```
!══════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════
!   Program:  SarTest, the interactive version of the Sar Monty Carlo preformance test of Sar --- 10/17/2001
!   The program writes the results on a user named output file; those results are summerized by the program "Errors".
!   ••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
    program SarTest
    implicit none
    character( 1) :: yn
    character( 2) :: trm
    character( 8) :: ofile
    integer(4) :: i,j,l, jr,lm,  it,isd,  n1,n2,nt
    integer(4),allocatable :: nc(:), af(:)
    real(8) :: zz, dt, xl, sm, se, fs, r0=0.0, p5=0.5, r3=3.0
    real(8),external :: Uni, Z0
    real(8),allocatable :: ff(:), lf(:),  cf(:), sf(:)
    integer(4) :: m, nm, mh(3), lr, lx, jx
    real(8) :: as,gk,rd,rl,dk,cv,sd,scv,ssd
    !••••••••••••••••••••••••••••••• SCVs •••••••••••••••••••••••••••••••••••••••••••••••
    data jr,jx,dk,zz,n1,n2/ 2, 20, 0.707, 0.45, 1000, 500000 /
    !••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
    write(*,"(/1x,'                       There are 4 possible species profiles as follows:')")
    write(*,"(/1x,'Profile              Description               L(∞)     K       BirthDate     BirthLength')")
    write(*,"( 1x,'───────    ──────────────────────────────    ─────     ────     ─────────     ───────────')")
    write(*,"( 1x,'  Y1       YellowTail, Old Values (inches)    24.00    0.30       0.445          0.088    ')")
    write(*,"( 1x,'  Y2       YellowTail, New Values (cm)        62.00    0.30       0.445          0.088    ')")
    write(*,"( 1x,'  G1       Gag, Old Values (inches)           49.21    0.13       0.164          0.034    ')")
    write(*,"( 1x,'  G2       Gag, New Values (cm)              119.00    0.13       0.164          0.086    ')")
    write(*,"(/1x,'Enter the Profile for this run:  ')"); read*,trm
    selectcase(trm)     !  L(∞)       K        t(b)      l(b)                              !
      case('Y1','y1'); as= 24.00; gk=0.30; rd=0.445; rl=0.088  ! Old Yellowtail in inches  !
      case('Y2','y2'); as= 62.00; gk=0.30; rd=0.445; rl=0.224  ! New Yellowtail  in cm     !
      case('G1','g1'); as= 49.21; gk=0.13; rd=0.164; rl=0.034  ! Old Gag Grouper in inches !
      case('G2','g2'); as=119.00; gk=0.16; rd=0.164; rl=0.086  ! New Gag Grouper in cm     !
      casedefault; print*,'The Profile code entered is not recognized.'; stop
    endselect
    write(*,"(/' The variance of length at age is required; σ[length] can be proportional to length (& thus age)')")
    write(*,"( ' so that a constant cv exists, or constant over length & age so that a constant variance exists.')")
    write(*,"(/' Pick one -- a constant cv or a constant sd : ')"); read*,trm

    if(trm=='cv')then; print*,' Enter the cv[length|age]: '; read*,cv; sd=r0
    else;              print*,' Enter the sd[length|age]: '; read*,sd; cv=r0; endif

    write(*,"(/' Which estimators?     LS numbers ? (y or n):    ')"); read*,yn; if(yn=='y')then; nm=nm+1; mh(nm)=1; endif
    write(*,"( '                   LS proportions ? (y or n):    ')"); read*,yn; if(yn=='y')then; nm=nm+1; mh(nm)=2; endif
    write(*,"( '          Weighted LS proportions ? (y or n):    ')"); read*,yn; if(yn=='y')then; nm=nm+1; mh(nm)=3; endif

    write(*,"(/' How many Monty Carlo trials are required?  ')"); read*,nt
    write(*,"(/' Enter a random number starting point or a zero for the code to pick one:  ')"); read*,isd

    if(isd <= 0 )then; call system_clock(isd,i,j); do while (isd > 99999); isd = isd*p5; enddo; endif
    write(*,"(/' Enter a filename for the output file : ')"); read*,ofile
    open(1,file=ofile,status='unknown')
    write(1,"(4f8.2,2i8,2f8.3,2i8,2f8.4,i8,i9)")as,gk,rd,rl,jr,jx,dk,zz,n1,n2,cv,sd,nt,isd
    close(1)
    allocate( nc(jr:jx), af(0:jx), cf(0:jx), sf(0:jx) )
    sm = as - Exp(-gk*(dk-Dint(dk)+jx-rd))*(as-rl)
    se=sd; if(cv > r0)se=cv*sm; lm=sm+r3*se; allocate(ff(0:lm))
    !••••••••••••••••••••••••••••••••••••••••••••••• trial loop •••••••••••••••••••••••••••••••••••••••••••••••┐
    do it=1,nt                                                                                                 !
      write(*,"(1x)")                                                                                          !
      write(*,"($,1x,'Beginning trial ',i4)")it                                                                !
      !•••••• computes yc abundance in the sample ••••••••••••••••••┐                                          !
      af=r0                                                         !                                          !
      do j=jr,jx                                                    !                                          !
        nc(j) = n1 + Uni(isd)*(n2-n1+1)  ! initial cohort strength  !                                          !
        af(j) = Nint(nc(j)*Exp(-zz*j))   ! nos(age j)               !                                          !
      enddo                                                         !                                          !
      !•••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••┘                                          !
      !••••••••••••••••••••• sample draw •••••••••••••••••••••••••••••••••••••••••••••••┐                      !
      1 ff=r0; lx=0; lr=999999                                                         !                      !
      do j=jr,jx                                                                       !                      !
        dt = dk+j-rd       ! dt= □t                                                    !                      !
        xl = as - Exp(-gk*dt)*(as-rl)                                                  !                      !
        if (cv > r0) sd = xl * cv                                                      !                      !
        do i=1,af(j)                                                                   !                      !
          l = Z0(isd,xl,sd)                                                            !                      !
          if( l > lm)then; deallocate(ff); lm=lm+15; allocate(ff(0:lm)); goto 1; endif !                      !
          ff(l)=ff(l)+1                                                                !                      !
          if(l < lr)lr=l                                                               !                      !
          if(l > lx)lx=l                                                               !                      !
        enddo                                                                          !                      !
      enddo                                                                            !                      !
      allocate( lf(lr:lx) )                                                            !                      !
      do l=lr,lx; lf(l)=ff(l); enddo                                                   !                      !
      write(*,"($,'. Sampling finished. ')")                                           !                      !
      if (cv > r0) sd = r0                                                             !                      !
