

Appendix 2 Herring bioenergetic model FORTRAN code for the base case.

```
C -----
C Bioenergetic herring model based on the paper of Rudstam (1988).
C Exploring the dynamics of herring consumption in the Baltic:
C Applications of an energetic model of fish growth.
C Kieler Meeresforsch Sonderth 6:312-322.
C
C Originally coded as difference equation in FORTRAN by Kenny Rose 26 Dec 01
C
C Corrected and changed to differential equation with Euler and
C Runge Kutta numerical integration scheme
C
C           01/03/02 Bernard A. Megrey
C
C Added observed and predicted size at age data
C           01/12/02 Bernard A. Megrey
C
C Added YOY formulations per Arrhenius (1998)
C           01/22/02 Bernard A. Megrey
C
C All relic code removed for general distribution at
C Nemuro 2002 workshop
C           01/25/02 Bernard A. Megrey
C
C -----
  program NemuroHerring

    include 'stuff.cmn'
    include 'state.cmn'
    include 'sizeaa.cmn'
    REAL NYEARS, NSTEPS, NSTEP

    OPEN(UNIT=11,FILE='nemuro.txt',STATUS='unknown')
    OPEN(UNIT=8,FILE='compareEuler.out',STATUS='unknown')
    OPEN(UNIT=9,FILE='sizeatage.out',STATUS='unknown')

C
C-----read in the 3 zoop groups from Nemuro output (7th, 9th and 11th columns)
C
  do 45 ii=1,731
    READ(11,999)id(ii),zop1(ii),zop2(ii),zop3(ii)
  999  FORMAT(1x,i3,1x,5(13x,1x),2(e13.6,1x,13x,1x),e13.6)

C
C----- take the first year for now
C
    IF(ii.le.365)then
      zoop1(ii)=zop1(ii)
      zoop2(ii)=zop2(ii)
      zoop3(ii)=zop3(ii)
    endif
```

```

45  continue

C-----convert Nemuro zoop in uM N/L to g ww/m3
C----- tt1 is conversion from uM N/liter to g ww/m3
C----- 14 ug N/uM * 1.0e-6 g/ug * 1 g dw/0.07 g N dw
C----- * 1 g ww/0.2 g dw *
C----- 1.e3 liters/m3
      do 55 i=1,365
          tt1=14.0*1.0e-6*(1.0/0.07)*(1.0/0.2)*1.0e3
          zoop1(i)=zoop1(i)*tt1
          zoop2(i)=zoop2(i)*tt1
          zoop3(i)=zoop3(i)*tt1

C----initial weight and age of newly metamphosed herring
      x(1)=0.2
      iage=0
      maxage=0
55  continue
C
C --- number of state variables
C
      nstate=1
C
C - time initialization
C
      TZERO = 0.0
      NYEARS = 11.0
      NSTEPS = 100.0
      TEND = 365.0*NYEARS
      dt= 1/NSTEPS
      TPRINT = 1.0
      TEPS = 1.0E-05

C
C --- MAIN TIME LOOP
C
      NSTEP=0.0
      DO WHILE (time .LT. TEND)
          NSTEP=NSTEP+1.0
          time= TZERO + NSTEP*dt
          CALL EULER(x,xdot,nstate,dt,time)
C      CALL KUTTA(x,xdot,nstate,dt,time)
          iday=int(amod(time,365.0))+1
          iyr=int(time/365.)+1
C
C----- update age every time iday 365 goes by and
C----- after NSTEP have gone by
C----- the NSTEPS test is to avoid incrementing iage
C----- every dt
C
          IF(iday.eq.365 .and. amod(NSTEP,NSTEPS).eq.0) then

```

```

    iage=iage+1
    jpage(iage)=iage
    psizeaa(iage)=x(1)
    if(iage .gt. maxpage) maxpage=iage
endif
C
C --- check for time to print
C
    IF(ABS(AMOD(time,TPRINT)) .LT. TEPS) THEN
    write(8,1001) time, x(1), wtemp, gcmx
1001  format(1x,4(f9.3,1x))
C    ENDIF
    ENDIF
    END DO
    kage=min(maxoage,maxpage)
C
C --- write predicted and observed size-at-age output
C
    do 73 i=1,kage
        write(9,1002) i, psizeaa(i), osizeaa(i)
73  continue
1002  format(1x,i4,2(1x,f9.3))

    close(8)
    close(9)
    STOP
    END
    SUBROUTINE DER(x,xdot,time)

C-----
C
C Herring bioenergetics differential equation process.
C Prey base are in units of micromoles N /m^3 and are
C converted
C via conversion factors
C
C programmed by BAM 01/05/02
C
C Added YOY formulations per Arrhenius (1998)
C 01/22/02 Bernard A. Megrey
C
C-----

    include 'stuff.cmn'
    include 'state.cmn'

C    WRITE(*,*) 'IN DER', time

    iday=int(amod(time,365.0))+1
    iyr=int(time/365.0)+1

```

```

C
C zero out xdot
C
      DO 15 i=1,nstate
        xdot(i)=0.0
      15 CONTINUE
C---- start age-0 on day 200
C---- jday is julian day (1,..., 400) but goes past 365
C---- iday is counter for day in model simulation
C---- jjday is julian day (i.e., jday reset for >365)
C---- I start on day 200; i you want different then change 200
C--- below and 165, which 365 minus the start day.
      jday=iday+200
      IF(jday.le.365)then
        jjday=jday
      else
        jjday=iday-165
      endif

C
C----- generate daily temperatures for a year -- made up
C
      t1=float(jjday)
      t2=12.75-10.99*cos(0.0172*t1)-6.63*sin(0.0172*t1)
      wtemp=t2-5.0
      IF(wtemp.le.1.0)wtemp=1.0
C      write(*,*) "wtemp",wtemp
C50  continue

C
C Herring = x(1)
C
C
C
C----- set vulnerabilities and k values for 3 zoop groups
C
      vul(1)=1.0
      vul(2)=1.0
      vul(3)=1.0

      k(1)=100.0
      k(2)=10.0
      k(3)=100.0
C
C-----if using constant p rather than functional response, set p
C-----here
C      p=0.6

C----- loop over years
C      do 100 iyr=1,9

```

```

C---- loop over days for each year
C    do 200 iday=1,365
C      write(*,*) 'iday jday jjday', iday,jday,jjday

C-----iday is running value of days in simulation (1,....., 2000)
C      iday=(iyr-1)*365+iday

      tt1=1.0/x(1)
      t1=0.0033*tt1**0.227
C      write(*,*) 't1 tt1', x(1), t1, tt1

c----***this is the new stuff from Arrhenius (1998) for YOY only***
c----- The 5.258 puts resp is in units of g zoop/g fish/day
c----- [13560 joules/gram oxygen]/4.18 joules/cal = 3244 cal/gO2
c-----[2580 joules/gram zoop]/4.18 joules/cal = 617 cal/g zoop
c----- 3244/617 = 5.258

      IF(iage.eq.0)then
        IF(wtemp.le.15.0)then
          v=5.76*EXP(0.0238*wtemp)*x(1)**0.386
        endif
        IF(wtemp.gt.15.0)then
          v=8.6*x(1)**0.386
        endif
        a=EXP((0.03-0.0*wtemp)*v)
        resp=t1*EXP(0.0548*wtemp)*a*5.258
      endif
c-----***back to the old equations for respiration for age-1 and
C-----older ***
      IF(iage.ge.1)then
        IF(wtemp.le.9.0)then
          u=3.9*x(1)**0.13*EXP(0.149*wtemp)
        else
          u=15.0*x(1)**0.13
        endif
        resp=t1*EXP(0.0548*wtemp)*EXP(0.03*u)*5.258
      endif
C      write(*,*) 'after new stuff'
C      IF(wtemp.lt.9.0)then
C        u=3.9*x(1)**0.13* exp(0.149*wtemp)
C      else
C        u=15.0*x(1)**0.13
C      endif
c-- --- 13,560 joules/g O2 1 cal/4.18 joules 1 g ww/5533 cal
C      resp=t1*EXP(0.0548*wtemp)*EXP(0.03*u)*0.59

C
C----- Thornton and Lessem (1978)temperature effect
c----**Arrhenius (1998) for age-0 changed te4 from 25 to 23 degrees***
      IF(iage.eq.0)then
        xk1=0.1

```

```

    xk2=0.98
    xk3=0.98
    xk4=0.01

    te1=1.0
    te2=15.0
    te3=17.0
    te4=23.0
endif
C
IF(iage.eq.1)then
    xk1=0.1
    xk2=0.98
    xk3=0.98
    xk4=0.01

    te1=1.0
    te2=15.0
    te3=17.0
    te4=25.0
endif
IF(iage.gt.1)then
    xk1=0.1
    xk2=0.98
    xk3=0.98
    xk4=0.01

    te1=1.0
    te2=13.0
    te3=15.0
    te4=23.0
endif
C
C- non age dependent temperature effect on consumption
C    xk1=0.1
C    xk2=0.98
C    xk3=0.98
C    xk4=0.01
C    te1=1.0
C    te2=13.0
C    te3=15.0
C    te4=23.0

tt5=(1.0/(te2-te1))
t5=tt5 * alog(0.98*(1.0-xk1)/(0.02*xk1))
t4=exp(t5*(wtemp-te1))

tt7 = 1.0/(te4-te3)
t7=tt7*alog(0.98*(1.0-xk4)/(0.02*xk4))
t6=exp(t7*(te4-wtemp))

```

```

gcta=(xk1*t4)/(1.0+xk1*(t4-1.0))
gctb=xk4*t6/(1.0+xk4*(t6-1.0))
gctemp=gcta * gctb
gcmax=0.642*tt1**0.256*gctemp
C
C----- no tempeature effect
C      gcmax=0.642*tt1**0.256*1.0

C----- either use fixed p or call functional response
C      con=p*gcmax
C      write(*,*) 'der time iday iyr jjday zoop123',time,iday,
C      iyr,
C      jjday, zoop1(jjday),zoop2(jjday), zoop3(jjday)
cnum=zoop1(jjday)*vul(1)/k(1)+zoop2(jjday)*vul(2)/k(2)
$      +zoop3(jjday)*vul(3)/k(3)
      c1=gcmax*zoop1(jjday)*vul(1)/k(1)
      c2=gcmax*zoop2(jjday)*vul(2)/k(2)
      c3=gcmax*zoop3(jjday)*vul(3)/k(3)
      con1=c1/(1.0+cnum)
      con2=c2/(1.0+cnum)
      con3=c3/(1.0+cnum)
      con=con1+con2+con3

C
C --- for comparison to Kenny's version
C
C      con=0.75*gcmax
C
C --- to tune to observed size at age data
C      con=0.48*gcmax
C
C --- egestion
C
C      f=0.16*con
C
C --- excretion
C      e=0.1*(con-f)
C
C --- Specific Dynamic Action
C
c----- *****Arrhenius (1998) changed SDA from 17.5% to 15% *****
      IF(iage.eq.0)sda=0.15*(con-f)
      IF(iage.ge.1)sda=0.175*(con-f)

c----- J/g ww 1 cal=4.18 J
C      con1=con*2580.0/5533.0
C      write(*,*) 'der',con1,resp,xdot(1)
C
C --- bioenergetics differential equation
C
      xdot(1)=(con- resp-f-e-sda)*x(1)*2580./5533.

```

```

IF(wtemp.le.1.0)xdot(1)=0.0
  t1=float(jjday)
  if(amod(t1,365.0).ge.152.0.and.amod(t1,365.0).le.156.0) then
write(*,*) 'in spawn'
  xdot(1)=(con-resp-f-e-sda-0.20)*x(1)*2580./5533.
endif

```

```

c----- update age every time day 365 goes by
c   IF(iday.eq.365) then
c       iage=iage+1
c       jage(iage)=iage
c       psizeaa(iage)=x(1)
c       write(*,*)'iday, iyr, jjday, iage, maxage',iday, iyr,
c   $   jjday, iage, maxage
c       if(iage .gt. maxage) maxage=iagec
c       endif

```

```

C   WRITE(*,*) 'OUT OF DER'
RETURN
END

```

```

SUBROUTINE EULER(x,xdot,nstate,dt,time)

```

```

C-----

```

```

C
C USE THE EULER METHOD TO SOLVE A SYSTEM OF NONLINEAR
C DIFFERENTIAL EQUATIONS. A SUBROUTINE DER IS NEEDED
C TO COMPUTE THE DERIVATIVES OF THE STATE VARIBALES
C
C X - STATE VARIABLE ARRAY
C XDOT - ARRAY OF DERIVATIVES OF STATE ARIABLES
C NSTATE - NUMBER OF STATE VARIABLES
C DT - TIME STEP
C TIME - CURRENT TIME
C
C-----

```

```

include 'stuff.cmn'
include 'state.cmn'

```

```

INTEGER I

```

```

CALL DER(x,xdot,time)

```

```

DO 10 i=1, nstate
  x(i)=x(i) + dt * xdot(i)
10 CONTINUE

```

```

RETURN
END

```

```

SUBROUTINE KUTTA(x,xdot,nstate,dt,time)

```

```

C-----
C
C USE THE 4TH ORDER RUNGE KUTTA METHOD TO SOLVE A SYSTEM OF NONLINEAR
C DIFFERENTIAL EQUATIONS. A SUBROUTINE DER IS NEEDED
C TO COMPUTE THE DERIVATIVES OF THE STATE VARIABLES
C
C X - STATE VARIABLE ARRAY
C XDOT - ARRAY OF DERIVATIVES OF STATE VARIABLES
C NSTATE - NUMBER OF STATE VARIABLES
C DT - TIME STEP
C TIME - CURRENT TIME
C
C programmed by Bernard A. Megrey 01/06/02
C
C-----
      include 'stuff.cmn'
      include 'state.cmn'

      INTEGER I
      REAL SUMDX(16), DTO2, XPLUS(16)
C      WRITE(*,*) 'IN KUTTA'

      DTO2 = dt/2.0

      CALL DER(x,xdot,time)

      DO 10 I=1, nstate
          XPLUS(I) = x(I) + DTO2 * xdot(I)
          SUMDX(I) = xdot(I)
10 CONTINUE

      CALL DER(XPLUS,xdot,time)

      DO 20 I=1, nstate
          XPLUS(I) = x(I) + DTO2 * xdot(I)
          SUMDX(I) = SUMDX(I) + 2.0 * xdot(I)
20 CONTINUE

      CALL DER(XPLUS,xdot,time)

      DO 30 I=1, nstate
          XPLUS(I) = x(I) + dt * xdot(I)
          SUMDX(I) = SUMDX(I) + 2.0 * xdot(I)
30 CONTINUE

      CALL DER(XPLUS,xdot,time)

      DO 40 I=1, nstate
          SUMDX(I) = SUMDX(I) + xdot(I)
          x(I) = x(I) + dt * SUMDX(I) / 6.0

