```
*************************************************************
*                                                           *
*   Calculates pure one loop correction to effective        *
*   potential with no special care of summation order.      *
*                                                           *
*                                                           *
*                    .true.      print to logcheck          *
*                  /                                        *
*   Logical lpr  --                                         *
*                  \                                        *
*                    .false.     no print to logcheck       *
*                                                           *
*   COMMON /ZIPER/    When we need some special sectors     *
*                                                           *
*   Fdmass  calculates field dependent squere mass          *
*           eigenvalues needed and puts them into           *
*           sqe [array] .                                   *
*                                                           *
*   vcf [array]  Appropriate coefficients according to      *
*                fdmass procedure.                          *
*   sf  [array]  Soft Parameters plus vevs at renormalization   *
*                scale.                                     *
*   Q            Renormalization scale.                     *
*   x1,x2        Independent variables.                     *
*                                                           *
*************************************************************
      SUBROUTINE dv1(iun,lpr,sf,x1,x2,ts,veff1)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      DIMENSION izip(48)
      DOUBLE PRECISION sf(nuu),vcf(48),sqe(48)
      LOGICAL lpr
      SAVE vcf
      COMMON /SCALE/ amgut,susy,amz,amw
      COMMON /NUEQ/ nuu
      DATA (vcf(jc),jc=1,48)
     # /3*-4.d0, 6*-12.d0, 6.d0, 3.d0, 8*-2.d0, 9*2.d0,
     # 12*6.d0, 8*1.d0/
      COMMON /wur/ wur,pi
      COMMON /ZIPER/ izip
CC
      call fdmass(sf,x1,x2,sqe)
      smax=fmax(sqe,12,48,48)
C
      ff1 = 0.d0
      do  i6=1,48
         if(izip(i6).ne.0)then
           psuvf = vcf(i6)*u(smax , sqe(i6) , ts )/(64.d0*pi*pi)
           ff1 = ff1+psuvf
C------------------------------------------------------------
C    Printing facilities
C
C    Print only the Logs of the "pure" 1-loop contribution
C------------------------------------------------------------
           Q=AMGUT*DEXP(-TS/2.D0)
           IF(LPR)
     #         write(iun,'(/,I3," MASS = ",d25.17,/,
   #    5x,"LOG = ",f23.17,/,5x,"PARTIAL SUM = ",D25.17)')
     #         I6,DSQRT(DABS(SQE(I6))),DLOG(DABS(SQE(I6))/Q**2),PSUVF
         endif
      enddo
C
      if(lpr)write(iun,'(/"TOTAL CF = ",f23.17)') ff1
```



```
      veff1=smax*smax*ff1
CC
      return
      END
```

```
C
C   Given the array arr of dimen. idim we find the largest entry
C   between the k - n places
C
      DOUBLE PRECISION FUNCTION fmax(arr,k,n,idim)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      DOUBLE PRECISION arr(idim)
      fmax=arr(k)
      do  ind=k+1,n
         if(arr(ind).gt.fmax)fmax=arr(ind)
      enddo
      return
      END
```

```
***************************************************************
*   Calculates the normalized value of the effective           *
*   potential for the TRUE minimum computed by                 *
*   itterations.                                               *
*                                                              *
*                                                              *
*   COMMON /SOFCOM/   :   Soft parameters and corrected        *
*                         vevs at Qrn.                         *
*   COMMON /RENTR/    :   Renormaliz. scale (for True min.)    *
*                                                              *
*   p(n)  :  Array of independent variables (x1,x2)            *
*                                                              *
*   No printing facilities in logcheck. (.false. everywhere)   *
*                                                              *
***************************************************************
       DOUBLE PRECISION FUNCTION ftr1(p)
       IMPLICIT DOUBLE PRECISION (a-h,o-z)
       PARAMETER (n=2,rscal=90000.d0,NU=39)
       DOUBLE PRECISION p(n),xcom(NU)
       COMMON /SOFCOM/ xcom
       COMMON /RENTR/ trn,qsl
C
       x1 = p(1)
       x2 = p(2)
         call vborn(98,.false.,xcom,x1,x2,veff0)
         call dv1(98,.false.,xcom,0.d0,0.d0,trn,wmeg)
         call dv1(98,.false.,xcom,x1,x2,trn,veff1)
C-------------------------------------------------------
C    Vacuum energy does not play any role for fixed Q
C    but for normalization reasons ...