      !••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••┘                      !
      !•••••••••••••••••••••••••••••• estimation ••••••••••••••••••••••••••••••••••••••••••••••••┐            !
      call Cutter(as,gk,rd,rl,dk,lr,lx,jx,lf,cf)                                                 !            !
      open(1,file=ofile,position='append')                                                       !            !
      write(1,"(1x,i4,'  ',16x,14x,99i8)")it,af                                                  !            !
      write(1,"(1x,i4,' 0',16x,14x,99i8)")it,Nint(cf)                                            !            !
      close(1)                                                                                   !            !
      do m=1,nm                                                                                  !            !
        write(*,"($,'Estimating with method',i2,'.  ')")mh(m)                                    !            !
        sf=cf; scv=cv; ssd=sd                                                                    !            !
        call Sar(mh(m),as,gk,rd,rl,dk,scv,ssd,lr,lx,jx,lf,sf,fs)                                 !            !
        open(1,file=ofile,position='append')                                                     !            !
        write(1,"(1x,i4,i2,E16.8,2f7.4,99i8)")it,mh(m),fs,scv,ssd,Nint(sf)                       !            !
        close(1)                                                                                 !            !
      enddo                                                                                      !            !
      open(1,file=ofile,position='append')                                                       !            !
      close(1)                                                                                   !            !
      deallocate( lf )                                                                           !            !
    enddo !••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••┘••••••••••••┘
    write(*,"(/1x,' *******  The Results Are In The File:',a10,'  *******')")ofile
    stop
    end program SarTest

!═════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════
!   Program:  Sarf, the non-interactive version of the Sar Monty Carlo preformance test of Sar.
!   Revised may, 2002.
!   The program appends the results on the end of a user named input file.
!   Those results are summerized by the program "Errof".
!   Free field Input file format Example:
!     119.0 0.16 0.086 0.164 0.707 0.45 cv 0.15 2 20 10000 20000 500 0 no yes no
!   Field Deffinitions:
!         1) L(∞)
!         2) von Berlanffy growth parameter K
!         3) birth length
!         4) birth date as a portion of a year
!         5) sampling (catch) date as a portion of a year
!         6) the instantaneous total mortality rate
!         7) cv or sd
!         8) the level for the cv or the sd
!         9) the age at recruitment
!        10) the last age to be simulated
!        11) the minimum number of fish in a cohort
!        12) the maximum number of fish in a cohort
!        13) number of Monte Carlo trials
!        14) random number start or 0 for the code to generate one
!        15) yes or no for unweighted least squares with the length frequency sample as numbers of fish at length
!        16) yes or no for unweighted least squares with proportional length frequency samples
!        17) yes or no for least squares weighted by variance estimates with proportional length frequency samples
!   ••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
    program Sarf
    implicit none
    character( 1) :: lsn,lsp,mle
character( 2) :: trm
character(20) :: ofile
integer(4) :: i,j,l, jr,lm,  it,isd,  n1,n2,nt
integer(4),allocatable :: nc(:), af(:)
real(8) :: zz, dt, xl, sm, se, fs, r0=0.0, p5=0.5, r3=3.0
real(8),external :: Uni, Z0
real(8),allocatable :: ff(:), lf(:), cf(:), sf(:)
integer(4) :: m, nm, mh(3), lr, lx, jx
real(8) :: as,gk,rd,rl,dk,cv,sd,scv,ssd
write(*,"(/' Enter the file name for the input/output file:  ')"); read*,ofile
open(1,file=ofile,status='unknown')
read(1,*)as,gk,rl,rd,dk,zz,trm,cv,jr,jx,n1,n2,nt,isd,lsn,lsp,mle
if(isd <= 0 )then; call system_clock(isd,i,j); do while (isd > 99999); isd = isd*p5; enddo; endif
write(*,"(/' The SCVs are:')")
write(*,"( '         L(∞)     = ',f7.2)")as
write(*,"( '         K        = ',f7.2)")gk
write(*,"( '         l(birth) = ',f8.3)")rl
write(*,"( '         t(birth) = ',f8.3)")rd
write(*,"( '         t(kill)  = ',f8.3)")dk
write(*,"( '         Z        = ',f7.2)")zz; if(trm=='cv')then; sd=r0;
write(*,"( '         cv[l]    = ',f7.2)")cv; else; sd=cv; cv=r0
write(*,"( '         sd[l]    = ',f7.2)")sd; endif
write(*,"( '         jr       = ',i4  )")jr
write(*,"( '         jx       = ',i4  )")jx
write(*,"( '         R-min    = ',i8  )")n1
write(*,"( '         R-max    = ',i8  )")n2
write(*,"( '         # trials = ',i4  )")nt
write(*,"( '         isd      = ',i8  )")isd;                             if(lsn=='y')then; nm=nm+1; mh(nm)=1;
write(*,"( '         Unweighted Least Squares Numbers')");         endif; if(lsp=='y')then; nm=nm+1; mh(nm)=2;
write(*,"( '         Unweighted Least Squares Proportions')");     endif; if(mle=='y')then; nm=nm+1; mh(nm)=3;
write(*,"( '         MLE (Weighted Least Squares) Proportions')"); endif
write(*,"(/' Proceed? ')"); pause
write(1,"(1x)")
close(1)
allocate( nc(jr:jx), af(0:jx), cf(0:jx), sf(0:jx) )
sm = as - Exp(-gk*(dk-Dint(dk)+jx-rd))*(as-rl)
se=sd; if(cv > r0)se=cv*sm; lm=sm+r3*se; allocate(ff(0:lm))
!••••••••••••••••••••••••••••••••••••••••••••••• trial loop ••••••••••••••••••••••••••••••••••••••••••••••┐
do it=1,nt                                                                                                !
  write(*,"(1x)")                                                                                         !
  write(*,"($,1x,'Beginning trial ',i4)")it                                                               !