```

```
40 CONTINUE
C  WRITE(*,*) 'OUT OF KUTTA'
  RETURN
  END
```

```
C=====
C
C Include file: state.cmn
C state variable declarations
C
C BAM 1/02/02
C-----
  REAL*4 dt, time, x(1), xdot(1)
  INTEGER*4 nstate
```

```
C=====
C
C Include file: stuff.cmn
C miscellaneous common block variables
C
C BAM 1/2/02
C-----
  COMMON /ZOO/ zoop1(365), zoop2(365), zoop3(365)
  COMMON /ISV/ wtemp, con, gcmx,f, e ,sda, con1, resp
  COMMON /TIMER/ iday, jday, iyr, iage
  REAL*4 zoop1,zoop2,zoop3,k(3),vul(3)
  REAL*4 zop1(731),zop2(731),zop3(731)
  INTEGER*4 id(731)
```

```
C=====
C
C Include file: sizeaa.cmn
C Size at age common block
C
C BAM 1/06/02
C-----
  COMMON /SIZEAA/ joage(25),jpage(25),psizeaa(25),osizeaa(25),
  $           maxoage,maxpage
C --- joage Observed age
C --- jpage Predicted age
C --- psizeaa Predicted size at age
C --- osizeaa Observed size at age

  INTEGER*4 joage, jpage, maxoage, maxpage
  REAL*4 psizeaa, osizeaa
  DATA maxoage /11/
  DATA joage /1,2,3,4,5,6,7,8,9,10,11,14*0/
```

C

C --- observed size at age Pacific herring data (wt) from the C

C --- 1973 year class

C --- as supplied by Doug Hay

C

DATA osizeaa /8.75,63.0,87.66,124.13,139.26,152.98,177.43,
\$ 185.78,187.64,195.0,208.73,14*0.0/

Appendix 3 Fully customized herring subroutine. See Appendix 2 for main program and common block including files.

SUBROUTINE DER(x,xdot,time)

```

C-----
C
C Herring bioenergetics differential equation process.
C Prey base are in units of micromoles N /m^3 and are converted
C via conversion factors
C
C   programmed by B.A. Megrey 01/05/02
C
C Added YOY formulations per Arrhenius 1998
C       01/22/02 B.A. Megrey

C modifications and customizations by F.E.Werner and R.A.Klumb 1/26/02
C while at the Nemuro workshop
C
C-----

      include 'stuff.cmn'
      include 'state.cmn'

C
C- calculate date and year
C
      iday=int(amod(time,365.0))+1
      iyr=int(time/365.)+1
C
C zeroout xdot
C
      DO 15 i=1,nstate
         xdot(i)=0.0
      15 CONTINUE
c
c---- start age-0 on day 200
c----   jday is julian day (1,..., 400) but goes past 365
c----   iday is counter for day in model simulation
c----   jjday is julian day (i.e., jday reset for >365)
c----   I start on day 200; i you want different then change 200
c----   below and 165, which 365 minus the start day.
      jday=iday+200
      IF(jday.le.365)then
         jjday=jday
      else
         jjday=iday-165
      endif
C
C----- generate daily temperatures for a year -- made up
C

```

```

t1=float(jjday) ! BaseCase T
t2=12.75-10.99*cos(0.0172*t1)-6.63*sin(0.0172*t1) ! BaseCase T
wtemp=t2-5.0 ! BaseCase T
IF(wtemp.le.1.0)wtemp=1.0 ! BaseCase T
  pi=acos(-1.)
  wtemp=7.717+(5.6796*0.5*(1.-cos(2.*pi*(t1-30.)/365.)))
C
C--- Herring weight state variable = x(1)
C--- weight affect on respiration
C
  tt1=1.0/x(1)
  t1=0.0033*tt1**0.227
C
C --- *****this is the new stuff from Arrhenius (1998) for YOY only*****
C --- The 5.258 puts resp (g oxygen/fish) into units of g zoop/g fish/day
C --- [13560 joules/gram oxygen]/4.18 joules/cal = 3244 cal/gO2
C --- [2580 joules/gram zoop]/4.18 joules/cal = 617 cal/g zoop
C --- so respiration in grams/oxygen/g fish/day is multiplied
C --- by 3244/617 = 5.258
C --- to get food energy equivalents of a gram of oxygen respired
C
c IF(iage.eq.0)then ! BASE
c IF(wtemp.le.15.0)then ! BASE
c v=5.76*EXP(0.0238*wtemp)*x(1)**0.386 ! BASE
c endif ! BASE
c IF(wtemp.gt.15.0)then ! BASE
c v=8.6*x(1)**0.386 ! BASE
c endif ! BASE
c a=EXP((0.03-0.0*wtemp)*v) ! BASE
c resp=t1*EXP(0.0548*wtemp)*a*5.258 ! BASE
c endif ! BASE
C
IF(iage.eq.0)then ! R.A. Klumb (26 Jan 2002)
  t1=0.00528*tt1**0.007 ! R.A. Klumb (26 Jan 2002)
  resp=t1*EXP(0.083*wtemp)*5.258 ! R.A. Klumb (26 Jan 2002)
end if ! R.A. Klumb (26 Jan 2002)
C
C --- *****back to the old equations for respiration for age-1 and
C --- older*****
C
IF(iage.ge.1)then
C Base IF(wtemp.le.9.0)then
IF(wtemp.le.9.655)then
u=3.9*x(1)**0.13*EXP(0.149*wtemp)
else
u=15.0*x(1)**0.13
endif
resp=t1*EXP(0.0548*wtemp)*EXP(0.03*u)*5.258
endif
C
C --- Thornton and Lessem temperature effect

```

C --- age dependent values
C --- *****Arrhenius (1998)for age-0 changed te4 from 25 to 23 degrees*****
C

```
IF(iage.eq.0)then
  xk1=0.1
  xk2=0.98
  xk3=0.98
  xk4=0.01
```

```
C Base      te1=1.0
C Base      te2=15.0
C Base      te3=17.0
C Base      te4=23.0
```

```
te1=8.0
te2=10.897
te3=11.310
te4=12.552
endif
```

C

```
IF(iage.eq.1)then
  xk1=0.1
  xk2=0.98
  xk3=0.98
  xk4=0.01
```

```
C Base      te1=1.0
C Base      te2=15.0
C Base      te3=17.0
C Base      te4=25.0
```

```
te1=8.0
te2=10.897
te3=11.310
te4=12.966
endif
```

```
IF(iage.gt.1)then
  xk1=0.1
  xk2=0.98
  xk3=0.98
  xk4=0.01
```

```
C Base      te1=1.0
C Base      te2=13.0
C Base      te3=15.0
C Base      te4=23.0
```

```
te1=8.0
te2=10.483
te3=10.897
```

```

    te4=12.552
endif

tt5=(1.0/(te2-te1))
t5=tt5 * alog(0.98*(1.0-xk1)/(0.02*xk1))
t4=exp(t5*(wtemp-te1))

tt7 = 1.0/(te4-te3)
t7=tt7*alog(0.98*(1.0-xk4)/(0.02*xk4))
t6=exp(t7*(te4-wtemp))

gcta=(xk1*t4)/(1.0+xk1*(t4-1.0))
gctb=xk4*t6/(1.0+xk4*(t6-1.0))
gctemp=gcta * gctb
gcmax=0.642*tt1**0.256*gctemp
C
C --- multispecies functional response
C --- usse either this or adjust little p
C
C----- set vulnerabilities and k values for 3 zoop groups
C
    vul(1)=1.0
    vul(2)=1.0
    vul(3)=1.0

    k(1)=0.3638
    k(2)=0.0364
    k(3)=0.3638

c    k(1)=2000.0
c    k(2)=200.0
c    k(3)=2000.0

cnum=zoop1(jjday)*vul(1)/k(1)+zoop2(jjday)*vul(2)/k(2)
$    +zoop3(jjday)*vul(3)/k(3)
    c1=gcmax*zoop1(jjday)*vul(1)/k(1)
    c2=gcmax*zoop2(jjday)*vul(2)/k(2)
    c3=gcmax*zoop3(jjday)*vul(3)/k(3)
    con1=c1/(1.0+cnum)
    con2=c2/(1.0+cnum)
    con3=c3/(1.0+cnum)
    con=con1+con2+con3
C
C-----if using constant p rather than functional response, set p here
C --- to tune to observed size at age data. If using functional response
C --- comment the next line out
C
    con=0.6375*gcmax
C
C --- egestion
C

```

```

f=0.16*con          ! Base Case
IF(iage.eq.0)f=0.125*con      ! Age-dependent – R.A. Klumb (26 Jan 2002)
C
C --- excretion
e=0.1*(con-f)        ! Base Case
IF(iage.eq.0)e=0.078*con    ! Age-dependent – R.A. Klumb (26 Jan 2002)
C
C --- Specific Dynamic Action
C
c----- *****Arrhenius (1998) age dependent SDA from 17.5% to 15% *****
IF(iage.eq.0)sda=0.15*(con-f) ! Base Case
IF(iage.eq.0)sda=0.125*(con-f) ! Age-dependent – R.A. Klumb (26 Jan 2002)
IF(iage.ge.1)sda=0.175*(con-f)
C
C --- use the ratio of calories/g of zoop (2580) to calories/g of fish (5533)
C
C --- bioenergetics differential equation - constant energy density for herring
C
C      xdot(1)=(con-resp-f-e-sda)*x(1)*2580./5533.    ! Base Case
C
C include seasonal variation of energy density for .ge. 2 yr olds
C
if(iage.ge.2)then
  enMar1=5750.
  jdMar1=60
  enOct1=9800.
  jdOct1=274
  if(jjday.lt.60)then
    delen=(enMar1-enOct1)/151
    en=enOct1+(90+jjday)*delen
  end if
  if(jjday.ge.60.and.jjday.lt.274)then
    delen=(enOct1-enMar1)/(jdOct1-jdMar1)
    en=enMar1+(jjday-jdMar1)*delen
  end if
  if(jjday.ge.274)then
    delen=(enMar1-enOct1)/151
    en=enOct1+(jjday-jdOct1)*delen
  end if
  else
    en=4460.
  end if
  xdot(1)=(con-resp-f-e-sda)*x(1)*2580./en

  IF(wtemp.le.1.0)xdot(1)=0.0
C
C --- Spawning section. Assume loose 20% of body weight/day
C      t1=float(jjday)
c      if(amod(t1,365.0) .ge. 152.0 .and.
c &      amod(t1,365.0) .le. 156.0) then
c      xdot(1)=(con-resp-f-e-sda-0.20)*x(1)*2580./5533.

```

```
c      endif
```

```
RETURN  
END
```

Appendix 4 NEMURO.FISH (NEMURO FORTRAN code supplied by Yasuhiro Yamanaka) with the herring bioenergetic model (base case) (supplied by Bernard Megrey and Ken Rose). The herring model is linked to NEMURO in a one-way static link.

```

|*****
! NEMURO model Jun 13, 2002 written by Yasuhiro Yamanaka
|*****
program NEMURO.FISH
  implicit none
! ..... Control for Time Ingration .....
  character(19) :: Cstart = '0001/07/20 00:00:00' ! Starting date
  character(19) :: Cend = '0011/07/21 00:00:00' ! Ending date
  character(19) :: Cstep = '0000/00/00 01:00:00' ! Time step
  character(19) :: Cmon = '0000/00/01 00:00:00' ! Monitor Interval
  character(19) :: CTime
  real(8) :: dt, TTime, Tbefore, Season, Tmon
  integer :: Iyr, Imon, Iday, Ihour, Imin, Isec
! ..... scale conversion .....
  real(8),parameter :: d2s = 86400.0d0 ! day ---> sec
  real(8),parameter :: mcr = 1.0d-6 ! micro
! ..... Prognostic Variables (with initial conditions) and Thier Source Term .....
  real(8) :: TPS = 0.1D-6, QPS ! Small Phytoplankton [molN/l]
  real(8) :: TPL = 0.1D-6, QPL ! Large Phytoplankton
  real(8) :: TZS = 0.1D-6, QZS ! Small Zooplankton
  real(8) :: TZL = 0.1D-6, QZL ! Large Zooplankton
  real(8) :: TZP = 0.1D-6, QZP ! Pradatory Zooplankton
  real(8) :: TNO3 = 5.0D-6, QNO3 ! Nitrate
  real(8) :: TNH4 = 0.1D-6, QNH4 ! Ammmonium
  real(8) :: TPON = 0.1D-6, QPON ! Particulate Organic Nitrogen
  real(8) :: TDON = 0.1D-6, QDON ! dissolved Organic Nitrogen
  real(8) :: TSiOH4 = 10.0D-6, QSiOH4 ! Silicate
  real(8) :: TOpal = 0.1D-7, QOpal ! Particulate Opal
! ..... Prognostic Variables (with initial conditions) and Thier Source Term .....
! real(8) :: THrr = 0.2D0, QHrrl ! Particulate Opal
! ..... Light Condition Parameters .....
  real(8),parameter :: alpha1 = 4.0D-2 ! Light Dissipation coefficient of sea water[/m]
  real(8),parameter :: alpha2 = 4.0D4 ! PS+PL Selfshading coefficientS+PL [l/molN/m]
  real(8),parameter :: IoptS = 0.15D0 ! PS Optimum Light Intensity S [ly/min]
  real(8),parameter :: IoptL = 0.15D0 ! PL Optimum Light Intensity [ly/min]
  integer,parameter :: LLN = 10 ! Number of sublayer for calculating of Lfc
  real(8) :: LfcS ! Light factor for PS
  real(8) :: LfcL ! Light factor for PL
  real(8) :: kappa, Lint, dLint, LfcUS, LfcUL, LfcDS, LfcDL
  integer :: L
! ..... biological Parameters .....
  real(8),parameter :: VmaxS = 0.4D0/d2s ! PS Maximum Photosynthetic rate @0degC [/s]
  real(8),parameter :: KNO3S = 1.0D-6 ! PS Half saturation constant for Nitrate [molN/l]
  real(8),parameter :: KNH4S = 0.1D-6 ! PS Half saturation constant for Ammonium [molN/l]
  real(8),parameter :: PusaiS = 1.5D6 ! PS Ammonium Inhibition Coefficient [l/molN]
  real(8),parameter :: KGppS = 6.93D-2 ! PS Temp. Coeff. for Photosynthetic Rate [degC]
  real(8),parameter :: MorPS0 = 5.85D4/d2s ! PS Mortality Rate @0degC [/s]

