      3.0000E+00       0.1000E+00       15.000E+01    -4     0
0.1000E+01
# effort observation error variance scalar
  1      1.0000E+00       0.1000E+00       0.5000E+01    -5     0
0.1000E+01
#========================================================================
# Specifications 2: process ERROR parameters
#========================================================================
#    best estimate (or central tendency of prior)
#    |                lower bound    upper bound
#    |                |                |              phase to estimate
(-1 = don't estimate)
```

```
#    |                |                |              |     prior density
(1= lognormal, 2=normal, 3=uniform)
#    |                |                |              |     |       prior
variance
#    |                |                |              |     |       |
#-----------------------------------------------------------------------
# overall variance (negative value indicates a CV)
   -0.5000E+00      -5.000E+00      -0.4000E-01     4     0
0.1000E+01
# recruitment process variation parameters (allows year to year
fluctuations)
#    correlation coefficient
    0.5000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
#    variance (should be log-scale variance if prior density = 1 or
arithmetic scale variance if prior = 2)
#      Note: this variance is NOT multiplied by the overall variance
parameter
    -0.25          0.0000E+00       0.1000E+21      -1     1
0.1000E+01
#    annual deviation parameters (last entry is arbitrary for deviations)
    0.0000E+00      -0.5000E+01       0.5000E+01     -4     1      -
0.4000E+00
# catchability process variation parameters (allows year to year
fluctuations)
#    correlation coefficients
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
```

```
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.1000E+21    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.1000E+21    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.1000E+21    -1     0
0.1000E+01
#    variance scalars (multiplied by overall variance)
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.9900E+00    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.1000E+21    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.1000E+21    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.1000E+21    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.1000E+21    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.1000E+21    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.1000E+21    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.1000E+21    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.1000E+21    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.1000E+21    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.1000E+21    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.1000E+21    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.1000E+21    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.1000E+21    -1     0
0.1000E+01
    0.0000E+00      -0.1000E-31       0.1000E+21    -1     0
0.1000E+01
#    annual deviation parameters (last entry is arbitrary for deviations)
```

```
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
    0.0000E+00      -0.5000E+01       0.5000E+01    -1     0
0.1000E+01
# effort process variation parameters (allows year to year fluctuations)
#    correlation coefficients
    0.300E+00       0.0000E+00       0.9900E+00    -1     0
0.1000E+01
    0.300E+00       0.0000E+00       0.9900E+00    -1     0
0.1000E+01
    0.300E+00       0.0000E+00       0.9900E+00    -1     0
0.1000E+01
    0.300E+00       0.0000E+00       0.9900E+00    -1     0
0.1000E+01
    0.300E+00       0.0000E+00       0.9900E+00    -1     0
0.1000E+01
#    variance (should be log-scale variance if prior density = 1 or
arithmetic scale variance if prior = 2)
```

```
#      Note: this variance is NOT multiplied by the overall variance
parameter
    700.2000         0.0000E+00       0.1000E+21    -1     0
0.1000E+01
    700.2000         0.0000E+00       0.1000E+21    -1     0
0.1000E+01
    700.2000         0.0000E+00       0.1000E+21    -1     0
0.1000E+01
    700.2000         0.0000E+00       0.1000E+21    -1     0
0.1000E+01
    700.2000         0.0000E+00       0.1000E+21    -1     0
0.1000E+01
#    annual deviation parameters (last entry is arbitrary for deviations)
    0.000E-03      -8.000E+00       0.8000E+01     2     1
0.1000E+01
    0.000E-03      -8.000E+00       0.8000E+01      2     1
0.1000E+01
    0.000E-03      -8.000E+00       0.8000E+01      2     1
0.1000E+01
    0.000E-03      -8.000E+00       0.8000E+01      2     1
0.1000E+01
    0.000E-03      -8.000E+00       0.8000E+01      2     1
0.1000E+01

/////////////////////////////////////////////////////////////////////////////
DATA_SECTION
/////////////////////////////////////////////////////////////////////////////

 // --------------- read data file ---------------//
 !! ad_comm::change_datafile_name("shark_spasm.dat");

 // general information
  init_ivector year(1,2)                              // first and last
year in analysis
  init_int nyears_deterministic                       // number of years
in the deterministic period (when F and R are constant)
  init_int nyears_proj                 //number of years to project
population (LIZ added 13 Feb 2006)
  init_number pup_start                  //starting point for pup survival
  init_ivector age(1,2)                               // first and last
age in analysis
  init_int nsteps                                     // number of steps
(time periods) in each year
  init_int overall_var_pdf                            // type of overall
variance (1=cv, 2=absolute scale)
  int nyears                                                  // number
of years in the simulation
  int nyears_stochastic                               // number of years
in the stochastic period (when F and R vary interannually)
  int n_eras                                          // number of time
periods when F can vary (nyears_stochastic+1)
  int nages                                           // number of age
classes
```

```
  int nes                                             // (n)umber of
(s)ets of (e) effort parameters
  int nqs                                             // (n)umber of
(s)ets of (q) catchability-related parameters
  int nss                                             // (n)umber of
(s)ets of (s) selectivity-related parameters
  int ncds                                            // (n)umber of
(s)ets of (cd) catch-data-related parameters
  int neds                                            // (n)umber of
(s)ets of (ed) effort-data-related parameters
  int nids                                            // (n)umber of
(s)ets of (id) index-data-related parameters
 !! nyears=year(2)-year(1)+1;
 !! nyears_stochastic=nyears-nyears_deterministic;
 !! n_eras=nyears_stochastic+1;
 !! nages=age(2)-age(1)+1;
 // spawning information
  init_int spawn_season
  init_vector p(1,nages)
  init_vector fecundity_input(1,nages)

 // catch information
 !! cout << "reading catches " << endl;
  init_int n_catch_series
  init_ivector effort_model_type(1,n_catch_series)   // (LIZ 14-feb-2006)
method of treating prehistoric effort: 0 = exact match to effort data, 1
= estimated constant, 2 = estimated linear
  init_ivector catch_pdf(1,n_catch_series)
  init_ivector catch_units(1,n_catch_series)
  init_ivector catch_first(1,n_catch_series)
  init_ivector catch_last(1,n_catch_series)
  init_ivector cvs(1,n_catch_series)
  init_ivector cqs(1,n_catch_series)
  init_ivector css(1,n_catch_series)
  init_ivector ces(1,n_catch_series)
 !! if(n_catch_series<=0) n_catch_series=-1;
  init_matrix catch_obs(1,nyears,1,n_catch_series+1)
  init_matrix catch_cv(1,nyears,1,n_catch_series+1)
 !! if(n_catch_series<=0) n_catch_series=0;

 // index (cpue) information
 !! cout << "reading indices " << endl;
  init_int n_index_series
  init_ivector index_pdf(1,n_index_series)
  init_ivector index_units(1,n_index_series)
  init_ivector index_first(1,n_index_series)
  init_ivector index_last(1,n_index_series)
  init_ivector index_scale(1,n_index_series)
  init_ivector ivs(1,n_index_series)
  init_ivector iqs(1,n_index_series)
  init_ivector iss(1,n_index_series)
 !! if(n_index_series<=0) n_index_series=-1;
  init_matrix index_obs(1,nyears,1,n_index_series+1)
  init_matrix index_cv(1,nyears,1,n_index_series+1)
```

```
 !! if(n_index_series<=0) n_index_series=0;

 // effort information
 !! cout << "reading effort " << endl;
  init_int n_effort_series
  init_ivector effort_pdf(1,n_effort_series)
  init_ivector effort_first(1,n_effort_series)
  init_ivector effort_last(1,n_effort_series)
  init_ivector effort_scale(1,n_effort_series)
  init_ivector evs(1,n_effort_series)
  init_ivector ees(1,n_effort_series)
 !! if(n_effort_series<=0) n_effort_series=-1;
  init_matrix effort_obs(1,nyears,1,n_effort_series+1)
  init_matrix effort_cv(1,nyears,1,n_effort_series+1)
 !! if(n_effort_series<=0) n_effort_series=0;

 // age composition information
 !! cout << "reading age composition " << endl;
  init_int n_agecomp_series
  init_int agecomp_begin_yr                                   // year
when age comp data first become available
  int      nyrs_agecomp
 !! nyrs_agecomp=year(2)-agecomp_begin_yr+1;
  init_ivector agecomp_pdf(1,n_agecomp_series)
  init_ivector agecomp_units(1,n_agecomp_series)
  init_ivector agecomp_first(1,n_agecomp_series)
  init_ivector agecomp_last(1,n_agecomp_series)
  init_matrix  agecomp_input(1,nyrs_agecomp*n_agecomp_series,1,nages+3)
// age composition data

 // --------------- read parameter file ---------------//
 !! ad_comm::change_datafile_name("shark_spasm.prm");
 !! cout << "reading parameter specifications " << endl;

  init_int n_par                         // number of process parameters
  init_ivector n_sets(1,6)                                // number of
sets of each parameter type
 !! nqs=n_sets(1); nes=n_sets(2); nss=n_sets(3); ncds=n_sets(4);
nids=n_sets(5); neds=n_sets(6);
  init_matrix par_specs(1,n_par,1,7)                 // specifications
for structural parameters
  init_vector o_var_specs(1,6)                            //
specifications for overall scale of variance
  init_vector r_rho_specs(1,6)                       // specifications
for r process error correlation coefficient
  init_vector r_var_specs(1,6)                       // specifications
for r process error relative variance
  init_vector r_dev_specs(1,6)                       // specifications
for r process error deviations
  init_matrix q_rho_specs(1,nqs,1,6)                 // specifications
for q process error correlation coefficient
  init_matrix q_var_specs(1,nqs,1,6)                 // specifications
for q process error relative variance
```

```
  init_matrix q_dev_specs(1,nqs,1,6)                // specifications
for q process error deviations
  init_matrix e_rho_specs(1,nes,1,6)                // specifications
for e process error correlation coefficient
  init_matrix e_var_specs(1,nes,1,6)                // specifications
for e process error relative variance
  init_matrix e_dev_specs(1,nes,1,6)                // specifications
for e process error deviations

 // --------- derived variables pertaining to parameters that are
constant (don't need to be differentiated)----------//

  int i; int ie; int n_series; int n_par_phase; int k;
  number delta; number half_delta; number spawn_time; vector
step_time(1,nsteps)
  vector ag(1,nages)
  ivector n_calls(1,1000)
  ivector npf(1,50); ivector nature(1,n_par);
  vector  best_guess(1,n_par);  number  o_var_best_guess;
  number  r_rho_best_guess       ;  number  r_var_best_guess       ;
number  r_dev_best_guess       ;
  vector  q_rho_best_guess(1,nqs);  vector  q_var_best_guess(1,nqs);
vector  q_dev_best_guess(1,nqs);
  vector  e_rho_best_guess(1,nes);  vector  e_var_best_guess(1,nes);
vector  e_dev_best_guess(1,nes);
  ivector iph(1,n_par);      int     o_var_iph;
  int     r_rho_iph;         int     r_var_iph;         int
r_dev_iph;
  ivector q_rho_iph(1,nqs);  ivector q_var_iph(1,nqs);  ivector
q_dev_iph(1,nqs);
  ivector e_rho_iph(1,nes);  ivector e_var_iph(1,nes);  ivector
e_dev_iph(1,nes);
  ivector pdf(1,n_par);      int     o_var_pdf;
  int     r_rho_pdf;         int     r_var_pdf;         int
r_dev_pdf;
  ivector q_rho_pdf(1,nqs);  ivector q_var_pdf(1,nqs);  ivector
q_dev_pdf(1,nqs);
  ivector e_rho_pdf(1,nes);  ivector e_var_pdf(1,nes);  ivector
e_dev_pdf(1,nes);
  vector  cv(1,n_par);       number  o_var_cv;
  number  r_rho_cv;          number  r_var_cv;          number  r_dev_cv;
  vector  q_rho_cv(1,nqs);   vector  q_var_cv(1,nqs);   vector
q_dev_cv(1,nqs);
  vector  e_rho_cv(1,nes);   vector  e_var_cv(1,nes);   vector
e_dev_cv(1,nes);
  ivector iqv(1,nqs);  ivector iev(1,nes);  ivector isv(1,nss)
  number  F_best_guess;
  int     last_iph;