  !•••••• computes yc abundance in the sample ••••••••••••••••••┐                                         !
  af=r0                                                         !                                         !
  do j=jr,jx                                                    !                                         !
    nc(j) = n1 + Uni(isd)*(n2-n1+1)  ! initial cohort strength  !                                         !
    af(j) = Nint(nc(j)*Exp(-zz*j))   ! nos(age j)               !                                         !
  enddo                                                         !                                         !
  !•••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••┘                                         !
  !••••••••••••••••••••• sample draw ••••••••••••••••••••••••••••••••••••••••┐                            !
  1 ff=r0; lx=0; lr=999999                                                   !                            !
  do j=jr,jx                                                                 !                            !
    dt = dk+j-rd       ! dt= □t                                              !                            !
    xl = as - Exp(-gk*dt)*(as-rl)                                            !                            !
    if (cv > r0) sd = xl * cv                                                !                            !
    do i=1,af(j)                                                             !                            !
      l = Z0(isd,xl,sd)                                                      !                            !
      if( l > lm)then                                                        !                            !
        deallocate(ff); lm=lm+15; allocate(ff(0:lm)); goto 1; endif         !                            !
      ff(l)=ff(l)+1                                                          !                            !
      if(l < lr)lr=l                                                         !                            !
      if(l > lx)lx=l                                                         !                            !
    enddo                                                                    !                            !
  enddo                                                                      !                            !
  allocate( lf(lr:lx) )                                                      !                            !
  do l=lr,lx; lf(l)=ff(l); enddo                                             !                            !
  write(*,"($,'. Sampling done; estimating with method(s):')")               !                            !
  if (cv > r0) sd = r0                                                       !                            !
  !••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••┘                            !
  !•••••••••••••••••••••••••••••• estimation ••••••••••••••••••••••••••••••••••••••••••••••••┐            !
```

```
      call Cutter(as,gk,rd,rl,dk,lr,lx,jx,lf,cf)                                                 !            !
      open(1,file=ofile,position='append')                                                       !            !
      write(1,"(1x,i4,'  ',16x,14x,99i8)")it,af                                                  !            !
      write(1,"(1x,i4,' 0',16x,14x,99i8)")it,Nint(cf)                                            !            !
      close(1)                                                                                   !            !
      do m=1,nm                                                                                  !            !
        write(*,"($,'  ',i2)")mh(m)                                                              !            !
        sf=cf; scv=cv; ssd=sd                                                                    !            !
        call Sar(mh(m),as,gk,rd,rl,dk,scv,ssd,lr,lx,jx,lf,sf,fs)                                 !            !
        open(1,file=ofile,position='append')                                                     !            !
        write(1,"(1x,i4,i2,E16.8,2f7.4,99i8)")it,mh(m),fs,scv,ssd,Nint(sf)                       !            !
        close(1)                                                                                 !            !
      enddo                                                                                      !            !
      deallocate( lf )                                                                           !            !
    enddo !••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••┴••••••••••••┘
    write(*,"(/1x,' *******  The Results Are In The File:',a10,'  *******')")ofile
    stop
    end program Sarf

!═════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════
!   Program:  Errors   reads the output produced by program "SarTest" and appends summarized
!                      estimation preformace statistics to the end of the file.  Oct 09 2001
!   ••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
    program Errors
    implicit none
    logical(1) :: cv, method(0:3)=(/.true.,.false.,.false.,.false./)
    character( 2) :: vl(2)=(/'cv','sd'/)
    character( 5) :: lin=' ────'
    character( 7) :: line='  ─────'
    character(10) :: ofile
    character( 5),allocatable :: dat(:)

    real(8) :: as,gk,rd,rl,dk,zz
    integer(4) :: jr,jx,n1,n2,nt,lr,lx,isd

    real(8) :: r0=0.0, r1=1.0, r2=2.0, r3=3.0, rp2=0.2, rp5=0.5, rd4=0.00005

    real(8) :: vr(2),vi(2),p0,p1,p2,q1,q2,av(0:3),vt(0:3),u,uc,u2,u1
    integer(4) :: h,i,j,k,l,m, nr,it,ni(0:3),nu(0:3)

    real(8),allocatable :: di(:), af(:,:), sf(:,:,:), sv(:,:), a(:), s(:), v(:), p(:,:)
    real(8),external :: Pr

    write(*,"(/' Enter the file name for the Sar MCT results: ')"); read*,ofile
    open(1,file=ofile,status='unknown')

    read(1,"(4f8.2,2i8,2f8.3,2i8,2f8.4,i8,i9)")as,gk,rd,rl,jr,jx,dk,zz,n1,n2,vr,nt,isd

    allocate( di(0:jx), af(nt,0:jx+1), sf(nt,0:3,0:jx), sv(nt,0:3), a(0:jx+1), s(0:jx+1), v(0:jx+1), dat(0:jx) )

    if(vr(1) > r0)then; cv=.true.; h=1; else; cv=.false.; h=2; endif
    nr=0; af=r0
    do
    1 read(1,"(1x,i4,2x,16x,14x,99f8.0)")it,di; nr=nr+1
      do j=0,jx; af(nr,j)=di(j); af(nr,jx+1)=af(nr,jx+1)+di(j); enddo
      do
        read(1,"(1x,i4,i2,16x,2f7.4,99f8.0)",end=2)i,m,vi,di
        if(i /= it)then; backspace 1; goto 1; endif
        method(m)=.true.