```

```

real(8),parameter :: KMorPS = 6.93D-2 ! PS Temp. Coeff. for Mortality [/degC]
real(8),parameter :: ResPS0 = 0.03D0/d2s ! PS Respiration Rate at @0degC [s]
real(8),parameter :: KResPS = 0.0519D0 ! PS Temp. Coeff. for Respiration [degC]
real(8),parameter :: GammaS = 0.135D0 ! PS Ratio of Extracell. Excret. to Photo. [(nodim)]
real(8),parameter :: VmaxL = 0.8D0/d2s ! PL Maximum Photosynthetic rate @0degC [s]
real(8),parameter :: KNO3L = 3.00D-6 ! PL Half satuation constant for Nitrate [molN/l]
real(8),parameter :: KNH4L = 0.30D-6 ! PL Half satuation constant for Ammonium [molN/l]
real(8),parameter :: KSiL = 6.00D-6 ! PL Half satuation constant for Silicate [molSi/l]
real(8),parameter :: PusaiL = 1.50D6 ! PL Ammonium Inhibition Coefficient [l/molN]
real(8),parameter :: KGppL = 6.93D-2 ! PL Temp. Coeff. for Photosynthetic Rate [degC]
real(8),parameter :: MorPL0 = 2.90D4/d2s ! PL Mortality Rate @0degC [s]
real(8),parameter :: KMorPL = 6.93D-2 ! PL Temp. Coeff. for Mortality [degC]
real(8),parameter :: ResPL0 = 0.03D0/d2s ! PL Respiration Rate at @0degC [s]
real(8),parameter :: KResPL = 0.0519D0 ! PL Temp. Coeff. for Respiration [degC]
real(8),parameter :: GammaL = 0.135D0 ! PL Ratio of Extracell. Excret. to Photo. [(nodim)]
real(8),parameter :: GRmaxS = 0.40D0/d2s ! ZS Maximum Rate of Grazing PS @0degC [s]
real(8),parameter :: KGraS = 6.93D-2 ! ZS Temp. Coeff. for Grazing [degC]
real(8),parameter :: LamS = 1.40D6 ! ZS Ivlev constant [l/molN]
real(8),parameter :: PS2ZSstar= 0.043D-6 ! ZS Threshold Value for Grazing PS [molN/l]
real(8),parameter :: AlphaZS = 0.70D0 ! ZS Assimilation Efficiency [(nodim)]
real(8),parameter :: BetaZS = 0.30D0 ! ZS Growth Efficiency [(nodim)]
real(8),parameter :: MorZS0 = 5.85D4/d2s ! ZS Mortality Rate @0degC [s]
real(8),parameter :: KMorZS = 0.0693D0 ! ZS Temp. Coeff. for Mortality [degC]
real(8),parameter :: GRmaxLps = 0.10D0/d2s ! ZL Maximum Rate of Grazing PS @0degC [s]
real(8),parameter :: GRmaxLpl = 0.40D0/d2s ! ZL Maximum Rate of Grazing PL @0degC [s]
real(8),parameter :: GRmaxLzs = 0.40D0/d2s ! ZL Maximum Rate of Grazing ZS @0degC [s]
real(8),parameter :: KGraL = 6.93D-2 ! ZL Temp. Coeff. for Grazing [degC]
real(8),parameter :: LamL = 1.4000D6 ! ZL Ivlev constant [l/molN]
real(8),parameter :: PS2ZLstar= 4.00D-8 ! ZL Threshold Value for Grazing PS [molN/l]
real(8),parameter :: PL2ZLstar= 4.00D-8 ! ZL Threshold Value for Grazing PL [molN/l]
real(8),parameter :: ZS2ZLstar= 4.00D-8 ! ZL Threshold Value for Grazing ZS [molN/l]
real(8),parameter :: AlphaZL = 0.70D0 ! ZL Assimilation Efficiency [(nodim)]
real(8),parameter :: BetaZL = 0.30D0 ! ZL Growth Efficiency [(nodim)]
real(8),parameter :: MorZL0 = 5.85D4/d2s ! ZL Mortality Rate @0degC [s]
real(8),parameter :: KMorZL = 0.0693D0 ! ZL Temp. Coeff. for Mortality [degC]
real(8),parameter :: GRmaxPpl = 0.20D0/d2s ! ZP Maximum rate of grazing PL @0degC [s]
real(8),parameter :: GRmaxPzs = 0.20D0/d2s ! ZP Maximum rate of grazing ZS @0degC [s]
real(8),parameter :: GRmaxPzl = 0.20D0/d2s ! ZP Maximum rate of grazing ZL @0degC [s]
real(8),parameter :: KGraP = 6.93D-2 ! ZP Temp. Coeff. for grazing [degC]
real(8),parameter :: LamP = 1.4000D6 ! ZP Ivlev constant [l/molN]
real(8),parameter :: PL2ZPstar= 4.00D-8 ! ZP Threshold Value for Grazing PL [molN/l]
real(8),parameter :: ZS2ZPstar= 4.00D-8 ! ZP Threshold Value for Grazing ZS [molN/l]
real(8),parameter :: ZL2ZPstar= 4.00D-8 ! ZP Threshold Value for Grazing ZL [molN/l]
real(8),parameter :: PusaiPL = 4.605D6 ! ZP Preference Coeff. for PL [l/molN]
real(8),parameter :: PusaiZS = 3.010D6 ! ZP Preference Coeff. for ZS [l/molN]
real(8),parameter :: AlphaZP = 0.70D0 ! ZP Assimilation Efficiency [(nodim)]
real(8),parameter :: BetaZP = 0.30D0 ! ZP Growth Efficiency [(nodim)]
real(8),parameter :: MorZP0 = 5.85D4/d2s ! ZP Mortality Rate @0degC [s]
real(8),parameter :: KMorZP = 0.0693D0 ! ZP Temp. Coeff. for Mortality [degC]
real(8),parameter :: Nit0 = 0.03D0/d2s ! NH4 Nitrification Rate @0degC [s]
real(8),parameter :: KNit = 0.0693D0 ! NH4 Temp. coefficient for Nitrification [degC]

```

```

real(8),parameter :: VP2N0 = 0.10D0/d2s ! PON Decomp. Rate to Ammonium @0degC [s]
real(8),parameter :: KP2N = 6.93D-2 ! PON Temp. Coeff. for Decomp. to Ammon. [degC]
real(8),parameter :: VP2D0 = 0.10D0/d2s ! PON Decomp. Rate to DON @0degC [s]
real(8),parameter :: KP2D = 6.93D-2 ! PON Temp. Coeff. for Decomp. to DON [degC]
real(8),parameter :: VD2N0 = 0.20D0/d2s ! DON Decomp. Rate to Ammonium @0degC [s]
real(8),parameter :: KD2N = 6.93D-2 ! DON Temp. Coeff. for Decomp. to Ammon. [degC]
real(8),parameter :: VO2S0 = 0.10D0/d2s ! Opal Decomp. Rate to Silicate @0degC [s]
real(8),parameter :: KO2S = 6.93D-2 ! Opal Temp. Coeff. for Decomp. to Silicate [degC]
real(8),parameter :: RSiN = 2.0D0 !Si/N ratio [molSi/molN]
real(8),parameter :: RCN = 106.0D0/16.0D0 !C/N ratio [molC/molN]
!
! ..... bottom boundary Condition .....
real(8),parameter :: setVPON = 40.0D0/d2s ! Settling velocity of PON [m/s]
real(8),parameter :: setVOpal = 40.0D0/d2s ! Settling velocity of Opal [m/s]
real(8),parameter :: TNO3d = 25.0d-6 ! Nitrate Concentraion in the Deep Layer [molN/l]
real(8),parameter :: TSiOH4d = 35.0d-6 ! Silicate Concentraion in the Deep Layer [molSi/l]
!
! ..... Paramters of ZL Vertical Migration .....
character(19) :: CZup='0000/04/01 00:00:00' ! Date Coming up to the Euphotic Layer
character(19) :: CZdwn='0000/09/01 00:00:00' ! Date Returning to the Deep Layer
real(8) :: TZup, TZdwn, TZLd, SVRate=0.2D0
integer :: IyrU, ImonU, IdayU, IhourU, IminU, IsecU
integer :: IyrD, ImonD, IdayD, IhourD, IminD, IsecD
!
real(8) :: GppPSn, GppNPSn, GppAPSn, RnewS, ResPSn, MorPSn, ExcPSn
real(8) :: GppPLn, GppNPLn, GppAPLn, RnewL, ResPLn, MorPLn, ExcPLn
real(8) :: GppPLsi, GppSiPLsi, ResPLsi, MorPLsi, ExcPLsi
real(8) :: GraPS2ZSn, GraPS2ZLn, GraPL2ZLn, GraPL2ZLsi, GraZS2ZLn
real(8) :: GraPL2ZPn, GraPL2ZPsi, GraZS2ZPn, GraZL2ZPn
real(8) :: EgeZSn, MorZSn, ExcZSn, EgeZLn, EgeZLsi, MorZLn, ExcZLn
real(8) :: EgeZPn, EgeZPsi, MorZPn, ExcZPn
real(8) :: DecP2N, DecP2D, DecD2N, DecO2S, Nit
real(8) :: ExpPON, ExpOpal, ExcNO3, ExcSiOH4
integer :: lt=0, nt
!
! ..... Environmental Condition .....
real(8) :: Temp ! Temperature [degC]
real(8) :: Lint0 ! Light Intencity at sea surface [ly/min]
real(8) :: MLD = 30.0d0 ! Mixed Layer Depth [m]
real(8) :: ExcTime = 1.0d0 / (100.0d0*d2s) ! Exch. Coeff. between Sur-Deep [s]
!
! ..... statement function & def. type of functions .....
real(8) :: cd2tt, nd2tt
character(19) :: tt2cd
real(8) :: Td, GraF, Mich, a, b, c
Td (a,b) = a * exp(b*Temp)
GraF(a,b,c) = MAX( 0.0D0, 1.0 - exp(a * (b - c)))
Mich(a,b) = b / ( a + b )
!
! ***** Initial Setting *****
! ..... for time control .....
TTime = cd2tt(Cstart) ! Starting Date
CTime = TT2CD(TTime) ! present time (charactor form)
dt = cd2tt(Cstep) - cd2tt('0000/00/00 00:00:00') ! Time Step (real8 form)

```

```

Tmon = cd2tt(Cmon) - cd2tt('0000/00/00 00:00:00') ! Monitor Interval (real8 form)
nt = NINT( ( cd2tt(Cend) - cd2tt(Cstart) ) / dt ) ! Total Time Steps
! ..... for Vertical Migration .....
TZup = CD2TT( CZup )
TZdwn = CD2TT( CZdwn )
call TT2ND(IyrU, ImonU, IdayU, IhourU, IminU, IsecU, TZup )
call TT2ND(IyrD, ImonD, IdayD, IhourD, IminD, IsecD, TZdwn)
TZLd = TZL ! ZL living in the deep layer at the initial condition
TZL = 0.0
! ..... File Open for monitoring output .....
open( 10, file='Results.csv', form='FORMATTED' )
write(10,'(A,13(", ", A))' ) 'Time(day)', &
      'NO3' , 'NH4' , 'PS' , 'PL' , &
      'ZS' , 'ZL' , 'ZP' , 'PON' , &
      'DON' , 'SiOH4' , 'Opal' , 'TotalN' , 'TotalSi'
write(10,'(A,11(", ", F8.4))' ) CTime, &
      TNO3/mcr, TNH4 /mcr, TPS /mcr, TPL /mcr, &
      TZS /mcr, TZL /mcr, TZP /mcr, TPON/mcr, &
      TDON/mcr, TSiOH4/mcr, TOpal/mcr
open( 11, file='Forcing.csv', form='FORMATTED' )
write(11,'(A,13(", ", A))' ) 'Time(day)', 'Lint0', 'TMP', 'MLD', 'ExcTime'
write(11,'(A,13(", ", 1PE10.4))' ) CTime, Lint0, Temp, MLD, ExcTime*d2s
!
! ***** Main Loop *****
do lt = 1, nt
! ..... time control (Season : 0 to 1, percentage in a year).....
  Tbefore = TTime ! one step before present time
  TTime = TTime + dt ! present time (real8 form)
  CTime = TT2CD(TTime) ! present time (character form)
  CALL TT2ND(Iyr, Imon, Iday, Ihour, Imin, Isec, TTime)
  Season = ( TTime - ND2TT(Iyr, 1, 1, 0, 0, 0) ) / &
           ( ND2TT(Iyr+1, 1, 1, 0, 0, 0) - ND2TT(Iyr, 1, 1, 0, 0, 0) )
!
! ..... Example of Boundray condition .....
  Lint0 = 0.1d0 * ( 1.0D0 + 0.3d0 * cos( 2.0d0*3.1415926536d0*(Season - 0.50D0) ) )
  Temp = 6.0D0 + 4.0d0 * cos( 2.0d0*3.1415926536d0*(Season - 0.65D0) )
  if (Temp .lt. 4.0 ) then
    MLD = MLD + dt * ( 150.0d0 - MLD ) / ( 100.0d0 * d2s )
    ExcTime = ExcTime + dt * ( 1.0d0 / ( 40.0d0*d2s ) - ExcTime ) / ( 100.0d0*d2s )
  else
    MLD = MLD + dt * ( 30.0d0 - MLD ) / ( 5.0d0 * d2s )
    ExcTime = ExcTime + dt * ( 1.0d0 / ( 100.0d0*d2s ) - ExcTime ) / ( 5.0d0*d2s )
  end if
!
! ..... Light Factors (LfcS, LfcL).....
  Lint = Lint0
  LfcDS = Lint/IoptS * exp(1.0D0 - Lint/IoptS)
  LfcDL = Lint/IoptL * exp(1.0D0 - Lint/IoptL)
  LfcS = 0.0D0
  LfcL = 0.0D0
  Kappa = alpha1 + alpha2 * ( TPS + TPL )