C
C    Tree + one loop - vacEnergy
C-------------------------------------------------------
        v = veff0-wmeg+veff1
C-------------------------------------------------------
C   Rescaling so as to have as small values as possible
C-------------------------------------------------------
         ftr1 = v/rscal**4
C
      return
      END
```

```
*******************************************************************
*                                                                 *
*   COMMON /RENTR/   Ren. scale (for TRUE)                        *
*   COMMON /SOFCOM/  Soft parameters and vevs at renorm. scalee   *
*                                                                 *
*   rmar1,rmar2      Margins to independent variables x1,x2.      *
*                                                                 *
*******************************************************************
       DOUBLE PRECISION FUNCTION ftr2(p)
       IMPLICIT DOUBLE PRECISION (a-h,o-z)
       PARAMETER (n=2,rscal=90000.d0,NU=39)
       LOGICAL lpr
       DOUBLE PRECISION p(n),xcom(NU)
       SAVE rmar1,rmar2
       COMMON /SOFCOM/ xcom
       COMMON /RENTR/ trn,qsl
       COMMON /queue/ lpr
       DATA rmar1,rmar2 /4.d3, 4.d3/
C
       x1 = p(1)
       x2 = p(2)
C--------------------------------------------
C   Defining a 'Box' for the function Veff
C     with given margins.
C--------------------------------------------
       if(dabs(x1).lt.rmar1.and.dabs(x2).lt.rmar2)then
         call vborn(98,.false.,xcom,x1,x2,veff0)
         call dv1(98,.false.,xcom,0.d0,0.d0,trn,wmeg)
C----------------------------------------------------
C   Enable printing only for pure one loop
C   contribution. This is decided inside samebsa.f
C   and transfered here through COMMON /queue/.
C----------------------------------------------------
         call dv1(98,lpr,xcom,x1,x2,trn,veff1)
C-----------------------------------------------------
C    Vacuum energy does not play any role for fixed Q
C    but for normalization reasons ...
C-----------------------------------------------------
         v = veff0-wmeg+veff1
C-----------------------------------------------------
C   Rescaling so as to have as small values as possible
C-----------------------------------------------------
         ftr2 = v/rscal**4
C-----------------------------
C   Extra printing facilities
C-----------------------------
         if(lpr)
     #    write(97,'(/"Vacen = ",d25.17,
  #  /,"Pure tree = ",d25.17,/,"Tree - Vacen = ",d25.17,/,
  #  "1-loop = ",d25.17,/,"v0/v1 = ",f10.5,/,"[v0-w]/v1 = ",
  #  f10.5,/,"VFULL = ",d25.17/)')
     #    wmeg,veff0,veff0-wmeg,veff1,veff0/veff1,
     #    (veff0-wmeg)/veff1,v
      else
         ftr2 = 1.d200
      endif
      return
      END
```

```
*==> trnf.f <==
**********************************************************
*   Supplies vborn module with some necessary constants  *
*   from the array of soft parameters (a)                *
*                                                        *
*   COMMON /WUR/  :  Transfers pi from evalsoft          *
*                                                        *
**********************************************************
      SUBROUTINE trnf(a,g,g2,sqm1,sqm2,sqm3)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      DOUBLE PRECISION a(nuu)
      COMMON /wur/ wur,pi
      COMMON /NUEQ/ nuu
C
      y1 = a(1)
      y2 = a(2)
      g = 4.d0*pi*dsqrt(3.d0*y1/5.d0)
      g2 = 4.d0*pi*dsqrt(y2)
      sqm1 = a(30)
      sqm2 = a(31)
      sqm3 = a(11)
C
      return
      END

*==> vborn.f <==
*******************************************************
*   Computes tree level neutral effective potential   *
*   given the soft parameter array sf(33)             *
*                                                     *
*   Logical lpr is for printing in logcheck.          *
*                                                     *
*   x1,x2  Independent variables                      *
*******************************************************
      SUBROUTINE vborn(iu,lpr,sf,x1,x2,vb)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      DOUBLE PRECISION sf(nuu)
      LOGICAL lpr
      COMMON /NUEQ/ nuu
C
      call trnf(sf,g,g2,sqm1,sqm2,sqm3)