        do j=0,jx; sf(nr,m,j)=di(j); enddo
        sv(nr,m)=vi(h)
      enddo
    enddo
    2 close(1)
    open(1,file=ofile,position='append')
    write(1,"(/1x,116('─'))")
    write(1,"( ' SCVs:      L(∞)      =',f7.2)")as
    write(1,"( '             K        =',f7.2)")gk
    write(1,"( '         L[birth]     =',f8.3)")rl
write(1,"( '         t[birth]     =',f8.3)")rd
write(1,"( '         t[sample]    =',f8.3)")dk
write(1,"( '             Z        =',f7.2)")zz
write(1,"(     10x, a2,'[l:j]     =',f8.2)")vl(h),vr(h)
write(1,"( '       recruitmet age =',i8  )")jr
write(1,"( '         maximum age  =',i8  )")jx
write(1,"( '   minimum cohort nos =',i8  )")n1
write(1,"( '   maximum cohort nos =',i8  )")n2
write(1,"( '    number of trials  =',i8  )")nt
write(1,"( '  random number start =',i8  )")isd
write(1,"(/1x,t27,'        Cutting                 LS - Numbers          LS - Proportions     Maximum Likelihood  ')")
write(1,"( 1x,t27,'──────────────────────  ─────────────────────  ─────────────────────  ─────────────────────')")
write(1,"( 1x,t7 ,'   Length At Age  ',15x,'# trials',15x,'# trials',15x,'# trials',15x,'# trials')")
write(1,"( 1x,t7 ,'──────────────',15x,'────────',15x,'────────',15x,'────────',15x,'────────')")
write(1,"( 1x,'Age    μ     σ      cv ',t25,4('  μ[ϵ]   σ[ϵ]  ϵ=1  ϵ=∞'))")
write(1,"( 1x,'───  ─────  ─────  ────',t25,4('  ────  ─────  ───  ───'))")
do j=0,jx+1
  a(j) = as - Exp(-gk*(dk+j-rd))*(as-rl)
  if(cv)then; v(j)=(a(j)*vr(h))**2; else; v(j)=vr(h)**2; endif; s(j)=Sqrt(v(j))
enddo
do i=0,jx
  ni=0; nu=0; av=r0; vt=r0
  do m=0,3
    if(.not. method(m))cycle
    do k=1,nr
      if    (sf(k,m,i) == af(k,i))then; cycle
      elseif(sf(k,m,i) == r0     )then; nu(m)=nu(m)+1
      elseif(af(k,  i) == r0     )then; ni(m)=ni(m)+1
      else; p0=Abs(af(k,i)-sf(k,m,i))/af(k,i); av(m)=av(m)+p0; vt(m)=vt(m)+p0**2; endif
    enddo
    vt(m)=Sqrt( (vt(m)-av(m)*av(m)/nr)/(nr-1) )
    av(m)=av(m)/nr
  enddo
  write(1,"(1x,i3,f7.1,f7.2,f6.2,4(f6.2,f7.2,2i5))")i,a(i),s(i),s(i)/a(i),(av(m),vt(m),nu(m),ni(m),m=0,3)
enddo
av=r0; vt=r0
do m=0,3
  if(.not. method(m))cycle
  do i=1,nr
    p0=r0
    do j=0,jx
      p0=p0 + Abs( af(i,j)-sf(i,m,j) )
    enddo
    p0 = p0/af(i,jx+1)
    av(m) = av(m) + p0
    vt(m) = vt(m) + p0**2
  enddo
  vt(m)=Sqrt( (vt(m)-av(m)*av(m)/nr)/(nr-1) )
  av(m) = av(m)/nr
enddo
write(1,"( 1x,'tot',t25,4(f6.2,f7.2,10x))")(av(m),vt(m),m=0,3)
ni=0; nu=0; av=r0; vt=r0; p0=vr(h)
do m=1,3
  if(.not. method(m))cycle
  do i=1,nr
    p1=(p0-sv(i,m))/p0
    av(m)=av(m)+p1
    vt(m)=vt(m)+p1**2
  enddo
  vt(m)=Sqrt( (vt(m)-av(m)*av(m)/nr)/(nr-1) )
  av(m)=av(m)/nr
enddo
write(1,"(/2x,a2,t48,3(f6.2,f7.2,10x))")vl(h),(av(m),vt(m),m=1,3)
write(1,"(1x,116('•'))")
write(1,"(/1x,'         n                                  ┌  n ┌                     ┐²        ┐½ ')")
write(1,"( 1x,'μ[ϵ(j)] = Σ ϵ(i,j) / n   and   σ[ϵ(j)] = │  Σ │ϵ(i,j) - μ[ϵ(j)]│ / [n-1]│. ')")
write(1,"( 1x,'        i=1                                └ i=1└                     ┘        ┘  ')")
write(1,"(/1x,t12,'n = number of trials,')")
write(1,"( 1x,t12,'ϵ(i,j) = |ß(i,j) - p(i,j)| / p(i,j) = the proportion of age j that were misclassified on trial i,')")
write(1,"( 1x,t12,'ß(i,j) = estimated numbers at age j on trial i,')")
write(1,"( 1x,t12,'p(i,j) = actual numbers at age j on trial i.')")
write(1,"(/1x,t12,'ϵ = ∞ if p(i,j) = 0  and ß(i,j) > 0.')")
write(1,"( 1x,t12,'ϵ = 1 if p(i,j) > 0  and ß(i,j) = 0.')")
write(1,"( 1x,t12,'ϵ = 1  and ϵ = ∞  are not included in μ[ϵ] & σ²[ϵ] computations.')")
write(1,"(/1x,'      n ┌                          ┐                                                            ')")
write(1,"( 1x,'tot = Σ │ Σ{|ß(j) - p(j)|/[Σ p(j)]}│ = The proportion of total numbers that were misclassified. ')")
write(1,"( 1x,'     i=1└ j                j       ┘                                                            ')")
write(1,"( 1x,2x,'   n')")
write(1,"(1x,a2,' = Σ ',a2,'[estimate] / ',a2,'-1 = The estimation error of ',a2,'[length at age]')")vl(h),vl(h),vl(h),vl(h)
write(1,"( 1x,2x,'  i=1')")
write(1,"(1x,116('—'))")
write(1,"( t2,109('—'))")
write(1,"(/t2,'                 Overlaps: the amount of the P matrix length frequency common to both ages.')")
