```
      subroutine cdbonn
c
c     alias  "CDBonn2000"
c
c*****************************************************************
c
c     FINAL VERSION: JANUARY 1, 2000
c     (FORTRAN improved by W. Schadow 7/23/00)
c
c     this potential is published in:
c     R. Machleidt, PRC 63, 024001 (2001).
c
c*****************************************************************
c
c     This code computes the
c     Charge-Dependent Bonn NN-Potential (`CDBonn2000'),
c     ------------------------------------------------
c     in momentum space.
c
c     This CD-Bonn potential includes CSB and CIB effects derived
c     from irreducible 2pi and pi-rho exchanges with nucleon
c     and delta(1232) intermediate states. CIB also includes
c     the effects of irreducible pi-gamma exchange.
c     Besides this, the usual CIB due to OPE is included.
c     Note that the latter is contained in any modern high-precision
c     potential, while the former is not.
c
c*****************************************************************
c
c
c     this package is self-contained and includes
c     all subroutines needed.
c     only cdbonn needs to be called by the user.
c     all codes are consistently in double precision.
c     when working on an UNIX system, it is crucial
c     to compile this code with the  -static  option.
c
c     NOTE: as compared to the earlier version of cdbonn, there is
c     a minor practical change in the code: this code does not
c     read-in anything. the type of potential to be used
c     (pp, np, or nn) is now controlled by a new common block,
c
c     common /cnn/ inn  ,
c     where
c     inn=1  means pp potential,
c     inn=2  means np potential, and
c     inn=3  means nn potential.
c
c     the user needs to include this common block in his/her code,
c     and specify which potential he/she wants. inn can be
c     changed at any time, i.e., all three potentials can be
c     used in one run, which is the advantage of this new regulation.
c
c
c*****************************************************************
c
c
c     author:      Ruprecht Machleidt
c     department of physics
c     university of idaho
c     moscow, idaho 83844
c     u. s. a.
c     e-mail: machleid@uidaho.edu
c
c     formerly:
c     institut fuer theoretische kernphysik der
```

```
c     universitaet bonn
c     nussallee 14-16
c     d - 5300  bonn, w. germany
c
c*********************************************************************
c
c     this version of the code uses the Legendre functions
c     ----------------------------------------------------
c     of the second kind for the partial wave decomposition
c     ----------------------------------------------------
c     and includes the meson-parameters in data statements.
c     ----------------------------------------------------
c
c
c     an earlier version of this code has been published in
c     "computational nuclear physics 2 -- nuclear reactions",
c     K. Langanke, J.A. Maruhn, and S.E. Koonin, eds.
c     (Springer, New York, 1993), Chapter 1, pp. 1-29.
c
c     This code is a slight modification of the earlier, published
c     version. However, the mathematical formulae, as well as the
c     general organization of this code is the same as described in
c     the above-referenced book-chapter.
c     in this version of the code, the integrals, Eqs. (1.68) of the
c     above reference, are solved analytically by means of the
c     Legendre functions of the second kind, see Eqs. (E.44) of
c     R. Machleidt et al., Phys. Rep. 149, 1 (1987).
c
c     Still, the above-referenced article may serve as a good
c     introduction into this code.
c
c*********************************************************************
c
c
      implicit real*8 (a-h,o-z)
c
c
      common /crdwrt/ kread,kwrite,kpunch,kda(9)
c
c     arguments and values of this subroutine:
c
      common /cpot/   v(6),xmev,ymev
      common /cstate/ j,heform,sing,trip,coup,endep,label
      common /cnn/ inn
c
c
c     this has been the end of the common-blocks containing
c     the arguments and values of this subroutine
c
c     specifications for these two common blocks
c
      logical heform,sing,trip,coup,endep
c
c     THE ABOVE FOUR COMMON BLOCKS IS ALL THE USER NEEDS
c     TO BE FAMILIAR WITH.
c
c
c     xmev and ymev are the final and initial relative momenta,
c     respectively, in units of mev/c.
c     v is the potential in units of mev**(-2).
c     concerning units and factor of pi etc.,
c     cf. with the partial-wave Lippmann-Schwinger equation, Eq. (1.32),
c     and with the phase shift relation, Eq. (1.41) of
c     R. Machleidt, in: Computational Nuclear Physics 2
c     -- Nuclear Reactions, Langanke et al., eds.
c     (Springer, New York, 1993), Chapter 1, pp. 1-29.
```

```
c
c     the partial-wave Lippmann-Schwinger equation for the
c     K-matrix reads:
c
c     K(q',q) = V(q',q) + M P \int dk k^2 V(q',k) K(k,q)/(q^2-k^2)
c
c     with M the nucleon mass in MeV and P denoting the principal value;
c     V(q',q) as provided by this code in common block /cpot/;
c     all momenta in MeV.
c
c     the phase-shift relation is:
c
c     tan \delta_L = -(pi/2) M q K_L(q,q)
c
c     with M and q in units of MeV, K_L in MeV**(-2) like V.
c
c
c     if heform=.true., v contains the 6 matrix elements
c     associated with one j in the helicity formalism
c     in the following order:
c     0v, 1v, 12v, 34v, 55v, 66v (for notation see above article).
c     if heform=.false., v contains the 6 matrix elements
c     associated with one j in the lsj formalism
c     in the following order:
c     0v(singlet), 1v(uncoupled triplet), v++, v--, v+-, v-+ (coupled)
c     (see above article for notation).
c     j is the total angular momentum. there is essentially no upper
c     limit for j.
c     sing, trip, and coup should in general be .true..
c     endep and label can be ignored.
c     it is customary, to set kread=5 and kwrite=6;
c     ignore kpunch and kda(9).
c     inn=1  means pp potential,
c     inn=2  means np potential, and
c     inn=3  means nn potential.
c
c
c     THIS IS ESSENTIALLY ALL THE USER NEEDS TO KNOW.
c
c***********************************************************************
c
c
c     common block for all ob-subroutines
c
      common /cobq/   vj(32,50),c(20,50),fff,ff,f(52),aa(96),ai(19,15),
     1     wnn(3),wdd(3),x,xx,y,yy,xy2,xxpyy,ex,ey,eem12,
     2     ez1,ez2,ct(96),wt(96),
     3     ic(20,50),ift(3),mint(3),maxt(3),nt,
     4     mge,mgg(12,3),mggo(12,3),ima(15,12,3),
     5     imaa(3),imea(3),ime,im,mc,m,mg,inter,ide,idde,
     6     indc(10,50),indpar(3),indxy
c
c     specifications for this common block
c
      logical indc,indxy,indpar
c
c
c     further specifications
c

      character*4 mesong(12)
      logical index

      dimension vl(4),adminv(4,4),ldminv(4),mdminv(4)

      data mesong/'0-  ','0-t ','0-st','0+  ','0+st',
```

```
     1     '1-  ','1-t ','1-tt','1-st','1-ss',
     2     '1+  ','2+  '/
      data pi/3.141592653589793d0/
      data index/.false./
      data jj/-1/
      data innn/-1/
      save
c
c
c
c
      inter=1
c
c
      if (inn.lt.1.or.inn.gt.3) then
         write (kwrite,19001) inn
19001    format (////' error in cdbonn: potential type  inn =',i3,
     1        '  unknown.'/' execution terminated.'////)
         stop
      endif
      if (j.lt.0) then
         write (kwrite,19002)
19002    format (////' error in cdbonn: total angular momentum j',
     1        ' is negative.'/' execution terminated.'////)
         stop
      endif
c
c
c
c
c     call obparq whenever j or inn has changed
c
c
      if (j.eq.jj.and.inn.eq.innn) go to 50
      innn=inn
c
c
c
      call obparq
c     -----------
c     -----------
c
c
      dwn=1.d0/wnn(inter)
c
c     prepare constant over-all factor
c
      fac=1.d0/(2.d0*pi)*dwn*dwn
c     --------------------------
c
c
      if (index) go to 30
      index=.true.
c
      iftgo=ift(inter)+1
      iman=imaa(inter)
      imen=imea(inter)
c
      imanm1=iman-1
c
      iman1=imanm1+1
      iman2=imanm1+2
      iman3=imanm1+3
      iman4=imanm1+4
      iman5=imanm1+5
      iman6=imanm1+6
```

```
      iman7=imanm1+7
      iman8=imanm1+8
      iman9=imanm1+9
      iman10=imanm1+10
      iman11=imanm1+11
      iman12=imanm1+12
      iman13=imanm1+13
      iman14=imanm1+14
      iman15=imanm1+15
      iman16=imanm1+16
      iman17=imanm1+17
      iman18=imanm1+18
c
c
c
c
 30   if (j.eq.jj) go to 50
      jj=j
      if (j.eq.0) go to 50
      aj=dble(j)
      aj1=dble(j+1)
      a2j1=dble(2*j+1)
      aaj6=dsqrt(aj*aj1)
c
c     coefficient matrix for the translations into lsj formalism
c
      adminv(1,1)=aj1
      adminv(1,2)=aj
      adminv(1,3)=-aaj6
      adminv(1,4)=-aaj6
      adminv(2,1)=aj
      adminv(2,2)=aj1
      adminv(2,3)=aaj6
      adminv(2,4)=aaj6
      adminv(3,1)=aaj6
      adminv(3,2)=-aaj6
      adminv(3,3)=aj1
      adminv(3,4)=-aj
      adminv(4,1)=aaj6
      adminv(4,2)=-aaj6
      adminv(4,3)=-aj
      adminv(4,4)=aj1
c
c     inversion
c
      call dminv (adminv,4,deter,ldminv,mdminv)
c
c
c
c
c
c
c
c     prepare expressions depending on x and y
c     ----------------------------------------
c     ----------------------------------------
c
c
c
c
 50   x=xmev*dwn
      y=ymev*dwn
      indxy=.false.
c
c
      if (xmev.lt.0.d0) then
```

```
         write (kwrite,19003)
19003    format (////' error in cdbonn: momentum xmev',
     1        ' is negative.'/' execution terminated.'////)
         stop
       endif
       if (ymev.lt.0.d0) then
         write (kwrite,19004)
19004    format (////' error in cdbonn: momentum ymev',
     1        ' is negative.'/' execution terminated.'////)
         stop
       endif
c
c
       xx=x*x
       yy=y*y
       xy2=x*y*2.d0
       xxpyy=xx+yy
       ex=dsqrt(1.d0+xx)
       ey=dsqrt(1.d0+yy)
       eem12=(ex*ey-1.d0)*2.d0
c
c
c
c
       xy=xy2*0.5d0
       ee=ex*ey
       ree=dsqrt(ee)
       eem1=ee-1.d0
       eme=ex-ey
       emeh=eme*0.5d0
       emehq=emeh*emeh
       eep1=ee+1.d0
       epe=ex+ey
       xxyy=xx*yy
c
c
c
c
c      prepare over-all factor
c
c
       go to (70,71,72),iftgo
c
c      no additional factor
c
 70    fff=fac
       go to 90
c
c      minimal relativity
c
 71    fff=fac/ree
       go to 90
c
c      factor m/e*m/e
c
 72    fff=fac/ee
c
c
c
c
c
c
 90    do 93 iv=1,6
 93       v(iv)=0.d0
          do 95 il=1,50
             do 95 iv=1,6
```

```
 95            vj(iv,il)=0.d0
c
c
c
c
c     contributions of mesons
c     -----------------------
c     -----------------------
c
c
c
c
               do 1995 img=1,mge
                  mg=mggo(img,inter)
                  if (mg.eq.0) go to 2000
                  if (mg.gt.7) go to 9000
                  me=mgg(mg,inter)
                  go to (100,9000,9000,400,9000,9000,700),mg
c
c
c
c     0-   , pseudo-scalar coupling
c     ----------------------------
c
c
c
c
 100              mc=1
c
                  ff=1.d0
                  f(1)=eem1
                  f(2)=-xy
                  f(3)=-f(1)
                  f(4)=-f(2)
                  f(5)=f(2)
                  f(6)=f(1)
                  f(7)=-eme
                  f(8)=-f(7)
c
                  call obstrq(1,1,me)
                  go to 1995
c
c
c
c
c     0+   , scalar coupling
c     ----------------------
c
c
c
c
 400              mc=1
c
                  ff=1.d0
                  f(1)=-eep1
                  f(2)=xy
                  f(3)=f(1)
                  f(4)=f(2)
                  f(5)=f(2)
                  f(6)=f(1)
                  f(7)=epe
                  f(8)=f(7)
c
                  call obstrq(1,1,me)
                  go to 1995
c
```

```
c
c
c
c     1-t , vector mesons
c     -------------------
c
c
c
c
c     vector-vector coupling
c
c
c
c
 700              mc=1
c
                  ff=2.d0
                  f(1)=eem1+ee
                  f(2)=0.d0
                  f(3)=ee
                  f(4)=xy
                  f(5)=xy2
                  f(6)=1.d0
                  f(7)=-ey
                  f(8)=-ex
c
                  call obstrq(1,1,me)
c
c
c
c
c     tensor-tensor coupling
c
c
c
c
                  mc=2
c
                  ff=0.25d0
                  f(1)=(3.d0*ee+1.d0)*xxpyy
                  f(2)=-(6.d0*ee+2.d0-xxpyy)*xy
                  f(3)=eem1*xxpyy+4.d0*xxyy
                  f(4)=-(4.d0*ee+xxpyy)*xy
                  f(5)=(4.d0-3.d0*xxpyy)*xy
                  f(6)=6.d0*xxyy-(ee+3.d0)*xxpyy
                  f(7)=(ex+3.d0*ey)*xx+eme*yy
                  f(8)=(ey+3.d0*ex)*yy-eme*xx
c     factors for additional terms
                  f(9)=-2.d0*xxyy
                  f(10)=eep1*xy2
                  f(11)=-epe*xy2
c
                  call obstrq(2,1,me)
c
c
c
c
c     vector-tensor coupling
c
c
c
c
                  mc=3
c
                  ff=1.d0
                  f(1)=xxpyy
```

```
                  f(2)=-xy2
                  f(3)=-f(1)
                  f(4)=-f(2)
                  f(5)=6.d0*xy
                  f(6)=3.d0*f(3)
                  f(7)=(ex*yy+3.d0*ey*xx)
                  f(8)=(ey*xx+3.d0*ex*yy)
c
                  call obstrq(1,1,me)
                  go to 1995
c
c
c
c
c     this has been the end of the contributions of mesons
c     ---------------------------------------------------
c
c
c
c
c
c     error exit
c     ----------
c
c
c
c
 9000             write (kwrite,19000) mesong(mg)
19000             format(////' error in cdbonn:   meson-group   ',a4,'  does not
exi
     1                 st in this program.'/' execution terminated.'