 LOCAL_CALCS
  // reformat parameter control matrices
```

```
  best_guess=column(par_specs,2);  iph=ivector(column(par_specs,5));
pdf=ivector(column(par_specs,6)); cv=column(par_specs,7);
nature=ivector(column(par_specs,1));
  o_var_best_guess=o_var_specs(1); o_var_iph=int(o_var_specs(4));
o_var_pdf=int(o_var_specs(5));    o_var_cv=o_var_specs(6);
  r_rho_best_guess=r_rho_specs(1); r_rho_iph=int(r_rho_specs(4));
r_rho_pdf=int(r_rho_specs(5));    r_rho_cv=r_rho_specs(6);
  r_var_best_guess=r_var_specs(1); r_var_iph=int(r_var_specs(4));
r_var_pdf=int(r_var_specs(5));    r_var_cv=r_var_specs(6);
  r_dev_best_guess=r_dev_specs(1); r_dev_iph=int(r_dev_specs(4));
r_dev_pdf=int(r_dev_specs(5));    r_dev_cv=r_dev_specs(6);
  q_rho_best_guess=column(q_rho_specs,1);
q_rho_iph=ivector(column(q_rho_specs,4));
q_rho_pdf=ivector(column(q_rho_specs,5));
q_rho_cv=column(q_rho_specs,6);
  q_var_best_guess=column(q_var_specs,1);
q_var_iph=ivector(column(q_var_specs,4));
q_var_pdf=ivector(column(q_var_specs,5));
q_var_cv=column(q_var_specs,6);
  q_dev_best_guess=column(q_dev_specs,1);
q_dev_iph=ivector(column(q_dev_specs,4));
q_dev_pdf=ivector(column(q_dev_specs,5));
q_dev_cv=column(q_dev_specs,6);
  e_rho_best_guess=column(e_rho_specs,1);
e_rho_iph=ivector(column(e_rho_specs,4));
e_rho_pdf=ivector(column(e_rho_specs,5));
e_rho_cv=column(e_rho_specs,6);
  e_var_best_guess=column(e_var_specs,1);
e_var_iph=ivector(column(e_var_specs,4));
e_var_pdf=ivector(column(e_var_specs,5));
e_var_cv=column(e_var_specs,6);
  e_dev_best_guess=column(e_dev_specs,1);
e_dev_iph=ivector(column(e_dev_specs,4));
e_dev_pdf=ivector(column(e_dev_specs,5));
e_dev_cv=column(e_dev_specs,6);
  // initialize number of parameters in each function type
  npf=1; for (int j=1; j<=4;j++) npf(j)=j; // constants and polynomials
  npf(5)=1; npf(6)=2; npf(7)=2; // knife-edge, logistic and gamma
selectivity curves
  npf(8)=8; npf(9)=3; // Chapman-Richards and Gompertz growth curves;
PLUS: 2 parameters to re-scale units if necessary (LIZ added 1/31/2006)
  npf(10)=2; // Beverton and Holt asymptotic recruitment
  npf(11)=2; // Ricker spawner-recruit
  npf(12)=2; // power
  npf(15)=5; // double logistic (LIZ added 8/18/2005)
  npf(16)=2; // exponential (LIZ added 4/25/2005)
  npf(22)=nages; //allows age-specific values
  delta=1./double(nsteps); half_delta=0.5*delta;
spawn_time=double(spawn_season-1)*delta;
  for (ie=1; ie<=nsteps; ie++) step_time(ie)=double(ie)*delta-half_delta;
  for (a=1; a<=nages; a++) ag(a)=double(a+age(1))-1.0;
   //cout << "Best Guess..." << endl;
   //cout << best_guess << endl;
  F_best_guess=0.05;
```

```
  last_iph=max(iph);
 END_CALCS

 // --------- derived variables pertaining to the data that are constant
(don't need to be differentiated)----------//

  matrix  n_agecomp_data(1,nyears,1,n_agecomp_series)            //
number of fish sampled for age composition
  3darray agecomp_obs(1,nages,1,nyears,1,n_agecomp_series)      // age
composition data
  vector  catch_delta(1,n_catch_series)
  vector  index_delta(1,n_index_series)
  vector  effort_avg(1,n_effort_series+1)
  vector  effort_min(1,n_effort_series+1)
  vector  n_effort_points(1,n_effort_series+1)
  vector  index_avg(1,n_index_series+1)
  vector  index_min(1,n_index_series+1)
  vector  n_index_points(1,n_index_series+1)
  vector  one_vector_age(1,nages)
  number  aic
  number  catch_max
  number  catch_min
  number  temp_dble
  number  n_data
  number  sumcomp

 LOCAL_CALCS
  // compute maximum total catch and averages (initial biomass ought to
be near the maximum catch divided by F(y=1)
  cout << "Averaging data" << endl;
  zero=0.0; one=1.0; n_calls=0; i_one=1; i_two=2; one_vector_age=one;
tiny_number=1.0e-32; huge_number=1.0e+32; two_pi=6.2831853;
  n_effort_points=0.0; n_index_points=0.0; effort_avg=0.0 ;
index_avg=0.0; catch_max=1.0; catch_min=10.0; index_min=1000.0;
effort_min=1000.0;
  for (y=1; y<=nyears;y++) {
    // compute maximum and minimum total catch
    temp_dble=0.0;
    for (series=1; series<=n_catch_series;series++) {
      if(y==1) catch_delta(series)=1/double(catch_last(series)-
catch_first(series)+1);
      if(catch_pdf(series)>0 && catch_obs(y,series)>0.0) {
        temp_dble+=catch_obs(y,series);
        if(catch_obs(y,series)<catch_min) catch_min=catch_obs(y,series);
      }
    }
    if(temp_dble>catch_max) catch_max=temp_dble;
    // compute average effort and average index
    for (series=1; series<=n_effort_series;series++)
      if(effort_obs(y,series)>=0.0) {
        if(effort_obs(y,series)>0.0 &&
effort_obs(y,series)<effort_min(series))
effort_min(series)=effort_obs(y,series);
```

```
        effort_avg(series) += effort_obs(y,series);
n_effort_points(series) += 1.0;
      }
    for (series=1; series<=n_index_series;series++) {
      if(y==1) index_delta(series)=one/double(index_last(series)-
index_first(series)+1);
      if(index_obs(y,series)>=0) {
        if(index_obs(y,series)>0.0 &&
index_obs(y,series)<index_min(series))
index_min(series)=index_obs(y,series);
        index_avg(series) += index_obs(y,series);
n_index_points(series) += 1.0;
      }
    }
  }
  //scale index and effort series
   cout << "Scaling" << endl;
   n_data=sum(n_index_points)+sum(n_effort_points);
   for (series=1; series<=n_index_series;series++)  { index_avg(series)
/= n_index_points(series)   ; index_min(series) /= 1000.0 ; } // so q ~
C/N and e~1
   for (series=1; series<=n_effort_series;series++) { effort_avg(series)
/= n_effort_points(series) ; effort_min(series) /= 1000.0; } // so e~1
and q ~ C/N


   for (y=1; y<=nyears;y++) {
     for (series=1; series<=n_effort_series;series++) {
       if(effort_pdf(series)==1 && effort_obs(y,series)>=0)
effort_obs(y,series)+=effort_min(series); // no zero effort for lognormal
       if(effort_scale(series)>0) effort_obs(y,series) /=
effort_avg(series);
     }

     for (series=1; series<=n_index_series;series++) {
       if(index_pdf(series)==1 && index_obs(y,series)>=0)
index_obs(y,series)+=index_min(series); // no zero indices for lognormal
       if(index_scale(series)>0)  index_obs(y,series) /=
index_avg(series)/catch_max;
     }

     for (series=1; series<=n_catch_series;series++) {
       if(catch_pdf(series)>=0 && catch_obs(y,series)>=0) {
        n_data += 1; if(catch_obs(y,series)<catch_min &&
catch_pdf(series)==1) catch_obs(y,series)=catch_min/10.0; // no zero
catches permitted for lognormal
       }
     }
   }
   catch_min=catch_min/100000.0+1.0e-10;
   n_series=n_index_series; if(n_catch_series>n_series)
n_series=n_catch_series;

  //format age composition data
```

```
   n_agecomp_data.initialize();
   for (y=1; y<=nyrs_agecomp; y++) {
    j=agecomp_begin_yr+y-year(1);
    for (i=1; i<=n_agecomp_series; i++) {
      k=(i-1)*nyrs_agecomp+y; n_agecomp_data(j,i)=agecomp_input(k,3);
sumcomp=0;
      for (a=1; a<=nages; a++) {
        if(agecomp_input(k,a+3)>=0) sumcomp+=agecomp_input(k,a+3);
        else if(n_agecomp_data(j,i)>0) {
          cout << "Error: There is a negative value entered in the age
composition data for " << endl;
          cout << "       series " << i << ", year " << j+year(1)-1 <<
endl; exit(0);
        }
      }
      if(sumcomp>0) for (a=1; a<=nages; a++)
agecomp_obs(a,j,i)=agecomp_input(k,a+3)/sumcomp;
      else          n_agecomp_data(j,i)=0;
    }
   }
 END_CALCS

/////////////////////////////////////////////////////////////////////////
PARAMETER_SECTION
 // Warning: all variables in this section must be floating point, not
integers
 //          integers may be declared locally by use of !! int i
etc..., but these will
 //          not apply outside the parameter section (whereas the ADMB
types number, vector
 //          and matrix are global)

/////////////////////////////////////////////////////////////////////////

 // --------- specify estimated parameters -----------------------------
-//

 // get parameter bounds and phases in proper formats
 LOCAL_CALCS
   cout << "specifying parameter bounds " << endl;
   dvector lb(1,n_par); lb=column(par_specs,3); dvector ub(1,n_par);
ub=column(par_specs,4);
   double lb_o_var; lb_o_var=o_var_specs(2); double ub_o_var;
ub_o_var=o_var_specs(3);
   double lb_r_rho; lb_r_rho=r_rho_specs(2); double ub_r_rho;
ub_r_rho=r_rho_specs(3);
   double lb_r_var; lb_r_var=r_var_specs(2); double ub_r_var;
ub_r_var=r_var_specs(3);
   double lb_r;     lb_r=r_dev_specs(2);     double ub_r;
ub_r=r_dev_specs(3);
   dvector lb_q_rho(1,nqs); lb_q_rho=column(q_rho_specs,2); dvector
ub_q_rho(1,nqs); ub_q_rho=column(q_rho_specs,3);
   dvector lb_q_var(1,nqs); lb_q_var=column(q_var_specs,2); dvector
ub_q_var(1,nqs); ub_q_var=column(q_var_specs,3);
```

```
   dvector lb_q(1,nqs);     lb_q=column(q_dev_specs,2);     dvector
ub_q(1,nqs);     ub_q=column(q_dev_specs,3);
   dvector lb_e_rho(1,nes); lb_e_rho=column(e_rho_specs,2); dvector
ub_e_rho(1,nes); ub_e_rho=column(e_rho_specs,3);
   dvector lb_e_var(1,nes); lb_e_var=column(e_var_specs,2); dvector
ub_e_var(1,nes); ub_e_var=column(e_var_specs,3);
   dvector lb_e(1,nes);     lb_e=column(e_dev_specs,2);     dvector
ub_e(1,nes);     ub_e=column(e_dev_specs,3);
   double lb_0;     lb_0=0.0001;            double ub_2;     ub_2=2.0;
 END_CALCS

 // set parameter vectors to be estimated
 !! cout << "specifying parameters " << endl;
  init_bounded_number_vector par_est(1,n_par,lb,ub,iph)
  init_bounded_number overall_var(lb_o_var,ub_o_var,o_var_iph)
  init_bounded_number r_rho(lb_r_rho,ub_r_rho,r_rho_iph)
  init_bounded_number r_var(lb_r_var,ub_r_var,r_var_iph)
  init_bounded_vector r_devs(2,n_eras,lb_r,ub_r,r_dev_iph)
  init_bounded_number_vector q_rho(1,nqs,lb_q_rho,ub_q_rho,q_rho_iph)
  init_bounded_number_vector q_var(1,nqs,lb_q_var,ub_q_var,q_var_iph)
  init_bounded_vector_vector q_devs(1,nqs,2,n_eras,lb_q,ub_q,q_dev_iph)
  init_bounded_number_vector e_rho(1,nes,lb_e_rho,ub_e_rho,e_rho_iph)
  init_bounded_number_vector e_var(1,nes,lb_e_var,ub_e_var,e_var_iph)
  init_bounded_vector_vector e_devs(1,nes,2,n_eras,lb_e,ub_e,e_dev_iph)

//  init_bounded_number Fspr20(lb_0,ub_2,last_iph)
//  init_bounded_number Fspr30(lb_0,ub_2,last_iph)
//  init_bounded_number Fspr40(lb_0,ub_2,last_iph)
//  init_bounded_number Fspr50(lb_0,ub_2,last_iph)
//  init_bounded_number Fspr60(lb_0,ub_2,last_iph)
//
 // --------- derived variables that are functions of the parameters and
therefore need derivatives ----------//

 // state variables
  vector r(1,nyears)
  matrix q(1,nyears,1,nqs);  matrix e(1,nyears,1,nes)

 // state (process) expectations (deterministic part)
  vector m(1,nages)
  vector exp_m(1,nages)
  vector fecundity(1,nages)
  matrix s(1,nages,1,nss)

 // observation error parameters
  vector c_d_var(1,ncds);  vector e_d_var(1,neds);  vector
i_d_var(1,nids)

 // likelihoods and priors
  vector catch_lklhd(1,n_catch_series);  vector
index_lklhd(1,n_index_series+1);  vector
effort_lklhd(1,n_effort_series+1); vector
agecomp_lklhd(1,n_agecomp_series+1)
  number r_lklhd
```

```
  vector q_lklhd(1,nqs);  vector e_lklhd(1,nes)
  number m_prior;  number r_prior;  number w_prior
  vector q_prior(1,nqs);  vector e_prior(1,nes);  vector s_prior(1,nss)
  vector c_d_prior(1,ncds);  vector i_d_prior(1,nids);  vector
e_d_prior(1,neds)
  number e_process_prior;  number r_process_prior;  number
q_process_prior
  number v_prior
  number f_penalty
  number n_penalty
  number plusage_penalty


 // misc. temporary variables
  number pred;  number var;  number spr0;  number sprphi;  number
survive;  number plus_age;  number catch_by_age;  number index_by_age;
number avg_F;
  vector function_parameter(1,6);  vector recruitment_parameter(1,6);
  vector growth_parameter(1,8);  //(LIZ 13 feb 2006); changed dim from 6
to 8
  vector s_latest(1,nages);  vector s_equilibrium(1,nages); vector
wbyage(1,nages)
  matrix total_catch(1,nages,1,nyears);  matrix
total_yield(1,nages,1,nyears)
  matrix average_n(1,nages,1,nyears);
  matrix catch_pred(1,nyears,1,n_catch_series);  matrix
index_pred(1,nyears,1,n_index_series+1);
  matrix effort_pred(1,nyears,1,n_effort_series+1)
  3darray agecomp_pred(1,nages,1,nyears,1,n_agecomp_series)
  vector ssb(1,nyears)
  //3darray f(1,nages,1,n_eras,1,n_catch_series)
  //3darray f_index(1,nages,1,n_eras,1,n_index_series)
  3darray f(1,nages,1,nyears,1,n_catch_series)               //changed
n_eras to nyears (LIZ 8/18/2005)
  3darray f_index(1,nages,1,nyears,1,n_index_series)         //changed
n_eras to nyears (LIZ 8/18/2005)
  3darray n(1,nages+1,1,nyears+1,1,nsteps+1)
  vector n_last(1,nages)
  vector w_last(1,nages)
  vector n_virg(1,nages)
  vector w_virg(1,nages)
  matrix  w(1,nages+1,1,nsteps)
  objective_function_value obj_func;