write(1,"(/t2,'                                                   Age                                      ')")
write(1,"( 1x,'Age',200i5)")(i,i=0,jx)
write(1,"( 1x,'———',200a5)")(lin,i=1,jx+1)
di=r0
do i=0,jx
  dat='     '
  do j=0,jx
    if(i==j)then; dat(i)=' 1.00'; cycle; endif
    if(v(i)==v(j))then; u=rp5*(a(i)+a(j))
      if(a(i) < a(j))then; uc=Pr(u,a(j),s(j)) + (r1-Pr(u,a(i),s(i)))
      else;                uc=Pr(u,a(i),s(i)) + (r1-Pr(u,a(j),s(j))); endif
    else
      u=v(i)*v(j)*((a(i)-a(j))**2-r2*(v(i)-v(j))*Log(Sqrt(v(j)/v(i))))
      q1=(a(j)*v(i)-a(i)*v(j)-Sqrt(u))/(v(i)-v(j)); q2=(a(j)*v(i)-a(i)*v(j)+Sqrt(u))/(v(i)-v(j))
      u1=Min(q1,q2); u2=Max(q1,q2)
      if(v(i) < v(j))then; uc=Pr(u1,a(i),s(i))+(Pr(u2,a(j),s(j))-Pr(u1,a(j),s(j)))+(r1-Pr(u2,a(i),s(i)))
      else;                uc=Pr(u1,a(j),s(j))+(Pr(u2,a(i),s(i))-Pr(u1,a(i),s(i)))+(r1-Pr(u2,a(j),s(j))); endif
    endif
    write(dat(j),"(f5.2)")uc
    if(uc >= rp2)di(j)=di(j)+r1
  enddo
  write(1,"(1x,i3,250a5)")i,dat
enddo
write(1,"(1x,3x,250i5)")Nint(di)
write(1,"(145('—'))")
deallocate (af, sf, sv, v)
lr=0; lx=r3*as
allocate( p(lr:lx,jr:jx) )
p=r0; di=r0
do j=jr,jx
  p1 = Pr(r0,a(j),s(j))
  do l=lr,lx
    u  = l+1
    p2 = Pr(u,a(j),s(j))
    p(l,j) = p2-p1
    p1=p2
  enddo
enddo
do l=lr,lx; do j=jr,jx
  if(p(l,j) 0.')")
write(1,"( 1x,t12,'ϵ = 1 if p(i,j) > 0  and ß(i,j) = 0.')")
write(1,"( 1x,t12,'ϵ = 1  and ϵ = ∞  are not included in μ[ϵ] & σ²[ϵ] computations.')")
write(1,"(/1x,'      n ┌                          ┐                                                            ')")
write(1,"( 1x,'tot = Σ │ Σ{|ß(j) - p(j)|/[Σ p(j)]}│ = The proportion of total numbers that were misclassified. ')")
write(1,"( 1x,'     i=1└ j                j       ┘                                                            ')")
write(1,"( 1x,2x,'   n')")
write(1,"( 1x,a2,' = Σ ',a2,'[estimate] / ',a2,'-1 = The estimation error of ',a2,'[length at age].')")vl,vl,vl,vl
write(1,"( 1x,2x,'  i=1')")
write(1,"(1x,116('—'))")
write(1,"( t2,109('—'))")
write(1,"(/t2,'                 Overlaps: the amount of the P matrix length frequency common to both ages.')")
write(1,"(/t2,'                                                   Age                                      ')")
```

```
    write(1,"( 1x,'Age',200i5)")(i,i=0,jx)
    write(1,"( 1x,'———',200a5)")(lin,i=1,jx+1)
    di=r0
    do i=0,jx
      dat='     '
      do j=0,jx
        if(i==j)then; dat(i)=' 1.00'; cycle; endif
        if(v(i)==v(j))then; u=rp5*(a(i)+a(j))
          if(a(i) < a(j))then; uc=Pr(u,a(j),s(j)) + (r1-Pr(u,a(i),s(i)))
          else;                uc=Pr(u,a(i),s(i)) + (r1-Pr(u,a(j),s(j))); endif
        else
          u=v(i)*v(j)*((a(i)-a(j))**2-r2*(v(i)-v(j))*Log(Sqrt(v(j)/v(i))))
          q1=(a(j)*v(i)-a(i)*v(j)-Sqrt(u))/(v(i)-v(j)); q2=(a(j)*v(i)-a(i)*v(j)+Sqrt(u))/(v(i)-v(j))
          u1=Min(q1,q2); u2=Max(q1,q2)
          if(v(i) < v(j))then; uc=Pr(u1,a(i),s(i))+(Pr(u2,a(j),s(j))-Pr(u1,a(j),s(j)))+(r1-Pr(u2,a(i),s(i)))
          else;                uc=Pr(u1,a(j),s(j))+(Pr(u2,a(i),s(i))-Pr(u1,a(i),s(i)))+(r1-Pr(u2,a(j),s(j))); endif
        endif
        write(dat(j),"(f5.2)")uc
        if(uc >= rp2)di(j)=di(j)+r1
      enddo
      write(1,"(1x,i3,250a5)")i,dat
    enddo
    write(1,"(1x,3x,250i5)")Nint(di)
    write(1,"(145('—'))")
    deallocate (af, sf, sv, v)
    lr=0; lx=r3*as
    allocate( p(lr:lx,jr:jx) )
    p=r0; di=r0
    do j=jr,jx
      p1 = Pr(r0,a(j),s(j))
      do l=lr,lx
        u  = l+1
        p2 = Pr(u,a(j),s(j))
        p(l,j) = p2-p1
        p1=p2
      enddo
    enddo
    do l=lr,lx; do j=jr,jx
      if(p(l,j) v1)goto 1
    Z0 = av + sd * x * Sqrt(-v2*Log(w)/w)
    if(Z0 > v0)return
    if(av > v0)goto 1
    print*,' A bad av was passed to Z0; av = ',av
    stop
    end function Z0

    function Pr(y,mu,sd)    ! rev 2001
    implicit none
    real(8) :: Pr,z,y,mu,sd,sdlm,abz,lmt=3.885,sum,fac,sign,r0=0.0,p5=0.5,r1=1.0
    real(8) :: xl(9)=(/.35366339,.70716781,1.06066017,1.41421357,1.76776695,2.12132034,2.47487373,2.82842712,3.18198052/)
    real(8) :: sqrt2=1.41421356237309504880168872420969807856967187537695
    real(8) :: sqrtpi=1.77245385090551590000000000 ! pi=3.14159265358979323846264338327950288419716939937511
    integer(4) :: h,i,k,m, n(9)=(/10,15,20,25,35,40,50,60,70/)
    z = y-mu; sdlm=sd*lmt
    if( z < -sdlm )then; Pr=r0
    elseif( z > sdlm )then; Pr=r1
    else; z=z/sd; abz=Abs(z)/sqrt2; h=1
      do while (abz.gt.xl(h)); h=h+1; enddo
      k=0; sum=0.0; sign=r1
      do i=1,n(h),2
        fac=1.; do m=1,k; fac=fac*m; enddo
        sum=sum+(sign*abz**i)/(i*fac); k=k+1; sign=sign*(-r1)
      enddo
      if(z > r0)then; Pr = p5 + sum/sqrtpi
      else;           Pr = p5 - sum/sqrtpi; endif
    endif
  end function Pr
!══════════════════════════════════════════════════════════════════════════════════════════════════════════════
```

```
!════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════
!