C----------------------------------
C    Tree level effective potential
C----------------------------------
      vb = sqm1*x1*x1 + sqm2*x2*x2 + 2.d0*sqm3*x1*x2
     # + (g*g + g2*g2)*dfx(x1,x2)*dfx(x1,x2) / 8.d0
C
      if(lpr)write(iu,'(3(f15.6/),2(f13.9/),//)')
     #   sqm1,sqm2,sqm3,g**2,g2**2
C
      return
      END
```

```
******************************************************************
*                                                                *
*   Logarithmic part of 1-loop corrections to Veff.              *
*   Some squares of field dependent masses (sqe) can be          *
*   negative or zero.                                            *
*                                                                *
*                                                                *
*          0                                     s = 0           *
*        /                                                       *
*   u =                                                          *
*        \                                                       *
*          s*s/(smax*smax) * {dlog(|s|/Q*Q) - 1.5 }    s =! 0    *
*                                                                *
*                                                                *
*                                                                *
*                                                                *
*                                                                *
*   s  :  Some general field dependent squere mass eigenvalue.   *
*   Q  :  Renormalization scale.                                 *
*                                                                *
******************************************************************
       DOUBLE PRECISION FUNCTION u(smax,s,ts)
       IMPLICIT DOUBLE PRECISION (a-h,o-z)
       COMMON /SCALE/ amgut,susy,amz,amw
C
       if(s.eq.0.d0)then
       u = 0.d0
       return
       endif
C      if(s.lt.0.d0)print*,'Negative sqe   ',s
       r1=s*s/(smax*smax)
       a1=dsqrt(dabs(s))
       a2=ts/2.d0 + dlog(a1) - dlog(amgut)
       u = r1/2.d0*(4.d0*a2-3.d0)
C
       return
       END
```

```
      SUBROUTINE xamebsa(kun,temptr,iiter,tfunk,stval,pb,yb)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      INTEGER NP,MP,MPNP
      INTEGER i,idum,iiter,j,jiter,ndim,nit
      DOUBLE PRECISION FTOL
      DOUBLE PRECISION temptr,tt,yb,ybb
      PARAMETER(NP=2,MP=3,MPNP=6,FTOL=1.0D-14)
      DOUBLE PRECISION p(MP,NP),x(NP),y(MP),xoff(NP),pb(NP),tfunk
      COMMON /ambsa/ tt,idum
      EXTERNAL tfunk
C-----------------------------------------------------
C  Each time control enters xamebsa initialize xoff,p
C-----------------------------------------------------
      do   jbv=1,NP
          xoff(jbv)=stval  ! *** Initial values near TRUE min. ***
      enddo
      do   kj1=1,MP
          do   kj2=1,NP
              p(kj1,kj2)=0.d0
          enddo
      enddo
C
      idum=-64
      ndim=NP
      do 11 j=2,MP
           p(j,j-1)=1.D0
  11  continue
      do 13 i=1,MP
           do 12 j=1,NP
                p(i,j)=p(i,j)+xoff(j)
                x(j)=p(i,j)
  12       continue
           y(i)=tfunk(x)
  13  continue
      yb=1.d30
C
      ybb=1.d30
      nit=0
      do 14 jiter=1,100
           iter=iiter
           temptr=temptr*0.8d0
           call amebsa(p,y,MP,NP,ndim,pb,yb,FTOL,tfunk,iter,temptr)
           nit=nit+iiter-iter
           if(yb.lt.ybb)then
              ybb=yb
              write(kun,'(i4,3x,d9.2,3x,2(d14.6,3x),d14.6)')
     #           nit,temptr,(pb(j),j=1,NP),yb
           endif
           if(iter.gt.0)goto 80
  14  continue
  80  write(kun,'(/1x,a)')'Vertices of final 3-D simplex and'
      write(kun,'(1x,a)')'function values at the vertices:'
      do 15 i=1,MP
           write(kun,'(i2,3x,2(d14.6,3x),d14.6)')
     #        i,(p(i,j),j=1,NP),y(i)
  15  continue
C
      write(kun,'(/,/,6X,"Best",3X,"Values",/)')
      write(kun,'("  tanb = ",f13.8)')pb(2)/pb(1)
C---------------------------------
C  Automatic format for comparison
C---------------------------------
```



```
      write(kun,*)
      write(kun,*)' v1 = ',pb(1)
      write(kun,*)' v2 = ',pb(2)
      write(kun,*)' Veff = ',yb
      write(kun,*)
      return
      END
```