 // equilibrium statistics
  number slope0; number spratio; number sprtemp; number sprold; number
yprtemp; number yprold; number ytemp; number yold
  number spr20;  number spr30;  number spr40;  number spr50;  number
spr60;  number spr01; number sprmax; //number sprmsy;  //spawning
potential ratio
  number ypr20;  number ypr30;  number ypr40;  number ypr50;  number
ypr60;  number ypr01; number yprmax; number yprmsy;  // yield per recruit
  number Rspr20; number Rspr30; number Rspr40; number Rspr50; number
Rspr60; number R01;   number Rmax;   number Rmsy;    // recruitment
```

```
  number Fspr20; number Fspr30; number Fspr40; number Fspr50; number
Fspr60;
  number F01;   number Fmax;   //number Fmsy;    // fishing mortality
  number Yspr20; number Yspr30; number Yspr40; number Yspr50; number
Yspr60; number Y01;   number Ymax;   number Ymsy;    // yield
  number Bspr20; number Bspr30; number Bspr40; number Bspr50; number
Bspr60; number B01;   number Bmax;   number Bmsy;    // spawning biomass
(fecundity)
  number BoverBspr20;  number BoverBspr30;  number BoverBspr40;  number
BoverBspr50;  number BoverBspr60;  number BoverB01;  number BoverBmax;
//number BoverBmsy;
  number FoverFspr20;  number FoverFspr30;  number FoverFspr40;  number
FoverFspr50;  number FoverFspr60;  number FoverF01;  number FoverFmax;
//number FoverFmsy;

 // standard deviation report variables
//  sdreport_number r0
//  sdreport_number Bcurrent
//  sdreport_number Fcurrent
  //sdreport_number pup_survival
  number steepness
  number alpha
  sdreport_number Bvirgin
  sdreport_vector B(1,nyears)
  sdreport_vector BoverBvirgin(1,nyears)

 // likeprof variables
  likeprof_number r0
  likeprof_number pup_survival
  likeprof_number Bcurrent             //SSB for current year
  likeprof_number Nmatcurrent            //virgin SS biomass
  likeprof_number Fcurrent
  likeprof_number B0           //virgin total biomass
  likeprof_number SSB0           //virgin SS biomass
  likeprof_number Nmat0           //virgin SS biomass
  likeprof_number Btot           //total biomass for current year
  likeprof_number Bdepletion            //B/B0 for current year
  likeprof_number SSBdepletion            //SSB/SSB0 for current year
  likeprof_number Nmatdepletion            //Nmat/Nmat0 for current year
  likeprof_number inflection;
  likeprof_number sprmsy;
  likeprof_number Fmsy;
  likeprof_number BoverBmsy
  likeprof_number FoverFmsy
  //likeprof_number Bvirgin
  //likeprof_number SSBref             //SSB/SSBmsy for current year

 !! cout << "initialize parameters" << endl;
  !! cout << best_guess << endl;

/////////////////////////////////////////////////////////////////////////
INITIALIZATION_SECTION
/////////////////////////////////////////////////////////////////////////
   par_est best_guess
```

```
   overall_var o_var_best_guess
   r_rho  r_rho_best_guess
   r_var  r_var_best_guess
   r_devs r_dev_best_guess
   q_rho  q_rho_best_guess
   q_var  q_var_best_guess
   q_devs q_dev_best_guess
   e_rho  e_rho_best_guess
   e_var  e_var_best_guess
   e_devs e_dev_best_guess
//   Fspr20  F_best_guess
//   Fspr30  F_best_guess
//   Fspr40  F_best_guess
//   Fspr50  F_best_guess
//   Fspr60  F_best_guess
//
/////////////////////////////////////////////////////////////////////////
PROCEDURE_SECTION
/////////////////////////////////////////////////////////////////////////
  define_parameters();
  calculate_biomass_and_predicted_catch();
  calculate_the_objective_function();

/////////////////////////////////////////////////////////////////////////
 // FUNCTION SECTION
 //  Warning: ADMB FUNCTIONS are unpredictable when they call other ADMB
FUNCTIONS.
 //           It is safer to simply write global functions in C++ (in the
GLOBALS_SECTION)
 //           and call these if you wish to nest the routines.

/////////////////////////////////////////////////////////////////////////

 //-------------------------------------------------------------------
FUNCTION define_parameters
 //--------------------------------------------------------------------
  int j, y, inow;
  current_ph=current_phase();
  n_calls(current_ph) += 1;

  //-------------compute expectations of state variables----------------
//
    if(n_calls(1)==1) cout << "expectations of state variables" << endl;

  i=1;
  // expected natural mortality rate by age
    if(n_calls(1)==1) cout << "     natural mortality" << endl;
    inow=i; m_prior=0.;

    //code for nature=22
      if(nature(inow)==22)  {
  //cout << "inside 22 loop " << endl;
        for ( j=1; j<=npf(nature(inow)); j++) {
         m(j) = best_guess(i);
```

```
         exp_m(j)=mfexp(-m(j)*half_delta);
  //cout << "m(j) " << m(j) << endl;
         i=i+1;
         } //end for-loop
       } //end if

      else {
    for ( j=1; j<=npf(nature(inow)); j++) {
      function_parameter(j)=par_est(i);
      if(pdf(i)>0 && iph(i)>0 && iph(i)<=current_ph)
m_prior+=neg_log_prior(function_parameter(j),best_guess(i),par_specs(i,3)
,par_specs(i,4),cv(i),pdf(i));
      i=i+1;
    }
    for ( a=1; a<=nages; a++) {
      m(a)=function_value(nature(i-
1),function_parameter,double(age(1)+a)-1);
      exp_m(a)=mfexp(-m(a)*half_delta);
    }
      }  //end else statement
      //cout << "i, inow  " <<  i << " , " << inow << endl;

  // expected recruitment parameters
    if(n_calls(1)==1) cout << "     recruitment" << endl;
    inow=i; r_prior=0.; irn=i;
    for ( j=1; j<=npf(nature(inow)); j++) {
      recruitment_parameter(j)=par_est(i);
      if(pdf(i)>0 && iph(i)>0 && iph(i)<=current_ph)
r_prior+=neg_log_prior(recruitment_parameter(j),best_guess(i),par_specs(i
,3),par_specs(i,4),cv(i),pdf(i));
      i=i+1;
    }
      if(n_calls(1)==1) {
       if(pup_start>0)  recruitment_parameter(2)=pup_start;  //(LIZ 13
feb 2006)
      }

  // expected growth/fecundity parameters
    if(n_calls(1)==1) cout << "     growth" << endl;
    inow=i; w_prior=0.; iwn=i;
    for ( j=1; j<=npf(nature(inow)); j++) {
      growth_parameter(j)=par_est(i);

      if(pdf(i)>0 && iph(i)>0 && iph(i)<=current_ph)
w_prior+=neg_log_prior(growth_parameter(j),best_guess(i),par_specs(i,3),p
ar_specs(i,4),cv(i),pdf(i));
      i=i+1;
    }

    for ( a=1; a<=nages-1; a++) {
      if(fecundity_input(a)>=0) fecundity(a)=fecundity_input(a); else
fecundity(a)=function_value(nature(i-
1),growth_parameter,ag(a)+spawn_time);
```

```
      for ( j=1; j<=nsteps; j++)
w(a,j)=function_value(nature(iwn),growth_parameter,ag(a)+step_time(j));
    }

    if(m(nages)>0) plus_age=age(2)+mfexp(-m(nages))/(1-mfexp(-m(nages)));
else plus_age=2*age(2);
    if(fecundity_input(nages)>=0)
fecundity(nages)=fecundity_input(nages); else
fecundity(nages)=function_value(nature(i-
1),growth_parameter,plus_age+spawn_time);


  // virgin spawner-per recruit
    spr0=spr(p,fecundity,m,one_vector_age,zero,spawn_time,nages);

  // expected q
    if(n_calls(1)==1) cout << "     catchability" << endl;
    q_prior=0.;
    for (set=1; set<=nqs; set++) {
      inow=i;
      for ( j=1; j<=npf(nature(inow)); j++) {
        function_parameter(j)=par_est(i);
        if(pdf(i)>0 && iph(i)>0 && iph(i)<=current_ph) q_prior(set) +=
neg_log_prior(function_parameter(j),best_guess(i),par_specs(i,3),par_spec
s(i,4),cv(i),pdf(i));
        i=i+1;
      }
      for ( y=1; y<=nyears; y++) q(y,set)=function_value(nature(i-
1),function_parameter,one);
    }

  // expected effort
    if(n_calls(1)==1) cout << "     effort" << endl;
    e_prior=0.;
    for (set=1; set<=nes; set++) {
      e(1,set)=par_est(i);
      if(pdf(i)>0 && iph(i)>0 && iph(i)<=current_ph) e_prior(set) +=
neg_log_prior(e(1,set),best_guess(i),par_specs(i,3),par_specs(i,4),cv(i),
pdf(i));
      i=i+1;
    }
    for (set=1; set<=nes; set++) {
      inow=i;
      for ( j=1; j<=npf(nature(inow)); j++) {
        function_parameter(j)=par_est(i);
        if(pdf(i)>0 && iph(i)>0 && iph(i)<=current_ph) e_prior(set) +=
neg_log_prior(function_parameter(j),best_guess(i),par_specs(i,3),par_spec
s(i,4),cv(i),pdf(i));
        i=i+1;
      }
      for ( y=nyears_deterministic+1; y<=nyears; y++)
e(y,set)=function_value(nature(i-1),function_parameter,double( (y-
nyears_deterministic-1)/nyears_stochastic ) );
    }
```

```
  // expected selectivity/vulnerability
    if(n_calls(1)==1) cout << "     vulnerability" << endl;
    s_prior=0.;
    for (set=1; set<=nss; set++) {
      inow=i;
      for ( j=1; j<=npf(nature(inow)); j++) {
        function_parameter(j)=par_est(i);
        if(pdf(i)>0 && iph(i)>0 && iph(i)<=current_ph) s_prior(set) +=
neg_log_prior(function_parameter(j),best_guess(i),par_specs(i,3),par_spec
s(i,4),cv(i),pdf(i));
        i=i+1;
      }
      for ( a=1; a<=nages; a++) s(a,set)=function_value(nature(i-
1),function_parameter,double(age(1)+a-1));
    }

    //cout << s << endl;

  //-------------expected relative observation variances---------//

    if(n_calls(1)==1) cout << "     observation variances" << endl;
    c_d_prior=0.;
    for (set=1; set<=ncds; set++) {
      c_d_var(set)=par_est(i);
      if(pdf(i)>0 && iph(i)>0 && iph(i)<=current_ph) c_d_prior(set) +=
neg_log_prior(c_d_var(set),best_guess(i),par_specs(i,3),par_specs(i,4),cv
(i),pdf(i));
      i=i+1;
    }

    i_d_prior=0.;
    for (set=1; set<=nids; set++) {
      i_d_var(set)=par_est(i);
      if(pdf(i)>0 && iph(i)>0 && iph(i)<=current_ph) i_d_prior(set) +=
neg_log_prior(i_d_var(set),best_guess(i),par_specs(i,3),par_specs(i,4),cv
(i),pdf(i));
      i=i+1;
    }

    e_d_prior=0.;
    for (set=1; set<=neds; set++) {
      e_d_var(set)=par_est(i);
      if(pdf(i)>0 && iph(i)>0 && iph(i)<=current_ph) e_d_prior(set) +=
neg_log_prior(e_d_var(set),best_guess(i),par_specs(i,3),par_specs(i,4),cv
(i),pdf(i));
      i=i+1;
    }

  //-------------overall scale of variance----------------//

    if(active(overall_var) && o_var_pdf>0)
v_prior=neg_log_prior(overall_var,o_var_best_guess,o_var_specs(2),o_var_s
pecs(3),o_var_cv,o_var_pdf);
```

```
  //-------------incorporate process errors----------------//

    if(n_calls(1)==1) cout << "priors for recruitment process parameters"
<< endl;
    r_process_prior=zero;
    if(active(r_rho) && r_rho_pdf>0)
r_process_prior+=neg_log_prior(r_rho,r_rho_best_guess,r_rho_specs(2),r_rh
o_specs(3),r_rho_cv,r_rho_pdf);
    if(active(r_var) && r_var_pdf>0)
r_process_prior+=neg_log_prior(r_var,r_var_best_guess,r_var_specs(2),r_va
r_specs(3),r_var_cv,r_var_pdf);

    if(n_calls(1)==1) cout << "priors for q process parameters" << endl;
    q_process_prior=zero;
    for (set=1; set<=nqs; set++) {
      if(active(q_rho(set)) && q_rho_pdf(set)>0)
q_process_prior+=neg_log_prior(q_rho(set),q_rho_best_guess(set),q_rho_spe
cs(set,2),q_rho_specs(set,3),q_rho_cv(set),q_rho_pdf(set));
      if(active(q_var(set)) && q_var_pdf(set)>0)
q_process_prior+=neg_log_prior(q_var(set),q_var_best_guess(set),q_var_spe
cs(set,2),q_var_specs(set,3),q_var_cv(set),q_var_pdf(set));
    }

    if(n_calls(1)==1) cout << "priors for effort process parameters" <<
endl;
    e_process_prior=zero;
    for (set=1; set<=nes; set++) {
      if(active(e_rho(set)) && e_rho_pdf(set)>0)
e_process_prior+=neg_log_prior(e_rho(set),e_rho_best_guess(set),e_rho_spe
cs(set,2),e_rho_specs(set,3),e_rho_cv(set),e_rho_pdf(set));
      if(active(e_var(set)) && e_var_pdf(set)>0)
e_process_prior+=neg_log_prior(e_var(set),e_var_best_guess(set),e_var_spe
cs(set,2),e_var_specs(set,3),e_var_cv(set),e_var_pdf(set));
    }

    if(n_calls(1)==1) cout << "catchability deviations" << endl;
    for (set=1; set<=nqs; set++) {
      if(q_dev_iph(set)>0 && q_dev_iph(set)<=current_ph) {
        //for (y=2; y<=n_eras; y++) {
              //I changed 2 to nyears_deterministic+1 and n_eras to
nyears in the y-loop (LIZ 8/18/2005)
        for (y=nyears_deterministic+1; y<=nyears; y++) {
          if(q_dev_pdf(set)==1) q(y,set)=q(y,set)*mfexp(q_devs(set,y));
else q(y,set)=q(y,set)+q_devs(set,y);
        }
      }
    }

    if(n_calls(1)==1) cout << "effort deviations" << endl;
    for (set=1; set<=nes; set++) {
      if(e_dev_iph(set)>0 && e_dev_iph(set)<=current_ph) {
        for (y=nyears_deterministic+1; y<=nyears; y++) {
          t=y-nyears_deterministic+1;
```