```

```

dLint = exp( -Kappa * (MLD/LLN) )
do L = 1, LLN
  LfcUS = LfcDS
  LfcUL = LfcDL
  Lint = Lint * dLint
  LfcDS = Lint/IoptS * exp( 1.0D0 - Lint/IoptS )
  LfcDL = Lint/IoptL * exp( 1.0D0 - Lint/IoptL )
  LfcS = LfcS + ( LfcUS + LfcDS ) * 0.5D0 / LLN
  LfcL = LfcL + ( LfcUL + LfcDL ) * 0.5D0 / LLN
end do
! ..... Photosynthesis of PS .....
GppNPSn = Mich( KNO3S, TNO3 ) * exp( - PusaiS * TNH4 )
GppAPSn = Mich( KNH4S, TNH4 )
GppPSn = Td(VmaxS, KGppS) * LfcS * TPS * ( GppNPSn + GppAPSn )
RnewS = GppNPSn / ( GppNPSn + GppAPSn )
ResPSn = Td( ResPS0, KResPS ) * TPS
MorPSn = Td( MorPS0, KMorPS ) * TPS * TPS
ExcPSn = GammaS * GppPSn
! ..... Photosynthesis of PL .....
GppNPLn = Mich( KNO3L, TNO3 ) * exp( - PusaiL * TNH4 )
GppAPLn = Mich( KNH4L, TNH4 )
GppSiPLsi = Mich( KSiL , TSiOH4 )
GppPLn = Td(VmaxL, KGppL) * LfcL * TPL * min( ( GppNPLn + GppAPLn ), GppSiPLsi )
RnewL = GppNPLn / ( GppNPLn + GppAPLn )
ResPLn = Td( ResPL0, KResPL ) * TPL
MorPLn = Td( MorPL0, KMorPL ) * TPL * TPL
ExcPLn = GammaL * GppPLn
! ..... Grazing PS, PL, ZS, ZL --> ZS, ZL, ZP .....
GraPS2ZSn = Td(GRmaxS, KGraS) * GraF(LamS,PS2ZSstar,TPS) * TZS
GraPS2ZLn = Td(GRmaxLps,KGraL) * GraF(LamL,PS2ZLstar,TPS) * TZL
GraPL2ZLn = Td(GRmaxLpl,KGraL) * GraF(LamL,PL2ZLstar,TPL) * TZL
GraZS2ZLn = Td(GRmaxLzs,KGraL) * GraF(LamL,ZS2ZLstar,TZS) * TZL
GraPL2ZPn = Td(GRmaxPpl,KGraP) * GraF(LamP,PL2ZPstar,TPL) * TZP * exp( -PusaiPL *(TZL
+ TZS))
GraZS2ZPn = Td(GRmaxPzs,KGraP) * GraF(LamP,ZS2ZPstar,TZS) * TZP * exp( -PusaiZS * TZL )
GraZL2ZPn = Td(GRmaxPzl,KGraP) * GraF(LamP,ZL2ZPstar,TZL) * TZP
! ..... Mortality, Excretion, Egestion for Zooplanktons
! ..... Commented out after Saito-san Meeting at 19 Jun, 2000 .....
! BetaZS = 0.3 ** ( 1.0 + Mich( TPL, TPS ) )
ExcZSn = (AlphaZS- BetaZS) * GraPS2ZSn
EgeZSn = (1.0 - AlphaZS) * GraPS2ZSn
MorZSn = Td( MorZS0, KMorZS ) * TZS * TZS
ExcZLn = (AlphaZL- BetaZL) * (GraPS2ZLn+GraPL2ZLn+GraZS2ZLn)
EgeZLn = (1.0 - AlphaZL) * (GraPS2ZLn+GraPL2ZLn+GraZS2ZLn)
MorZLn = Td( MorZL0, KMorZL ) * TZL * TZL
ExcZPn = (AlphaZP- BetaZP) * (GraPL2ZPn+GraZS2ZPn+GraZL2ZPn)
EgeZPn = (1.0 - AlphaZP) * (GraPL2ZPn+GraZS2ZPn+GraZL2ZPn)
MorZPn = Td( MorZP0, KMorZP ) * TZP * TZP
! ..... Decomposition PON, DON, Opal ---> NH4, DON, SiOH4 .....
DecP2N = Td( VP2N0 , KP2N ) * TPON
DecP2D = Td( VP2D0 , KP2D ) * TPON

```

```

DecD2N = Td( VD2N0 , KD2N ) * TDON
DecO2S = Td( VO2S0 , KO2S ) * TOpal
Nit     = Td( Nit0 , KNit ) * TNH4
!
..... silica fluxes .....
GppPLsi = GppPLn * RSiN
ResPLsi  = ResPLn * RSiN
MorPLsi  = MorPLn * RSiN
ExcPLsi  = ExcPLn * RSiN
GraPL2ZLsi = GraPL2ZLn * RSiN
GraPL2ZPsi = GraPL2ZPn * RSiN
EgeZLsi  = GraPL2ZLsi
EgeZPsi  = GraPL2ZPsi
!
!
..... Tendency Terms for biological processes .....
QNO3 = -( GppPSn - ResPSn ) * RnewS &
      -( GppPLn - ResPLn ) * RnewL + Nit
QNH4 = -( GppPSn - ResPSn ) * (1.0 - RnewS) &
      -( GppPLn - ResPLn ) * (1.0 - RnewL) &
      - Nit + DecP2N + DecD2N + ExcZSn + ExcZLn + ExcZPn
QPS   = GppPSn - ResPSn - MorPSn - ExcPSn - GraPS2ZSn - GraPS2ZLn
QPL   = GppPLn - ResPLn - MorPLn - ExcPLn - GraPL2ZLn - GraPL2ZPn
QZS   = GraPS2ZSn - GraZS2ZLn - MorZSn - ExcZSn - EgeZSn - GraZS2ZPn
QZL   = GraPL2ZLn + GraZS2ZLn - MorZLn - ExcZLn - EgeZLn + GraPS2ZLn - GraZL2ZPn
QZP   = GraPL2ZPn + GraZS2ZPn - MorZPn - ExcZPn - EgeZPn + GraZL2ZPn
QPON  = MorPSn + MorPLn + MorZSn + MorZLn + MorZPn &
      + EgeZPn + EgeZSn + EgeZLn - DecP2N - DecP2D
QDON  = ExcPSn + ExcPLn + DecP2D - DecD2N
QSiOH4 = -GppPLsi + ResPLsi + ExcPLsi + DecO2S
QOpal = MorPLsi + EgeZLsi + EgeZPsi - DecO2S
!
!
..... Exchange Fluxes between the Surface and Deep Layers .....
ExpPON = setVPON / MLD * TPON
ExpOpal = setVOpal / MLD * TOpal
ExcNO3 = ExcTime * ( TNO3d - TNO3 )
ExcSiOH4 = ExcTime * ( TSiOH4d - TSiOH4 )
QNO3 = QNO3 + ExcNO3
QSiOH4 = QSiOH4 + ExcSiOH4
QPON = QPON - ExpPON
QOpal = QOpal - ExpOpal
!
!
..... Time Integration with Forward Scheme .....
TNO3 = TNO3 + dt * QNO3
TNH4 = TNH4 + dt * QNH4
TPS = TPS + dt * QPS
TPL = TPL + dt * QPL
TZS = TZS + dt * QZS
TZL = TZL + dt * QZL
TZP = TZP + dt * QZP
TPON = TPON + dt * QPON
TDON = TDON + dt * QDON
TSiOH4 = TSiOH4 + dt * QSiOH4

```

```

      TOpal = TOpal + dt * QOpal
!
!
! ..... Vertical Migration of ZL .....
TZdwn = ND2TT(Iyr, ImonD, IdayD ,IhourD, IminD, IsecD )
TZup  = ND2TT(Iyr, ImonU, IdayU ,IhourU, IminU, IsecU )
if ( (Tbefore .lt. TZdwn).and.(TTime .ge. TZdwn) ) then
  TZLd = TZL
  TZL = 0.0
  write(*,*) '*** Down ***', CTime
end if
if ( (Tbefore .lt. TZup).and.(TTime .ge. TZup) ) then
  TZL = SVRate * TZLd
  write(*,*) '*** UP ***', CTime
end if
!
! call Herring(TTime, Tbefore, TZS, TZL, TZP, Temp)
!
! ..... Monitor .....
if ( int(TTime/Tmon).ne. int(Tbefore/Tmon) ) then
!   write(*,'(A,13(", ", F8.4))') CTime, Season
!   write(10,'(A,11(", ", F8.4))') CTime, &
!     TNO3/mcr, TNH4 /mcr, TPS /mcr, TPL /mcr, &
!     TZS /mcr, TZL /mcr, TZP /mcr, TPON/mcr, &
!     TDON/mcr, TSioH4/mcr, TOpal/mcr
!   write(11,'(A,13(", ", 1PE10.4))') CTime, Lint0, Temp, MLD, ExcTime*d2s
!   end if
end do
!
! close(10); close(11)
!
! stop
! end
!*****
! Subroutine Herring(TTime, Tbefore, TZS, TZL, TZP, Temp)
!
! implicit none
! real(8)      :: TTime, Tbefore, TZS, TZL, TZP, Temp
! real(8),parameter :: d2s    = 86400.0d0  ! day ---> sec
! integer      :: Iyr, Imon, Iday, Ihour, Imin, Isec
! character(19)  :: CAge ='0000/07/19 00:00:00' ! Date of Aging ( JJday = 200 )
! character(19)  :: CTime
! real(8)      :: TAge
! integer      :: iage = 0
! integer, save  :: IyrA, ImonA, IdayA ,IhourA, IminA, IsecA
! integer      :: JJday
! real(8)      :: ZooP1, ZooP2, ZooP3, tt1
! real(8)      :: t1,t2,wtemp
! real(8)      :: x(1) =0.2d0, xdot(1)
! real(8)      :: cd2tt, nd2tt
! character(19)  :: tt2cd
! real(8)      :: vul(3), k(3)

```

```

integer(4)    :: id(365)
real(8)      :: zop1(365), zop2(365), zop3(365)
real(8):: v, a, u, resp
real(8) :: xk1,xk2,xk3,xk4,te1,te2,te3,te4,tt5,t5,t4,tt7,t7,t6, gcta,gctb,gctemp,gcmax
real(8) :: cnum,c1,c2,c3,con1,con2,con3,con
real(8) :: f,e, sda
!
integer, save  :: First = 1
!
! =====
if ( First .eq. 1 ) then; First = 0
  TAge = CD2TT( CAge )
  call TT2ND(IyrA, ImonA, IdayA ,IhourA, IminA, IsecA ,TAge )
  open( 20, file='Herring.csv', form='FORMATTED' )
!
!!!!   OPEN(UNIT=111,FILE='nemuro.txt',STATUS='unknown')
!!!!   ----read in the 3 zoop groups from Nemuro output last 3 columns
!!!!   do JJday=1,365
!!!!     READ(111,999)id(JJday),zop1(JJday),zop2(JJday),zop3(JJday)
!!!! 999   FORMAT(1x,i3,1x,3(e13.6,1x))
!!!!   end do
end if
! =====
!
CTime = TT2CD(TTime)                ! present time (charactor form)
CALL TT2ND(Iyr, Imon, Iday ,Ihour, Imin, Isec ,TTime)
JJday = 1 + ( TTime - ND2TT(Iyr ,1,1,0,0,0) ) / d2s
!
!-----convert Nemuro zoop in uM N/L to g ww/m3
!----- tt1 is conversion from uM N/liter to g ww/m3
!----- 14 ug N/uM * 1.0e-6 g/ug * 1 g dw/0.07 g N dw * 1 g ww/0.2 g dw *
!----- 1.0e3 liters/m3
!
tt1=14.0*1.0e-6*(1.0/0.07)*(1.0/0.2)*1.0e3
zoop1 = TZS*tt1 *1.0d6
zoop2 = TZL*tt1 *1.0d6
zoop3 = TZP*tt1 *1.0d6
!!!! zoop1 = zop1(JJday) * tt1
!!!! zoop2 = zop2(JJday) * tt1
!!!! zoop3 = zop3(JJday) * tt1
!
! ..... Temperature Seting .....
!
t1=float(jjday)
t2=12.75-10.99*cos(0.0172*t1)-6.63*sin(0.0172*t1)
wtemp=t2-5.0
IF(wtemp.le.1.0)wtemp=1.0
!
! write(*,*) TT2CD(cd2tt('0002/01/01 00:00:00')+200.0*86400.0)
! stop
! ..... Aging of Herring .....