     2                 ////)
                  stop
c
c
c
c
 1995          continue
```

```
c
c
c
c
c     add up contributions of mesons
c     ------------------------------
c
c
c
c
 2000          continue

               do 2005 iv=1,6
 2005             v(iv)=vj(iv,iman1)+vj(iv,iman3)
c
c
                  if (j.eq.1) then
                     v(1)=vj(1,iman1)+vj(1,iman4)
                  end if

c
c
                  if (j.eq.2) then
                     do 2007 iv=3,6
 2007                   v(iv)=vj(iv,iman2)+vj(iv,iman3)
                     end if
c
c
                     if (mod(j,2).eq.1) go to 2020
c
c
c     j even
c     ------
c
                     v(1)=v(1)+vj(1,iman5)+vj(1,iman6)
                     v(2)=v(2)+vj(2,iman9)+vj(2,iman10)
c
                     if (j.eq.2) then
c
c     the pions for 3P2-3F2
                        do 2014 iv=3,6
 2014                      v(iv)=v(iv)+vj(iv,iman7)+vj(iv,iman8)
                        else
c
c     the pions in all other T=1 coupled cases
                           do 2015 iv=3,6
 2015                         v(iv)=v(iv)+vj(iv,iman5)+vj(iv,iman6)
                           end if
                           go to 2030
c
c
c     j odd
c     -----
c
 2020                      v(1)=v(1)+vj(1,iman9)+vj(1,iman10)
                           v(2)=v(2)+vj(2,iman5)+vj(2,iman6)
c
                           do 2025 iv=3,6
 2025                         v(iv)=v(iv)+vj(iv,iman9)+vj(iv,iman10)
c
c
c     for all j
c     _________
c
 2030                      v(1)=v(1)+vj(1,iman11)+vj(1,iman12)
                           v(2)=v(2)+vj(2,iman13)+vj(2,iman14)
```

```
                                v(3)=v(3)+vj(3,iman15)+vj(3,iman16)
                                v(4)=v(4)+vj(4,iman17)+vj(4,iman18)

                                do 2035 iv=5,6
 2035                              v(iv)=v(iv)+vj(iv,iman17)+vj(iv,iman18)
c
c
c
c
                                  if (j.eq.0.or..not.heform) go to 4000
c
c
c     translation into (combinations of) helicity states
c
c
                                  do 3005 i=1,4
 3005                             vl(i)=v(i+2)
c
                                    do 3020 ii=1,4
                                       iii=ii+2
                                       v(iii)=0.d0
c
                                       do 3015 i=1,4
 3015                 v(iii)=v(iii)+adminv(ii,i)*vl(i)
 3020                v(iii)=v(iii)*a2j1
c
c
c
c
 4000                                    return
                                         end
```

```
      subroutine obparq
c
c         obparq provides the parameters for the
c         charge-dependent Bonn potential.
c
c
       implicit real*8 (a-h,o-z)
c
c
       common /crdwrt/ kread,kwrite,kpunch,kda(9)
c
       common /cstate/ j,heform,sing,trip,coup,endep,label
       logical heform,sing,trip,coup,endep
c
c
c         common block for all ob-subroutines
c
       common /cobq/   vj(32,50),c(20,50),fff,ff,f(52),aa(96),ai(19,15),
     1                 wnn(3),wdd(3),x,xx,y,yy,xy2,xxpyy,ex,ey,eem12,
     2                 ez1,ez2,ct(96),wt(96),
     3                 ic(20,50),ift(3),mint(3),maxt(3),nt,
     4                 mge,mgg(12,3),mggo(12,3),ima(15,12,3),
     5                 imaa(3),imea(3),ime,im,mc,m,mg,inter,ide,idde,
     6                 indc(10,50),indpar(3),indxy
c
c          specifications for this common block
c
       logical indc,indxy,indpar
c
c
       common /cnn/ inn
c
c
c         further specifications
c
```

```
       dimension cc(5)
       character*4 name(4),nname(15)
       dimension wscale(3)
       integer imga(3)

       character*4 nucnuc(3)
c       integer nucnuc(3)

       character*4 cut,end
c       integer cut,end

       character*4 two
c       integer two

       logical index

       character*4 mesong(12)
c       integer mesong(12)

       character*4 ntab1(4,10)
c       integer ntab1(4,10)

       character*4 ntab2(4)
```

```
c      integer ntab2(4)

       dimension tab1(5,10,3)
       dimension tab2(5,4,7,3)

       data cut/'cut '/,end/'end '/
       data two/'2   '/
       data mesong/'0-  ','0-t ','0-st','0+  ','0+st',
      1            '1-  ','1-t ','1-tt','1-st','1-ss',
      2            '1+  ','2+  '/
       data nucnuc/'BNpp','BNnp','BNnn'/
       data index/.false./
       data innn/-1/
c
c
c
c
c        parameter tables
c        ----------------
c        ----------------
c
c
c        identification labels
c        ---------------------

       data ntab1/
      1 '1-t ',' ','rho','    ',
      2 'cut ',' ','    ','    ',
      3 '1-t ',' ','rho','    ',
      4 'cut ',' ','    ','    ',
      5 '1-t ',' ','ome','ga  ',
      6 'cut ',' ','    ','    ',
      7 '1-t ',' ','ome','ga  ',
      8 '0-  ','2','pio','ns  ',
      9 '0-  ','2','pio','ns  ',
      * '0-  ','2','pio','ns  '/
c

       data ntab2/
      1 '0+  ','2','sig','mas '/
c
c
c        global parameters
c        -----------------
c        -----------------

       data tab1/
c
c proton-proton potential
c ------------------------
      1        0.84    , 6.1       , 769.9     , 1.        , 0.,
      2        2.      , 0.        , 2.        , 1310.     , 0.1,
      3        0.84    , 6.1       , 769.9     , 1.        , 0.,
      4        2.      , 0.        , 2.        , 1310.     , 0.,
      5       20.0     , 0.        , 781.94    , 0.        , 0.,
      6        2.      , 0.        , 2.        , 1500.     , 0.,
      7       20.0     , 0.        , 781.94    , 0.        , 0.,
c t=1:
      8       13.6     , 134.9764  , 0.0       , 139.56995 , 1720.,
      9       13.6     , 134.9764  , 0.0       , 139.56995 , 3000.,
c t=0:
      *        0.0     , 134.9764  , 0.0       , 139.56995 , 1720.,
c
c neutron-proton potential
c ------------------------
      1        0.84    , 6.1       , 769.9     , 1.        , 0.,
```

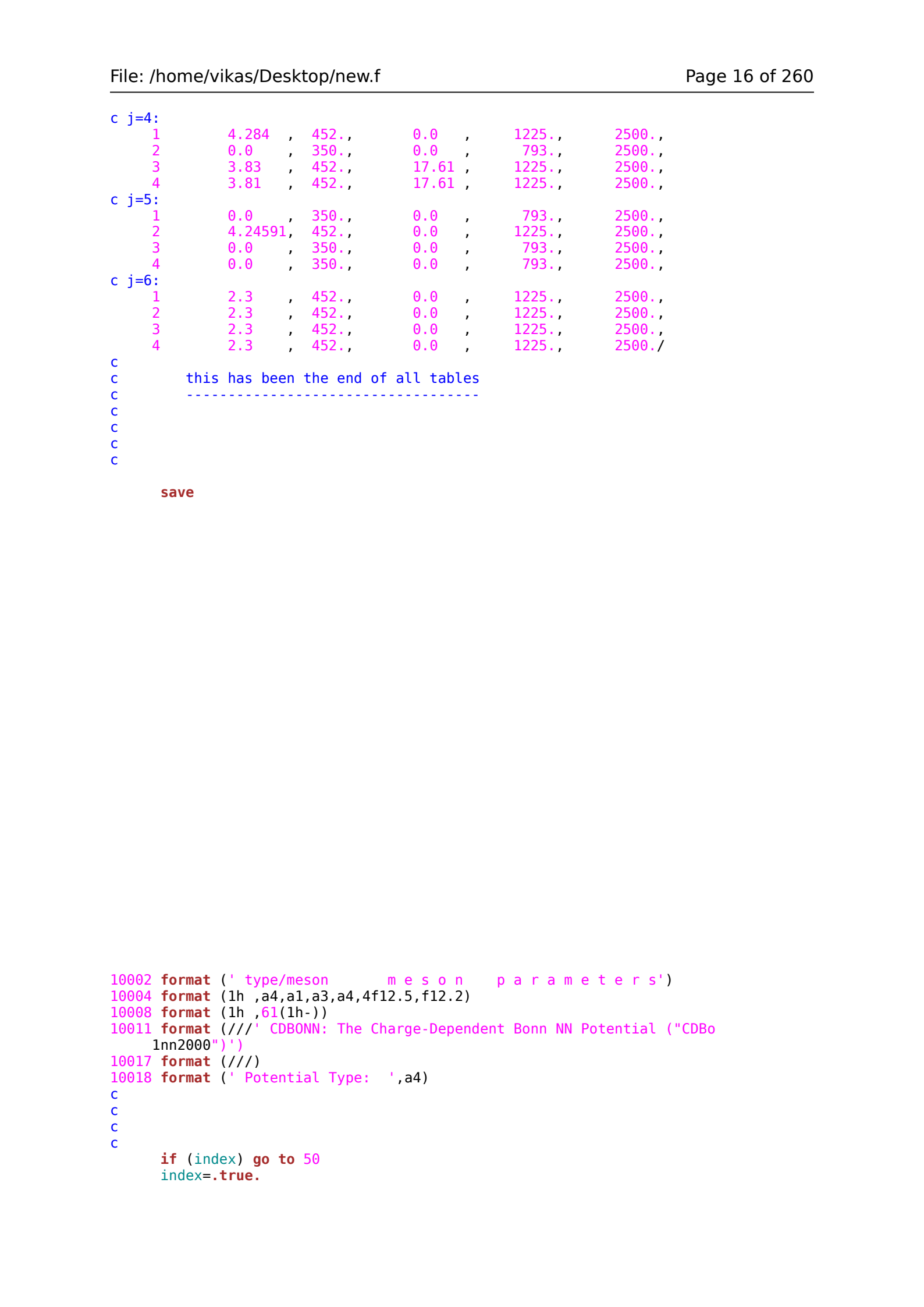
```
     2        2.       , 0.        , 2.        , 1310.     , 0.1,
     3        0.862    , 6.1       , 769.9     , 1.        , 0.,
     4        2.       , 0.        , 2.        , 1310.     , 0.,
     5       20.0      , 0.        , 781.94    , 0.        , 0.,
     6        2.       , 0.        , 2.        , 1500.     , 0.,
     7       20.0      , 0.        , 781.94    , 0.        , 0.,
c t=1:
     8      -13.6      , 134.9764  , 27.2      , 139.56995 , 1720.,
     9      -13.6      , 134.9764  , 27.2      , 139.56995 , 3000.,
c t=0:
     *      -13.6      , 134.9764  , -27.2     , 139.56995 , 1720.,
c
c neutron-neutron potential
c -------------------------
     1        0.84     , 6.1       , 769.9     , 1.        , 0.,
     2        2.       , 0.        , 2.        , 1310.     , 0.1,
     3        0.844    , 6.1       , 769.9     , 1.        , 0.,
     4        2.       , 0.        , 2.        , 1310.     , 0.,
     5       20.0      , 0.        , 781.94    , 0.        , 0.,
     6        2.       , 0.        , 2.        , 1500.     , 0.,
     7       20.0      , 0.        , 781.94    , 0.        , 0.,
c t=1:
     8       13.6      , 134.9764  , 0.0       , 139.56995 , 1720.,
     9       13.6      , 134.9764  , 0.0       , 139.56995 , 3000.,
c t=0:
     *        0.0      , 134.9764  , 0.0       , 139.56995 , 1720./
c
c
c         partial-wave dependent parameters
c         --------------------------------
c         --------------------------------

      data tab2/
c
c proton-proton potential
c -----------------------
c j=0:
     1        4.24591,  452.,       17.61,      1225.,      2500.,
     2        0.0    ,  350.,       0.0  ,       793.,      2500.,
     3        7.866  ,  560.,       17.61,      1225.,      2500.,
     4        0.0    ,  452.,       0.0  ,      1225.,      2500.,
c j=1:
     1        0.0    ,  350.,       0.0  ,      1225.,      2500.,
     2        2.303  ,  424.,       17.61,      1225.,      2500.,
     3        0.0    ,  350.,       0.0  ,       793.,      2500.,
     4        0.0    ,  350.,       0.0  ,       793.,      2500.,
c j=2:
     1        2.225  ,  400.,       190.7,      1225.,      2500.,
     2        0.0    ,  350.,       0.0  ,      1225.,      2500.,
     3        1.5    ,  452.,       56.21,       793.,      2500.,
     4        4.166  ,  470.,       24.80,      1225.,      2500.,
c j=3:
     1        0.0    ,  350.,       0.0  ,       793.,      2500.,
     2        1.5    ,  452.,       74.44,       793.,      2500.,
     3        0 0    ,  350.,       0.0  ,       793.,      2500.,
     4        0.0    ,  452.,       0.0  ,       793.,      2500.,
c j=4:
     1        4.24591,  452.,       0.0  ,      1225.,      2500.,
     2        0.0    ,  350.,       0.0  ,       793.,      2500.,
     3        3.8    ,  452.,       17.61,      1225.,      2500.,
     4        3.8    ,  452.,       17.61,      1225.,      2500.,
c j=5:
     1        0.0    ,  350.,       0.0  ,       793.,      2500.,
     2        4.24591,  452.,       0.0  ,      1225.,      2500.,
     3        0.0    ,  350.,       0.0  ,       793.,      2500.,
     4        0.0    ,  350.,       0.0  ,       793.,      2500.,
```

```
c j=6:
     1        2.3    ,  452.,       0.0  ,      1225.,      2500.,
     2        2.3    ,  452.,       0.0  ,      1225.,      2500.,
     3        2.3    ,  452.,       0.0  ,      1225.,      2500.,
     4        2.3    ,  452.,       0.0  ,      1225.,      2500.,
c
c neutron-proton potential
c -------------------------
c j=0:
     1        3.96451,  452.,       22.50007,   1225.,      2500.,
     2        0.0    ,  350.,       0.0      ,   793.,      2500.,
     3        7.866  ,  560.,       5.8      ,   1225.,      2500.,
     4        0.0    ,  452.,       0.0      ,   1225.,      2500.,
c j=1:
     1        0.81   ,  350.,       71.5     ,   1225.,      2500.,
     2        2.346  ,  424.,       19.22    ,   1225.,      2500.,
     3        0.575  ,  350.,       0.0      ,   793.,      2500.,
     4        0.51673,  350.,       14.01164,    793.,      2500.,
c j=2:
     1        2.236  ,  400.,       189.7    ,   1225.,      2500.,
     2        0.53   ,  350.,       154.5    ,   1225.,      2500.,
     3        1.573  ,  452.,       56.21    ,   793.,      2500.,
     4        4.194  ,  470.,       24.562   ,   1225.,      2500.,
c j=3:
     1        0.73   ,  350.,       0.0      ,   793.,      2500.,