```
          if(e_dev_pdf(set)==1) e(y,set)=e(y,set)*mfexp(e_devs(set,t));
else e(y,set)=e(y,set)+e_devs(set,t);
        }
      }
      //LIZ 14-feb-2006: making effort-type fleet specific;
    //for (a=1; a<=n_catch_series; a++)  {

    //cout << "catch series " << a << "   effort type  " <<
effort_model_type(a) << endl;

      for ( y=1; y<=nyears_deterministic; y++) {
        if(effort_model_type(set)<=0)      e(y,set) = effort_obs(y,set);
        else if(effort_model_type(set)==1) e(y,set) = e(1,set) ;
        else                          e(y,set) = e(1,set) + (
e(nyears_deterministic+1,set) - e(1,set) )*(y-1)/nyears_deterministic;
      }
     //}  //end loop on catch series
    }  //end loop on effort sets

    //cout << e << endl;

 //-----------------------------------------------------------------------
FUNCTION calculate_biomass_and_predicted_catch
 // Integrate the population dynamics over n time steps per year
 //-----------------------------------------------------------------------

  catch_pred=0.0; index_pred=0.0 ; ssb=0; agecomp_pred.initialize();
plusage_penalty=0; n_penalty=0;

n_virg=0.0;w_virg=0.0;n_last=0.0;w_last=0.0;Btot=0.0;B0=0.0;SSB0=0.0;Nmat
0=0.0;Nmatcurrent=0.0;
  total_catch.initialize(); total_yield.initialize();
average_n.initialize();

  if(n_calls(1)==1)  cout << "Calculating fishing mortality" << endl;
  for (series=1; series<=n_catch_series;series++)
    if(catch_pdf(series)>0) {
      for (y=1; y<=nyears;y++)
f(nages,y,series)=e(y,ces(series))*q(y,cqs(series));
      for (a=1; a<=nages;a++) {
        pred=s(a,css(series))*catch_delta(series);
        for (y=1; y<=nyears;y++) {
            f(a,y,series)=pred*f(nages,y,series);
  //cout << y << " " << a << " " << series << " " << pred << " " <<
f(nages,y,series) << endl;
        } //end for loop on years
      } // end age loop
    } // end if
  for (series=1; series<=n_index_series;series++)
    if(index_pdf(series)>0)
      for (a=1; a<=nages;a++) {
        pred=s(a,iss(series))*index_delta(series);
        for (y=1; y<=nyears;y++)
f_index(a,y,series)=pred*q(y,iqs(series));
```

```
      }

  //cout << "recruitment pars " << recruitment_parameter << endl;
  pup_survival=recruitment_parameter(2);
  alpha=pup_survival*spr0;
  recruitment_parameter(2)=alpha;
  steepness=alpha/(alpha+4.0);
  //cout << "recruitment pars " << recruitment_parameter << endl;

  if(n_calls(1)==1) cout << "Calculating virgin population structure" <<
endl;
  r=recruitment_parameter(1); Bvirgin=spr0*r(1); if(age(1)==0)
ssb(1)=Bvirgin;
  n(1,1,1)=r(1);
  for (a=2; a<=nages; a++) {
    n(a,1,1)=n(a-1,1,1)*mfexp(-m(a-1));
    if(a==nages) n(a,1,1)=n(a,1,1)/(one-mfexp(-m(a)));
  }

  if(n_calls(1)==1) cout << "Calculating time trajectory of population
structure" << endl;
  for (y=1; y<=nyears; y++)  {

    // distinguish historical period (no process errors) from modern
epoch (has process errors)
    if(y<=nyears_deterministic) t=1; else t=y-nyears_deterministic+1;

    // update recruitment
    if(y>age(1))
r(y)=function_value(nature(irn),recruitment_parameter,ssb(y-
age(1))/Bvirgin); // x-year-olds in year x+1 were produced in year 1 (for
which one can compute the ssb)
    if(t>1 && active(r_devs)) {if(r_dev_pdf==1)
r(y)=r(y)*mfexp(r_devs(t)); else r(y)=r(y)+r_devs(t); }
    n(1,y,1)=r(y);
    if(age(1)==0 && y==1) ssb(1)=0;  // don't need this anymore (it gets
recalculated)

    // update abundance and accumulate catches/indices after time step
delta
    for (a=1; a<=nages; a++) {

      for (int j=1; j<=nsteps; j++)  {
        average_n(a,y)+=n(a,y,j);
        //cout << a << " " << y << " " << j << " " << n(a,y,j) << endl;

        // spawning at beginning of step (month/season)
        if(j==spawn_season) {
          if(a==nages && fecundity_input(a)<0)
fecundity(a)=function_value(nature(iwn),growth_parameter,plus_age+spawn_t
ime);
        //changed "s(sby)+=p(a)*fecundity(a)*n(a,y,j)" to this as per
Xinsheng's e-mail from 1/31/2013
        if(y==1)ssb(y)=Bvirgin;
```

```
          else ssb(y)+=p(a)*fecundity(a)*n(a,y,j);
        }

        // then natural mortality until mid-interval
        n(a,y,j+1)=n(a,y,j)*exp_m(a);

  //cout << "exp_m(a) " <<  exp_m(a) << endl;

        // then indices and catches
        for (series=1; series<=n_series; series++) {
          //if(series<=n_index_series && index_pdf(series)>0 &&
index_obs(y,series)>=0 && j>=index_first(series) &&
j<=index_last(series)) {
          if(series<=n_index_series && index_pdf(series)>0 &&
j>=index_first(series) && j<=index_last(series)) {
            index_by_age = (f_index(a,y,series))*n(a,y,j+1);
            if(index_units(series)==1) index_pred(y,series) +=
index_by_age;
            else {
              if(a==nages)
w(a,j)=function_value(nature(iwn),growth_parameter,plus_age+step_time(j))
;
              index_pred(y,series) += index_by_age*w(a,j);
            }
          }
          if(series<=n_catch_series && catch_pdf(series)>0 &&
j>=catch_first(series) && j<=catch_last(series)) {
            catch_by_age = f(a,y,series)*n(a,y,j+1);
            n(a,y,j+1) = posfun(n(a,y,j+1)-catch_by_age,one,n_penalty);
            if(a==nages)
w(a,j)=function_value(nature(iwn),growth_parameter,plus_age+step_time(j))
;
            total_catch(a,y)+=catch_by_age;
total_yield(a,y)+=catch_by_age*w(a,j);
            if(catch_units(series)==2) catch_pred(y,series) +=
catch_by_age*w(a,j); else catch_pred(y,series) += catch_by_age;
          }
          if(series<=n_agecomp_series && j>=agecomp_first(series) &&
j<=agecomp_last(series)) {
            if(series<=n_catch_series) agecomp_pred(a,y,series) +=
catch_by_age;
            else                       agecomp_pred(a,y,series) +=
index_by_age;
          }
        } // end series loop

        // then natural mortality until end of interval
        n(a,y,j+1)=n(a,y,j+1)*exp_m(a);

      } // end j loop

      n(a+1,y+1,1)=n(a,y,nsteps+1) ; // This is the abundance at the
begining of the next year
      if(a==nages) {
```

```
        n(a,y+1,1) += n(a+1,y+1,1); // plus-group
        plus_age=posfun((age(2)*n(a-
1,y,nsteps+1)+(plus_age+one)*n(a,y,nsteps+1))/n(a,y+1,1),double(nages),pl
usage_penalty);
      }
      average_n(a,y)+=n(a,y,nsteps+1);

       if(y==1) {
          n_virg(a)=n(a,y,1);
          w_virg(a)=w(a,1);
          B0+=n_virg(a)*w_virg(a);
          SSB0+=n_virg(a)*fecundity(a)*p(a);
          Nmat0+=n_virg(a)*p(a);

          }
       if(y==nyears) {
          n_last(a)=n(a,y,nsteps);
          w_last(a)=w(a,nsteps);
          Btot+=n_last(a)*w_last(a);
          Nmatcurrent+=n_last(a)*p(a);
          }
    } // end age loop

    //  compute the predicted effort
    for (series=1; series<=n_effort_series; series++)
if(effort_pdf(series)>0) effort_pred(y,series) = e(y,ees(series));

  } // end year loop

  // sdreport variables
    // Projections and equilibrium statistics based on overall
selectivity during last year

  if (sd_phase) {
  average_n=average_n/double(nsteps+1);
    r0=recruitment_parameter(1);
    alpha=recruitment_parameter(2);
    for (a=1; a<=nages; a++) {
      if(average_n(a,nyears)>tiny_number)
s_latest(a)=total_catch(a,nyears)/average_n(a,nyears);
      else s_latest(a)=1.0;
        }  //end for-loop
    Fcurrent=max(s_latest); Bcurrent=ssb(nyears);

//Btot=sum(elem_prod(n_last,w_last));B0=sum(elem_prod(n_virg,w_virg));


//Btot=sum(elem_prod(n(a,nyears,nsteps),w(a,nsteps)));B0=sum(elem_prod(n(
a,1,1),w(a,1)));
    B=ssb; BoverBvirgin=B/Bvirgin;  SSBdepletion=BoverBvirgin(nyears);
Bdepletion=Btot/B0;  Nmatdepletion=Nmatcurrent/Nmat0;


   s_equilibrium=s_latest;
```

```
    if (last_phase()) {
      for (a=1; a<=nages; a++)
wbyage(a)=total_yield(a,nyears)/total_catch(a,nyears);

     // Compute equilibrium statistics
      if(n_calls(1)==1) cout << "Calculating equilibrium statistics" <<
endl;
      Fspr20=-9;  Fspr30=-9; Fspr40=-9; Fspr50=-9; Fspr60=-9; F01=-9;
Fmax=-9; Fmsy=-9;
      spr20=-9;    spr30=-9;  spr40=-9;  spr50=-9;  spr60=-9; spr01=-9;
sprmax=-9; sprmsy=-9;
      pred=0.001; yold=0; yprold=0; sprold=tiny_number;
      for (a=1; a<=nages; a++)
wbyage(a)=total_yield(a,nyears)/total_catch(a,nyears);
      if(Fcurrent>0) s_latest=s_latest/Fcurrent;
      slope0=0.1*ypr(wbyage,m,s_latest,pred,nages)/pred;
      while (1) {
       plus_age=age(2)+mfexp(-pred-m(nages))/(1-mfexp(-pred-m(nages)));
       if(fecundity_input(nages)<0)
fecundity(nages)=function_value(nature(iwn),growth_parameter,plus_age+spa
wn_time);

wbyage(nages)=function_value(nature(iwn),growth_parameter,plus_age+0.5);
       sprtemp=spr(p,fecundity,m,s_latest,pred,spawn_time,nages);
if(sprtemp<=0) sprtemp=0.0000001;
       spratio=sprtemp/spr0;
       yprtemp=ypr(wbyage,m,s_latest,pred,nages);

ytemp=yprtemp*equilibrium_ssb(nature(irn),recruitment_parameter,sprtemp,s
pr0)/sprtemp;
       if     (Fspr60<0 && spratio<0.6) {Fspr60=pred-0.001; ypr60=yprold;
spr60=sprold; Yspr60=yold; }
       else if(Fspr50<0 && spratio<0.5) {Fspr50=pred-0.001; ypr50=yprold;
spr50=sprold; Yspr50=yold; }
       else if(Fspr40<0 && spratio<0.4) {Fspr40=pred-0.001; ypr40=yprold;
spr40=sprold; Yspr40=yold; }
       else if(Fspr30<0 && spratio<0.3) {Fspr30=pred-0.001; ypr30=yprold;
spr30=sprold; Yspr30=yold; }
       else if(Fspr20<0 && spratio<0.2) {Fspr20=pred-0.001; ypr20=yprold;
spr20=sprold; Yspr20=yold; }
       if(F01<0 && ((yprtemp-yprold)/0.001)<=slope0) {F01=pred-0.001;
spr01=sprold; ypr01=yprold;Y01=yold; }
       if(Fmax<0 && yprtemp<=yprold)   { Fmax=pred-0.001; sprmax=sprold;
yprmax=yprold; Ymax=yold;  }
       if(Fmsy<0 && ytemp<=yold)       { Fmsy=pred-0.001; sprmsy=sprold;
yprmsy=yprold; Ymsy=yold;  }
       yprold=yprtemp; sprold=sprtemp; yold=ytemp;
       pred=pred+0.001;
       if(pred>3.0 || (Fspr20>=0 && Fmax>=0 && Fmsy>=0) ) break;
      }