```

```

TAge = ND2TT(Iyr, ImonA, IdayA ,IhourA, IminA, IsecA )
if ( (Tbefore .lt. TAge).and.(TTime .ge. TAge) ) then
    write(*,*) '*** Aging +1 of Herring ***', CTime
    iage = iage + 1
end if
!
!--- Herring weight state variable = x(1)
!
!----- set vulnerabilities and k values for 3 zoop groups
!
    vul(1) = 1.0; vul(2) = 1.0; vul(3) = 1.0
    k (1) = 0.3638; k (2) = 0.0364; k (3) = 0.3638
!
! --- weight affect on respiration
!
    tt1 = 1.0 / x(1)
    t1 = 0.0033 * tt1**0.227
! --- *****this is the new stuff from Ahhrenius for YOY only*****
! --- The 5.258 puts resp (g oxygen/fish) into units of g zoop/g fish/day
! --- [13560 joules/gram oxygen]/4.18 joules/cal = 3244 cal/gO2
! --- [2580 joules/gram zoop]/4.18 joules/cal = 617 cal/g zoop
! --- so respiration in grams/oxygen/g fish/day is multiplied by 3244/617 = 5.258
! --- to get food energy equivalents of a gram of oxygen respired
!
    IF (iage .eq. 0 )then
        IF(wtemp.le.15.0)then
            v = 5.76 * exp( 0.0238 * wtemp ) * x(1)**0.386
        else
            v = 8.6 * x(1)**0.386
        endif
        a=EXP((0.03-0.0*wtemp)*v)
        resp=t1*EXP(0.0548*wtemp)*a*5.258
! --- *****back to the old equations for respiration for age-1 and older*****
    else ! (iage .gt. 0)
        IF (wtemp.le.9.0)then
            u=3.9*x(1)**0.13*EXP(0.149*wtemp)
        else
            u=15.0*x(1)**0.13
        endif
        resp=t1*EXP(0.0548*wtemp)*EXP(0.03*u)*5.258
    endif
!C
!C --- Thornton and Lessem temperature effect
!C --- age dependent values
!C --- *****Arrhenius for age-0 he changed te4 from 25 to 23 degrees*****
!C
    if ( iage .eq. 0 ) then
        xk1 = 0.1; xk2 = 0.98; xk3 = 0.98; xk4 = 0.01
        te1 = 1.0; te2 = 15.0; te3 = 17.0; te4 = 23.0
    else if ( iage .eq. 1 ) then
        xk1 = 0.1; xk2 = 0.98; xk3 = 0.98; xk4 = 0.01

```

```

    te1 = 1.0; te2 = 15.0; te3 = 17.0; te4 = 25.0
else if( iage .gt. 1 ) then
    xk1 = 0.1; xk2 = 0.98; xk3 = 0.98; xk4 = 0.01
    te1 = 1.0; te2 = 13.0; te3 = 15.0; te4 = 23.0
endif
!
tt5 = ( 1.0 / ( te2 - te1 ) )
t5 = tt5 * log( 0.98 * ( 1.0 - xk1 ) / ( 0.02 * xk1 ) )
t4 = exp( t5 * ( wtemp - te1 ) )
!
tt7 = 1.0 / ( te4 - te3 )
t7 = tt7 * log( 0.98 * ( 1.0 - xk4 ) / ( 0.02 * xk4 ) )
t6 = exp( t7 * ( te4 - wtemp ) )
!
gcta = ( xk1 * t4 ) / ( 1.0 + xk1 * ( t4 - 1.0 ) )
gctb = xk4 * t6 / ( 1.0 + xk4 * ( t6 - 1.0 ) )
gctemp= gcta * gctb
gcmx = 0.642 * tt1**0.256 * gctemp
!
! --- multispecies functional response
! --- usse either this or adjust little p
!
cnum=zoop1 * vul(1)/k(1) + zoop2*vul(2)/k(2) +zoop3 * vul(3)/k(3)
c1=gcmx*zoop1*vul(1)/k(1)
c2=gcmx*zoop2*vul(2)/k(2)
c3=gcmx*zoop3*vul(3)/k(3)
con1=c1/(1.0+cnum)
con2=c2/(1.0+cnum)
con3=c3/(1.0+cnum)
con= con1+con2+con3
!
!-----if using constant p rather than functional response, set p here
! --- to tune to observed size at age data
!   con=0.425*gcmx
!
! --- egestion
!
!       f=0.16*con
!
! --- excretion
!       e=0.1*(con-f)
!
!
! --- Specific Dynamic Action
!
!c----- *****Arrhenius age dependent SDA from 17.5% to 15% *****
IF ( iage .eq. 0 ) then
    sda=0.15*(con-f)
else
    sda=0.175*(con-f)
end if

```

```

!C
!C --- use the ratio of calories/g of zoop (2580) to calories/g of fish (5533)
!C
!C --- bioenergetics differential equation
!C
      xdot(1)=(con-resp-f-e-sda)*x(1)*2580./5533.
!
      IF(wtemp.le.1.0)xdot(1)=0.0
!C
!C --- Spawning section. Assume loose 20% of boso weight/day
!C      t1=float(jjday)
!      if( mod(JJday,365) .ge. 152.0 .and. mod(JJday,365) .le. 156.0) then
!          xdot(1)=(con-resp-f-e-sda-0.20)*x(1)*2580./5533.
!          write(*,*) '### Spawning ###'
!      endif
!
!      if (iage .eq. 1 ) then
!          write(*,*) JJday, wtemp, x(1), xdot(1)
!          stop
!      end if
!      write(*,'(A,I4,3(1PE14.5))') Ctime, JJday, wtemp, x(1), xdot(1)
!
!      Time Integration
!
      x(1) = x(1) + 3600.0d0 /d2s * xdot(1)
!
!      .... for Check .....
      if ( int(TTime/d2s) .ne. int(Tbefore/d2s) ) then
!!          write(*,'(A,I4,3(1PE14.5))') Ctime, JJday, wtemp, x(1), xdot(1)
!!          stop
!!          write(*,*) TZS, zop1(JJday), TZL,zop2(JJday), TZP,zop3(JJday)
!!          write(*,*) TZP*1.0d6, zop3(JJday)
!!          write(20,'(A,11(" ", F12.4))') CTime, x(1), wtemp, gcmx
!!          end if
!
!      return
!
!      stop
!      end
!*****
!* Utilities for Date Control Writtien by Yasuhiro Yamanaka (galapen@ees.hokudai.ac.jp) *
!*****
!      exp. 1997/12/31 23:59:59 --> 6.223158719900000E+10
!      exp. 0000/01/01 00:00:00 --> 0.000000000000000E+00
!*****
      real(8) function CD2TT( Cdate )
!
!      integer      :: Iyr, Imon, Iday , Ihour, Imin, Isec
!      real(8)      :: ND2TT
!      character(19) :: Cdate
!

```

```

if ( len( Cdate ) .ne. 19 ) then
  write(*,*) '### Length of date is no good ###'
  stop
end if
read (Cdate( 1: 4),*) Iyr
read (Cdate( 6: 7),*) Imon
read (Cdate( 9:10),*) Iday
read (Cdate(12:13),*) Ihour
read (Cdate(15:16),*) Imin
read (Cdate(18:19),*) Isec
!
CD2TT = ND2TT(Iyr, Imon, Iday , Ihour, Imin, Isec)
!
return
end function
!*****
! exp. 6.223158719900000E+10 --> 1997/12/31 23:59:59
!*****
character(19) function TT2CD(tt)
!
integer :: Iyr, Imon, Iday , Ihour, Imin, Isec
real(8) :: tt
!
call TT2ND( Iyr, Imon, Iday, Ihour, Imin, Isec , tt )
!
write(TT2CD,'(I4.4,5(A,I2.2))') Iyr, '/', Imon, '/', Iday, &
  ', Ihour, ':', Imin, ':', Isec
!
return
end function
!*****
! exp. 1997,12,31,23,59,59 --> 6.223158719900000E+10
!*****
real(8) function ND2TT(Iyr, Imon, Iday, Ihour, Imin, Isec)
!
integer :: IM2D(12,0:1) = &
  reshape( (/ 0,31,59,90,120,151,181,212,243,273,304,334, &
    0,31,60,91,121,152,182,213,244,274,305,335 /), (/12,2/) )
integer :: Iyr, Imon, Iday, Ihour, Imin, Isec
integer :: Iy4, Iy1, Ileaf, Im, Itt
!
!
Iy4 = 1461 * ( Iyr / 4 )
Iy1 = 365 * mod( Iyr, 4 )
!
if ( mod( Iyr, 4 ) .ne. 0 ) then
  Ileaf = 0
else
  Ileaf = 1
end if
Im = IM2D( Imon, Ileaf)

```

```

!
  Itt = Iy4 + Iy1 + Im + Iday - Ileap
!
  ND2TT = Ihour * 3600 + Imin * 60 + Isec
  ND2TT = ND2TT + Itt * 86400.0D0
!
  return
end function
!*****
! exp. 6.223158719900000E+10 --> 1997,12,31,23,59,59
!*****
subroutine TT2ND(
    &
    Iyr , Imon , Iday , Ihour, Imin, Isec, & !O & I
    tt )
!
integer :: Iyr, Imon, Iday , Ihour, Imin, Isec
integer :: Itt, Iy, Iy4, Iyd, Iy1, Ileap, Imd, Im, Its
integer :: IM2D(12,0:1) = &
    reshape( (/ 0,31,59,90,120,151,181,212,243,273,304,334, &
    0,31,60,91,121,152,182,213,244,274,305,335 /), (/12,2/) )
integer :: IY2D(4) = (/0,366,731,1096/)
real(8) :: tt, tt0, ND2TT
!
!
! ..... ITT [day] .....
  Itt = 1 + tt / 86400.0D0
!
  Iy4 = (Itt-1) / 1461
  Iyd = Itt - Iy4 * 1461
  do IY = 1, 4
    if ( IY2D(Iy) + 1 .le. Iyd ) then
      Iy1 = Iy
    end if
  end do
!
  Iyr = Iy4 * 4 + Iy1 - 1
  if ( mod(Iyr,4) .ne. 0 ) then
    Ileap = 0
  else
    Ileap = 1
  end if
  IMD = IYD - IY2D(IY1)
!
  do IM = 1, 12
    if ( IM2D(IM,Ileap)+1 .le. IMD ) then
      IMON = IM
    end if
  end do
  IDAY = IMD - IM2D(IMON,Ileap)
!
  TT0 = ND2TT(IYR, IMON, IDAY ,0,0,0)

```

```
ITS = nint( TT - TT0 )
Ihour = ITS / 3600
Imin = ( ITS - Ihour * 3600 ) / 60
Isec = ITS - Ihour * 3600 - Imin * 60
!  
return  
end subroutine  
!
```

Appendix 5 NEMURO.FISH (NEMURO FORTRAN code supplied by Yasuhiro Yamanaka) with the saury bioenergetic model (base case) (supplied by Bernard Megrey and Ken Rose and modified by “team saury”). The saury model is linked to NEMURO in a one-way static link.

```

!*****
! NEMURO model   Jun 13, 2002  written by Yasuhiro Yamanaka
!               modified by Masahiko Fujii
!               Shin-ichi Ito
!*****

program NEMURO
implicit none
! ..... Control for Time Ingration .....
character(19)  :: Cstart = '0001/02/01 00:00:00' ! Starting date
character(19)  :: Cend   = '0003/02/01 00:00:00' ! Ending date
character(19)  :: Cstep  = '0000/00/00 01:00:00' ! Time step
character(19)  :: Cmon   = '0000/00/01 00:00:00' ! Monitor Interval
character(19)  :: CTime
real(8)        :: dt, TTime, Tbefore, Season, Tmon
integer        :: Iyr, Imon, Iday, Ihour, Imin, Isec
! ..... scale conversion .....
real(8),parameter :: d2s    = 86400.0d0  ! day ---> sec
real(8),parameter :: mcr    = 1.0d-6     ! micro
! ..... Prognostic Variables (with initial conditions) and Thier Source Term .....
real(8)          :: TPS    = 0.1D-6, QPS  ! Small Phytoplankton [molN/l]
real(8)          :: TPL    = 0.1D-6, QPL  ! Large Phytoplankton
real(8)          :: TZS    = 0.1D-6, QZS  ! Small Zooplankton
real(8)          :: TZL    = 0.1D-6, QZL  ! Large Zooplankton
real(8)          :: TZP    = 0.1D-6, QZP  ! Pradatory Zooplankton
real(8)          :: TNO3   = 5.0D-6, QNO3 ! Nitrate
real(8)          :: TNH4   = 0.1D-6, QNH4 ! Ammmonium
real(8)          :: TPON   = 0.1D-6, QPON ! Particulate Organic Nitrogen
real(8)          :: TDON   = 0.1D-6, QDON ! dissolved Organic Nitrogen
real(8)          :: TSiOH4 = 10.0D-6, QSiOH4 ! Silicate
real(8)          :: TOpal  = 0.1D-7, QOpal ! Particulate Opal
! ..... Prognostic Variables (with initial conditions) and Thier Source Term .....
! real(8)          :: THrr  = 0.2D0, QHrrl ! Particulate Opal
! ..... Light Condition Parameters .....
real(8),parameter :: alpha1 = 4.0D-2     ! Light Dissipation coefficient of sea water[/m]
real(8),parameter :: alpha2 = 4.0D4      ! PS+PL Selfshading coefficientS+PL [l/molN/m]
real(8),parameter :: IoptS  = 0.15D0     ! PS Optimum Light Intensity S [ly/min]
real(8),parameter :: IoptL  = 0.15D0     ! PL Optimum Light Intensity [ly/min]
integer,parameter :: LLN     = 10        ! Number of sublayer for calculating of Lfc
real(8)          :: LfcS     ! Light factor for PS
real(8)          :: LfcL     ! Light factor for PL
real(8)          :: kappa, Lint, dLint, LfcUS, LfcUL, LfcDS, LfcDL
integer          :: L
! ..... biological Parameters .....
real(8),parameter :: VmaxS   = 0.4D0/d2s ! PS Maximum Photosynthetic rate @0degC [l/s]
real(8),parameter :: KNO3S   = 1.0D-6    ! PS Half saturation constant for Nitrate [molN/l]
real(8),parameter :: KNH4S   = 0.1D-6    ! PS Half saturation constant for Ammonium [molN/l]