     2        1.53   ,  452.,       74.85    ,   793.,      2500.,
     3        0.29   ,  350.,       0.0      ,   793.,      2500.,
     4        3.4    ,  452.,       0.0      ,   793.,      2500.,
c j=4:
     1        4.27591,  452.,       0.0      ,   1225.,      2500.,
     2        0.62   ,  350.,       0.0      ,   793.,      2500.,
     3        3.85   ,  452.,       17.61    ,   1225.,      2500.,
     4        3.8115 ,  452.,       17.61    ,   1225.,      2500.,
c j=5:
     1        0.51673,  350.,       0.0      ,   793.,      2500.,
     2        4.24591,  452.,       0.0      ,   1225.,      2500.,
     3        0.96   ,  350.,       0.0      ,   793.,      2500.,
     4        0.96   ,  350.,       0.0      ,   793.,      2500.,
c j=6:
     1        2.3    ,  452.,       0.0      ,   1225.,      2500.,
     2        2.3    ,  452.,       0.0      ,   1225.,      2500.,
     3        2.3    ,  452.,       0.0      ,   1225.,      2500.,
     4        2.3    ,  452.,       0.0      ,   1225.,      2500.,
c
c neutron-neutron potential
c --------------------------
c j=0:
     1        4.26338,  452.,       17.540,      1225.,      2500.,
     2        0.0    ,  350.,       0.0    ,      793.,      2500.,
     3        7.892  ,  560.,       16.747,      1225.,      2500.,
     4        0.0    ,  452.,       0.0    ,      1225.,      2500.,
c j=1:
     1        0.0    ,  350.,       0.0    ,      1225.,      2500.,
     2        2.326  ,  424.,       17.61  ,      1225.,      2500.,
     3        0.0    ,  350.,       0.0    ,      793.,      2500.,
     4        0.0    ,  350.,       0.0    ,      793.,      2500.,
c j=2:
     1        2.241  ,  400.,       190.7  ,      1225.,      2500.,
     2        0.0    ,  350.,       0.0    ,      1225.,      2500.,
     3        1.522  ,  452.,       56.28  ,      793.,      2500.,
     4        4.180  ,  470.,       24.737,      1225.,      2500.,
c j=3:
     1        0.0    ,  350.,       0.0    ,      793.,      2500.,
     2        1.53   ,  452.,       74.44  ,      793.,      2500.,
     3        0 0    ,  350.,       0.0    ,      793.,      2500.,
     4        0.0    ,  452.,       0.0    ,      793.,      2500.,
```

```
c j=4:
     1        4.284   ,  452.,       0.0    ,     1225.,      2500.,
     2        0.0     ,  350.,       0.0    ,      793.,      2500.,
     3        3.83    ,  452.,       17.61  ,     1225.,      2500.,
     4        3.81    ,  452.,       17.61  ,     1225.,      2500.,
c j=5:
     1        0.0     ,  350.,       0.0    ,      793.,      2500.,
     2        4.24591 ,  452.,       0.0    ,     1225.,      2500.,
     3        0.0     ,  350.,       0.0    ,      793.,      2500.,
     4        0.0     ,  350.,       0.0    ,      793.,      2500.,
c j=6:
     1        2.3     ,  452.,       0.0    ,     1225.,      2500.,
     2        2.3     ,  452.,       0.0    ,     1225.,      2500.,
     3        2.3     ,  452.,       0.0    ,     1225.,      2500.,
     4        2.3     ,  452.,       0.0    ,     1225.,      2500./
c
c        this has been the end of all tables
c        -----------------------------------
c
c
c
c

      save
```



```
10002 format (' type/meson       m e s o n    p a r a m e t e r s')
10004 format (1h ,a4,a1,a3,a4,4f12.5,f12.2)
10008 format (1h ,61(1h-))
10011 format (///' CDBONN: The Charge-Dependent Bonn NN Potential ("CDBo
     1nn2000")')
10017 format (///)
10018 format (' Potential Type:  ',a4)
c
c
c
c

      if (index) go to 50
      index=.true.
```

```
c
      x=-1.d0
      y=-1.d0
c
c
c
c
c        maxima of certain indices related to the dimension as follows:
c        dimension c(mme,imee),ic(mice,imee),indc(mindce,imee),
c                  mgg(mge,3),mggo(mge,3),mesong(mge),vj(32,imee),
c                  ima(mee,mge,3)
c
      mge=12
      mee=15
      mme=20
      mice=20
      mindce=10
      imee=50
c        mme always ge mice, mindce
      imb=1
      endep=.false.
c
c
c        set all meson-parameters and indices to zero or .false.
c
      do 1 int=1,3
      imga(int)=0
      indpar(int)=.false.
      do 1 mgx=1,mge
      mgg(mgx,int)=0
    1 mggo(mgx,int)=0
c
c
      do 2 il=1,imee
      do 2 mm=1,mme
      if (mm.le.mindce) indc(mm,il)=.false.
      if (mm.le.mice) ic(mm,il)=0
    2 c(mm,il)=0.d0
c
c
c        write headline
c
      write (kwrite,10011)
      write (kwrite,10008)
      write (kwrite,10008)
      write (kwrite,10017)
c
c
      ift(inter)=1
c
c        scaling mass
c
      wscale(inter)=938.27231d0
c
c
c
c
   50 if (inn.eq.innn) go to 55
      innn=inn
c
c
      imga(inter)=0
      do 11 mgx=1,mge
      mgg(mgx,inter)=0
   11 mggo(mgx,inter)=0
c
```

```
      ime=0
      line=0
      jj=-1
c
c
c
c
c**** write (kwrite,10018) nucnuc(inn)
c      label=nucnuc(inn)
c
c
      if (inn.eq.1) then
      wn1=938.27231d0
      wn2=938.27231d0
      else
      if (inn.eq.2) then
      wn1=939.56563d0
      wn2=938.27231d0
      else
      wn1=939.56563d0
      wn2=939.56563d0
      end if
      end if
c
c
      wnn(inter)=dsqrt(wn1*wn2)
c
c
      wn=wnn(inter)
      wnq=wn*wn
      dwn=1.d0/wn
      dwnq=dwn*dwn
c
c
c        write headline for meson parameters
c
c**** write (kwrite,10008)
c**** write (kwrite,10008)
c**** write (kwrite,10002)
c**** write (kwrite,10008)
c**** write (kwrite,10008)
      go to 61
c
c
c
c
   55 ime=10
      mgx=mggo(imga(inter),inter)
      mgg(mgx,inter)=0
      mggo(imga(inter),inter)=0
      imga(inter)=imga(inter)-1
c
c        write headline for meson parameters
c
c**** write (kwrite,10008)
c**** write (kwrite,10008)
c**** write (kwrite,10002)
c**** write (kwrite,10008)
c
c
c
c
c        read, write, and store meson parameters
c        ---------------------------------------
c        ---------------------------------------
c
```

```
c
c
   61 if (ime.eq.18) go to 2000
c
c
c        instead of reading, get meson parameters from data tables
c        ---------------------------------------------------------
c
c
      if (ime.lt.10) then
      line=line+1
      do 63 i=1,5
      if (i.le.4) then
      name(i)=ntab1(i,line)
      end if
   63 cc(i)=tab1(i,line,inn)
c
c
      else
      if (j.eq.jj) then
      line=line+1
      else
      line=1
      jj=j
      j1=j+1
      if (j.gt.6) j1=7
      end if
c
      do 65 i=1,5
      if (i.le.4) then
      name(i)=ntab2(i)
      end if
   65 cc(i)=tab2(i,line,j1,inn)
      end if
c
c
c        check if record just read contains cut-off parameters
c
      if (name(1).eq.cut) go to 80
c
c
c
c
c        write meson-parameters
c        ----------------------
c
c
c
c
c**** write (kwrite,10004) name,cc
c
c        find out number of meson-group mg
c
      do 73 mg=1,mge
      if (name(1).eq.mesong(mg)) go to 74
   73 continue
      go to 9000
c
c
   74 if (name(2).eq.two) go to 1000
c
c
c
c
c        store meson parameters, which are no cut-off parameters
c        -------------------------------------------------------
```

```
c
c
c
c
      ime=ime+1
      if (ime.gt.imee) go to 9011
      mgg(mg,inter)=mgg(mg,inter)+1
      m=mgg(mg,inter)
      if (m.gt.mee) go to 9001
      ima(m,mg,inter)=ime
      if (m.ne.1) go to 76
      imga(inter)=imga(inter)+1
      mggo(imga(inter),inter)=mg
   76 continue
c
c        store coupling constant g**2/4pi
      c(1,ime)=cc(1)
c        store coupling constant f*g/4pi
      c(3,ime)=cc(1)*cc(2)*wn/wscale(inter)
c        store coupling constant f**2/4pi
      c(2,ime)=cc(2)*c(3,ime)*wn/wscale(inter)
c        store meson mass squared in units of nucleon mass squared
      c(4,ime)=cc(3)*cc(3)*dwnq
c
c        get iso-spin
      icc=cc(4)
      if (icc.ne.0.and.icc.ne.1) go to 9004
c         store isospin as logical constant
      if (icc.eq.1) indc(1,ime)=.true.
c        store parameter for meson propagator (iprop)
      ic(1,ime)=cc(5)
      if (ic(1,ime).ne.0) go to 9005
c
c        index values for further storing
      mi=4
      mm=5
      go to 61
c
c
c
c
c        write cut-off parameters
c        ------------------------
c
c
c
c
   80 continue
c**** write (kwrite,10004) name,cc
c
c
      if (ime.eq.1) eps=cc(5)
c
c
c
c        if cutoff type = 0, ignore cutoff
      if (cc(1).eq.0.d0) go to 61
c
c
c
c
c        store cut-off parameters
c        ------------------------
c
c
c
```

```
c
c        store type of cut-off
      ic(mi,ime)=cc(1)
      ityp=ic(mi,ime)
      if (ityp.ne.2) go to 9002
c        store and test type of denominator of cut-off
      ic(mi+1,ime)=cc(2)
      if (ic(mi+1,ime).ne.0) go to 9006
c
c
c        cut-off of monopole/dipole type
c        ******************************
c
c
c        store and test exponent of cut-off
      ic(mi+2,ime)=cc(3)
      if (ic(mi+2,ime).lt.0) go to 9009
      if (ic(mi+2,ime).gt.0) go to 101
c        exponent is zero, omit cut-off
      ic(mi,ime)=0
      ic(mi+1,ime)=0
      go to 999
  101 if (ic(mi+2,ime).ne.2) go to 9012
c        store first cut-off mass
      c(mm,ime)=(cc(4)+eps)**2*dwnq
c        store second cut-off mass
      c(mm+1,ime)=(cc(4)-eps)**2*dwnq
      mi=mi+3
      mm=mm+2
c
c
c
c
c        end cut-offs
c        ************
c
c        test dimensions
  999 if (mi.gt.mice.or.mm.gt.mme) go to 9010
c
c
      go to 61
c
c
c
c
c        two mesons on one input line
c        ----------------------------
c
c
c        store input parameters and set defaults
c
c
 1000 do 1995 ii=1,2
      ime=ime+1
      if (ime.gt.imee) go to 9011
      mgg(mg,inter)=mgg(mg,inter)+1
      m=mgg(mg,inter)
      if (m.gt.mee) go to 9001
      ima(m,mg,inter)=ime
      if (m.ne.1) go to 1076
      imga(inter)=imga(inter)+1
      mggo(imga(inter),inter)=mg
 1076 continue
c
c        store coupling constant g**2/4pi
      if (ii.eq.1) then
```

```
      c(1,ime)=cc(1)
      else
      c(1,ime)=cc(3)
      end if
c
c        scale the pi-NN coupling constant
      if (ime.ge.5.and.ime.le.10) then
      c(1,ime)=c(1,ime)*(wn/wscale(inter))**2
      end if
c
c        set coupling constant f*g/4pi
      c(3,ime)=0.d0
c        set coupling constant f**2/4pi
      c(2,ime)=0.d0
c        store meson mass squared in units of nucleon mass squared
      if (ii.eq.1) then
      c(4,ime)=cc(2)*cc(2)*dwnq
      else
      c(4,ime)=cc(4)*cc(4)*dwnq
      end if
c
c         set isospin-0 as logical constant
      indc(1,ime)=.false.