Bspr20=equilibrium_ssb(nature(irn),recruitment_parameter,spr20,spr0);
Rspr20=Bspr20/spr20;
```

```
Bspr30=equilibrium_ssb(nature(irn),recruitment_parameter,spr30,spr0);
Rspr30=Bspr30/spr30;

Bspr40=equilibrium_ssb(nature(irn),recruitment_parameter,spr40,spr0);
Rspr40=Bspr40/spr40;

Bspr50=equilibrium_ssb(nature(irn),recruitment_parameter,spr50,spr0);
Rspr50=Bspr50/spr50;

Bspr60=equilibrium_ssb(nature(irn),recruitment_parameter,spr60,spr0);
Rspr60=Bspr60/spr60;
      B01
=equilibrium_ssb(nature(irn),recruitment_parameter,spr01,spr0);    R01
=B01   /spr01;
      Bmax
=equilibrium_ssb(nature(irn),recruitment_parameter,sprmax,spr0);   Rmax
=Bmax  /sprmax;
      Bmsy
=equilibrium_ssb(nature(irn),recruitment_parameter,sprmsy,spr0);   Rmsy
=Bmsy  /sprmsy;


     if(Bspr20 >0) BoverBspr20 =Bcurrent/Bspr20 ; else BoverBspr20 =-9.0;
     if(Bspr30 >0) BoverBspr30 =Bcurrent/Bspr30 ; else BoverBspr30 =-9.0;
     if(Bspr40 >0) BoverBspr40 =Bcurrent/Bspr40 ; else BoverBspr40 =-9.0;
     if(Bspr50 >0) BoverBspr50 =Bcurrent/Bspr50 ; else BoverBspr50 =-9.0;
     if(Bspr60 >0) BoverBspr60 =Bcurrent/Bspr60 ; else BoverBspr60 =-9.0;
     if(B01    >0) BoverB01    =Bcurrent/B01    ; else BoverB01    =-9.0;
     if(Bmax   >0) BoverBmax   =Bcurrent/Bmax   ; else BoverBmax   =-9.0;
     if(Bmsy   >0) BoverBmsy   =Bcurrent/Bmsy   ; else BoverBmsy   =-9.0;
     if(Bmsy   >0)   inflection=Bmsy/Bvirgin;  else inflection=-9.0;
     if(Fspr20 >0) FoverFspr20 =Fcurrent/Fspr20 ; else FoverFspr20 =-9.0;
     if(Fspr30 >0) FoverFspr30 =Fcurrent/Fspr30 ; else FoverFspr30 =-9.0;
     if(Fspr40 >0) FoverFspr40 =Fcurrent/Fspr40 ; else FoverFspr40 =-9.0;
     if(Fspr50 >0) FoverFspr50 =Fcurrent/Fspr50 ; else FoverFspr50 =-9.0;
     if(Fspr60 >0) FoverFspr60 =Fcurrent/Fspr60 ; else FoverFspr60 =-9.0;
     if(F01    >0) FoverF01    =Fcurrent/F01    ; else FoverF01    =-9.0;
     if(Fmax   >0) FoverFmax   =Fcurrent/Fmax   ; else FoverFmax   =-9.0;
     if(Fmsy   >0) FoverFmsy   =Fcurrent/Fmsy   ; else FoverFmsy   =-9.0;

    }// last_phase loop
  }// sd_phase loop

 //-----------------------------------------------------------------------
FUNCTION calculate_the_objective_function
 //-----------------------------------------------------------------------
  catch_lklhd=0; index_lklhd=0.;  effort_lklhd=0.; agecomp_lklhd=0.;
obj_func=0.; f_penalty=0;

 if(n_calls(1)==1) cout << "Calculating objective function" << endl;
 // ---------------observation errors----------------

  for(y=1; y<=nyears; y++) {
```

```
    for(series=1; series<=n_catch_series; series++)
if(catch_pdf(series)>0  && catch_obs(y,series)>=0)
catch_lklhd(series)+=neg_log_lklhd(catch_obs(y,series),catch_pred(y,serie
s)+catch_min,catch_cv(y,series)*c_d_var(cvs(series))*overall_var,catch_pd
f(series),overall_var_pdf);
    for(series=1; series<=n_index_series; series++) {

if(index_pdf(series)==1  && index_obs(y,series)>0)
index_lklhd(series)+=neg_log_lklhd(index_obs(y,series),index_pred(y,serie
s)+index_min(series),index_cv(y,series)*i_d_var(ivs(series))*overall_var,
index_pdf(series),overall_var_pdf);
                                                     else
if(index_pdf(series)>0  && index_obs(y,series)>=0)
index_lklhd(series)+=neg_log_lklhd(index_obs(y,series),index_pred(y,serie
s),index_cv(y,series)*i_d_var(ivs(series))*overall_var,index_pdf(series),
overall_var_pdf);
                                                    }
    for(series=1; series<=n_effort_series; series++)
if(effort_pdf(series)>0 && effort_obs(y,series)>=0)
effort_lklhd(series)+=neg_log_lklhd(effort_obs(y,series),effort_pred(y,se
ries)+effort_min(series),effort_cv(y,series)*e_d_var(evs(series))*overall
_var,effort_pdf(series),overall_var_pdf);

    for(series=1; series<=n_agecomp_series; series++) {
      if(n_agecomp_data(y,series)>0) {
        pred=0;
        for(a=1; a<=nages; a++) pred+=agecomp_pred(a,y,series);
        for(a=1; a<=nages; a++) {if(pred>0)
agecomp_pred(a,y,series)/=pred; else agecomp_pred(a,y,series)=0; }
        for(a=1; a<=nages; a++) {
          if(agecomp_pdf(series)==8) { // Fournier's robustified normal
distribution
            var=( agecomp_pred(a,y,series)*(1-agecomp_pred(a,y,series)) +
0.1/nages )/n_agecomp_data(y,series);

agecomp_lklhd(series)+=neg_log_lklhd(agecomp_obs(a,y,series),agecomp_pred
(a,y,series),var,agecomp_pdf(series),overall_var_pdf);
          }
          else if(agecomp_pdf(series)==2) { // least-squares
            var=1;

agecomp_lklhd(series)+=neg_log_lklhd(agecomp_obs(a,y,series)*n_agecomp_da
ta(y,series),agecomp_pred(a,y,series)*n_agecomp_data(y,series),var,agecom
p_pdf(series),2);
          }
          else if(agecomp_pdf(series)>0) { // multinomial distribution
            if(agecomp_obs(a,y,series)>0)
agecomp_lklhd(series)+=n_agecomp_data(y,series)*neg_log_lklhd(agecomp_obs
(a,y,series),agecomp_pred(a,y,series)/agecomp_obs(a,y,series),var,agecomp
_pdf(series),overall_var_pdf);
          }
        }
      }
    }
```

```
  }
  if(n_catch_series>0) obj_func+=sum(catch_lklhd);
  if(n_index_series>0) obj_func+=sum(index_lklhd);
  if(n_effort_series>0) obj_func+=sum(effort_lklhd);
  if(n_agecomp_series>0) obj_func+=sum(agecomp_lklhd);

 // ---------------Process errors--------------------
  if(active(r_devs)) {
    r_lklhd=square(r_devs(2));
    for(t=3; t<=n_eras; t++) r_lklhd += square(r_devs(t)-r_rho*r_devs(t-
1));
    r_lklhd=0.5*(r_lklhd/r_var+double(n_eras-1)*log(r_var));
    obj_func +=r_lklhd;
  }

  for (set=1; set<=nes; set++) {
    if(e_dev_iph(set)>0 && e_dev_iph(set)<=current_ph) {
      e_lklhd(set)=square(e_devs(set,2));
      for(t=3; t<=n_eras; t++) e_lklhd(set) += square(e_devs(set,t)-
e_rho(set)*e_devs(set,t-1));
      e_lklhd(set)=0.5*(e_lklhd(set)/e_var(set)+(n_eras-
1)*log(e_var(set)));
      obj_func += e_lklhd(set);
    }
  }

  for (set=1; set<=nqs; set++) {
    if(q_dev_iph(set)>0 && q_dev_iph(set)<=current_ph) {
      if(overall_var_pdf==1 && q_dev_pdf(set)==1 && overall_var<zero)
var=log(1.0+square(q_var(set)*overall_var));
      else if(overall_var_pdf==2 && q_dev_pdf(set)==2 &&
overall_var>zero) var=q_var(set)*overall_var;
      else
var=get_variance(q(nyears_deterministic+1,set),q_var(set)*overall_var,q_d
ev_pdf(set),overall_var_pdf);
      q_lklhd(set)=square(q_devs(2,set));
      for(t=3; t<=n_eras; t++) q_lklhd(set) += square(q_devs(set,t)-
q_rho(set)*q_devs(set,t-1));
      q_lklhd(set)=0.5*(q_lklhd(set)/var+(n_eras-1)*log(var));
      obj_func += q_lklhd(set);
    }
  }

 // ---------------Bayesian priors-------------------//
  obj_func +=
m_prior+r_prior+w_prior+v_prior+sum(q_prior)+sum(e_prior)+sum(s_prior)+r_
process_prior+q_process_prior+e_process_prior;

 // --------------penalty to avoid getting stuck in 'extreme fishing'
solutions (i.e., mining with near zero F and very high N or
hyperproductivity with ver)--------------------//
//  if (!last_phase()) {
//    if(current_ph <= 2)      f_penalty +=
10.0*norm2(elem_div(total_catch,average_n)-.3);
```

```
//    else if(current_ph <= 3) f_penalty +=
1.0*norm2(elem_div(total_catch,average_n)-.3);
//    else                     f_penalty +=
0.1*norm2(elem_div(total_catch,average_n)-.3);
//  }
// added following lines from template used by Liz in 2007
         for (series=1; series<=n_catch_series; series++) {
    for (y=22; y<=56; y++)  {
       if(catch_obs(y,series)>1.0) {
        if(current_ph <= 3) f_penalty += 10.0*pow(
(log(catch_obs(y,series)/catch_pred(y,series) )  ),2);
        if(current_ph = 4) f_penalty += 1.0*pow(
(log(catch_obs(y,series)/catch_pred(y,series) )  ),2);
        //if(last_phase() ) f_penalty += 3.0*pow(
(log(catch_obs(y,series)/catch_pred(y,series) )  ),2);
//        if(current_ph <= 3) f_penalty += 10.0*pow(
(log(catch_obs(y,1)/catch_pred(y,1) )  ),2);
//        if(current_ph = 4) f_penalty += 1.0*pow(
(log(catch_obs(y,1)/catch_pred(y,1) )  ),2);
//        if(last_phase() ) f_penalty += 1.0*pow(
(log(catch_obs(y,1)/catch_pred(y,1) )  ),2);
//        if(current_ph <= 3) f_penalty += 10.0*pow(
(log(catch_obs(y,2)/catch_pred(y,2) )  ),2);
//        if(current_ph = 4) f_penalty += 1.0*pow(
(log(catch_obs(y,2)/catch_pred(y,2) )  ),2);
//        if(last_phase() ) f_penalty += 1.0*pow(
(log(catch_obs(y,2)/catch_pred(y,2) )  ),2);
//        if(current_ph <= 3) f_penalty += 10.0*pow(
(log(catch_obs(y,3)/catch_pred(y,3) )  ),2);
//        if(current_ph = 4) f_penalty += 1.0*pow(
(log(catch_obs(y,3)/catch_pred(y,3) )  ),2);
//        if(last_phase() ) f_penalty += 1.0*pow(
(log(catch_obs(y,3)/catch_pred(y,3) )  ),2);
//        if(current_ph <= 3) f_penalty += 10.0*pow(
(log(catch_obs(y,4)/catch_pred(y,4) )  ),2);
//        if(current_ph = 4) f_penalty += 1.0*pow(
(log(catch_obs(y,4)/catch_pred(y,4) )  ),2);
//        if(last_phase() ) f_penalty += 1.0*pow(
(log(catch_obs(y,4)/catch_pred(y,4) )  ),2);
//        if(current_ph <= 3) f_penalty += 10.0*pow(
(log(catch_obs(y,5)/catch_pred(y,5) )  ),2);
//        if(current_ph = 4) f_penalty += 1.0*pow(
(log(catch_obs(y,5)/catch_pred(y,5) )  ),2);
//        if(last_phase() ) f_penalty += 1.0*pow(
(log(catch_obs(y,5)/catch_pred(y,5) )  ),2);
//        if(current_ph <= 3) f_penalty += 10.0*pow(
(log(catch_obs(y,6)/catch_pred(y,6) )  ),2);
//        if(current_ph = 4) f_penalty += 1.0*pow(
(log(catch_obs(y,6)/catch_pred(y,6) )  ),2);
//        if(last_phase() ) f_penalty += 1.0*pow(
(log(catch_obs(y,6)/catch_pred(y,6) )  ),2);
            }
       }
    }
```

```
  //    }
        // if(current_ph = 5) f_penalty +=
10.0*norm2(elem_div(total_catch,average_n)-.3);
        // if(last_phase() ) f_penalty +=
1.0*norm2(elem_div(total_catch,average_n)-.3);
      //         f_penalty += 10.0*norm2(  elem_div(
column(catch_obs,6),average_n )-.3  ) ;
    //if(current_ph <= 2)      f_penalty +=
10.0*norm2(elem_div(total_catch,average_n)-.3);
     //else if(current_ph <= 3) f_penalty +=
1.0*norm2(elem_div(total_catch,average_n)-.3);
     //else                     f_penalty +=
0.1*norm2(elem_div(total_catch,average_n)-.3);
  // }

  obj_func+=f_penalty+100.0*(plusage_penalty+n_penalty);