!                          Subroutine   SAR  --   Stochastic Ageing Routine.
!
!  Converts A Length Frequency To A Cohort Frequency Via Conditional Least Squares or Conditional Maximum Likelihood by
!                                         jx       lx
!  finding [a] in [p]*[a]=[l] such that  Σ a(j) = Σ l(i) & a(j) ≥ 0
!                                        j=0       l=lr
!  Encods 3 estimators of the numbers by cohort and either an age invariant cv[length], or a age constant σ²[length],
!  conditional on L(∞), K, birthday, and kill date.
!
!  Linear Least Squares conditional on L(∞), K, and cv[length] and applied to numbers at length was first proposed by
!  Bartoo & Parker (1983) following the origional idea presented by Clark (1981) as a maximum likelihood proceedure
!  using proportional distributions to eleminate bias in age key application.  Shephard (1985) suggested using non
!  linear least squares on transformed numbers at length. Unpublished software (ETTA 1986) implemented non-linear least
!  squares on transformed proportional distributions. Sar continues that effort by including the variance of length at
!  age in the list of estimable parameters, estimating cohort numbers (using the birthday and kill date as additional
!  data) rather than numbers at age, and by presenting weighted least squares/maximum likelihood estimates. Parrack,2000.
!
!
!    •••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
!    INPUT VARIABLES:
!        mh = a code for the estimator to be used.
!             1: Unweighted Least Squares using Numbers at length.
!             2: Unweighted Least Squares using proportions of Numbers at length.
!             3: Weighted Least Squares (ML if sampled WR) using Proportions of numbers At Length.
!                Weights are the large sample (n ≥ 25) variance of the proportion at length.
!                Since                        X ~ N(NP, NPQ), where X is the numbers at a length,
!                for large n (≥ 25)           p ~ N(P,PQ), where p = X/n and n is the sample size,
!                thus                         s²[p(l)] = p(l)•(1-p(l))/n, l=lr,lx for p(l) > 0.
!                Terms where X=0 or p(l)=0 are not included in WEIGHTED function evaluations (mh=3); they ARE included .
!                in unweighted function evaluations (mh ≤ 2)
!                Notes: (a) Methods 1 & 2 yield equal results if v=1 for method 1 and v≈5E5 for method 2
!                       (b) if v=1 method 2 is faster & yields better results.
!    •••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
!         as = L(∞).
!         gk = K, the Brody growth coefficient.
!         rd = reference day when growth began (birthday, recruitment day); a real number (Sept 15 = 0.707.).
!         rl = length on rd; the reference length (length at birth, length at recruitment).
!         dk = The date that [lf] occured expressed as a real number, i.e., April 15, 1963 = 1963.288
!         lr = minimum length in [lf].
!         lx = maximum length in [lf].
!         jx = max age; oldest in [af] -- your choice.
!         lf = numbers at length.
!         cv = coefficient of variation of length at age. If cv > 0. then the variance of length at age is
!              proportional to age so that the cv is age inviariant but the variance increases with age.
!         sd = standard deviation of length at age. If sd > 0. then the variance of length at age is
!              constant over age (age inviariant).
!    •••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
!    INPUT - OUTPUT VARIABLES:
!         af = Numbers by cohort from cohort Int[da] to Int[da]+jx;  af(0) = the frequency of cohort Int[dk],
!              af(1) cohort Int[dk]+1, cf(j) cohort Int[dk]+j etc.. [af] contains the starting point (1st guess)
!              when Sar is called and the solution upon return.
!    •••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
!    OUTPUT VARIABLES:
!         fs = the function evaluation at the solution, [af].
!•••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••

    subroutine Sar(mh,as,gk,rd,rl,dk,cv,sd,lr,lx,jx,lf,af,fs)
    implicit none
    real(8) :: snp=223D-306,  mac=2D-16, bnp=179D306
    real(8) :: p25=.25, p38=.38196602, p61=.61803399, r0=0.0, r1=1.0, r175=1.75
    real(8) :: as, gk, rd, rl, cv, sd, dk, fs,   tn, sm,   tol, f1,f2,v1,v2,v3,x0,x1,x2,x3
    real(8) :: lf(lr:lx), pl(lr:lx), v(lr:lx), p(lr:lx,0:jx), af(0:jx)
    real(8) :: s(jx+1), b(jx+1), t(jx+1), d(jx+1), e(jx+1), a(jx+1,jx+1), u(jx+1,jx+1), w(jx+1,jx+1), x(jx+1,jx+1)
    integer(4) :: mh,lr,lx,jx,    i,j,k,l,np
    tol=Sqrt(mac); np=jx+1
    tn=Sum(lf); x0=snp*tn; do j= 0,jx; af(j)=Max(x0,af(j)); enddo
```

```
  selectcase(mh)
    case(1); pl=lf; v=r1
    case(2); do j= 0,jx; af(j)=af(j)/tn; enddo;  do l=lr,lx; pl(l)=lf(l)/tn; enddo; v=r1
    case(3); do j= 0,jx; af(j)=af(j)/tn; enddo;  do l=lr,lx; pl(l)=lf(l)/tn; enddo; v=r0
             do l=lr,lx; if(pl(l) > r0)v(l)=r1/(pl(l)*(r1-pl(l))/tn); enddo
    casedefault; mh=0; return
  endselect
  do j=0,jx; s(j+1)=Log(af(j)); enddo
  if(cv > r0)then; v2=cv; else; v2=sd; endif  !••••••• line minimization •••••••••••••••••┐
  v1=v2*p25; v3=v2*r175                                                                   !
  x1=Sol(v1); x2=Sol(v2); x3=Sol(v3)                                                      !