```

```

real(8),parameter :: PusaiS = 1.5D6 ! PS Ammonium Inhibition Coefficient [l/molN]
real(8),parameter :: KGppS = 6.93D-2 ! PS Temp. Coeff. for Photosynthetic Rate [/degC]
real(8),parameter :: MorPS0 = 5.85D4/d2s ! PS Mortality Rate @0degC [s]
real(8),parameter :: KMorPS = 6.93D-2 ! PS Temp. Coeff. for Mortality [/degC]
real(8),parameter :: ResPS0 = 0.03D0/d2s ! PS Respiration Rate at @0degC [s]
real(8),parameter :: KResPS = 0.0519D0 ! PS Temp. Coeff. for Respiration [/degC]
real(8),parameter :: GammaS = 0.135D0 ! PS Ratio of Extracell. Excret. to Photo. [(nodim)]
real(8),parameter :: VmaxL = 0.8D0/d2s ! PL Maximum Photosynthetic rate @0degC [s]
real(8),parameter :: KNO3L = 3.00D-6 ! PL Half saturation constant for Nitrate [molN/l]
real(8),parameter :: KNH4L = 0.30D-6 ! PL Half saturation constant for Ammonium [molN/l]
real(8),parameter :: KSiL = 6.00D-6 ! PL Half saturation constant for Silicate [molSi/l]
real(8),parameter :: PusaiL = 1.50D6 ! PL Ammonium Inhibition Coefficient [l/molN]
real(8),parameter :: KGppL = 6.93D-2 ! PL Temp. Coeff. for Photosynthetic Rate [/degC]
real(8),parameter :: MorPL0 = 2.90D4/d2s ! PL Mortality Rate @0degC [s]
real(8),parameter :: KMorPL = 6.93D-2 ! PL Temp. Coeff. for Mortality [/degC]
real(8),parameter :: ResPL0 = 0.03D0/d2s ! PL Respiration Rate at @0degC [s]
real(8),parameter :: KResPL = 0.0519D0 ! PL Temp. Coeff. for Respiration [/degC]
real(8),parameter :: GammaL = 0.135D0 ! PL Ratio of Extracell. Excret. to Photo. [(nodim)]
real(8),parameter :: GRmaxS = 0.40D0/d2s ! ZS Maximum Rate of Grazing PS @0degC [s]
real(8),parameter :: KGraS = 6.93D-2 ! ZS Temp. Coeff. for Grazing [/degC]
real(8),parameter :: LamS = 1.40D6 ! ZS Ivlev constant [l/molN]
real(8),parameter :: PS2ZSstar= 0.043D-6 ! ZS Threshold Value for Grazing PS [molN/l]
real(8),parameter :: AlphaZS = 0.70D0 ! ZS Assimilation Efficiency [(nodim)]
real(8),parameter :: BetaZS = 0.30D0 ! ZS Growth Efficiency [(nodim)]
real(8),parameter :: MorZS0 = 5.85D4/d2s ! ZS Mortality Rate @0degC [s]
real(8),parameter :: KMorZS = 0.0693D0 ! ZS Temp. Coeff. for Mortality [/degC]
real(8),parameter :: GRmaxLps = 0.10D0/d2s ! ZL Maximum Rate of Grazing PS @0degC [s]
real(8),parameter :: GRmaxLpl = 0.40D0/d2s ! ZL Maximum Rate of Grazing PL @0degC [s]
real(8),parameter :: GRmaxLzs = 0.40D0/d2s ! ZL Maximum Rate of Grazing ZS @0degC [s]
real(8),parameter :: KGraL = 6.93D-2 ! ZL Temp. Coeff. for Grazing [/degC]
real(8),parameter :: LamL = 1.4000D6 ! ZL Ivlev constant [l/molN]
real(8),parameter :: PS2ZLstar= 4.00D-8 ! ZL Threshold Value for Grazing PS [molN/l]
real(8),parameter :: PL2ZLstar= 4.00D-8 ! ZL Threshold Value for Grazing PL [molN/l]
real(8),parameter :: ZS2ZLstar= 4.00D-8 ! ZL Threshold Value for Grazing ZS [molN/l]
real(8),parameter :: AlphaZL = 0.70D0 ! ZL Assimilation Efficiency [(nodim)]
real(8),parameter :: BetaZL = 0.30D0 ! ZL Growth Efficiency [(nodim)]
real(8),parameter :: MorZL0 = 5.85D4/d2s ! ZL Mortality Rate @0degC [s]
real(8),parameter :: KMorZL = 0.0693D0 ! ZL Temp. Coeff. for Mortality [/degC]
real(8),parameter :: GRmaxPpl = 0.20D0/d2s ! ZP Maximum rate of grazing PL @0degC [s]
real(8),parameter :: GRmaxPzs = 0.20D0/d2s ! ZP Maximum rate of grazing ZS @0degC [s]
real(8),parameter :: GRmaxPzl = 0.20D0/d2s ! ZP Maximum rate of grazing ZL @0degC [s]
real(8),parameter :: KGraP = 6.93D-2 ! ZP Temp. Coeff. for grazing [/degC]
real(8),parameter :: LamP = 1.4000D6 ! ZP Ivlev constant [l/molN]
real(8),parameter :: PL2ZPstar= 4.00D-8 ! ZP Threshold Value for Grazing PL [molN/l]
real(8),parameter :: ZS2ZPstar= 4.00D-8 ! ZP Threshold Value for Grazing ZS [molN/l]
real(8),parameter :: ZL2ZPstar= 4.00D-8 ! ZP Threshold Value for Grazing ZL [molN/l]
real(8),parameter :: PusaiPL = 4.605D6 ! ZP Preference Coeff. for PL [l/molN]
real(8),parameter :: PusaiZS = 3.010D6 ! ZP Preference Coeff. for ZS [l/molN]
real(8),parameter :: AlphaZP = 0.70D0 ! ZP Assimilation Efficiency [(nodim)]
real(8),parameter :: BetaZP = 0.30D0 ! ZP Growth Efficiency [(nodim)]
real(8),parameter :: MorZP0 = 5.85D4/d2s ! ZP Mortality Rate @0degC [s]

```

```

real(8),parameter :: KMorZP = 0.0693D0 ! ZP Temp. Coeff. for Mortality [degC]
real(8),parameter :: Nit0 = 0.03D0/d2s ! NH4 Nitrification Rate @0degC [s]
real(8),parameter :: KNit = 0.0693D0 ! NH4 Temp. coefficient for Nitrification [degC]
real(8),parameter :: VP2N0 = 0.10D0/d2s ! PON Decomp. Rate to Ammonium @0degC [s]
real(8),parameter :: KP2N = 6.93D-2 ! PON Temp. Coeff. for Decomp. to Ammon. [degC]
real(8),parameter :: VP2D0 = 0.10D0/d2s ! PON Decomp. Rate to DON @0degC [s]
real(8),parameter :: KP2D = 6.93D-2 ! PON Temp. Coeff. for Decomp. to DON [degC]
real(8),parameter :: VD2N0 = 0.20D0/d2s ! DON Decomp. Rate to Ammonium @0degC [s]
real(8),parameter :: KD2N = 6.93D-2 ! DON Temp. Coeff. for Decomp. to Ammon. [degC]
real(8),parameter :: VO2S0 = 0.10D0/d2s ! Opal Decomp. Rate to Silicate @0degC [s]
real(8),parameter :: KO2S = 6.93D-2 ! Opal Temp. Coeff. for Decomp.to Silicate[degC]
real(8),parameter :: RSiN = 2.0D0 !Si/N ratio [molSi/molN]
real(8),parameter :: RCN = 106.0D0/16.0D0 !C/N ratio [molC/molN]
! ..... bottom boundary Condition .....
real(8),parameter :: setVPON = 40.0D0/d2s ! Settling velocity of PON [m/s]
real(8),parameter :: setVOpal = 40.0D0/d2s ! Settling velocity of Opal [m/s]
real(8),parameter :: TNO3d = 25.0d-6 ! Nitrate Concentraion in the Deep Layer [molN/l]
real(8),parameter :: TSiOH4d = 35.0d-6 ! Silicate Concentraion in the Deep Layer [molSi/l]
! ..... Paramters of ZL Vertical Migration .....
character(19) :: CZup='0000/04/01 00:00:00' ! Date Coming up to the Euphotic Layer
character(19) :: CZdwn='0000/09/01 00:00:00' ! Date Returning to the Deep Layer
real(8) :: TZup, TZdwn, TZLd, SVRate=0.2D0
integer :: IyrU, ImonU, IdayU ,IhourU, IminU, IsecU
integer :: IyrD, ImonD, IdayD ,IhourD, IminD, IsecD
!
real(8) :: GppPSn, GppNPSn, GppAPSn, RnewS, ResPSn, MorPSn, ExcPSn
real(8) :: GppPLn, GppNPLn, GppAPLn, RnewL, ResPLn, MorPLn, ExcPLn
real(8) :: GppPLsi, GppSiPLsi, ResPLsi, MorPLsi, ExcPLsi
real(8) :: GraPS2ZSn, GraPS2ZLn, GraPL2ZLn, GraPL2ZLsi, GraZS2ZLn
real(8) :: GraPL2ZPn, GraPL2ZPsi, GraZS2ZPn, GraZL2ZPn
real(8) :: EgeZSn, MorZSn, ExcZSn, EgeZLn, EgeZLsi, MorZLn, ExcZLn
real(8) :: EgeZPn, EgeZPsi, MorZPn, ExcZPn
real(8) :: DecP2N, DecP2D, DecD2N, DecO2S, Nit
real(8) :: ExpPON, ExpOpal,ExcNO3, ExcSiOH4
integer :: lt=0 , nt
!
! ..... Environmental Condition .....
real(8) :: Temp ! Temperature [degC]
real(8) :: Lint0 ! Light Intencity at sea surface [ly/min]
real(8) :: MLD = 30.0d0 ! Mixed Layer Depth [m]
real(8) :: ExcTime = 1.0d0 / (100.0d0*d2s) ! Exch. Coeff. between Sur-Deep [s]
! ..... statement function & def. type of functions .....
real(8) :: cd2tt, nd2tt
character(19) :: tt2cd
real(8) :: Td, GraF, Mich, a, b, c
Td (a,b) = a * exp(b*Temp)
GraF(a,b,c) = MAX( 0.0D0, 1.0 - exp(a * (b - c)))
Mich(a,b) = b / ( a + b )
!
! ***** Initial Setting *****
! ..... for time control .....

```

```

TTime = cd2tt(Cstart)                ! Starting Date
CTime = TT2CD(TTime)                 ! present time (character form)
dt = cd2tt(Cstep) - cd2tt('0000/00/00 00:00:00') ! Time Step (real8 form)
Tmon = cd2tt(Cmon) - cd2tt('0000/00/00 00:00:00') ! Monitor Interval (real8 form)
nt = NINT( ( cd2tt(Cend) - cd2tt(Cstart) ) / dt ) ! Total Time Steps
! ..... for Vertical Migration .....
TZup = CD2TT( CZup )
TZdwn = CD2TT( CZdwn )
call TT2ND(IyrU, ImonU, IdayU, IhourU, IminU, IsecU, TZup )
call TT2ND(IyrD, ImonD, IdayD, IhourD, IminD, IsecD, TZdwn)
TZLd = TZL ! ZL living in the deep layer at the initial condition
TZL = 0.0
! ..... File Open for monitoring output .....
open( 10, file='Results.csv', form='FORMATTED' )
write(10,'(A,13(", ", A))' ) 'Time(day)', &
      'NO3' , 'NH4' , 'PS' , 'PL' , &
      'ZS' , 'ZL' , 'ZP' , 'PON' , &
      'DON' , 'SiOH4' , 'Opal' , 'TotalN' , 'TotalSi'
write(10,'(A,11(", ", F8.4))' ) CTime, &
      TNO3/mcr, TNH4 /mcr, TPS /mcr, TPL /mcr, &
      TZS /mcr, TZL /mcr, TZP /mcr, TPON/mcr, &
      TDON/mcr, TSiOH4/mcr, TOpal/mcr
open( 11, file='Forcing.csv', form='FORMATTED' )
write(11,'(A,13(", ", A))' ) 'Time(day)', 'Lint0', 'TMP', 'MLD', 'ExcTime'
write(11,'(A,13(", ", 1PE10.4))' ) CTime, Lint0, Temp, MLD, ExcTime*d2s
!
! ***** Main Loop *****
do It = 1, nt
! ..... time control (Season : 0 to 1, percentage in a year).....
Tbefore = TTime                ! one step before present time
TTime = TTime + dt             ! present time (real8 form)
CTime = TT2CD(TTime)          ! present time (character form)
CALL TT2ND(Iyr, Imon, Iday, Ihour, Imin, Isec, TTime)
Season = ( TTime - ND2TT(Iyr, 1,1,0,0,0) ) / &
         ( ND2TT(Iyr+1,1,1,0,0,0) - ND2TT(Iyr, 1,1,0,0,0) )
!
! ..... Example of Boundray condition .....
Lint0 = 0.1d0 * ( 1.0D0 + 0.3d0 * cos( 2.0d0*3.1415926536d0*(Season - 0.50D0) ) )
Temp = 6.0D0 + 4.0d0 * cos( 2.0d0*3.1415926536d0*(Season - 0.65D0) )
if (Temp .lt. 4.0 ) then
MLD = MLD + dt * ( 150.0d0 - MLD ) / ( 100.0d0 * d2s )
ExcTime = ExcTime + dt * ( 1.0d0/( 40.0d0*d2s) - ExcTime ) / (100.0d0*d2s)
else
MLD = MLD + dt * ( 30.0d0 - MLD ) / ( 5.0d0 * d2s )
ExcTime = ExcTime + dt * ( 1.0d0/(100.0d0*d2s) - ExcTime ) / ( 5.0d0*d2s)
end if
!
! ..... Light Factors (LfcS, LfcL).....
Lint = Lint0
LfcDS = Lint/IoptS * exp(1.0D0 - Lint/IoptS)
LfcDL = Lint/IoptL * exp(1.0D0 - Lint/IoptL)