/////////////////////////////////////////////////////////////////////////
REPORT_SECTION   // uses regular C++ code
/////////////////////////////////////////////////////////////////////////
  cout << "Writing report" << endl;
  n_par_phase=initial_params::nvarcalc(); // number of active parameters
  double aic=2.0*(value(obj_func)+double(n_par_phase));
  report.setf(ios::right, ios::adjustfield);
  report.setf(ios::scientific, ios::floatfield);
  report << "-------------------------------------------------------------
--------" << endl;
  report << "LIKELIHOOD RESULTS" << endl;
  report << "-------------------------------------------------------------
--------" << endl;
  report << "AIC                 : " << setw(12) << setprecision(5) <<
aic << endl;
  report << "data points         : " << setw(12) << setprecision(5) <<
int(n_data) << endl;
  report << "estimated parameters: " << setw(12) << setprecision(5) <<
n_par_phase << endl;
  if(n_data<(n_par_phase+2)) {
    report << "AICc (small sample) : " << " undefined (too few data)" <<
endl;
  }
  else {
    double aicc=aic+2.0*double(n_par_phase*(n_par_phase+1)/(n_data-
n_par_phase-1));
    report << "AICc (small sample) : " << setw(12) << setprecision(5) <<
aicc << endl;
  }
  report << "OBJECTIVE FUNCTION  : " << setw(12) << setprecision(5) <<
obj_func << endl;
  report << "  Observation errors: " << endl;
  report << "    catch           : " << setw(12) << setprecision(5) <<
catch_lklhd    << endl;
  report << "    effort          : " ;
```

```
    for (series=1; series<=n_effort_series; series++) report << " " <<
setw(12) << setprecision(5) << effort_lklhd(series) ; report << endl;
  report << "    indices         : " ;
  for (series=1; series<=n_index_series; series++) report << " " <<
setw(12) << setprecision(5) << index_lklhd(series) ;  report << endl;
  report << "    age composition : " ;
  for (series=1; series<=n_agecomp_series; series++) report << " " <<
setw(12) << setprecision(5) << agecomp_lklhd(series) ; report << endl;
  report << "  Process errors    : " << endl;
  report << "    r recruitment   : " << setw(12) << setprecision(5) <<
r_lklhd  << endl;
  report << "    q catchability  : " ;
    for(set=1; set<=nqs; set++) report << setw(12) << setprecision(5) <<
q_lklhd(set) << " "; report << endl ;
  report << "    e effort        : " ;
    for(set=1; set<=nes; set++) report << setw(12) << setprecision(5) <<
e_lklhd(set) << " "; report << endl ;
  report << "  Priors            : " << endl;
  report << "    m natural mort. : " << setw(12) << setprecision(5) <<
m_prior  << endl;
  report << "    r recruitment   : " << setw(12) << setprecision(5) <<
r_prior  << endl;
  report << "    r process error : " << setw(12) << setprecision(5) <<
r_process_prior << endl;
  report << "    b historical F  : " << setw(12) << setprecision(5) <<
e_prior  << endl;
  report << "    k growth        : " << setw(12) << setprecision(5) <<
w_prior  << endl;
  report << "    q catchability  : " << setw(12) << setprecision(5) <<
q_prior  << endl;
  report << "    q process error : " << setw(12) << setprecision(5) <<
q_process_prior << endl;
  report << "    e effort        : " << setw(12) << setprecision(5) <<
e_prior  << endl;
  report << "    e process error : " << setw(12) << setprecision(5) <<
e_process_prior << endl;
  report << "    catch variance  : " << setw(12) << setprecision(5) <<
c_d_prior << endl;
  report << "    effort variance : " << setw(12) << setprecision(5) <<
e_d_prior << endl;
  report << "    index variance  : " << setw(12) << setprecision(5) <<
i_d_prior << endl;
  report << "    over-all var.   : " << setw(12) << setprecision(5) <<
v_prior << endl;
  report << "  Penalties         : " << endl;
  report << "    Negative abund. : " << setw(12) << setprecision(5) <<
n_penalty << endl;
  report << "    Plus-age        : " << setw(12) << setprecision(5) <<
plusage_penalty << endl;
  report << "    Fishing mort.   : " << setw(12) << setprecision(5) <<
f_penalty << endl;
  report << "                " << endl;
  if(overall_var<zero) report << "OVERALL %CV         : " << setw(12) <<
setprecision(5) << -100.0*overall_var << endl;
```

```
  else                 report << "OVERALL VARIANCE    : " << setw(12) <<
setprecision(5) << overall_var << endl;
  report << "               " << endl;
  report << "LIFE-TIME REPRODUCTIVE RATE: " << setw(12) <<
setprecision(5) << alpha << endl;
  report << "STEEPNESS:  " << setw(12) << setprecision(4) <<
alpha/(alpha+4) << endl;
  report << "PUP-SURVIVAL: " << setw(12) << setprecision(4) <<
pup_survival << endl;
  report << "               " << endl;
  report << "NUMBER OF FUNCTION EVALUATIONS (THIS PHASE): " << setw(12)
<< setprecision(5) << n_calls(current_ph) << endl;
  report << "NUMBER OF FUNCTION EVALUATIONS (CUMULATIVE): " << setw(12)
<< setprecision(5) << sum(n_calls) << endl;
  report << "               " << endl; report << "               " <<
endl;
  report << "   *****            " << endl; report << "  Inflection-point
(SSBmsy/SSB0)     " << inflection << endl;
  report << " Btot  " << Btot << " B0  " << B0<< " Btot/B0 " <<
Bdepletion << "  SSB/SSB0  " << SSBdepletion << endl;

  report << "-------------------------------------------------------------
--------" << endl;
  report << "MANAGEMENT BENCHMARKS" << endl;
  report << "Type           F          Y         Y/R        SSB
S/R          R" << endl;
  report << "-------------------------------------------------------------
--------" << endl;
  report.setf(ios::scientific, ios::floatfield);
  report << "VIRGIN   " << setw(12) << setprecision(4) << zero    << " "
<< zero    << " " << zero   << " " << spr0*recruitment_parameter(1) << "
" << spr0 << " " << recruitment_parameter(1) << endl;
  report << "MSY      " << setw(12) << setprecision(4) << Fmsy   << " "
<< Ymsy   << " " << yprmsy << " " << Bmsy    << " " << sprmsy << " " <<
Rmsy   << endl;
  report << "MAX Y/R  " << setw(12) << setprecision(4) << Fmax   << " "
<< Ymax   << " " << yprmax << " " << Bmax    << " " << sprmax << " " <<
Rmax   << endl;
  report << "F0.1     " << setw(12) << setprecision(4) << F01    << " "
<< Y01    << " " << ypr01  << " " << B01     << " " << spr01  << " " <<
R01    << endl;
  report << "20% SPR  " << setw(12) << setprecision(4) << Fspr20 << " "
<< Yspr20 << " " << ypr20  << " " << Bspr20  << " " << spr20  << " " <<
Rspr20 << endl;
  report << "30% SPR  " << setw(12) << setprecision(4) << Fspr30 << " "
<< Yspr30 << " " << ypr30  << " " << Bspr30  << " " << spr30  << " " <<
Rspr30 << endl;
  report << "40% SPR  " << setw(12) << setprecision(4) << Fspr40 << " "
<< Yspr40 << " " << ypr40  << " " << Bspr40  << " " << spr40  << " " <<
Rspr40 << endl;
  report << "50% SPR  " << setw(12) << setprecision(4) << Fspr50 << " "
<< Yspr50 << " " << ypr50  << " " << Bspr50  << " " << spr50  << " " <<
Rspr50 << endl;
```

```
  report << "60% SPR  " << setw(12) << setprecision(4) << Fspr60 << " "
<< Yspr60 << " " << ypr60  << " " << Bspr60  << " " << spr60  << " " <<
Rspr60 << endl;
  report << "               " << endl; report << "               " <<
endl;

  report << "------------------------------------------------------------
--------" << endl;
  report << "PRESENT CONDITION OF STOCK" << endl;
  report << "Type          F          SSB" << endl;
  report << "------------------------------------------------------------
--------" << endl;
//      if(Bspr20 >0) BoverBspr20 =Bcurrent/Bspr20 ; else BoverBspr20 =-
9.0;
//      if(Bspr30 >0) BoverBspr30 =Bcurrent/Bspr30 ; else BoverBspr30 =-
9.0;
//      if(Bspr40 >0) BoverBspr40 =Bcurrent/Bspr40 ; else BoverBspr40 =-
9.0;
//     if(Bspr50 >0) BoverBspr50 =Bcurrent/Bspr50 ; else BoverBspr50 =-
9.0;
//     if(Bspr60 >0) BoverBspr60 =Bcurrent/Bspr60 ; else BoverBspr60 =-
9.0;
//     if(B01    >0) BoverB01    =Bcurrent/B01    ; else BoverB01    =-
9.0;
//     if(Bmax   >0) BoverBmax   =Bcurrent/Bmax   ; else BoverBmax   =-
9.0;
//     if(Bmsy   >0) BoverBmsy   =Bcurrent/Bmsy   ; else BoverBmsy   =-
9.0;
//     if(Fspr20 >0) FoverFspr20 =Fcurrent/Fspr20 ; else FoverFspr20 =-
9.0;
//     if(Fspr30 >0) FoverFspr30 =Fcurrent/Fspr30 ; else FoverFspr30 =-
9.0;
//     if(Fspr40 >0) FoverFspr40 =Fcurrent/Fspr40 ; else FoverFspr40 =-
9.0;
//     if(Fspr50 >0) FoverFspr50 =Fcurrent/Fspr50 ; else FoverFspr50 =-
9.0;
//     if(Fspr60 >0) FoverFspr60 =Fcurrent/Fspr60 ; else FoverFspr60 =-
9.0;
//     if(F01    >0) FoverF01    =Fcurrent/F01    ; else FoverF01    =-
9.0;
//     if(Fmax   >0) FoverFmax   =Fcurrent/Fmax   ; else FoverFmax   =-
9.0;
//     if(Fmsy   >0) FoverFmsy   =Fcurrent/Fmsy   ; else FoverFmsy   =-
9.0;
  report.setf(ios::scientific, ios::floatfield);
  report << "CURRENT  " << setw(12) << setprecision(4) << Fcurrent
<< " " << Bcurrent      << endl;
  report << " /MSY    " << setw(12) << setprecision(4) << FoverFmsy
<< " " << BoverBmsy     << endl;
  report << " /MAX Y/R" << setw(12) << setprecision(4) << FoverFmax
<< " " << BoverBmax     << endl;
  report << " /F0.1   " << setw(12) << setprecision(4) << FoverF01
<< " " << BoverB01      << endl;
```

```
  report << " /20% SPR" << setw(12) << setprecision(4) << FoverFspr20
<< " " << BoverBspr20   << endl;
  report << " /30% SPR" << setw(12) << setprecision(4) << FoverFspr30
<< " " << BoverBspr30   << endl;
  report << " /40% SPR" << setw(12) << setprecision(4) << FoverFspr40
<< " " << BoverBspr40   << endl;
  report << " /50% SPR" << setw(12) << setprecision(4) << FoverFspr50
<< " " << BoverBspr50   << endl;
  report << " /60% SPR" << setw(12) << setprecision(4) << FoverFspr60
<< " " << BoverBspr60   << endl;
  report << "               " << endl; report << "               " <<
endl;

  report << "-------------------------------------------------------------
--------" << endl;
  report << "ABUNDANCE ESTIMATES by age" << endl;
  report << "Year" << " ";
  report.setf(ios::fixed, ios::floatfield);
  for (a=1; a<=nages-1; a++) report << setw(8) << setprecision(0) <<
a+age(1)-1 << "     ";
  report << setw(8) << setprecision(0) << nages+age(1)-1 << endl;
  report << "-------------------------------------------------------------
--------" << endl;
  for (y=1; y<=nyears; y++) {
    report.setf(ios::fixed, ios::floatfield);
    report << setw(4) << setprecision(0) << y+year(1)-1 << "  ";
    report.setf(ios::scientific, ios::floatfield);
    for (a=1; a<=nages-1; a++) report << setw(12) << setprecision(4) <<
n(a,y,1) << " ";
    report << setw(12) << setprecision(4) << n(nages,y,1) << endl;
  }
  report << "               " << endl; report << "               " <<
endl;

  report << "-------------------------------------------------------------
--------" << endl;
  report << "FISHING MORTALITY RATE ESTIMATES by age" << endl;
  report << "Year" << " ";
  report.setf(ios::fixed, ios::floatfield);
  for (a=1; a<=nages-1; a++) report << setw(8) << setprecision(0) <<
a+age(1)-1 << "     ";
  report << setw(8) << setprecision(0) << nages+age(1)-1 << endl;
  report << "-------------------------------------------------------------
--------" << endl;
  for (y=1; y<=nyears; y++) {
    report.setf(ios::fixed, ios::floatfield);
    report << setw(4) << setprecision(0) << y+year(1)-1 << "  ";
    report.setf(ios::scientific, ios::floatfield);
    for (a=1; a<=nages-1; a++) report << setw(12) << setprecision(4) <<
total_catch(a,y)/average_n(a,y) << " ";
    report << setw(12) << setprecision(4) <<
total_catch(nages,y)/average_n(nages,y) << endl;
  }
```