c        set parameter for meson propagator (iprop=0)
      ic(1,ime)=0
c
c        index values for further storing
      mi=4
      mm=5
c
c
c        store and set cut-off parameters
c
c        set type of cut-off
      ic(mi,ime)=2
c        set type of denominator of cut-off
      ic(mi+1,ime)=0
c
c
c        cut-off of monopole/dipole type
c        ******************************
c
c
c        set exponent of cut-off
      ic(mi+2,ime)=2
c        store first cut-off mass
      c(mm,ime)=(cc(5)+eps)**2*dwnq
c        store second cut-off mass
      c(mm+1,ime)=(cc(5)-eps)**2*dwnq
      mi=mi+3
      mm=mm+2
c
c
c
c
c        end cut-offs
c        ************
c
c        test dimensions
      if (mi.gt.mice.or.mm.gt.mme) go to 9010
c
 1995 continue
c
      go to 61
c
c
```

```
c
c
c        end of mesons for one j
c        -----------------------
c
c
 2000 imaa(inter)=imb
      imea(inter)=ime
c**** write (kwrite,10008)
c**** write (kwrite,10008)
c
      return
c
c
c
c
c        errors
c        ------
c        ------
c
c
c
c
 9000 write (kwrite,19000) name(1)
19000 format (/////' error in cdbonn:  meson-group   ',a4,'   does not
     1exist in this program.'/' execution terminated.'////)
      go to 9999
c
c
 9001 write (kwrite,19001)
19001 format (/////' error in cdbonn:   too many mesons within a meson-g
     1roup with respect to'/' the given dimensions.    execution termina
     2ted.'////)
      go to 9999
c
c
 9002 write (kwrite,19002) cc(1)
19002 format (/////' error in cdbonn:  cut-off type',f10.4,'  does not e
     1xist in this program.'/' execution terminated.'////)
      go to 9999
c
c
 9004 write (kwrite,19004) cc(4)
19004 format (/////' error in cdbonn:  isospin has the non-permissible
     1value',f10.4,'  .'/' execution terminated.'////)
      go to 9999
c
c
 9005 write (kwrite,19005) cc(5)
19005 format (/////' error in cdbonn:    iprop has the non-permissible
     1value',f10.4,'  .'/' execution terminated.'////)
      go to 9999
c
c
 9006 write (kwrite,19006) cc(2)
19006 format (/////' error in cdbonn: the index for the denominator of
     1 the cut-off has the'/' non-permissible value',f10.4,' . execution
     2 terminated.'////)
      go to 9999
c
c
 9009 write (kwrite,19009)
19009 format (/////' error in cdbonn:   the exponent of the cut-off is
     1less than zero.'/' execution terminated.'////)
      go to 9999
c
```

```
c
 9010 write (kwrite,19010)
19010 format (/////' error in cdbonn: too many cut-off parameters with
     1 respect to the given'/' dimensions. execution terminated.'////)
       go to 9999
c
c
 9011 write (kwrite,19011)
19011 format (/////' error in cdbonn:  too many mesons with respect to
     1 the dimensions given'/' in this program. execution terminated.'
     2////)
       go to 9999
c
c
 9012 write (kwrite,19012)
19012 format (/////' error in cdbonn:   the exponent of the cut-off is
     1not two.'/' execution terminated.'////)
       go to 9999
c
c
 9999 stop
      end
```

```
      subroutine obstrq (icase,max,mex)
c
c        obstrq computes the structure of one-boson-exchanges
c
c
      implicit real*8 (a-h,o-z)
c
c
c        common blocks
c
      common /cstate/ j,heform,sing,trip,coup,endep,label
      logical heform,sing,trip,coup,endep
c
c
c        common block for all ob-subroutines
c
      common /cobq/   vj(32,50),c(20,50),fff,ff,f(52),aa(96),ai(19,15),
```

```
     1               wnn(3),wdd(3),x,xx,y,yy,xy2,xxpyy,ex,ey,eem12,
     2               ez1,ez2,ct(96),wt(96),
     3               ic(20,50),ift(3),mint(3),maxt(3),nt,
     4               mge,mgg(12,3),mggo(12,3),ima(15,12,3),
     5               imaa(3),imea(3),ime,im,mc,m,mg,inter,ide,idde,
     6               indc(10,50),indpar(3),indxy
c
c         specifications for this common block
c
      logical indc,indxy,indpar
c
c
      common /cnn/ inn
c
c
c     further specifications
c
      dimension vv(32)
      dimension tt(2,3)
      logical index
      logical indiso
      data jj/-1/
      data index/.false./
      save
c
c
c
c
      if (index) go to 50
      index=.true.
c
c
      tt(1,1)=1.d0
      tt(2,1)=-3.d0
c
      do 1 ii=2,3
      do 1 i=1,2
    1 tt(i,ii)=1.d0
c
c
c
c
c
   50 do 1095 m=max,mex
      im=ima(m,mg,inter)
c
      if (mg.le.5.and.c(mc,im).eq.0.d0) go to 1095
c
c
      if (mc.ne.1) go to 60
c
c
c
c
c        call integrals
c        --------------
c
c
c
c
      call obaiq
c
c
c
c
   60 continue
```

```
c
      if (c(mc,im).eq.0.d0) go to 1095
c
c
c
c
c        nn-nn helicity amplitudes
c        -------------------------
c
c
c        vv(1), ..., vv(6) contain in the following order:
c        0v, 1v, 12v, 34v, 55v, 66v.
c
c
c        basic structure
c
c
  100 ive=6
c
      vv(1)=f(1)*ai(1,m)+f(2)*ai(2,m)
      vv(2)=f(3)*ai(1,m)+f(4)*ai(3,m)
      vv(3)=f(5)*ai(1,m)+f(6)*ai(2,m)
      vv(4)=f(4)*ai(1,m)+f(3)*ai(3,m)
      vv(5)=f(7)*ai(4,m)
      vv(6)=f(8)*ai(4,m)
c
c
      go to (1000,120),icase
c
c
c        additional terms for the case of tensor-tensor coupling
c
c
  120 vv(1)=vv(1)+f(9)*ai(5,m)
      vv(2)=vv(2)+f(10)*ai(2,m)+f(9)*ai(6,m)
      vv(3)=vv(3)+f(10)*ai(5,m)
      vv(4)=vv(4)+f(9)*ai(2,m)+f(10)*ai(6,m)
         e1=f(11)*ai(7,m)
      vv(5)=vv(5)+e1
      vv(6)=vv(6)+e1
      go to 1000
c
c
c
c
c        set certain cases to zero
c
 1000 if (j.ne.0) go to 1021
      vv(2)=0.d0
      vv(4)=0.d0
      vv(5)=0.d0
      vv(6)=0.d0
c
 1021 mmod=mod(j,2)
      if (.not.sing.or.(mmod.eq.1.and.inn.ne.2)) vv(1)=0.d0
      if (.not.trip.or.(mmod.eq.0.and.inn.ne.2)) vv(2)=0.d0
      if (coup.and.(mmod.eq.0.or.inn.eq.2)) go to 1030
      do 1025 iv=3,6
 1025 vv(iv)=0.d0
c
 1030 continue
c
c
c        transformation into lsj-formalism
c
      if (j.eq.jj) go to 1035
```

```
      jj=j
      aj=dble(j)
      aj1=dble(j+1)
      d2j1=1.d0/dble(2*j+1)
      arjj1=dsqrt(aj*aj1)
c
 1035 v3=vv(3)
      v4=vv(4)
      v5=vv(5)
      v6=vv(6)
      v34=arjj1*(v3-v4)
      v56=arjj1*(v5+v6)
      vv(3)=d2j1*(aj1*v3+aj*v4-v56)
      vv(4)=d2j1*(aj*v3+aj1*v4+v56)
      vv(5)=d2j1*(v34+aj1*v5-aj*v6)
      vv(6)=d2j1*(v34-aj*v5+aj1*v6)
c
c        after transformation into lsj formalism,
c        vv(3), ..., vv(6) contain:
c        v++, v--, v+-, v-+.
c
c
c
c
c        multiply with factors
c        ---------------------
c
c
c
c
      is=mod(j,2)+1
      it=mod(is,2)+1
      indiso=indc(1,im)
c        get coupling constant
      cmc=c(mc,im)
      fc=fff*ff*cmc
      do 1045 iv=1,ive
c
c        multiply with coupling-constant and factors fff and ff
c
      vv(iv)=vv(iv)*fc
c
c        multiply with isospin factor
c
      if (.not.indiso) go to 1045
      if (iv.eq.2) go to 1043
      vv(iv)=vv(iv)*tt(is,inter)
      go to 1045
 1043 vv(iv)=vv(iv)*tt(it,inter)
c
c
c        add up in case of several couplings for one meson and store
 1045 vj(iv,im)=vj(iv,im)+vv(iv)
c
c
 1095 continue
c
c
      return
      end
```

```
      subroutine obaiq
c
c        obaiq performs the integration over angle theta
c        (necessary for the partial wave decomposition)
c        in analytic form by using the Legendre functions of the
c        second kind.
c
c
      implicit real*8 (a-h,o-z)
c
      common /cstate/ j,heform,sing,trip,coup,endep,label
      logical heform,sing,trip,coup,endep
c
c
c        common block for all ob-subroutines
c
      common /cobq/   vj(32,50),c(20,50),fff,ff,f(52),aa(96),ai(19,15),
     1                wnn(3),wdd(3),x,xx,y,yy,xy2,xxpyy,ex,ey,eem12,
     2                ez1,ez2,ct(96),wt(96),
     3                ic(20,50),ift(3),mint(3),maxt(3),nt,
     4                mge,mgg(12,3),mggo(12,3),ima(15,12,3),
     5                imaa(3),imea(3),ime,im,mc,m,mg,inter,ide,idde,
     6                indc(10,50),indpar(3),indxy
c
c         specifications for this common block
c
      logical indc,indxy,indpar
c
c
      dimension gi(5,7)
      logical index
      data jj/-1/
      data index/.false./
      save
```

```
c
c
c
c
      if (index) go to 50
      index=.true.
c
      sqr2=dsqrt(2.d0)
c
c
c
c
c
   50 if (j.eq.jj) go to 70
      jj=j
c
c
      aj=dble(j)
      aj1=dble(j+1)
      dj1=1.d0/aj1
      ajdj1=aj*dj1
      aaj=dsqrt(ajdj1)
c
c
      if (j.eq.0) then
      delj0=1.d0
      else
      delj0=0.d0
      end if
c
      if (j.eq.1) then
      delj1=1.d0
      else
      delj1=0.d0
      end if
c
c
   70 continue
c
c
c
c
      mi=4
      mm=3
      ityp=ic(mi,im)
      nexp=ic(mi+2,im)
c
      nterms=nexp+1
      if (ityp.eq.0) nterms=1
c
c
      do 555 i=1,nterms
      mmi=mm+i
c
c        calculate the argument for the legendre function
c
      if (x.eq.y) then
      zstamm=1.d0
      zdelta=c(mmi,im)/xy2
      else
      zstamm=(xxpyy+c(mmi,im))/xy2
      zdelta=0.d0
      end if
c
      z=zstamm+zdelta
c
```

```
c
c        call legendre functions of the second kind
c
      if (j.eq.0) then
c
      call legen2 (qj,qjp1,zzq1m,1,zstamm,zdelta)
      qjm1=0.d0
c
      else
c
      call legen2 (qjm1,qj,zzq1m,j,zstamm,zdelta)
c
      end if
c
c
      gi(i,1)=qj
c
      if (j.eq.0) then
      gi(i,2)=qjp1
      else
      gi(i,2)=z*qj-delj0
      end if
c
      gi(i,3)=ajdj1*z*qj+dj1*qjm1
      gi(i,4)=aaj*(z*qj-qjm1)
c
      if (j.eq.1) then
      gi(i,5)=zzq1m
      gi(i,6)=0.5d0*(zzq1m+qj)
      gi(i,7)=0.5d0*sqr2*(zzq1m-qj)
      else
      gi(i,5)=z*gi(i,2)-delj1/3.d0
      gi(i,6)=z*gi(i,3)-2.d0*delj1/3.d0
      gi(i,7)=z*gi(i,4)+sqr2*delj1/3.d0
      end if
c
c
      if (i.eq.1) then
      fact=1.d0
      else
      ix=1
      if (i.eq.3) ix=-1
      fact=(c(mmi+ix,im)-c(4,im))/(c(mmi,im)-c(mmi+ix,im))
      end if
c
c
      do 545 ii=1,7
      gi(i,ii)=fact*gi(i,ii)
  545 continue
c
c
  555 continue
c
c
      do 725 ii=1,7
      ai(ii,m)=0.d0
      do 715 i=1,nterms
  715 ai(ii,m)=ai(ii,m)+gi(i,ii)
  725 continue
c
c
      dxy=2.d0/xy2
      do 2015 ii=1,7
 2015 ai(ii,m)=ai(ii,m)*dxy
c
c
```

```
c
c
      return
      end









      subroutine legen2 (qjm1,qj,zzq1m,j,zstamm,zdelta)
c
c**** legendre-funktionen der zweiten art ****
c
c**** notation: qjm1 = Q_{J-1}, qj = Q_J, z=zstamm+zdelta,
c**** in the case of j=1, this program also provides
c****                    zzq1m =  z*z*Q_1-1./3.
c
c
c     author:
c              R. Machleidt
c              Institut fuer Theoretische Kernphysik
c              der Universitaet Bonn
c              Nussallee 14-16
c              D-5300 Bonn
c              W. Germany
c
c              original version: April 1972
c              last revision: April 1995
c
c
c**** genauigkeit:
c**** j kleiner gleich 10   15 stellen
c**** j gleich  11 bis 30   mindestens 13 stellen
c**** j gleich 31 bis 100   mindestens 12 stellen
c**** eine dimension der koeffizienten von  me=40000  c(2,40001)
c**** ist gut bis j=50;
c**** fuer j=100 wird  me=150000  c(2,150001) benoetigt.