1 if ( x2 > x1  )then; v3=v2; x3=x2; v2=v1; x2=x1; v1=v2*p25;  x1=Sol(v1); goto 1         !
  elseif(x2 > x3)then; v1=v2; x1=x2; v2=v3; x2=x3; v3=v2*r175; x3=Sol(v3); goto 1; endif  !
  x0=v1; x3=v3                                                                            !
  if(Abs(v3-v2) > Abs(v2-v1))then; x1=v2; x2=v2+p38*(v3-v2)                               !
  else; x2=v2; x1=v2-p38*(v2-v1); endif                                                   !
  f1=Sol(x1)                                                                              !
  f2=Sol(x2)                                                                              !
  do while ( Min( Abs(x0-x1), Abs(x2-x1), Abs(x3-x2) ) > tol )                            !
    if(f2.lt.f1)then; x0=x1; x1=x2; x2=p61*x1+p38*x3; f1=f2; f2=Sol(x2)                   !
    else; x3=x2; x2=x1; x1=p61*x2+p38*x0; f2=f1; f1=Sol(x1); endif                        !
  enddo                                                                                   !
  if(f1.lt.f2)then; v2=x1; else; v2=x2; endif  !••••••••••••••••••••••••••••••••••••••••••┘
  if( cv > r0)then; cv=v2; else; sd=v2; endif; fs=Sol(v2)
  do j=1,np; b(j)=Exp(b(j)); enddo; sm=Sum(b); do j=1,np; af(j-1)=Anint(tn*b(j)/sm); enddo
  !write(*,"(1x)")
  contains
    function Sol(vt)
      implicit none
      real(8) :: Sol,vt,ds,da,dt,ex,s2,f1,f2,ft,lamda,df,  xfl,  yh,dv,z,  p001=0.001, p5=.5, r14=14., huge=170D300
      p=r0; ds=vt; da=dk-Dint(dk)  !••••••• [p] •••••••┐
      do j=0,jx                                        !
        dt = da+j-rd                                   ! da= day killed
        ex = as - Exp(-gk*dt)*(as-rl)                  ! j = age index
        if(cv > r0)ds = vt*ex                          ! dt= □t if growth began on nearest rd
        s2 = lr;  f1 = Pr(s2,ex,ds)                    ! f1= integral -∞ to lr
        do l=lr,lx                                     ! f2= integral -∞ to l+1
          s2 = l+1                                     !  p= integral  l to l+1
          f2 = Pr(s2,ex,ds)                            !
          p(l,j) = f2-f1                               !
          f1=f2                                        !
        enddo                                          !
      enddo                                            !
      if(cv > r0)ds = r0  !••••••••••••••••••••••••••••┘
      b=s; Sol=r0  !•••••••••••••••••••••••••••••••••••••• Marquardt •••••••••••••••••••••••••••••••••••••••••••┐
      do l=lr,lx !••••••••••••••••• Óunc[b] ••••••••••••••••••••••┐                                             !
        xfl=r0; do j=0,jx; xfl = xfl + p(l,j)*Exp(b(j+1)); enddo  !                                             !
        Sol = Sol + v(l)*(pl(l)-xfl)**2                           !                                             !
      enddo  !••••••••••••••••••••••••••••••••••••••••••••••••••••┘                                             !
      lamda=p001                                                                                                !
    1 e=r0; a=r0                                                                                                !
      do l=lr,lx                                                                                                !
        yh=r0                                                                                                   !
        do j=1,np                                                                                               !
          d(j)=p(l,j-1)*Exp(b(j))                                                                               !
          yh=yh+d(j)                                                                                            !
        enddo                                                                                                   !
        dv=pl(l)-yh                                                                                             !
        do j=1,np                                                                                               !
          do k=1,j                                                                                              !
            a(j,k)=a(j,k)+d(j)*d(k)*v(l)                                                                        !
          enddo                                                                                                 !
          e(j)=e(j) + dv*d(j)*v(l)                                                                              !
        enddo                                                                                                   !
      enddo                                                                                                     !
      do i=2,np; do k=1,i-1; a(k,i)=a(i,k); enddo; enddo                                                        !
    2 u=a; do j=1,np; u(j,j)=a(j,j)+lamda; enddo  ! u = [diag of A] * (1+lamda) or lamda                         !
      w=u; x=r0; do i=1,np; x(i,i)=1.0; enddo                                                                   !
```

```
        do i=1,np                                                                                                 !
          z=w(i,i); do j=1,np; w(i,j)=w(i,j)/z; x(i,j)=x(i,j)/z; enddo                                            !
          do k=1,i-1;  z=w(k,i); do j=1,np; w(k,j)=w(k,j)-w(i,j)*z; x(k,j)=x(k,j)-x(i,j)*z; enddo; enddo          !
          do k=i+1,np; z=w(k,i); do j=1,np; w(k,j)=w(k,j)-w(i,j)*z; x(k,j)=x(k,j)-x(i,j)*z; enddo; enddo          !
        enddo                                                                                                     !
        d=r0; do i=1,np; do j=1,np; d(i)=d(i)+x(i,j)*e(j); enddo; enddo                                           !
        do j=1,np; t(j)=b(j)+d(j); enddo                                                                          !
        do j=1,np; if(t(j) > r14)exit; enddo !•••••• Óunc[t] •••••••••┐                                          !
        if(j > np)then                                                !                                           !
          ft=r0                                                       !                                           !
          do l=lr,lx                                                  !                                           !
            xfl=r0; do j=0,jx; xfl = xfl + p(l,j)*Exp(t(j+1)); enddo  !                                           !
            ft = ft + v(l)*(pl(l)-xfl)**2                             !                                           !
          enddo                                                       !                                           !
        else; ft=huge; endif !••••••••••••••••••••••••••••••••••••••••┘                                          !
        if(ft <= r0)then; b=t; Sol=ft; return; endif                                                              !
        df=Sol-ft                                                                                                 !