```
  report << "               " << endl; report << "               " <<
endl;

  report << "-------------------------------------------------------------
--------" << endl;
  report << "CATCH ESTIMATES IN NUMBERS by age" << endl;
  report << "Year" << " ";
  report.setf(ios::fixed, ios::floatfield);
  for (a=1; a<=nages-1; a++) report << setw(8) << setprecision(0) <<
a+age(1)-1 << "     ";
  report << setw(8) << setprecision(0) << nages+age(1)-1 << endl;
  report << "-------------------------------------------------------------
--------" << endl;
  for (y=1; y<=nyears; y++) {
    report.setf(ios::fixed, ios::floatfield);
    report << setw(4) << setprecision(0) << y+year(1)-1 << "  ";
    report.setf(ios::scientific, ios::floatfield);
    for (a=1; a<=nages; a++) report << setw(12) << setprecision(4) <<
total_catch(a,y) << " "; report << endl;
  }
  report << "               " << endl; report << "               " <<
endl;

  report << "-------------------------------------------------------------
--------" << endl;
  report << "CATCH ESTIMATES IN WEIGHT by age" << endl;
  report << "Year" << " ";
  report.setf(ios::fixed, ios::floatfield);
  for (a=1; a<=nages-1; a++) report << setw(8) << setprecision(0) <<
a+age(1)-1 << "     ";
  report << setw(8) << setprecision(0) << nages+age(1)-1 << endl;
  report << "-------------------------------------------------------------
--------" << endl;
  for (y=1; y<=nyears; y++) {
    report.setf(ios::fixed, ios::floatfield);
    report << setw(4) << setprecision(0) << y+year(1)-1 << "  ";
    report.setf(ios::scientific, ios::floatfield);
    for (a=1; a<=nages; a++) report << setw(12) << setprecision(4) <<
total_yield(a,y) << " "; report << endl;
  }
  report << "               " << endl; report << "               " <<
endl;

  report << "-------------------------------------------------------------
--------" << endl;
  report << "SPAWNING BIOMASS ESTIMATES" << endl;
  report << "Year" << " " << endl;
  report.setf(ios::fixed, ios::floatfield);
  report << "-------------------------------------------------------------
--------" << endl;
  for (y=1; y<=nyears; y++) {
    report.setf(ios::fixed, ios::floatfield);
    report << setw(4) << setprecision(0) << y+year(1)-1 << "  ";
    report.setf(ios::scientific, ios::floatfield);
```

```
    report << setw(12) << setprecision(4) << ssb(y) << endl;
  }
  report << "               " << endl; report << "               " <<
endl;

  report << "-------------------------------------------------------------
----------------" << endl;
  report << "CATCH ESTIMATES" << endl;
  report << "Series" << "  Year" << "    Observed" << "    Predicted" <<
"   Variance" << "    Catchability" << "  Effort" << endl;
  report << "-------------------------------------------------------------
----------------" << endl;
  for(series=1; series<=n_catch_series; series++) {
    report.setf(ios::fixed, ios::floatfield);
    if(catch_pdf(series)==0)
      report << setw(4) << setprecision(0) << series << "    " << "Not
used" << endl;
    else {
      for (y=1; y<=nyears; y++) {
        if(y<=nyears_deterministic) t=1; else t=y-nyears_deterministic+1;
        report.setf(ios::fixed, ios::floatfield);
        report << setw(4) << setprecision(0) << series << "    ";
        report << setw(4) << setprecision(0) << y+year(1)-1 << "  ";
        report.setf(ios::scientific, ios::floatfield);
        report << setw(12) << setprecision(4) << catch_obs(y,series);
        report << setw(12) << setprecision(4) << catch_pred(y,series);
        report << setw(12) << setprecision(4) <<
get_variance(catch_pred(y,series)+catch_min,catch_cv(y,series)*c_d_var(cv
s(series))*overall_var,catch_pdf(series),overall_var_pdf) << " ";
        report << setw(12) << setprecision(4) << q(y,cqs(series)) << " ";
        report << setw(12) << setprecision(4) << e(y,ces(series)) <<
endl;
      }
    }
  }
  report << "               " << endl; report << "               " <<
endl;
  report << "-------------------------------------------------------------
--------" << endl;
  report << "EFFORT ESTIMATES" << endl;
  report << "Series" << "  Year" << "    Observed" << "    Predicted" <<
"   Variance" << endl;
  report << "-------------------------------------------------------------
--------" << endl;
  if(n_effort_series<=0) report << "  None used" << endl;
  for(series=1; series<=n_effort_series; series++) {
    report.setf(ios::fixed, ios::floatfield);
    if(effort_pdf(series)==0)
      report << setw(4) << setprecision(0) << series << "    " << "Not
used" << endl;
    else {
      for (y=1; y<=nyears; y++) {
        report.setf(ios::fixed, ios::floatfield);
        report << setw(4) << setprecision(0) << series << "    ";
```

```
        report << setw(4) << setprecision(0) << y+year(1)-1 << "  ";
        report.setf(ios::scientific, ios::floatfield);
        report << setw(12) << setprecision(4) << effort_obs(y,series);
        report << setw(12) << setprecision(4) << effort_pred(y,series);
        report << setw(12) << setprecision(4) <<
get_variance(effort_pred(y,series)+effort_min(series),effort_cv(y,series)
*e_d_var(evs(series))*overall_var,effort_pdf(series),overall_var_pdf) <<
endl;
      }
    }
  }
  report << "               " << endl; report << "               " <<
endl;
  report << "-------------------------------------------------------------
--------" << endl;
  report << "INDEX (CPUE) ESTIMATES" << endl;
  report << "Series" << "  Year" << "    Observed" << "    Predicted" <<
"   Variance" << "     Catchability" << endl;
  report << "-------------------------------------------------------------
--------" << endl;
  if(n_index_series<=0) report << "  None used" << endl;
  for(series=1; series<=n_index_series; series++) {
    report.setf(ios::fixed, ios::floatfield);
    if(index_pdf(series)==0)
      report << setw(4) << setprecision(0) << series << "    " << "Not
used" << endl;
    else {
      for (y=1; y<=nyears; y++) {
        if(y<=nyears_deterministic) t=1; else t=y-nyears_deterministic+1;
        report.setf(ios::fixed, ios::floatfield);
        report << setw(4) << setprecision(0) << series << "    ";
        report << setw(4) << setprecision(0) << y+year(1)-1 << "  ";
        report.setf(ios::scientific, ios::floatfield);
        report << setw(12) << setprecision(4) << index_obs(y,series);
        report << setw(12) << setprecision(4) << index_pred(y,series);
        report << setw(12) << setprecision(4) <<
get_variance(index_pred(y,series)+index_min(series),index_cv(y,series)*i_
d_var(ivs(series))*overall_var,index_pdf(series),overall_var_pdf) ;
        //report << setw(12) << setprecision(4) << q(t,iqs(series)) << "
" << endl;
         // changed t to y (LIZ 8/18/2005)
        report << setw(12) << setprecision(4) << q(y,iqs(series)) << " "
<< endl;
      }
    }
  }
  report << "               " << endl; report << "               " <<
endl;
  report << "-------------------------------------------------------------
--------" << endl;
  report << "AGE COMPOSITION ESTIMATES" << endl;
  report << "Series" << "  Year     N" << "    Predicted age composition"
<< endl;
```

```
  report << "----------------------------------------------------------------------------
--------" << endl;
  if(n_agecomp_series<=0) report << "  None used" << endl;
  for(series=1; series<=n_agecomp_series; series++) {
    report.setf(ios::fixed, ios::floatfield);
    if(agecomp_pdf(series)<0)
      report << setw(4) << setprecision(0) << series << "    " << "Not
used" << endl;
    else {
      for (y=1; y<=nyears; y++) {
        if(n_agecomp_data(y,series)>0) {
          report.setf(ios::fixed, ios::floatfield);
          report << setw(4) << setprecision(0) << series << "    ";
          report << setw(4) << setprecision(0) << y+year(1)-1 << "  ";
          report << setw(4) << setprecision(0) <<
n_agecomp_data(y,series) << "  pred: " ;
          report.setf(ios::scientific, ios::floatfield) ;
          for (a=1; a<=nages; a++) report << setw(12) << setprecision(4)
<< agecomp_pred(a,y,series);
          report << endl;
          report.setf(ios::fixed, ios::floatfield);
          report << setw(4) << setprecision(0) << series << "    ";
          report << setw(4) << setprecision(0) << y+year(1)-1 << "  ";
          report << setw(4) << setprecision(0) <<
n_agecomp_data(y,series) << "  obsd: " ;
          report.setf(ios::scientific, ios::floatfield) ;
          for (a=1; a<=nages; a++) report << setw(12) << setprecision(4)
<< agecomp_obs(a,y,series);
          report << endl;
        }
      }
    }
  }
  report << "               " << endl; report << "               " <<
endl;
  report << "----------------------------------------------------------------------------
--------" << endl;
  report << "WEIGHT ESTIMATES by age (  yield(age,year)/catch(age,year)
)" << endl;
  report << "Year" << " ";
  report.setf(ios::fixed, ios::floatfield);
  for (a=1; a<=nages-1; a++) report << setw(8) << setprecision(0) <<
a+age(1)-1 << "     ";
  report << setw(8) << setprecision(0) << nages+age(1)-1 << endl;
  report << "----------------------------------------------------------------------------
--------" << endl;
  for (y=1; y<=nyears; y++) {
    report.setf(ios::fixed, ios::floatfield);
    report << setw(4) << setprecision(0) << y+year(1)-1 << "  ";
    report.setf(ios::scientific, ios::floatfield);
    for (a=1; a<=nages; a++) {
      if(total_catch(a,y)>0) wbyage(a)=total_yield(a,y)/total_catch(a,y);
else wbyage(a)=w(a,1);
```

```
      if(a<nages) report << setw(12) << setprecision(4) << wbyage(a) << "
";
      else        report << setw(12) << setprecision(4) << wbyage(nages)
<< " " << endl;
    }
  }
  report << "------------------------------------------------------------
--------" << endl;
  report << "SELECTIVITY AT AGE" << endl;
  for (y=1; y<=nss; y++) {
    report.setf(ios::fixed, ios::floatfield);
    report << setw(4) << setprecision(0) << y << "  ";
    report.setf(ios::scientific, ios::floatfield);
    for (a=1; a<=nages; a++) report << setw(12) << setprecision(4) <<
s(a,y) << " " ;
    report << "               " << endl;
  }
  report << "               " << endl; report << "               " <<
endl;

 #include "SB_make_Rfile.cxx"
////////////////////////////////////////////////////////////////////////
RUNTIME_SECTION
////////////////////////////////////////////////////////////////////////
 convergence_criteria 1.e-2,1.e-3,1.e-5
 maximum_function_evaluations 200,500,1000

////////////////////////////////////////////////////////////////////////
TOP_OF_MAIN_SECTION
////////////////////////////////////////////////////////////////////////
 // set buffer sizes etc...
 arrmblsize=700000;
 gradient_structure::set_MAX_NVAR_OFFSET(500);
 gradient_structure::set_NUM_DEPENDENT_VARIABLES(50000);
 gradient_structure::set_GRADSTACK_BUFFER_SIZE(10000000);
 gradient_structure::set_CMPDIF_BUFFER_SIZE(40000000);

////////////////////////////////////////////////////////////////////////
GLOBALS_SECTION
////////////////////////////////////////////////////////////////////////
 #include "D:\Enric SYX\My Documents\My
Data\Private\ADMB\03_SPASM\fishgraph_Enric_11_04_2011\admodel.h"
 #include "D:\Enric SYX\My Documents\My
Data\Private\ADMB\03_SPASM\fishgraph_Enric_11_04_2011\admb2r.cpp"

  double zero, one, tiny_number, huge_number, two_pi;
  int
imv,imd,iwv,iwd,iwn,irv,ird,irn,i_one,i_two,current_ph,series,set,y,t,a;

 //-----------------------------------------------------------------------
-------------
  dvariable neg_log_lklhd(dvariable obs,dvariable pred,dvariable var,int
pdf,int overall_var_pdf)
 // compute generic negative log-likelihood formulae
```

```
//------------------------------------------------------------------------
  {
    dvariable answer, alph, beta;

    if( obs<0.0 || (obs==0 && (pdf==1 || pdf==3 || pdf==7)) )
      answer=0.0; // no data or process
    else {
      switch(pdf) {
       case 1: // lognormal
          if(pred<=0) {answer=square(pred)*huge_number; break;}
          if(var<0)                     var=log(1.0+square(var)) ;
// convert cv to variance on log scale
            else if(overall_var_pdf==2) var=log(1.0+var/square(pred)); //
convert observation variance to log scale
            else if(overall_var_pdf==0) var=1.0;                    //
automatic equal weighting
          if(obs==pred) answer= 0.5*log(var);
          else          answer= 0.5*( square(log(obs/pred))/var +
log(var) );
          break;
        case 2: // normal
          if(var<0)                     var=square(var)*square(pred); //
convert cv to variance on observation scale
            else if(overall_var_pdf==1) var=square(pred)*(mfexp(var)-1);
// convert log-scale variance to observation scale
            else if(overall_var_pdf==0) var=1.0;                    //
automatic equal weighting
          answer= 0.5*( square(obs-pred)/var + log(var) );
          break;
        case 3: // Multinomial (pred is expected proportion, obs is
observed frequency)
          if(pred<=0) {answer=square(pred)*huge_number; break;}
          answer= -obs*log(pred+0.000001);
          break;
        case 4: // Poisson (pred is expected value, obs is observed)
          if(pred<=0) {answer=square(pred)*huge_number; break;}
          answer= pred-obs*log(pred+0.000001);
          break;
        case 5: // Chi-Square
          answer= square(obs-pred)/(pred+1.0);
          break;
        case 6: // laplace (double exponential)--check this
          if(var<0)                     var=log(1.0+square(var)) ;
// convert cv to variance on log scale
            else if(overall_var_pdf==2) var=log(1.0+var/square(pred)); //
convert observation variance to log scale
            else if(overall_var_pdf==0) var=1.0;                    //
automatic equal weighting
          var=sqrt(var);
          if(obs==pred) answer=log(var);
          else answer= log(sqrt(2.0))*sfabs((obs-pred)/var) + log(var) ;
          break;
        case 7: // gamma
          if(pred<=0) {answer=square(pred)*huge_number; break;}
```