```

```

LfcS = 0.0D0
LfcL = 0.0D0
Kappa = alpha1 + alpha2 * ( TPS + TPL )
dLint = exp( -Kappa * (MLD/LLN) )
do L = 1, LLN
  LfcUS = LfcDS
  LfcUL = LfcDL
  Lint = Lint * dLint
  LfcDS = Lint/IoptS * exp( 1.0D0 - Lint/IoptS )
  LfcDL = Lint/IoptL * exp( 1.0D0 - Lint/IoptL )
  LfcS = LfcS + ( LfcUS + LfcDS ) * 0.5D0 / LLN
  LfcL = LfcL + ( LfcUL + LfcDL ) * 0.5D0 / LLN
end do
!
..... Photosynthesis of PS .....
GppNPSn = Mich( KNO3S, TNO3 ) * exp( - PusaiS * TNH4 )
GppAPSn = Mich( KNH4S, TNH4 )
GppPSn = Td(VmaxS, KGppS) * LfcS * TPS * ( GppNPSn + GppAPSn )
RnewS = GppNPSn / ( GppNPSn + GppAPSn )

ResPSn = Td( ResPS0, KResPS ) * TPS
MorPSn = Td( MorPS0, KMorPS ) * TPS * TPS
ExcPSn = GammaS * GppPSn
!
..... Photosynthesis of PL .....
GppNPLn = Mich( KNO3L, TNO3 ) * exp( - PusaiL * TNH4 )
GppAPLn = Mich( KNH4L, TNH4 )
GppSiPLsi = Mich( KSiL , TSiOH4 )
GppPLn = Td(VmaxL, KGppL) * LfcL * TPL * min( ( GppNPLn + GppAPLn ), GppSiPLsi )
RnewL = GppNPLn / ( GppNPLn + GppAPLn )
ResPLn = Td( ResPL0, KResPL ) * TPL
MorPLn = Td( MorPL0, KMorPL ) * TPL * TPL
ExcPLn = GammaL * GppPLn
!
..... Grazing PS, PL, ZS, ZL --> ZS, ZL, ZP .....
GraPS2ZSn = Td(GRmaxS, KGraS) * GraF(LamS,PS2ZSstar,TPS) * TZS
GraPS2ZLn = Td(GRmaxLps,KGraL) * GraF(LamL,PS2ZLstar,TPS) * TZL
GraPL2ZLn = Td(GRmaxLpl,KGraL) * GraF(LamL,PL2ZLstar,TPL) * TZL
GraZS2ZLn = Td(GRmaxLzs,KGraL) * GraF(LamL,ZS2ZLstar,TZS) * TZL
GraPL2ZPn = Td(GRmaxPpl,KGraP) * GraF(LamP,PL2ZPstar,TPL) * TZP * exp( -PusaiPL *(TZL
+ TZS))
GraZS2ZPn = Td(GRmaxPzs,KGraP) * GraF(LamP,ZS2ZPstar,TZS) * TZP * exp( -PusaiZS * TZL )
GraZL2ZPn = Td(GRmaxPzl,KGraP) * GraF(LamP,ZL2ZPstar,TZL) * TZP
!
..... Mortality, Excretion, Egestion for Zooplanktons
!
..... Commented out after Saito-san Meeting at 19 Jun, 2000 .....
!
BetaZS = 0.3 ** ( 1.0 + Mich( TPL, TPS ) )
ExcZSn = (AlphaZS- BetaZS) * GraPS2ZSn
EgeZSn = (1.0 - AlphaZS) * GraPS2ZSn
MorZSn = Td( MorZS0, KMorZS ) * TZS * TZS
ExcZLn = (AlphaZL- BetaZL) * (GraPS2ZLn+GraPL2ZLn+GraZS2ZLn)
EgeZLn = (1.0 - AlphaZL) * (GraPS2ZLn+GraPL2ZLn+GraZS2ZLn)
MorZLn = Td( MorZL0, KMorZL ) * TZL * TZL
ExcZPn = (AlphaZP- BetaZP) * (GraPL2ZPn+GraZS2ZPn+GraZL2ZPn)
EgeZPn = (1.0 - AlphaZP) * (GraPL2ZPn+GraZS2ZPn+GraZL2ZPn)

```

$$\text{MorZPn} = \text{Td}(\text{MorZP0}, \text{KMorZP}) * \text{TZP} * \text{TZP}$$

! Decomposition PON, DON, Opal ---> NH4, DON, SiOH4

$$\text{DecP2N} = \text{Td}(\text{VP2N0}, \text{KP2N}) * \text{TPON}$$

$$\text{DecP2D} = \text{Td}(\text{VP2D0}, \text{KP2D}) * \text{TPON}$$

$$\text{DecD2N} = \text{Td}(\text{VD2N0}, \text{KD2N}) * \text{TDON}$$

$$\text{DecO2S} = \text{Td}(\text{VO2S0}, \text{KO2S}) * \text{TOpal}$$

$$\text{Nit} = \text{Td}(\text{Nit0}, \text{KNit}) * \text{TNH4}$$

! silica fluxes

$$\text{GppPLsi} = \text{GppPLn} * \text{RSiN}$$

$$\text{ResPLsi} = \text{ResPLn} * \text{RSiN}$$

$$\text{MorPLsi} = \text{MorPLn} * \text{RSiN}$$

$$\text{ExcPLsi} = \text{ExcPLn} * \text{RSiN}$$

$$\text{GraPL2ZLsi} = \text{GraPL2ZLn} * \text{RSiN}$$

$$\text{GraPL2ZPsi} = \text{GraPL2ZPn} * \text{RSiN}$$

$$\text{EgeZLsi} = \text{GraPL2ZLsi}$$

$$\text{EgeZPsi} = \text{GraPL2ZPsi}$$

!

! Tendency Terms for biological processes

$$\text{QNO3} = -(\text{GppPSn} - \text{ResPSn}) * \text{RnewS} \ \&$$

$$\quad -(\text{GppPLn} - \text{ResPLn}) * \text{RnewL} + \text{Nit}$$

$$\text{QNH4} = -(\text{GppPSn} - \text{ResPSn}) * (1.0 - \text{RnewS}) \ \&$$

$$\quad -(\text{GppPLn} - \text{ResPLn}) * (1.0 - \text{RnewL}) \ \&$$

$$\quad - \text{Nit} + \text{DecP2N} + \text{DecD2N} + \text{ExcZSn} + \text{ExcZLn} + \text{ExcZPn}$$

$$\text{QPS} = \text{GppPSn} - \text{ResPSn} - \text{MorPSn} - \text{ExcPSn} - \text{GraPS2ZSn} - \text{GraPS2ZLn}$$

$$\text{QPL} = \text{GppPLn} - \text{ResPLn} - \text{MorPLn} - \text{ExcPLn} - \text{GraPL2ZLn} - \text{GraPL2ZPn}$$

$$\text{QZS} = \text{GraPS2ZSn} - \text{GraZS2ZLn} - \text{MorZSn} - \text{ExcZSn} - \text{EgeZSn} - \text{GraZS2ZPn}$$

$$\text{QZL} = \text{GraPL2ZLn} + \text{GraZS2ZLn} - \text{MorZLn} - \text{ExcZLn} - \text{EgeZLn} + \text{GraPS2ZLn} - \text{GraZL2ZPn}$$

$$\text{QZP} = \text{GraPL2ZPn} + \text{GraZS2ZPn} - \text{MorZPn} - \text{ExcZPn} - \text{EgeZPn} + \text{GraZL2ZPn}$$

$$\text{QPON} = \text{MorPSn} + \text{MorPLn} + \text{MorZSn} + \text{MorZLn} + \text{MorZPn} \ \&$$

$$\quad + \text{EgeZPn} + \text{EgeZSn} + \text{EgeZLn} - \text{DecP2N} - \text{DecP2D}$$

$$\text{QDON} = \text{ExcPSn} + \text{ExcPLn} + \text{DecP2D} - \text{DecD2N}$$

$$\text{QSiOH4} = \text{GppPLsi} + \text{ResPLsi} + \text{ExcPLsi} + \text{DecO2S}$$

$$\text{QOpal} = \text{MorPLsi} + \text{EgeZLsi} + \text{EgeZPsi} - \text{DecO2S}$$

!

! Exchange Fluxes between the Surface and Deep Layers

$$\text{ExpPON} = \text{setVPON} / \text{MLD} * \text{TPON}$$

$$\text{ExpOpal} = \text{setVOpal} / \text{MLD} * \text{TOpal}$$

$$\text{ExcNO3} = \text{ExcTime} * (\text{TNO3d} - \text{TNO3})$$

$$\text{ExcSiOH4} = \text{ExcTime} * (\text{TSiOH4d} - \text{TSiOH4})$$

$$\text{QNO3} = \text{QNO3} + \text{ExcNO3}$$

$$\text{QSiOH4} = \text{QSiOH4} + \text{ExcSiOH4}$$

$$\text{QPON} = \text{QPON} - \text{ExpPON}$$

$$\text{QOpal} = \text{QOpal} - \text{ExpOpal}$$

!

! Time Integration with Forward Scheme

$$\text{TNO3} = \text{TNO3} + \text{dt} * \text{QNO3}$$

$$\text{TNH4} = \text{TNH4} + \text{dt} * \text{QNH4}$$

$$\text{TPS} = \text{TPS} + \text{dt} * \text{QPS}$$

$$\text{TPL} = \text{TPL} + \text{dt} * \text{QPL}$$

$$\text{TZS} = \text{TZS} + \text{dt} * \text{QZS}$$

$$\text{TZL} = \text{TZL} + \text{dt} * \text{QZL}$$

```

TZP = TZP + dt * QZP
TPON = TPON + dt * QPON
TDON = TDON + dt * QDON
TSiOH4 = TSiOH4 + dt * QSiOH4
TOpal = TOpal + dt * QOpal
!
! ..... Vertical Migration of ZL .....
TZdwn = ND2TT(Iyr, ImonD, IdayD ,IhourD, IminD, IsecD )
TZup = ND2TT(Iyr, ImonU, IdayU ,IhourU, IminU, IsecU )
if ( (Tbefore .lt. TZdwn).and.(TTime .ge. TZdwn) ) then
  TZLd = TZL
  TZL = 0.0
  write(*,*) '*** Down ***', CTime
end if
if ( (Tbefore .lt. TZup).and.(TTime .ge. TZup) ) then
  TZL = SVRate * TZLd
  write(*,*) '*** UP ***', CTime
end if
!
call Saury(TTime, Tbefore, TZS, TZL, TZP, Temp)
!
! ..... Monitor .....
if ( int(TTime/Tmon).ne. int(Tbefore/Tmon) ) then
!   write(*,'(A,13(" ", F8.4))') CTime, Season
!   write(10,'(A,11(" ", F8.4))') CTime, &
!     TNO3/mcr, TNH4 /mcr, TPS /mcr, TPL /mcr, &
!     TZS /mcr, TZL /mcr, TZP /mcr, TPON/mcr, &
!     TDON/mcr, TSiOH4/mcr, TOpal/mcr
!   write(11,'(A,13(" ", 1PE10.4))') CTime, Lint0, Temp, MLD, ExcTime*d2s
!   end if
end do
!
close(10); close(11)
!
stop
end
!*****
Subroutine Saury(TTime, Tbefore, TZS, TZL, TZP, Temp)
!
implicit none
real(8)      :: TTime, Tbefore, TZS, TZL, TZP, Temp
real(8),parameter :: d2s    = 86400.0d0 ! day ---> sec
integer      :: Iyr, Imon, Iday, Ihour, Imin, Isec
character(19) :: CAge ='0000/03/01 00:00:00' ! Date of Aging ( JJday = 200 )
character(19) :: CTime
character(19) :: CAge2 ='0000/07/01 00:00:00' ! Date of Aging ( JJday = 200 )
character(19) :: CTime2
real(8)      :: TAge
real(8)      :: TAge2
integer      :: iage = 0
integer, save  :: IyrA, ImonA, IdayA ,IhourA, IminA, IsecA

```

```

integer, save  :: IyrB, ImonB, IdayB ,IhourB, IminB, IsecB
integer       :: JJday
real(8)      :: ZooP1, ZooP2, ZooP3, tt1
real(8)      :: t1,t2,wtemp
real(8)      :: x(1)=0.2d0, xdot(1)
real(8)      :: cd2tt, nd2tt
character(19) :: tt2cd
real(8)      :: vul(3), k(3)
integer(4)   :: id(365)
real(8)      :: zop1(365), zop2(365), zop3(365)
real(8):: v, a, u, resp
real(8) :: xk1,xk2,xk3,xk4,te1,te2,te3,te4,tt5,t5,t4,tt7,t7,t6, gcta,gctb,gctemp,gcmax
real(8) :: cnum,c1,c2,c3,con1,con2,con3,con
real(8) :: f,e, sda
real(8) :: phalf
!
integer, save  :: First = 1
!
PHalf=0.100
!
=====
if ( First .eq. 1 ) then; First = 0
  TAge = CD2TT( CAge )
  call TT2ND(IyrA, ImonA, IdayA ,IhourA, IminA, IsecA ,TAge )
  TAge2 = CD2TT( CAge2 )
  call TT2ND(IyrB, ImonB, IdayB ,IhourB, IminB, IsecB ,TAge2 )
  open( 20, file='saury.csv', form='FORMATTED' )
!
!!!!   OPEN(UNIT=111,FILE='nemuro.txt',STATUS='unknown')
!!!!   -----read in the 3 zoop groups from Nemuro output last 3 columns
!!!!   do JJday=1,365
!!!!     READ(111,999)id(JJday),zop1(JJday),zop2(JJday),zop3(JJday)
!!!!   999   FORMAT(1x,i3,1x,3(e13.6,1x))
!!!!   end do
end if
!
=====
!
CTime = TT2CD(TTime)                ! present time (charactor form)
CALL TT2ND(Iyr, Imon, Iday ,Ihour, Imin, Isec ,TTime)
JJday = 1 + ( TTime - ND2TT(Iyr ,1,1,0,0,0) ) / d2s
!
!-----convert Nemuro zoop in uM N/L to g ww/m3
!----- tt1 is conversion from uM N/liter to g ww/m3
!----- 14 ug N/uM * 1.0e-6 g/ug * 1 g dw/0.07 g N dw * 1 g ww/0.2 g dw
!----- 1.0e3 liters/m3
!
tt1=14.0*1.0e-6*(1.0/0.07)*(1.0/0.2)*1.0e3
zoop1 = TZS*tt1 *1.0d6
zoop2 = TZL*tt1 *1.0d6