c
c
      implicit real*8 (a-h,o-z)
      common /crdwrt/ kread,kwrite,kpunch,kda(9)
      dimension c(2,40001)

      data tolr/1.d-16/
      data me/40000/
      data jj/-1/
      save
c
c
c**** berechnung des arguments ****
      z=zstamm+zdelta
c
      qjm1=0.d0
```

```
      qj=0.d0
      zzq1m=0.d0
      if (j.lt.0) go to 123
      if (z.le.1.d0) go to 113
c
c
c**** fallunterscheidung ****
      if (j.ne.0) go to 2
      if (z-10.d0) 10,10,11
    2 if (j.ne.1) go to 3
      if (z-1.5d0) 10,10,11
    3 if (j.ne.2) go to 4
      if (z-1.2d0) 10,10,11
    4 zcut=1.d0+dble(j)**(-2.d0)
      if (z-zcut) 10,10,11
c
c**** rekursive berechnung mit dem logarithmus ****
   10 zdel=zstamm-1.d0
      zdel  =zdel+zdelta
      zz=2.d0/zdel  +1.d0
      qjm1=0.5d0*dlog(zz)
      if (j.eq.0) then
      qj=qjm1
      qjm1=0.d0
      return
      end if
      qj=z*qjm1-1.d0
      if (j.eq.1) then
      zzq1m=z*z*qj-1.d0/3.d0
      return
      end if
      do  7 i=2,j
      qq=dble(2*i-1)/dble(i)*z*qj-dble(i-1)/dble(i)*qjm1
      qjm1=qj
    7 qj=qq
      return
c
c**** berechnung mit reihe ****
c**** der laufende index m ist immer mue plus eins ****
   11 zqinv=z**(-2)
      zzqinv=1.d0
      qjm1=0.d0
      qj=0.d0
      zzq1m=0.d0
      if (j.eq.jj) go to 12
      jj=j
      ma=1
      go to 14
   12 do 13 m=1,mme
      cz1=c(1,m)*zzqinv
      cz= c(2,m)*zzqinv
      qjm1=cz1+qjm1
      qj=  cz+qj
      if (j.eq.1) then
      if (m.eq.1) then
      zzq1m=0.d0
      else
      zzq1m=zzq1m+cz
      end if
      if (cz.lt.tolr*zzq1m) go to 62
      go to 13
      end if
      if (cz.lt.tolr*qj) go to 62
   13 zzqinv=zzqinv*zqinv
      ma=mme+1
c
```

```
c**** verteiler ****
   14 if (j.le.1) go to 20
      if (j.eq.2) go to 30
      if (mod(j,2)) 50,40,50
c
c**** die faelle j gleich null und j gleich eins ****
   20 if (ma.ne.1) go to 22
      ma=2
      c(1,1)=1.d0
      c(2,1)=1.d0/3.d0
      qjm1=c(1,1)
      qj=c(2,1)
      zzq1m=0.d0
      zzqinv=zqinv
   22 do 21 m=ma,me
      c(1,m)=c(2,m-1)
      c(2,m)=1.d0/dble(2*m+1)
      cz1=c(1,m)*zzqinv
      cz= c(2,m)*zzqinv
      qjm1=cz1+qjm1
      qj=  cz+qj
      if (j.eq.1) then
      zzq1m=zzq1m+cz
      if (cz.lt.tolr*zzq1m) go to 61
      go to 21
      end if
      if (cz.lt.tolr*qj) go to 61
   21 zzqinv=zzqinv*zqinv
      go to 60
c
c**** fall j gleich zwei ****
   30 do 31 m=ma,me
      m2=2*m
      c(1,m)=1.d0/dble(m2+1)
      c(2,m)=c(1,m)*dble(m2)/dble(m2+3)
      qjm1=   c(1,m)*zzqinv+qjm1
      cz=     c(2,m)*zzqinv
      qj=cz+qj
      if (cz.lt.tolr*qj) go to 61
   31 zzqinv=zzqinv*zqinv
      go to 60
c
c**** fall j ist gerade ****
   40 do 41 m=ma,me
      m2=2*m
c**** zaehler ****
      aehler=1.d0
      ka=m2
      kez=m2+j-4
      do 42 k=ka,kez,2
   42 aehler=aehler*dble(k)
c**** nenner ****
      aenner=1.d0
      ka=m2+j-1
      ken=m2+2*j-3
      do 43 k=ka,ken,2
   43 aenner=aenner*dble(k)
      c(1,m)=aehler/aenner
      c(2,m)=c(1,m)*dble(kez+2)/dble(ken+2)
      qjm1=   c(1,m)*zzqinv+qjm1
      cz=     c(2,m)*zzqinv
      qj=cz+qj
      if (cz.lt.tolr*qj) go to 61
   41 zzqinv=zzqinv*zqinv
      go to 60
c
```

```
c**** fall j ist ungerade ****
   50 do 51 m=ma,me
      m2=2*m
c**** zaehler ****
      aehler=1.d0
      ka=m2
      ke=m2+j-3
      do 52 k=ka,ke,2
   52 aehler=aehler*dble(k)
      if (m.ne.1) go to 55
      m2=0
      go to 54
   56 m2=2
   55 c(1,m)=aehler/aenner
c**** nenner ****
   54 aenner=1.d0
      ka=m2+j
      ke=m2+2*j-1
      do 53 k=ka,ke,2
   53 aenner=aenner*dble(k)
      if (m2) 57,56,57
   57 c(2,m)=aehler/aenner
      qjm1=   c(1,m)*zzqinv+qjm1
      cz=     c(2,m)*zzqinv
      qj=cz+qj
      if (cz.lt.tolr*qj) go to 61
   51 zzqinv=zzqinv*zqinv
c
c
   60 mme=me
      write (kwrite,1131)
 1131 format (/////' warning in legen2. the dimension for the'/
     1' coefficients is too small. the Legendre function of the'/
     2' second kind may be inaccurate.'/////)
      go to 62
c
   61 mme=m
c
c**** schlussrechnung ****
   62 zmj1=z**(-j-1)
      if (j.eq.0) go to 68
      qj=qj*zmj1
      qjm1=qjm1*zmj1*z
      return
   68 qj=qjm1*zmj1
      qjm1=0.d0
      return
c
c**** fehlermeldung ****
  113 write (kwrite,1130)
 1130 format (/////' error in legen2. the argument of the'/
     1' Legendre function of the second kind is smaller or'/
     2' equal one. the function is set to zero.'/
     3' results may be wrong.'/////)
      return
  123 write (kwrite,1230)
 1230 format (/////' error in legen2. the parameter j of the'/
     1' Legendre function of the second kind is smaller zero.'/
     2' the function is set to zero.'/
     3' results may be wrong.'/////)
      return
      end
```

```
c***********************************************************************
c name:    dminv
c        programmbibliothek rhrz bonn        28/11/78       dminv
c                                            fortran iv     ibm 370/168
c
c purpose:
c
c invert a matrix
c
c usage:   call dminv (a,n,d,l,m)
c
c parameters:
c
c a:       input matrix, destroyed in computation and replaced by
c          resultant inverse.
c          double precision required.
c
c n:       order of matrix a
c
c d:       resultant determinant
c          double precision required.
c
c l:       work vector of length n
c
c m:       work vector of length n
c
c remarks: matrix a must be a general matrix
c
c method:
c
c the standard gauss-jordan method is used. the determinant
c is also calculated. a determinant of zero indicates that
c the matrix is singular.
c
c programs required:
c          none
c
c author:  ibm, ssp iii
c
c***********************************************************************
      subroutine dminv (a,n,d,l,m)
      implicit real*8 (a-h,o-z)
      dimension a(1),l(1),m(1)
```

```
c
c
c        search for largest element
c
      d=1.d0
      nk=-n
      do 80 k=1,n
      nk=nk+n
      l(k)=k
      m(k)=k
      kk=nk+k
      biga=a(kk)
      do 20 j=k,n
      iz=n*(j-1)
      do 20 i=k,n
      ij=iz+i
   10 if (dabs(biga)-dabs(a(ij)))  15,20,20
   15 biga=a(ij)
      l(k)=i
      m(k)=j
   20 continue
c
c        interchange rows
c
      j=l(k)
      if(j-k) 35,35,25
   25 ki=k-n
      do 30 i=1,n
      ki=ki+n
      hold=-a(ki)
      ji=ki-k+j
      a(ki)=a(ji)
   30 a(ji) =hold
c
c        interchange columns
c
   35 i=m(k)
      if(i-k) 45,45,38
   38 jp=n*(i-1)
      do 40 j=1,n
      jk=nk+j
      ji=jp+j
      hold=-a(jk)
      a(jk)=a(ji)
   40 a(ji) =hold
c
c        divide column by minus pivot (value of pivot element is
c        contained in biga)
c
   45 if(biga) 48,46,48
   46 d=0.d0
      return
   48 do 55 i=1,n
      if(i-k) 50,55,50
   50 ik=nk+i
      a(ik)=a(ik)/(-biga)
   55 continue
c
c        reduce matrix
c
      do 65 i=1,n
      ik=nk+i
      hold=a(ik)
      ij=i-n
      do 65 j=1,n
      ij=ij+n
```

```
      if(i-k) 60,65,60
   60 if(j-k) 62,65,62
   62 kj=ij-i+k
      a(ij)=hold*a(kj)+a(ij)
   65 continue
c
c        divide row by pivot
c
      kj=k-n
      do 75 j=1,n
      kj=kj+n
      if(j-k) 70,75,70
   70 a(kj)=a(kj)/biga
   75 continue
c
c        product of pivots
c
      d=d*biga
c
c        replace pivot by reciprocal
c
      a(kk)=1.d0/biga
   80 continue
c
c        final row and column interchange
c
      k=n
  100 k=(k-1)
      if(k) 150,150,105
  105 i=l(k)
      if(i-k) 120,120,108
  108 jq=n*(k-1)
      jr=n*(i-1)
      do 110 j=1,n
      jk=jq+j
      hold=a(jk)
      ji=jr+j
      a(jk)=-a(ji)
  110 a(ji) =hold
  120 j=m(k)
      if(j-k) 100,100,125
  125 ki=k-n
      do 130 i=1,n
      ki=ki+n
      hold=a(ki)
      ji=ki-k+j
      a(ki)=-a(ji)
  130 a(ji) =hold
      go to 100
  150 return
      end
c**************** this is the end of the program cdbonn **********************
```

```
c ***************************************************************
c Argonne v18 potential package
c
c prepared 1 Sept.94 by R.B.Wiringa, Physics Division,
c Argonne National Laboratory, Argonne, IL 60439
c e-mail: wiringa@theory.phy.anl.gov
c
c reference:
c  "An accurate nucleon-nucleon potential with charge-independence
c   breaking" by R.B.Wiringa, V.G.J.Stoks, and R.Schiavilla,
c   Physical Review C51, 38 (1995)
c
c this file contains 4 subroutines:
c   subroutine av18pw(l,s,j,t,t1z,t2z,r,vpw)
c   subroutine av18op(r,vnn)
c   subroutine empot(r,vem)
c   subroutine consts(hc,mpi0,mpic,mp,mn,alpha,mup,mun)
c
c av18pw gives the full potential in a particular partial wave
c av18op gives the strong interaction part in operator format
c empot  gives the electromagnetic part in operator format
c consts gives values of fundamental constants and masses used
c
c notes:
c 1) empot does not include the energy-dependence of the Coulomb
c    interaction used in eq.(4), i.e., it uses alpha, not alpha'.
c 2) the vacuum polarization in empot is a short-range approximation
c    to eq.(7) suitable for bound states, but not for scattering.
c 3) these subroutines should be compiled with a compiler option
c    that forces all floating point constants to be evaluated at
c    real*8 significance, e.g., on an IBM RS6000 the xlf compiler
c    option qdpc=e should be used, while on a Cray no action is needed;
c    if such an option is not available and the default precision is
c    real*4 (32 bits), then all constants should be explicitly
c    converted to double precision by appending a D0.
c
c *id* av18pw ************************************************
c subroutine for partial-wave projection of argonne v18 potential
```

```
c calls subroutines av18op, empot, consts
c ----------------------------------------------------------------------
c arguments for av18pw
c l:   orbital angular momentum of pair (0,1,2,...)
c s:   total spin of pair (0 or 1)
c j:   total angular momentum of pair (0,1,2,...)