```
          if(var<0)                     var=square(var)*square(pred);
// convert cv to variance on observation scale
            else if(overall_var_pdf==1) var=square(pred)*(mfexp(var)-1);
// convert log-scale variance to observation scale
            else if(overall_var_pdf==0) var=1.0;                    //
automatic equal weighting
          alph=square(pred)/var; beta=var/pred;
          answer= alph*log(beta)-(alph-1)*log(obs)+obs/beta+gammln(alph);
          break;
        case 8: // Fournier robust normal (variance must be calculated
externally)
          answer= 0.5*log(two_pi*var)-log( mfexp(-square(obs-
pred)/(2.0*var)) + 0.01 );
          break;
        default: // no such pdf accomodated
          cout << "The pdf must be either 1 (lognormal), 2 (normal), 3
(multinomial), 4 (Poisson), " << endl;
          cout << "                       5 (Chi-Square), 6 (Laplace), 7
(gamma) or 8 (robustified normal) " << endl;
          cout << "Presently it is " << pdf << endl;
          exit(0);
      }
    }
    return answer;
  }

 //----------------------------------------------------------------------
  dvariable neg_log_prior(dvariable obs,dvariable pred,double
lower,double upper,dvariable var,int pdf)
 //--------------------------------------------------------------------
  {
    int oldcount;
    dvariable answer;
    dvariable alph, beta;

    // compute generic pdf's
    switch(pdf) {
        case 1: // lognormal
          if(pred<=0) answer=square(pred)*huge_number;
          else if(obs/pred<=0) answer=square(obs/pred)*huge_number;
          else {
            if(var<0) var=log(1.0+square(var)) ;      // convert cv to
variance on log scale
            answer= 0.5*( square(log(obs/pred))/var + log(var) );
          }
          break;
        case 2: // normal
          if(var<0 && pred!=0) var=square(var*pred);      // convert cv
to variance on observation scale
          else if(var<0) var=var*tiny_number;             // cv not
really appropriate if predicted value close to zero
          answer= 0.5*( square(obs-pred)/var + log(var) );
          break;
        case 3: // uniform
```

```
          if(pred>=lower && pred<=upper) answer= log(upper-lower);
          else answer=huge_number;
          break;
        case 4: // uniform on log-scale
          if(pred>=lower && pred<=upper) answer= log(log(upper/lower));
          else answer=huge_number;
          break;
        case 5: // gamma
          if(pred==zero) answer=huge_number;
          else if(obs/pred<=0) answer=huge_number;
          else {
            if(obs<0) {pred=pred*-1.0; obs=obs*-1.0;} // negative of
parameter value considered gama distributed
            if(var<0) var=square(var*pred);           // convert cv to
variance on observation scale
            alph=pred*pred/var; beta=var/pred;
            answer= alph*log(beta)-(alph-
1)*log(obs)+obs/beta+gammln(alph);
          }
          break;
        case 6: // beta
          if(var<0) var=square(var*pred);      // convert cv to variance
on observation scale
          var=var/square(upper-lower);         // rescale variance to
beta (0,1) scale
          pred=(pred-lower)/(upper-lower);     // rescale prediction to
beta (0,1) scale
          obs=(obs-lower)/(upper-lower);       // rescale observation to
beta (0,1) scale
          alph=(pred*pred-pred*pred*pred-pred*var)/var; beta=alph*(1/obs-
1);
          if(pred>=0 && pred<=1) answer= (1-alph)*log(obs)+(1-
beta)*log(1-obs)-gammln(alph+beta)+gammln(alph)+gammln(beta);
          else answer=huge_number;
          break;
        default: // no such pdf accomodated
          cout << "The prior must be either 1(lognormal), 2(normal),
3(uniform), 5(gamma) or 6(beta)." << endl;
          cout << "Presently it is " << pdf << endl;
          exit(0);
    }
    return answer;
  }

 //-----------------------------------------------------------------------
  dvariable function_value(int nature, dvar_vector par_func, dvariable
obs)
 //-----------------------------------------------------------------------
{
    dvariable answer;

    // constants
    if(nature==1 || nature==13 || nature==14 || nature==50)
      return par_func(1);
```

```
    // polynomial of degree nature-1
    else if( nature<5) {
      answer=0.0;
      for(int j=1; j<nature; j++) {
        answer=answer+par_func(j)*pow(obs,j-1);
      }
      return answer+par_func(nature)*pow(obs,nature-1);  // trick to
avoid calculating the derivative of the final sum twice
    }

    // knife edge selectivity function
    else if( nature==5) {
      if(obs < par_func(1) ) return 0; else return 1;
    }

    // logisitic selectivity function
    else if( nature==6) {
      return 1/(1+mfexp(-(obs-par_func(1))/par_func(2)));
    }

    // gamma selectivity function
    else if( nature==7) {
      //return pow((mfexp(1-
obs/par_func(2))*obs/par_func(2)),1.0/square(par_func(1))-1.0);
        return
pow((obs/(par_func(1)*par_func(2))),par_func(1))*exp(par_func(1)-
(obs/par_func(2)));
    }

    // Chapman-Richards growth function (reduces to vonB with
par_func(4)=1
    else if( nature==8) {
      //if(par_func(5)<=0 || par_func(1) <=0 || (1-par_func(4)*mfexp(-
par_func(2)*(obs-par_func(3))))<=0) cout << "Error in growth parameters"
<< endl; //LIZ commented out 5/23/2004;
      // original line of code:
      //return
mfexp(log(par_func(5))+par_func(6)*(log(par_func(1))+log(1-
par_func(4)*mfexp(-par_func(2)*(obs-par_func(3))))/par_func(4))) ;
      answer=par_func(1)*(1-par_func(4)*exp(-
par_func(2)/par_func(4)*(obs-par_func(3)))) ; // von bert
      answer=par_func(7)*answer+par_func(8) ;  // convert  units
      answer=par_func(5)*pow(answer,par_func(6)); // convert  L to W
      return answer;
    }

    // Gompertz growth function
    else if( nature==9) {
      return par_func(1)*mfexp(-mfexp(-par_func(2)*(obs-par_func(3))));
    }

    // Beverton and Holt asymptotic function
    else if( nature==10) {
```

```
      return par_func(1)*obs*par_func(2)/(one+(par_func(2)-one)*obs);
    }

    // Ricker dome-shaped function
    else if( nature==11) {
      if(par_func(2)>0) return mfexp(log(par_func(1))+log(obs)+(one-
obs)*log(par_func(2)));
      else return mfexp(log(par_func(1))+log(obs)+(one-obs)*log(1));

    }

    // power function y=a*x**b
    else if( nature==12) {
      return par_func(1)*pow(obs,par_func(2));
    }

    // double logistic function     (LIZ added 8/18/2005)
     else if( nature==15)  {
        return (1/(1+mfexp(-(obs-par_func(1))/par_func(2))))*(1-
(1/(1+mfexp(-(obs-par_func(3))/par_func(4)))))/par_func(5) ;
    }

    // exponential function of form:  par_func(1)*exp(par_func(2)*obs)
     else if ( nature==16)  {
        return  par_func(1)*exp(par_func(2)*obs) ;
    }

    // invalid function type
    else {
      cout << "No such function type accomodated" << endl; exit(0);
      return answer;
    }
  }

 //----------------------------------------------------------------------
  dvariable get_variance(dvariable pred,dvariable var,int pdf,int
overall_var_pdf)
 //----------------------------------------------------------------------
{
    switch(pdf) {
      case 1: // autocorrelated lognormal
        if(var<0)                     var=log(1.0+square(var)) ;
// convert cv to variance on log scale
          else if(overall_var_pdf==2) var=log(1.0+var/square(pred));
// convert observation variance to log scale
          else if(overall_var_pdf==0) var=1.0;
// automatic equal weighting
        break;
      case 2: // autocorrelated normal
        if(var<0)                     var=square(var)*square(pred);
// convert cv to variance on observation scale
          else if(overall_var_pdf==1) var=square(pred)*(mfexp(var)-1);
// convert log-scale variance to observation scale
```

```
          else if(overall_var_pdf==0) var=1.0;
// automatic equal weighting
        break;
      default: // no such pdf accomodated
        exit(0);
    }
    return value(var);
  }

 //-----------------------------------------------------------------------
  dvariable spr(dvar_vector pp, dvar_vector ww, dvar_vector mm,
dvar_vector ss, dvariable ff, dvariable tau ,int na)
 //  Computes equilibrium spawn per recruit
 //-----------------------------------------------------------------------
{
    dvariable answer;
    dvariable survive;
    dvariable zz;
    survive=1;
    answer=0;
    for (a=1; a<na; a++) {
      zz=mm(a)+ff*ss(a);
      answer+=pp(a)*ww(a)*mfexp(-zz*tau)*survive;
      survive=survive*mfexp(-zz);
    }
    zz=mm(na)+ff*ss(na);
    return answer+pp(na)*ww(na)*mfexp(-zz*tau)*survive/(1-mfexp(-zz));
  }

 //------------------------------------------------------------------------
dvariable ypr(dvar_vector ww, dvar_vector mm, dvar_vector ss, dvariable
ff,int na)
 //  Computes equilibrium yield per recruit
 //------------------------------------------------------------------------
  {
    dvariable answer;
    dvariable survive;
    dvariable zz;
    survive=1;
    answer=0;
    for (a=1; a<na; a++) {
      zz=mm(a)+ff*ss(a);
      answer+=ww(a)*ss(a)*(1-mfexp(-zz))*survive/zz;
      survive=survive*mfexp(-zz);
    }
    zz=mm(na)+ff*ss(na);
    return ff*(answer+ww(na)*ss(na)*survive/zz);
  }

 //------------------------------------------------------------------------
  dvariable equilibrium_ssb(int nature, dvar_vector par_func, dvariable
sprvalue, dvariable spr0)
 //  Computes equilibrium spawning biomass
 //-----------------------------------------------------------------------
```

```
  {
    dvariable spratio;
    if(sprvalue<=zero) sprvalue=tiny_number;
    spratio=sprvalue/spr0;
    if(par_func(2)>1.0/spratio) {
      // Beverton and Holt asymptotic function
      if( nature==10)      return spr0*par_func(1)*(par_func(2)*spratio-
1.0)/(par_func(2)-1.0);      // Beverton and Holt asymptotic function
      else if( nature==11) return spr0*par_func(1)*(1.0 +
log(spratio)/log(par_func(2)));            // Ricker dome
    }
    else
    return -9.0;
  }

 //-----------------------------------------------------------------------
  dvariable goldensection(int typ, dvariable bf, dvar_vector ww,
dvar_vector mm, dvar_vector ss, int na, dvar_vector mat, dvar_vector fec,
dvariable tau, dvariable spr00, int sr_nature, dvar_vector par_func)
 // vars being passed:        ...        ref pt          weight
mort          s_equil.      nages      maturity          fecundity
spawn time       spr0         sr nature        sr-pars
 //  Computes F's at maximum equilibrium yield per recruit and MSY
 //-----------------------------------------------------------------------
  {
    dvariable y1, y2, f0, f1, f2, f3, af, cf, sprtemp,sprt, slope0;
    double g1, g2;
    int iter;
    af=0.0001; cf=3.0; g1=0.618034; g2=0.381966;
    if(typ==i_two) {
      for (iter=1; iter<29; iter++) {
        cf=cf-0.1;
        sprt=spr(mat, fec, mm, ss, cf, tau, na);
        sprtemp=spr(mat, fec, mm, ss, cf, tau, na)/spr00;
y1=equilibrium_ssb(sr_nature,par_func,sprt,spr00)/sprt;
        if(y1>0) break;
      }
    }
    if(bf>(cf-0.1)) bf=bf-(bf-cf+0.1);
    f0=af; f3=cf;

    if(fabs(cf-bf)>fabs(bf-af)) { f1=bf; f2=bf+g2*(cf-bf); }
    else { f2=bf; f1=bf-g2*(bf-af); }
    y1= -ypr(ww, mm, ss, f1, na); y2= -ypr(ww, mm, ss, f2, na); // yield
per recruit
    if(typ==3) { slope0=0.1*ypr(ww, mm, ss, 0.001, na);
y1=fabs(slope0+y1+ypr(ww, mm, ss, f1-0.001, na));
y2=fabs(slope0+y2+ypr(ww, mm, ss, f2-0.001, na)); }
    if(typ==i_two) {
       sprt=spr(mat, fec, mm, ss, f1, tau, na) ;
       sprtemp=spr(mat, fec, mm, ss, f1, tau, na)/spr00;
y1=y1*equilibrium_ssb(sr_nature,par_func,sprt,spr00)/sprt;
       sprtemp=spr(mat, fec, mm, ss, f2, tau, na)/spr00;
y2=y2*equilibrium_ssb(sr_nature,par_func,sprt,spr00)/sprt;
```

```
    }
    for (iter=1; iter<21; iter++) {
      if(y2<y1) {
        f0=f1; f1=f2; f2=g1*f1+g2*f3; y1=y2; y2= -ypr(ww, mm, ss, f2, 
na);
        if(typ==3) y2=fabs(slope0+y2+ypr(ww, mm, ss, f2-0.001, na));
        if(typ==i_two) {sprt=spr(mat, fec, mm, ss, f2, tau, na); 
sprtemp=spr(mat, fec, mm, ss, f2, tau, na)/spr00; 
y2=y2*equilibrium_ssb(sr_nature,par_func,sprt,spr00)/sprt; }
      }
      else      {
        f3=f2; f2=f1; f1=g1*f2+g2*f0; y2=y1; y1= -ypr(ww, mm, ss, f1, 
na);
        if(typ==3) y1=fabs(slope0+y1+ypr(ww, mm, ss, f1-0.001, na));
        if(typ==i_two) {sprt=spr(mat, fec, mm, ss, f1, tau, na); 
sprtemp=spr(mat, fec, mm, ss, f1, tau, na)/spr00; 
y1=y1*equilibrium_ssb(sr_nature,par_func,sprt,spr00)/sprt; }
      }
    }
    if(y1<y2) return f1;
    else return f2;
  }
```