```

```

zoop3 = TZP*tt1 *1.0d6
!!!! zoop1 = zop1(JJday) * tt1
!!!! zoop2 = zop2(JJday) * tt1
!!!! zoop3 = zop3(JJday) * tt1
!
! ..... Temperature Seting .....
!
t1=float(jjday)
t2=12.75-10.99*cos(0.0172*t1)-6.63*sin(0.0172*t1)
wtemp=t2-5.0
IF(wtemp.le.1.0)wtemp=1.0

! write(*,*) TT2CD(cd2tt('0002/01/01 00:00:00')+200.0*86400.0)
! stop
! ..... Aging of saury .....
TAge = ND2TT(Iyr, ImonA, IdayA ,IhourA, IminA, IsecA )
if ( (Tbefore .lt. TAge).and.(TTime .ge. TAge) .and. (iage.eq.0)) then
    write(*,*) '*** Aging +1 of saury ***', CTime
    iage = iage + 1
end if
TAge2 = ND2TT(Iyr, ImonB, IdayB ,IhourB, IminB, IsecB )
if ( (Tbefore .lt. TAge2).and.(TTime .ge. TAge2) .and. (iage.eq.1) ) then
    write(*,*) '*** Aging +1 of saury ***', CTime
    iage = iage + 1
end if
!
!--- saury weight state variable = x(1)
!
!----- set vulnerabilities and k values for 3 zoop groups
!
if ( iage .eq. 0 ) then
    vul(1) = 1.0; vul(2) = 0.0; vul(3) = 0.0
    k (1) = phalf; k (2) = phalf; k (3) = phalf
else if ( iage .eq. 1 ) then
    vul(1) = 1.0; vul(2) = 1.0; vul(3) = 0.0
    k (1) = phalf; k (2) = phalf; k (3) = phalf
else
    vul(1) = 0.0; vul(2) = 1.0; vul(3) = 1.0
    k (1) = phalf; k (2) = phalf; k (3) = phalf
endif
!
! --- weight affect on respiration
!
tt1 = 1.0 / x(1)
t1 = 0.0033 * tt1**0.227
! --- *****this is the new stuff from Ahhrenius for YOY only*****
! --- The 5.258 puts resp (g oxygen/fish) into units of g zoop/g fish/day
! --- [13560 joules/gram oxygen]/4.18 joules/cal = 3244 cal/gO2
! --- [2580 joules/gram zoop]/4.18 joules/cal = 617 cal/g zoop
! --- so respiration in grams/oxygen/g fish/day is multiplied by 3244/617 = 5.258
! --- to get food energy equivalentents of a gram of oxygen respired

```

```

!
!ccc  IF (iage .eq. 0 )then
!ccc    IF(wtemp.le.15.0)then
!ccc      v = 5.76 * exp( 0.0238 * wtemp ) * x(1)**0.386
!ccc    else
!ccc      v = 8.6 * x(1)**0.386
!ccc    endif
!ccc    a=EXP((0.03-0.0*wtemp)*v)
!ccc    resp=t1*EXP(0.0548*wtemp)*a*5.258
! --- *****back to the old equations for respiration for age-1 and older*****
!ccc  else ! (iage .gt. 0)
!c    IF (wtemp.le.9.0)then
      IF (wtemp.le.12.0)then
!C      u=3.9*x(1)**0.13*EXP(0.149*wtemp)
      u=2.0*x(1)**0.33*EXP(0.149*wtemp)
      else
!c      u=15.0*x(1)**0.13
      u=11.7*x(1)**0.33
      endif
      resp=t1*EXP(0.0548*wtemp)*EXP(0.03*u)*5.258
!ccc  endif
!C
!C --- Thornton and Lessem temperature effect
!C --- age dependent values
!C --- *****Arrhenius for age-0 he changed te4 from 25 to 23 degrees*****
!C
      if ( iage .eq. 0 ) then
!c      xk1 = 0.1; xk2 = 0.98; xk3 = 0.98; xk4 = 0.01
!c      te1 = 1.0; te2 = 15.0; te3 = 17.0; te4 = 23.0
      xk1 = 0.1; xk2 = 0.98; xk3 = 0.98; xk4 = 0.5
      te1 = 5.0; te2 = 20.0; te3 = 26.0; te4 = 30.0
      else if ( iage .eq. 1 ) then
!c      xk1 = 0.1; xk2 = 0.98; xk3 = 0.98; xk4 = 0.01
!c      te1 = 1.0; te2 = 15.0; te3 = 17.0; te4 = 25.0
      xk1 = 0.1; xk2 = 0.98; xk3 = 0.98; xk4 = 0.5
      te1 = 5.0; te2 = 16.0; te3 = 20.0; te4 = 30.0
      else if( iage .gt. 1 ) then
!c      xk1 = 0.1; xk2 = 0.98; xk3 = 0.98; xk4 = 0.01
!c      te1 = 1.0; te2 = 13.0; te3 = 15.0; te4 = 23.0
      xk1 = 0.1; xk2 = 0.98; xk3 = 0.98; xk4 = 0.5
      te1 = 5.0; te2 = 16.0; te3 = 20.0; te4 = 30.0
      endif
!
      tt5 = ( 1.0 / ( te2 - te1 ) )
      t5 = tt5 * log( xk2 * ( 1.0 - xk1 ) / ( (1.0-xk2) * xk1 ) )
      t4 = exp( t5 * ( wtemp - te1 ) )
!
      tt7 = 1.0 / ( te4 - te3 )
      t7 = tt7 * log( xk3 * ( 1.0 - xk4 ) / ( (1.0-xk3) * xk4 ) )
      t6 = exp( t7 * ( te4 - wtemp ) )
!

```

```

gcta = ( xk1 * t4 ) / ( 1.0 + xk1 * ( t4 - 1.0 ) )
gctb = xk4 * t6 / ( 1.0 + xk4 * ( t6 - 1.0 ) )
gctemp= gcta * gctb
!c  gcmx = 0.642 * tt1**0.256 * gctemp
gcmx = 0.6 * tt1**0.256 * gctemp
!
! --- multispecies functional response
! --- usse either this or adjust little p
!
cnum=zoop1 * vul(1)/k(1) + zoop2*vul(2)/k(2) +zoop3 * vul(3)/k(3)
c1=gcmx*zoop1*vul(1)/k(1)
c2=gcmx*zoop2*vul(2)/k(2)
c3=gcmx*zoop3*vul(3)/k(3)
con1=c1/(1.0+cnum)
con2=c2/(1.0+cnum)
con3=c3/(1.0+cnum)
con= con1+con2+con3
!
!----if using constant p rather than functional response, set p here
! --- to tune to observed size at age data
!  con=0.425*gcmx
!
! --- egestion
!
f=0.16*con
!
! --- excretion
e=0.1*(con-f)
!
!
! --- Specific Dynamic Action
!
!c----- *****Arrhenius age dependent SDA from 17.5% to 15% *****
IF ( iage .eq. 0 ) then
sda=0.15*(con-f)
else
sda=0.175*(con-f)
end if
!C
!C --- use the ratio of calories/g of zoop (2580) to calories/g of fish (5533)
!C
!C --- bioenergetics differential equation
!C
xdot(1)=(con-resp-f-e-sda)*x(1)*2580./5533.
!
IF(wtemp.le.1.0)xdot(1)=0.0
!C
!C --- Spawning section. Assume loose 20% of bosal weight/day
!C  t1=float(jjday)
!  if( mod(jjday,365) .ge. 152.0 .and. mod(jjday,365) .le. 156.0) then
!  xdot(1)=(con-resp-f-e-sda-0.20)*x(1)*2580./5533.

```

```

!     write(*,*) '### Spawning ###'
!     endif
!
!     if (iage .eq. 1 ) then
!       write(*,*) JJday, wtemp, x(1), xdot(1)
!       stop
!     end if
!     write(*, '(A,I4,3(1PE14.5))') Ctime, JJday, wtemp, x(1), xdot(1)
!
!     Time Integration
!
!     x(1) = x(1) + 3600.0d0 /d2s * xdot(1)
!
!     ..... for Check .....
!     if ( int(TTime/d2s) .ne. int(Tbefore/d2s) ) then
!!       write(*, '(A,I4,3(1PE14.5))') Ctime, JJday, wtemp, x(1), xdot(1)
!!       stop
!!       write(*,*) TZS, zop1(JJday), TZL,zop2(JJday), TZP,zop3(JJday)
!!       write(*,*) TZP*1.0d6, zop3(JJday)
!       write(20, '(A,11(" ", F12.4))') CTime, x(1), wtemp, gcmx
!     end if
!
!     return
!
!     stop
!     end
!*****
!* Utilities for Date Control  Writtien by Yasuhiro Yamanaka (galapen@ees.hokudai.ac.jp) *
!*****
!   exp. 1997/12/31 23:59:59 --> 6.223158719900000E+10
!   exp. 0000/01/01 00:00:00 --> 0.000000000000000E+00
!*****
!   real(8) function CD2TT( Cdate )
!
!   integer    :: Iyr, Imon, Iday , Ihour, Imin, Isec
!   real(8)    :: ND2TT
!   character(19) :: Cdate
!
!   if ( len( Cdate ) .ne. 19 ) then
!     write(*,*) '### Length of date is no good ###'
!     stop
!   end if
!   read (Cdate( 1: 4),*) Iyr
!   read (Cdate( 6: 7),*) Imon
!   read (Cdate( 9:10),*) Iday
!   read (Cdate(12:13),*) Ihour
!   read (Cdate(15:16),*) Imin
!   read (Cdate(18:19),*) Isec
!
!   CD2TT = ND2TT(Iyr, Imon, Iday , Ihour, Imin, Isec)
!
!

```

```

return
end function
|*****
! exp. 6.223158719900000E+10 --> 1997/12/31 23:59:59
|*****
character(19) function TT2CD(tt)
!
integer :: Iyr, Imon, Iday , Ihour, Imin, Isec
real(8) :: tt
!
call TT2ND( Iyr, Imon, Iday, Ihour, Imin, Isec , tt )
!
write(TT2CD,'(I4.4,5(A,I2.2))') Iyr, '/', Imon, '/', Iday, &
      ' ', Ihour, ':', Imin, ':', Isec
!
return
end function
|*****
! exp. 1997,12,31,23,59,59 --> 6.223158719900000E+10
|*****
real(8) function ND2TT(Iyr, Imon, Iday, Ihour, Imin, Isec)
!
integer :: IM2D(12,0:1) = &
      reshape( (/ 0,31,59,90,120,151,181,212,243,273,304,334, &
                0,31,60,91,121,152,182,213,244,274,305,335 /), (/12,2/) )
integer :: Iyr, Imon, Iday, Ihour, Imin, Isec
integer :: Iy4, Iy1, Ileap, Im, Itt
!
!
Iy4 = 1461 * ( Iyr / 4 )
Iy1 = 365 * mod( Iyr, 4 )
!
if( mod( Iyr, 4 ) .ne. 0 ) then
  Ileap = 0
else
  Ileap = 1
end if
Im = IM2D( Imon, Ileap)
!
Itt = Iy4 + Iy1 + Im + Iday - Ileap
!
ND2TT = Ihour * 3600 + Imin * 60 + Isec
ND2TT = ND2TT + Itt * 86400.0D0
!
return
end function
|*****
! exp. 6.223158719900000E+10 --> 1997,12,31,23,59,59
|*****
subroutine TT2ND(
      &
      Iyr , Imon , Iday , Ihour, Imin, Isec, & !O & I

```

```

        tt )
!
integer :: Iyr, Imon, Iday , Ihour, Imin, Isec
integer :: Itt, Iy, Iy4, Iyd, Iy1, Ileap, Imd, Im, Its
integer :: IM2D(12,0:1) = &
    reshape( (/ 0,31,59,90,120,151,181,212,243,273,304,334, &
        0,31,60,91,121,152,182,213,244,274,305,335 /), (/12,2/) )
integer :: IY2D(4) = (/0,366,731,1096/)
real(8) :: tt, tt0, ND2TT
!
!
! ..... ITT [day] .....
Itt = 1 + tt / 86400.0D0
!
Iy4 = (Itt-1) / 1461
Iyd = Itt - Iy4 * 1461
do IY = 1, 4
    if ( IY2D(Iy) + 1 .le. Iyd ) then
        Iy1 = Iy
    end if
end do
!
Iyr = Iy4 * 4 + Iy1 - 1
if ( mod(Iyr,4) .ne. 0 ) then
    Ileap = 0
else
    Ileap = 1
end if
IMD = IYD - IY2D(IY1)
!
do IM = 1, 12
    if ( IM2D(IM,ILEAP)+1 .le. IMD ) then
        IMON = IM
    end if
end do
IDAY = IMD - IM2D(IMON,ILEAP)
!
TT0 = ND2TT(IYR, IMON, IDAY ,0,0,0)
ITS = nint( TT - TT0 )
Ihour = ITS / 3600
Imin = ( ITS - Ihour * 3600 ) / 60
Isec = ITS - Ihour * 3600 - Imin * 60
!
return
end subroutine
!

```