c t:   total isospin of pair (0 or 1)
c t1z: isospin of particle 1 (1 for p, -1 for n)
c t2z:    "    "     "     2 (1 for p, -1 for n)
c r:   separation in fm
c v:   returned potential in MeV (2x2 array)
c      (includes all strong and em terms)
c ----------------------------------------------------------------------
c order of terms in v(l,m):
c      single channel                coupled channel (l=j-1,s=1)
c      v(1,1) = v(l,s,j,t,t1z,t2z)   v(1,1) = v(l,s,j,t,t1z,t2z)
c      v(2,1) = 0                    v(2,1) = v(l<->l+2)
c      v(1,2) = 0                    v(1,2) = v(l<->l+2)
c      v(2,2) = 0                    v(2,2) = v(l+2,s,j,t,t1z,t2z)
c ----------------------------------------------------------------------
      subroutine av18pw(l,s,j,t,t1z,t2z,r,vpw)
c      subroutine av18pw(l,s,j,t,r,vpw)
      implicit real*8 (a-h,o-z)
      implicit integer*4 (i-n)
      integer*4 l,s,j,t,t1z,t2z,s1ds2,t1dt2,t12
      dimension vnn(18),vem(14),vpw(2,2)
c ------------------------
c strong interaction terms
c ------------------------
      call av18op(r,vnn)
      s1ds2=4*s-3
      t1dt2=4*t-3
      t12=3*t1z*t2z-t1dt2
      vc=vnn(1)+t1dt2*vnn(2)+s1ds2*vnn(3)+s1ds2*t1dt2*vnn(4)
     &  +t12*vnn(15)+s1ds2*t12*vnn(16)+(t1z+t2z)*vnn(18)
      vt=vnn(5)+t1dt2*vnn(6)+t12*vnn(17)
      vls=vnn(7)+t1dt2*vnn(8)
      vl2=vnn(9)+t1dt2*vnn(10)+s1ds2*vnn(11)+s1ds2*t1dt2*vnn(12)
      vls2=vnn(13)+t1dt2*vnn(14)


C       modified v18 to v14 only

c      vc=vnn(1)+t1dt2*vnn(2)+s1ds2*vnn(3)+s1ds2*t1dt2*vnn(4)
c      vt=vnn(5)+t1dt2*vnn(6)
c      vls=vnn(7)+t1dt2*vnn(8)
c      vl2=vnn(9)+t1dt2*vnn(10)+s1ds2*vnn(11)+s1ds2*t1dt2*vnn(12)
c      vls2=vnn(13)+t1dt2*vnn(14)

c ---------------------
c electromagnetic terms
c ---------------------
      call empot(r,vem)
      if (t1z+t2z) 10,20,30
   10 vc=vc+s1ds2*vem(7)
      vt=vt+vem(10)
      go to 40
   20 vc=vc+vem(5)+s1ds2*vem(8)
      vt=vt+vem(11)
      vls=vls+vem(14)
      go to 40
   30 vc=vc+vem(1)+vem(2)+vem(3)+vem(4)+s1ds2*vem(6)
      vt=vt+vem(9)
      vls=vls+vem(12)
   40 continue
      ncc=1
```

```
      if (s.eq.1.and.j.gt.l) ncc=2
      if (ncc.eq.1) then
        s12=0.
        if (s.eq.1.and.l.eq.j) s12=2.
        if (l.eq.(j+1)) s12=-2.*(j+2.)/(2.*j+1.)
        ls=(j*(j+1)-l*(l+1)-s*(s+1))/2
        vpw(1,1)=vc+s12*vt+ls*vls+l*(l+1)*vl2+ls**2*vls2
        vpw(2,1)=0
        vpw(1,2)=0
        vpw(2,2)=0
      else if (ncc.eq.2) then
        s12m=-2.*(j-1.)/(2.*j+1.)
        s12=sqrt(36.*j*(j+1))/(2.*j+1.)
        s12p=-2.*(j+2.)/(2.*j+1.)
        lsm=j-1
        lsp=-(j+2)
        vpw(1,1)=vc+s12m*vt+lsm*vls+l*(l+1)*vl2+lsm**2*vls2
        vpw(2,1)=s12*vt
        vpw(1,2)=s12*vt
        vpw(2,2)=vc+s12p*vt+lsp*vls+(l+2)*(l+3)*vl2+lsp**2*vls2
      end if
      return
      end
c *id* av18op ************************************************************
c subroutine for strong interaction part of argonne v18 potential
c in operator format
c calls subroutine consts
c ---------------------------------------------------------------------
c arguments for av18pot
c r:   separation in fm
c vnn: output potential in MeV (18 component array)
c ---------------------------------------------------------------------
c order of operators l in vnn(l):
c l:    1=1                              2=t1.t2
c       3=s1.s2                          4=(s1.s2)(t1.t2)
c       5=S12 [=3(s1.r)(s2.r)-s1.s2]     6=S12(t1.t2)
c       7=L.S                            8=L.S(t1.t2)
c       9=L**2                          10=L**2(t1.t2)
c      11=L**2(s1.s2)                   12=L**2(s1.s2)(t1.t2)
c      13=(L.S)**2                      14=(L.S)**2(t1.t2)
c      15=T12 [=3*t1z*t2z-t1.t2]        16=(s1.s2)T12
c      17=S12*T12                       18=t1z+t2z
c where s1=sigma_1, t1=tau_1, t1z=tau_1(z), etc.
c ---------------------------------------------------------------------
      subroutine av18op(r,vnn)
      implicit real*8 (a-h,o-z)
      implicit integer*4 (i-n)
      dimension vnn(18)
      real*8 mpi0,mpic,mp,mn,mup,mun
      real*8 mpi,mu0,muc,mu
      data small/1e-4/
      do 5 l=1,18
        vnn(l)=0
    5 continue
      call consts(hc,mpi0,mpic,mp,mn,alpha,mup,mun)
      mpi=(mpi0+2.*mpic)/3.
      mu0=mpi0/hc
      muc=mpic/hc
      mu=mpi/hc
      fsq=.075
      cpi=2.1
      rws=.5
      aiws=5.
      x=mu*r
      x0=mu0*r
      xc=muc*r
```

```
      if (r.le.small) then
        tpi=3*cpi**2*r/mu**3
        ypi0=(mpi0/mpic)**2*(mpi0/3)*cpi*r/mu0
        tpi0=3*cpi*ypi0/mu0**2
        ypic=(mpic/3)*cpi*r/muc
        tpic=3*cpi*ypic/muc**2
      else
        rcut=1-exp(-cpi*r*r)
        ypi=exp(-x)*rcut/x
        tpi=(1+(3+3/x)/x)*ypi*rcut
        ypi0=(mpi0/mpic)**2*(mpi0/3)*exp(-x0)*rcut/x0
        tpi0=(1+(3+3/x0)/x0)*ypi0*rcut
        ypic=(mpic/3)*exp(-xc)*rcut/xc
        tpic=(1+(3+3/xc)/xc)*ypic*rcut
      end if
      ypi0=fsq*ypi0
      ypic=fsq*ypic
      tpi0=fsq*tpi0
      tpic=fsq*tpic
      tpi2=tpi*tpi
      ws=1/(1+exp((r-rws)*aiws))
      ws0=1/(1+exp(-rws*aiws))
      wsp=ws*(1+aiws*exp(-rws*aiws)*ws0*r)
      wsx=ws*x
      wsx2=wsx*x
      dypi00=(mpi0/mpic)**2*(mpi0/3)*cpi/mu0
      dypic0=(mpic/3)*cpi/muc
      ypi0p=ypi0-fsq*dypi00*ws*r/ws0
      ypicp=ypic-fsq*dypic0*ws*r/ws0
      ypi=(ypi0+2*ypic)/3
      tpi=(tpi0+2*tpic)/3
      p11pp=  -7.62701*tpi2+1815.4920*wsp+1847.8059*wsx2+ypi0p
      p11np=  -7.62701*tpi2+1813.5315*wsp+1847.8059*wsx2-ypi0p+2*ypicp
      p11nn=  -7.62701*tpi2+1811.5710*wsp+1847.8059*wsx2+ypi0p
      pt1pp=   1.07985*tpi2 -190.0949*wsx -811.2040*wsx2+tpi0
      pt1np=   1.07985*tpi2 -190.0949*wsx -811.2040*wsx2-tpi0+2*tpic
      pt1nn=   1.07985*tpi2 -190.0949*wsx -811.2040*wsx2+tpi0
      pls1=    -.62697*tpi2 -570.5571*wsp +819.1222*wsx2
      pl211=    .06709*tpi2 +342.0669*wsp -615.2339*wsx2
      pls21=    .74129*tpi2   +9.3418*wsp -376.4384*wsx2
      p10=    -8.62770*tpi2+2605.2682*wsp +441.9733*wsx2-ypi0p-2*ypicp
      pt0=    1.485601*tpi2-1126.8359*wsx +370.1324*wsx2-tpi0-2*tpic
      pls0=     .10180*tpi2  +86.0658*wsp -356.5175*wsx2
      pl210=   -.13201*tpi2 +253.4350*wsp   -1.0076*wsx2
      pls20=    .07357*tpi2 -217.5791*wsp  +18.3935*wsx2
      p01pp= -11.27028*tpi2+3346.6874*wsp-3*ypi0p
      p01np= -10.66788*tpi2+3126.5542*wsp-3*(-ypi0p+2*ypicp)
      p01nn= -11.27028*tpi2+3342.7664*wsp-3*ypi0p
      pl201=    .12472*tpi2  +16.7780*wsp
      p00=    -2.09971*tpi2+1204.4301*wsp-3*(-ypi0p-2*ypicp)
      pl200=   -.31452*tpi2 +217.4559*wsp
      p11=(p11pp+p11nn+p11np)/3
      p11cd=(.5*(p11pp+p11nn)-p11np)/6
      p11cs=(p11pp-p11nn)/4
      pt1=(pt1pp+pt1nn+pt1np)/3
      pt1cd=(.5*(pt1pp+pt1nn)-pt1np)/6
      pt1cs=(pt1pp-pt1nn)/4
      p01=(p01pp+p01nn+p01np)/3
      p01cd=(.5*(p01pp+p01nn)-p01np)/6
      p01cs=(p01pp-p01nn)/4
      vnn(1)=.0625*(9*p11+3*p10+3*p01+p00)
      vnn(2)=.0625*(3*p11-3*p10  +p01-p00)
      vnn(3)=.0625*(3*p11  +p10-3*p01-p00)
      vnn(4)=.0625*(  p11  -p10  -p01+p00)
      vnn(5)=.25*(3*pt1+pt0)
      vnn(6)=.25*(  pt1-pt0)
```

```
      vnn(7)=.25*(3*pls1+pls0)
      vnn(8)=.25*(  pls1-pls0)
      vnn(9)= .0625*(9*pl211+3*pl210+3*pl201+pl200)
      vnn(10)=.0625*(3*pl211-3*pl210+  pl201-pl200)
      vnn(11)=.0625*(3*pl211+  pl210-3*pl201-pl200)
      vnn(12)=.0625*(  pl211-  pl210-  pl201+pl200)
      vnn(13)=.25*(3*pls21+pls20)
      vnn(14)=.25*(  pls21-pls20)
      vnn(15)=.25*(3*p11cd+p01cd)
      vnn(16)=.25*(  p11cd-p01cd)
      vnn(17)=pt1cd
      vnn(18)=p01cs
      return
      end
c *id* empot ************************************************************
c subroutine for electromagnetic part of Argonne v18 potential
c calls subroutine consts
c ------------------------------------------------------------------------
c arguments for empot
c r:    input separation in fm
c vem:  output potential in MeV (14 component array)
c ------------------------------------------------------------------------
c order of operators in vem(l)
c l:    1=C1    (pp)          2=DF    (pp)          3=C2      (pp)
c       4=VP    (pp)                                5=C1      (np)
c       6=s1.s2 (pp)          7=s1.s2 (nn)          8=s1.s2   (np)
c       9=S12   (pp)         10=S12   (nn)         11=S12     (np)
c      12=L.S   (pp)         13=L.S   (nn)         14=L.S     (np)
c C1 = one-photon-exchange Coulomb with form factor
c C2 = two-photon-exchange Coulomb
c DF = Darwin-Foldy
c VP = vacuum polarization (short-range approximation)
c all other terms from magnetic moment (MM) interactions
c ------------------------------------------------------------------------
      subroutine empot(r,vem)
      implicit real*8 (a-h,o-z)
      implicit integer*4 (i-n)
      dimension vem(14)
      real*8 mpi0,mpic,mp,mn,mup,mun
      real*8 kr,me,mr
      data small/1e-5/
      call consts(hc,mpi0,mpic,mp,mn,alpha,mup,mun)
      b=4.27
      br=b*r
      pi=acos(-1.)
      me=0.510999
      mr=mp*mn/(mp+mn)
      gamma=0.577216
      beta=.0189
      if (r.lt.small) then
        fcoulr=5*b/16
        ftr3=b**3*br**2/720
        flsr3=b**3/48
        kr=me*small/hc
      else
        fcoulr=(1-(1+11*br/16+3*br**2/16+br**3/48)*exp(-br))/r
        ftr3=(1-(1+br+br**2/2+br**3/6+br**4/24+br**5/144)*exp(-br))/r**3
        flsr3=(1-(1+br+br**2/2+7*br**3/48+br**4/48)*exp(-br))/r**3
        kr=me*r/hc
      end if
      fivp=-gamma+5./6.+abs(log(kr))+6*pi*kr/8
      fdelta=b**3*(1+br+br**2/3)*exp(-br)/16
      fnpr=b**3*(15+15*br+6*br**2+br**3)*exp(-br)/384
      vem(1)=alpha*hc*fcoulr
      vem(2)=-alpha*hc**3*fdelta/(4*mp**2)
      vem(3)=-vem(1)**2/mp
```

```
      vem(4)=2*alpha*vem(1)*fivp/(3*pi)
      vem(5)=alpha*hc*beta*fnpr
      vem(6)=-alpha*hc**3*mup**2*fdelta/(6*mp**2)
      vem(7)=-alpha*hc**3*mun**2*fdelta/(6*mn**2)
      vem(8)=-alpha*hc**3*mup*mun*fdelta/(6*mn*mp)
      vem(9)=-alpha*hc**3*mup**2*ftr3/(4*mp**2)
      vem(10)=-alpha*hc**3*mun**2*ftr3/(4*mn**2)
      vem(11)=-alpha*hc**3*mup*mun*ftr3/(4*mp*mn)
      vem(12)=-alpha*hc**3*(4*mup-1)*flsr3/(2*mp**2)
      vem(13)=0
      vem(14)=-alpha*hc**3*mun*flsr3/(2*mn*mr)
      return
      end
c *id* consts ************************************************************
c subroutine for constants in av18 potential
c ------------------------------------------------------------------------
c arguments for consts
c hc:    output value for hbar*c (MeV-fm)
c mpi0:    "      "    "  neutral pion mass (MeV)
c mpic:    "      "    "  charged pion mass (MeV)
c mp:      "      "    "  proton mass (MeV)
c mn:      "      "    "  neutron mass (MeV)
c alpha:   "      "    "  electromagnetic constant alpha
c mup:     "      "    "  proton magnetic moment (nm)
c mun:     "      "    "  neutron magnetic moment (nm)
c ------------------------------------------------------------------------
      subroutine consts(hc,mpi0,mpic,mp,mn,alpha,mup,mun)
      real*8 hc,mpi0,mpic,mp,mn,alpha,mup,mun
      hc=197.327053
      mpi0=134.9739
      mpic=139.5675
      mp=938.27231
      mn=939.56563
      alpha=1./137.035989
      mup= 2.7928474
      mun=-1.9130427
      return
      end









      subroutine av8(l,s,j,t,r,vpw)
      implicit real*8 (a-h,o-z)
      implicit integer*4 (i-n)
      integer*4 l,s,j,t,s1ds2,t1dt2
      dimension vnn(8),vpw(2,2)

      call av8op(r,vnn)
      s1ds2=4*s-3
      t1dt2=4*t-3
      vc=vnn(1)+t1dt2*vnn(2)+s1ds2*vnn(3)+s1ds2*t1dt2*vnn(4)
```

```
      vt=vnn(5)+t1dt2*vnn(6)
      vls=vnn(7)+t1dt2*vnn(8)
      ncc=1
      if (s.eq.1.and.j.gt.l) ncc=2
      if (ncc.eq.1) then
        s12=0.