        if(df < r0)then; lamda=lamda*2.; if(lamda < bnp)goto 2;  !write(*,"($'o')")                               !
        else; b=t; Sol=ft; lamda=lamda*p5; if(df > tol)goto 1 ;  !write(*,"($'•')")                               !
        endif  !•••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••┘
      end function Sol
      function Pr(y,mu,sd)
        implicit none
        real(8) :: z,Pr,y,mu,sd,sdlm,abz,lmt=3.885,sum,fac,sign,r0=0.0,p5=0.5,r1=1.0
        real(8) :: xl(9)=(/.35366339,.70716781,1.06066017,1.41421357,1.76776695,2.12132034,2.47487373,2.82842712,3.18198052/)
        real(8) :: sqrt2=1.41421356237309504880168872420969807856967187537695
        real(8) :: sqrtpi=1.77245385090551590000000000 ! pi=3.14159265358979323846264338327950288419716939937511
        integer(4) :: h,i,k,m, n(9)=(/10,15,20,25,35,40,50,60,70/)
        z = y-mu; sdlm=sd*lmt
        if( z < -sdlm )then; Pr=r0
        elseif( z > sdlm )then; Pr=r1
        else; z=z/sd; abz=Abs(z)/sqrt2; h=1
          do while (abz.gt.xl(h)); h=h+1; enddo
          k=0; sum=0.0; sign=r1
          do i=1,n(h),2
            fac=1.; do m=1,k; fac=fac*m; enddo
            sum=sum+(sign*abz**i)/(i*fac); k=k+1; sign=sign*(-r1)
          enddo
          if(z > r0)then; Pr = p5 + sum/sqrtpi
          else;           Pr = p5 - sum/sqrtpi; endif
        endif
      end function Pr
  end subroutine Sar

!══════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════
!
!                                     subroutine:  Cutter
!
!  Converts a length frequency [lf] taken on date dk to a cohort frequency [af] by cutting the length frequency
!           via Von Berlanffy growth equation parameters.  (Revised from the 4/78 F77 source of Parrack).
!
!  input:  as  = L(∞).
!          gk  = K, the Brody growth coefficient.
!          rd  = The growth reference day when growth began; birth day; recruitment day.
!          rl  = The length on rd; the reference length; length at birth; length at recruitment.
!          dk  = The date that [lf] occured (date killed) expressed as a real number, i.e., April 15, 1963 = 1963.288
!          lr  = minimum length in [lf].
!          lx  = maximum length in [lf].
!          jx  = max age; oldest in [af].
!         [lf] = Numbers at length.
! output: [af] = Numbers by cohort from cohort Int[da] to Int[da]+jx. af(0) contains the frequency of
!                cohort Int[da], af(1) the frequency of cohort Int[da]+1, af(j) cohort Int[da]+j, etc..
!••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
    subroutine Cutter(as,gk,rd,rl,dk,lr,lx,jx,lf,af)
    implicit none
    real(8) :: as, gk, rd, rl, dk, lf(lr:lx), af(0:jx)
    integer(4) :: lr,lx,jx
    integer(4) :: h,iy,j,jf,js,l,lm
    real(8),allocatable :: f(:)
real(8) :: s, dt, cd, vx, vy, ymn, sx, sy, xx, xy, vn, z
real(8) :: d0=0.0, p1=0.1, p5=0.5, d1=1.0, huge=175D300
l = as-p1; lm = Min( l, lx )
allocate( f(lr:lm) ); f=d0
do l=lr,lm; f(l)=lf(l); enddo
do l=lm+1,lx;  f(lm)=f(lm)+lf(l); enddo
iy = dk; af=d0; jf=0; js=jx
do l=lr,lm  !••••••••••••••••• Cuter •••••••••••••••••••••••┐
  s  = l                                                    !
  dt = Log( (as-rl)/(as-s) ) / gk                           !
  cd = dk - dt ! computed reference date                    !
  h  = cd      ! cohort if cd near rd                       !
  vx = cd-h                                                 !
  if( Abs(vx-rd) > p5 )then                                 !
    if(rd > vx )then; h=h-1                                 !
    else; h=h+1; endif                                      !
  endif                                                     !
  j=iy-h                                                    !
  if(j < 0 )j=0                                             !
  if(j > jx)j=jx                                            !
  af(j) = af(j)+f(l)                                        !
  if( js < jx )cycle                                        !
  if( j <= jf+1 .or. l == lr )then; jf=j; cycle; endif      !
  js=jf                                                     !
enddo !•••••••••••••••••••••••••••••••••••••••••••••••••••••┘
deallocate(f) !••••••••••••••••••••••••• hole filler ••••••••••••••••••••••••••••••••••••••┐
s=d0; do j=js+1,jx; s=s+af(j); enddo; if(s <= d0)return ! s = Σ C after last good age      !
jf=0; do j=1,js; if( af(j) > af(jf) )jf=j; enddo  ! jf = full recruitment age              !
ymn=huge; do j=jf,js; if(af(j) > d0)ymn = Min(ymn,af(j)); enddo                            !
if(ymn < d1)then; ymn=d1-ymn; else; ymn=d0; endif                                          !
sx=d0; xx=d0; sy=d0; xy=d0; vn=d0                                                          !
do j=jf,js                                                                                 !
  if( af(j) <= d0 )cycle                                                                   !
  vy=Log(af(j)+ymn); vx=j; sx=sx+vx; sy=sy+vy; xx=xx+vx**2; xy=xy+vx*vy; vn=vn+d1          !
enddo                                                                                      !
z=(xy-sx*sy/vn)/(xx-sx*sx/vn); if(z > d0)z=-p5  ! full z from Log[Catch(i,j)] over j       !
allocate( f(js:jx) ); f=d0; sx=d0                                                          !
if(af(js) > d0)then; f(js)=af(js)                                                          !
else; f(js)=s; endif                                                                       !
do j=js+1,jx; f(j)=f(j-1)*Exp(z*(j-js)); sx=sx+f(j); enddo                                 !
do j=js+1,jx; af(j)=s*f(j)/sx; enddo   !•••••••••••••••••••••••••••••••••••••••••••••••••••┘
deallocate( f )
end subroutine Cutter
```