        if (s.eq.1.and.l.eq.j) s12=2.
        if (l.eq.(j+1)) s12=-2.*(j+2.)/(2.*j+1.)
        ls=(j*(j+1)-l*(l+1)-s*(s+1))/2
        vpw(1,1)=vc+s12*vt+ls*vls
        vpw(2,1)=0
        vpw(1,2)=0
        vpw(2,2)=0
      else if (ncc.eq.2) then
        s12m=-2.*(j-1.)/(2.*j+1.)
        s12=sqrt(36.*j*(j+1))/(2.*j+1.)
        s12p=-2.*(j+2.)/(2.*j+1.)
        lsm=j-1
        lsp=-(j+2)
        vpw(1,1)=vc+s12m*vt+lsm*vls
        vpw(2,1)=s12*vt
        vpw(1,2)=s12*vt
        vpw(2,2)=vc+s12p*vt+lsp*vls
      end if

      end
```

```
      subroutine av8op(r,vv)
      implicit real*8 (a-h,o-z)
      implicit integer*4 (i-n)
      dimension vv(8)

      yc(t)=exp(-t)/x
      yt(t)=(1+3/t+3/t**2)*exp(-t)/x

      do i=1,8
          vv(i)=0.D0
      enddo

      u=.7
      x=u*r
```

```
      y1=yc(x)
      y2=yc(2*x)
      y3=yc(3*x)
      y4=yc(4*x)
      y6=yc(6*x)
      y7=yc(7*x)
      yr=yt(x)-(12/x+3/x**2)*y4
      if (r.le.1e-5) yr=23.5/x
      hr=10.463
      vv(1)=          -19.874*y2+135.21*y3-1432.3*y4+4196.4*y6+1215.8*y7
      vv(2)=           19.874*y2-135.21*y3+319.52*y4-1082.3*y6 +405.3*y7
      vv(3)=           46.241*y2-135.21*y3 -64.78*y4+1398.8*y6-1215.8*y7
      vv(4)=(hr/3)*y1-46.241*y2+135.21*y3+244.06*y4-360.76*y6 -405.3*y7
      vv(5)=                    -26.194*y3+87.943*y4-418.38*y6
      vv(6)=(hr/3)*yr            -8.731*y3-87.943*y4+418.38*y6
      vv(7)=                               177.23*y4-2233.9*y6
      vv(8)=                              -177.23*y4+159.75*y6

      end
```

```
      subroutine argonp
c
c**** The ARGONNE V18 NEUTRON-PROTON potential in momentum space
c
c**** this is the argonne v18 code that works; use it.    5/25/95
c
c**** NOTE: before this code is called, q(97),n1  of Common block /cpts/
c****       have to be defined;
c****       q(i) are all the momenta for which the potential
c****       may be called in the course of the calculation;
c****       the maximum number of momenta to be considered
c****       is n1 = 50;
c****       when calling the code, ix and iy have to be defined
c****       (besides, of course, xmev and ymev and j);
c****       if xmev=q(i) andc****          ymev=q(k) then
```

```
c****          ix=i and
c****          iy=k.
c****       it is assumed that the points q(97) change only if
c****       the point q(n1) changes.
c
c**** interface routine to adjust the ARGONNE POTENTIAL CODE by Wiringa
c**** (see attached) to the Bonn application routines.
c
c     May 25, 1995
c
c
c
      implicit real*8 (a-h,o-z)
      common /crdwrt/ kread,kwrite,kpunch,kda(9)
      common/cpot/ v(6),xmev,ymev
      common/cstate/ j,heform,sing,trip,coup,endep,label
      common /cpts/   q(97),c,n1,ix,iy
      logical heform,sing,trip,coup,endep
      dimension vv(6)
      dimension vl(4),adminv(4,4),ldminv(4),mdminv(4)
      dimension xkp(97),vkk(97,97,25)
      character*4 name(3),nname(15)
      data pi/3.141592653589793d0/
      logical index
      data index/.false./
      data jj/-1/
      data qq0mev/-1.d0/
      data nn1/-1/
      data uf/197.32705d0/
c
c
      if (index) go to 50
      index=.true.
c
c
c        write Argonne potential
c
      write (kwrite,10000)
10000 format (///' Argonne V18 neutron-proton potential (1995)'/
     1          ' ------------------------------------------'///)
c      label='arnp'
c
c
      endep=.false.
c
c
      wp=938.27231d0
      wn=939.56563d0
c
      wnp=2.d0*wp*wn/(wp+wn)
c
      fa=uf/wnp/2.d0
c
c
   50 if (j.eq.0.or.j.eq.jj) go to 70
      jj=j
c
c
      if (j.ge.5) write (kwrite,19001)
19001 format (///' warning. argonne potential code does not provide'/
     1' a potential for j greater 4, except for 3G5, EP5, and 3I5.'/
     2' the potential is set to zero where no potential is provided.'/
     3' execution continued.'///)
c
c
c
```

```
      aj=dfloat(j)
      aj1=dfloat(j+1)
      a2j1=dfloat(2*j+1)
      aaj6=dsqrt(aj*aj1)
c
c        coefficient matrix for the translations into lsj formalism
c
      adminv(1,1)=aj1
      adminv(1,2)=aj
      adminv(1,3)=-aaj6
      adminv(1,4)=-aaj6
      adminv(2,1)=aj
      adminv(2,2)=aj1
      adminv(2,3)=aaj6
      adminv(2,4)=aaj6
      adminv(3,1)=aaj6
      adminv(3,2)=-aaj6
      adminv(3,3)=aj1
      adminv(3,4)=-aj
      adminv(4,1)=aaj6
      adminv(4,2)=-aaj6
      adminv(4,3)=-aj
      adminv(4,4)=aj1
c
c       inversion
c
      call dminv (adminv,4,deter,ldminv,mdminv)
c
c
c
c
   70 if (q(n1).eq.qq0mev.and.n1.eq.nn1) go to 90
      qq0mev=q(n1)
      nn1=n1
c
      nkp=n1
      do 71 i=1,n1
   71 xkp(i)=q(i)
c
c
c        call argonne potential
c
c

      call av18npk (nkp,xkp,vkk)
c
c
c
c
   90 do 95 iv=1,6
   95 vv(iv)=0.d0
      if (xmev.gt.4000..or.ymev.gt.4000.) go to 2000

c
c
      if (j.gt.5) go to 2000
c
c
      j1=j+1
      go to (100,110,120,130,140,150),j1
c
c
c        j = 0
c
  100 vv(1)=vkk(ix,iy,1)
      vv(2)=0.d0
```

```
      vv(3)=vkk(ix,iy,6)
      vv(4)=0.d0
      vv(5)=0.d0
      vv(6)=0.d0
      go to 2000
c
c
c        j = 1
c
  110 vv(1)=vkk(ix,iy,5)
      vv(2)=vkk(ix,iy,7)
      vv(3)=vkk(ix,iy,4)
      vv(4)=vkk(ix,iy,2)
      vv(5)=vkk(ix,iy,3)
      vv(6)=vkk(iy,ix,3)
      go to 1000
c
c
c        j = 2
c
  120 vv(1)=vkk(ix,iy,11)
      vv(2)=vkk(ix,iy,12)
      vv(3)=vkk(ix,iy,10)
      vv(4)=vkk(ix,iy,8)
      vv(5)=vkk(ix,iy,9)
      vv(6)=vkk(iy,ix,9)
      go to 1000
c
c
c        j = 3
c
  130 vv(1)=vkk(ix,iy,16)
      vv(2)=vkk(ix,iy,17)
      vv(3)=vkk(ix,iy,15)
      vv(4)=vkk(ix,iy,13)
      vv(5)=vkk(ix,iy,14)
      vv(6)=vkk(iy,ix,14)
      go to 1000
c
c
c        j = 4
c
  140 vv(1)=vkk(ix,iy,21)
      vv(2)=vkk(ix,iy,22)
      vv(3)=vkk(ix,iy,20)
      vv(4)=vkk(ix,iy,18)
      vv(5)=vkk(ix,iy,19)
      vv(6)=vkk(iy,ix,19)
      go to 1000
c
c
c        j = 5
c
  150 vv(1)=0.d0
      vv(2)=0.d0
      vv(3)=vkk(ix,iy,25)
      vv(4)=vkk(ix,iy,23)
      vv(5)=vkk(ix,iy,24)
      vv(6)=vkk(iy,ix,24)
      go to 1000
c
c
c
c
 1000 if (.not.heform) go to 2000
c
```

```
c
c         translation into (combination of) helicity states
c
c
      do 1005 i=1,4
 1005 vl(i)=vv(i+2)
c
      do 1020 ii=1,4
      iii=ii+2
      vv(iii)=0.d0
c
      do 1015 i=1,4
 1015 vv(iii)=vv(iii)+adminv(ii,i)*vl(i)
 1020 vv(iii)=vv(iii)*a2j1
c
c
c
c
c
c         over-all factors
c
 2000 do 2005 iv=1,6
 2005 v(iv)=vv(iv)*fa/(xmev*ymev)
c
c
      return
      end
c
c
c**************** this is the argonne v18 np code as obtained from
c**************** Wiringa on Jan. 19, 1995.
c**************** nothing has been changed,
c****************     except, maybe, alpha=0, see 'c****   '
c
c
c *id* av18npk ************************************************************
c momentum space matrix elements of v18(np) potential
c using Bessel transforms
c ---------------------------------------------------------------------
c arguments for av18npk
c nkp: # of k values (up to 50)
c xkp(nkp): k values
c vkk(nkp,nkp,l): v(k,k',l)
c                 where l=partial wave # given by array channel
c ---------------------------------------------------------------------
      subroutine av18npk(nkp,xkp,vkk)
      implicit real*8 (a-h,o-z)
      implicit integer*4 (i-n)
      real*8 mpi,mpi0,mpic,mp,mn,mr,mup,mun,mu0,muc,mu
      dimension nrl(6),nru(6),v(50),xkp(97),bkp(7,97),vkk(97,97,25)
      character*3 channel(25)
      data channel/'1S0','3S1','EP1','3D1','1P1','3P0','3P1','3P2'
     &            ,'EP2','3F2','1D2','3D2','3D3','EP3','3G3','1F3'
     &            ,'3F3','3F4','EP4','3H4','1G4','3G4','3G5','EP5'
     &            ,'3I5'/
c -------------
c set constants
c -------------
      data ndub/6/,nrl/1,129,193,257,321,385/
     &          ,nru/128,192,256,320,384,448/
      pi=acos(-1.)
      hc=197.327053
      mp=938.27231
      mn=939.56563
      mr=mp*mn/(mp+mn)
      h2m=hc**2/(2*mr)
```

```
      h2mfac=4/(3*pi*h2m)

      wp=938.27231d0
      wn=939.56563d0
c
      wnp=2.d0*wp*wn/(wp+wn)
c
      fa_com=mr/hc

c
c**** watch it: alpha, maybe, set to zero
      alpha=1./137.035989
c**** alpha=0.d0
c
      beta=.0189
      b=4.27
      mup= 2.7928474
      mun=-1.9130427
      mpi0=134.9739
      mpic=139.5675
      mpi=(mpi0+2*mpic)/3
      mu0=mpi0/hc
      muc=mpic/hc
      mu=mpi/hc
      fsq=.075
      cpi=2.1
      rws=.5
      aiws=5.
      rt1=sqrt(8.)
      rt2=sqrt(216.)/5.
      rt3=sqrt(432.)/7.
      rt4=sqrt(720.)/9.
      rt5=sqrt(1080.)/11.
      t5th=2./5.
      e5th=8./5.
      f7th=4./7.
      t7th=10./7.
      s9th=6./9.
      tw9th=12./9.
      e11th=8./11.
      ft11th=14./11.
c --------------------
c zero matrix elements
c --------------------
      do 10 l=1,25
      do 10 kp=1,nkp
      do 10 k=1,nkp
        vkk(k,kp,l)=0
   10 continue
c ---------------------------------
c start integrations
c simpson's rule using doubling grid
c r=0->1 fm: dr=1/128 fm
c   1->2        1/64
c   2->4        1/32
c   4->8        1/16
c   8->16       1/8
c  16->32       1/4
c ---------------------------------
      r=0.
      dr=1./128.
      isimp=1
      do 2000 nd=1,ndub
        nr1=nrl(nd)
        nr2=nru(nd)
        do 1000 nr=nr1,nr2
```

```
          r=r+dr
          drx=(3+isimp)*dr*h2mfac
          if (nr.eq.nr2) drx=1.5*drx
c --------------------
c calculate potentials
c --------------------
c electromagnetic terms
c ----------------------
          br=b*r
          fnpr=b**3*(15+15*br+6*br**2+br**3)*exp(-br)/384
          fdelta=b**3*(1+br+br**2/3)*exp(-br)/16
          ftr3=(1-(1+br+br**2/2+br**3/6+br**4/24+br**5/144)*exp(-br))
     &         /r**3
          flsr3=(1-(1+br+br**2/2+7*br**3/48+br**4/48)*exp(-br))/r**3
          vc1np=alpha*hc*beta*fnpr
          vmmnps=-alpha*hc**3*mup*mun*fdelta/(6*mn*mp)
          vmmnpt=-alpha*hc**3*mup*mun*ftr3/(4*mp*mn)
          vmmnpls=-alpha*hc**3*mun*flsr3/(2*mn*mr)
c ------------------
c strong interaction
c ------------------
          x=mu*r
          x0=mu0*r
          xc=muc*r
          rcut=1-exp(-cpi*r**2)
          ypi=rcut*exp(-x)/x
          tpi=(1+3/x+3/x**2)*ypi*rcut
          tpi2=tpi*tpi
          ypi0=fsq*(mpi0/mpic)**2*(mpi0/3)*rcut*exp(-x0)/x0
          ypic=fsq*(mpic/3)*rcut*exp(-xc)/xc
          tpi0=(1+(3+3/x0)/x0)*ypi0*rcut
          tpic=(1+(3+3/xc)/xc)*ypic*rcut
          ws=1/(1+exp((r-rws)*aiws))
          ws0=1/(1+exp(-rws*aiws))
          wsp=ws*(1+aiws*exp(-rws*aiws)*ws0*r)
          wsx=ws*x
          wsx2=wsx*x
          dypi00=(mpi0/mpic)**2*(mpi0/3)*cpi/mu0
          dypic0=(mpic/3)*cpi/muc
          ypi0p=ypi0-fsq*dypi00*ws*r/ws0
          ypicp=ypic-fsq*dypic0*ws*r/ws0


c -------
c s=0,t=1
c -------
          vc= -10.66788*tpi2+3126.5542*wsp-3*(-ypi0p+2*ypicp)
     &         +vc1np-3*vmmnps
          vl2=   .12472*tpi2  +16.7780*wsp
          v(1)=vc
          v(11)=vc+6*vl2
          v(21)=vc+20*vl2
c -------
c s=1,t=0
c -------
          vc=  -8.62770*tpi2+2605.2682*wsp +441.9733*wsx2-ypi0p-2*ypicp
     &         +vc1np+vmmnps
          vt=  1.485601*tpi2-1126.8359*wsx +370.1324*wsx2-tpi0-2*tpic
     &         +vmmnpt
          vls=   .10180*tpi2  +86.0658*wsp -356.5175*wsx2
     &         +vmmnpls
          vl2=  -.13201*tpi2 +253.4350*wsp   -1.0076*wsx2
          vls2=  .07357*tpi2 -217.5791*wsp  +18.3935*wsx2
          v(2)=vc
          v(3)=rt1*vt
          v(4)=vc-2*vt-3*vls+6*vl2+9*vls2
```

```
          v(12)=vc+2*vt-vls+6*vl2+vls2
          v(13)=vc-f7th*vt+2*vls+6*vl2+4*vls2
          v(14)=rt3*vt
          v(15)=vc-t7th*vt-5*vls+20*vl2+25*vls2
          v(22)=vc+2*vt-vls+20*vl2+vls2
          v(23)=vc-e11th*vt+4*vls+20*vl2+16*vls2
          v(24)=rt5*vt
          v(25)=vc-ft11th*vt-7*vls+42*vl2+49*vls2
c -------
c s=0,t=0
c -------
          vc=  -2.09971*tpi2+1204.4301*wsp-3*(-ypi0p-2*ypicp)
     &         +vc1np-3*vmmnps
          vl2=  -.31452*tpi2 +217.4559*wsp
          v(5)=vc+2*vl2
          v(16)=vc+12*vl2
c -------
c s=1,t=1
c -------
          vc=  -7.62701*tpi2+1813.5315*wsp+1847.8059*wsx2-ypi0p+2*ypicp
     &         +vc1np+vmmnps
          vt=   1.07985*tpi2 -190.0949*wsx -811.2040*wsx2-tpi0+2*tpic
     &         +vmmnpt
          vls=  -.62697*tpi2 -570.5571*wsp +819.1222*wsx2
     &         +vmmnpls
          vl2=   .06709*tpi2 +342.0669*wsp -615.2339*wsx2
          vls2=  .74129*tpi2   +9.3418*wsp -376.4384*wsx2
          v(6)=vc-4*vt-2*vls+2*vl2+4*vls2
          v(7)=vc+2*vt-vls+2*vl2+vls2
          v(8)=vc-t5th*vt+vls+2*vl2+vls2
          v(9)=rt2*vt
          v(10)=vc-e5th*vt-4*vls+12*vl2+16*vls2
          v(17)=vc+2*vt-vls+12*vl2+vls2
          v(18)=vc-s9th*vt+3*vls+12*vl2+9*vls2
          v(19)=rt4*vt
          v(20)=vc-tw9th*vt-6*vls+30*vl2+36*vls2

c --------------------------
c calculate bessel functions
c --------------------------
          do 100 k=1,nkp
            x=xkp(k)*r
            xi=1./x
            bkp(1,k)=sin(x)
            bkp(2,k)=bkp(1,k)*xi-cos(x)
            bkp(3,k)=3*bkp(2,k)*xi-bkp(1,k)
            bkp(4,k)=5*bkp(3,k)*xi-bkp(2,k)
            bkp(5,k)=7*bkp(4,k)*xi-bkp(3,k)
            bkp(6,k)=9*bkp(5,k)*xi-bkp(4,k)
            bkp(7,k)=11*bkp(6,k)*xi-bkp(5,k)
  100     continue
c --------------------
c sum matrix elements
c --------------------
          do 200 kp=1,nkp
          do 200 k=kp,nkp
            vkk(k,kp,1)=vkk(k,kp,1)+v(1)*bkp(1,k)*bkp(1,kp)*drx
            vkk(k,kp,2)=vkk(k,kp,2)+v(2)*bkp(1,k)*bkp(1,kp)*drx
            vkk(k,kp,3)=vkk(k,kp,3)+v(3)*bkp(3,k)*bkp(1,kp)*drx
            vkk(k,kp,4)=vkk(k,kp,4)+v(4)*bkp(3,k)*bkp(3,kp)*drx
            vkk(k,kp,5)=vkk(k,kp,5)+v(5)*bkp(2,k)*bkp(2,kp)*drx
            vkk(k,kp,6)=vkk(k,kp,6)+v(6)*bkp(2,k)*bkp(2,kp)*drx
            vkk(k,kp,7)=vkk(k,kp,7)+v(7)*bkp(2,k)*bkp(2,kp)*drx
            vkk(k,kp,8)=vkk(k,kp,8)+v(8)*bkp(2,k)*bkp(2,kp)*drx
            vkk(k,kp,9)=vkk(k,kp,9)+v(9)*bkp(4,k)*bkp(2,kp)*drx
            vkk(k,kp,10)=vkk(k,kp,10)+v(10)*bkp(4,k)*bkp(4,kp)*drx
```

```
            vkk(k,kp,11)=vkk(k,kp,11)+v(11)*bkp(3,k)*bkp(3,kp)*drx
            vkk(k,kp,12)=vkk(k,kp,12)+v(12)*bkp(3,k)*bkp(3,kp)*drx
            vkk(k,kp,13)=vkk(k,kp,13)+v(13)*bkp(3,k)*bkp(3,kp)*drx
            vkk(k,kp,14)=vkk(k,kp,14)+v(14)*bkp(5,k)*bkp(3,kp)*drx
            vkk(k,kp,15)=vkk(k,kp,15)+v(15)*bkp(5,k)*bkp(5,kp)*drx
            vkk(k,kp,16)=vkk(k,kp,16)+v(16)*bkp(4,k)*bkp(4,kp)*drx
            vkk(k,kp,17)=vkk(k,kp,17)+v(17)*bkp(4,k)*bkp(4,kp)*drx
            vkk(k,kp,18)=vkk(k,kp,18)+v(18)*bkp(4,k)*bkp(4,kp)*drx
            vkk(k,kp,19)=vkk(k,kp,19)+v(19)*bkp(6,k)*bkp(4,kp)*drx
            vkk(k,kp,20)=vkk(k,kp,20)+v(20)*bkp(6,k)*bkp(6,kp)*drx
            vkk(k,kp,21)=vkk(k,kp,21)+v(21)*bkp(5,k)*bkp(5,kp)*drx
            vkk(k,kp,22)=vkk(k,kp,22)+v(22)*bkp(5,k)*bkp(5,kp)*drx
            vkk(k,kp,23)=vkk(k,kp,23)+v(23)*bkp(5,k)*bkp(5,kp)*drx
            vkk(k,kp,24)=vkk(k,kp,24)+v(24)*bkp(7,k)*bkp(5,kp)*drx
            vkk(k,kp,25)=vkk(k,kp,25)+v(25)*bkp(7,k)*bkp(7,kp)*drx
            if (k.eq.kp) go to 200
            vkk(kp,k,3)=vkk(kp,k,3)+v(3)*bkp(3,kp)*bkp(1,k)*drx
            vkk(kp,k,9)=vkk(kp,k,9)+v(9)*bkp(4,kp)*bkp(2,k)*drx
            vkk(kp,k,14)=vkk(kp,k,14)+v(14)*bkp(5,kp)*bkp(3,k)*drx
            vkk(kp,k,19)=vkk(kp,k,19)+v(19)*bkp(6,kp)*bkp(4,k)*drx
            vkk(kp,k,24)=vkk(kp,k,24)+v(24)*bkp(7,kp)*bkp(5,k)*drx
  200     continue
c ------------------------------
c set remaining matrix elements
c ------------------------------
          do 210 kp=2,nkp
          kpm=kp-1
          do 210 k=1,kpm
          do 210 l=1,25
            if (l.eq.3.or.l.eq.9.or.l.eq.14.or.l.eq.19.or.l.eq.24)
     &         go to 210
            vkk(k,kp,l)=vkk(kp,k,l)
  210     continue
          isimp=-isimp
 1000   continue
c --------------------
c double the grid size
c --------------------
        dr=2*dr
 2000 continue
      return
      end
```

```
      subroutine reid93
c
c**** The REID93 POTENTIAL in momentum space
c**** as constructed by the Nijmegen group.
c
c**** interface routine to adjust the REID93P POTENTIAL CODE
c**** (see attached) to the Bonn application routines.
c
c     R. Machleidt;
c     November 12, 1994
c
c
c
      implicit real*8 (a-h,o-z)
      common /crdwrt/ kread,kwrite,kpunch,kda(9)
      common/cpot/ v(6),xmev,ymev
      common/cstate/ j,heform,sing,trip,coup,endep,label
      COMMON/EMANHP/PHNAME
      logical heform,sing,trip,coup,endep
      CHARACTER PHNAME*3, TYPE*2
      REAL*8 VPOT(2,2)
      dimension vv(6)
      dimension vl(4),adminv(4,4),ldminv(4),mdminv(4)
      character*4 name(3),nname(15)
      data pi/3.141592653589793d0/
      data uf/197.327053d0/
      logical index
      data index/.false./
      data jj/-1/
c
c


      kread=5
      KWRITE=6


      if (index) go to 50
      index=.true.
c
c
c        read in parameters for reid93 potential
c
      write (kwrite,10000)
10000 format (///' Reid93 Potential'/
     1           ' ----------------')
      read (kread,10001) nname
10001 format (15a4)
      write (kwrite,10002) nname
10002 format (' ',15a4)
c        TYPE = 'PP', 'NN', 'NP', or 'PN'.
      read (kread,10005) name,TYPE
```

```
10005 format (2a4,a2,a2)
      write (kwrite,10006) name,TYPE
10006 format (' ',2a4,a2,a2)
      read (kread,10009) name,label
10009 format (2a4,a2,a4)
      write (kwrite,10010) name,label
10010 format (' ',2a4,a2,a4///)
c
c
      endep=.false.
      fa=1./(2.d0*pi*pi)
c
c
   50 if (j.eq.0.or.j.eq.jj) go to 90
      jj=j
c
c
      if (j.gt.9) write (kwrite,19001)
19001 format (///' warning. the reid93 potential is not'/
     1' defined for j greater 9.'/
     2' the potential is set to zero.'/
     3' execution continued.'///)
c
c
c
      aj=dfloat(j)
      aj1=dfloat(j+1)
      a2j1=dfloat(2*j+1)
      aaj6=dsqrt(aj*aj1)
c
c        coefficient matrix for the translations into lsj formalism
c
      adminv(1,1)=aj1
      adminv(1,2)=aj
      adminv(1,3)=-aaj6
      adminv(1,4)=-aaj6
      adminv(2,1)=aj
      adminv(2,2)=aj1
      adminv(2,3)=aaj6
      adminv(2,4)=aaj6
      adminv(3,1)=aaj6
      adminv(3,2)=-aaj6
      adminv(3,3)=aj1
      adminv(3,4)=-aj
      adminv(4,1)=aaj6
      adminv(4,2)=-aaj6
      adminv(4,3)=-aj
      adminv(4,4)=aj1
c
c       inversion
c
      call dminv (adminv,4,deter,ldminv,mdminv)
c
c
c
c
   90 do 95 iv=1,6
   95 vv(iv)=0.d0
c
c
      if (j.gt.9) go to 2000
c
c
      j1=j+1
      do 295 i=1,3
      if (i.eq.1.and..not.sing) go to 295
```

```
      if (i.eq.2.and..not.trip) go to 295
      if (i.eq.3.and..not.coup) go to 295
c
c
      go to (100,110,120,130,140,150,160,170,180,190),j1
c
c
c        j = 0
c
  100 go to (101,102,295),i
  101 phname='1S0'
      go to 200
  102 phname='3P0'
      go to 200
c
c
c        j = 1
c
  110 go to (111,112,113),i
  111 phname='1P1'
      go to 200
  112 phname='3P1'
      go to 200
  113 phname='3C1'
      go to 200
c
c
c        j = 2
c
  120 go to (121,122,123),i
  121 phname='1D2'
      go to 200
  122 phname='3D2'
      go to 200
  123 phname='3C2'
      go to 200
c
c
c        j = 3
c
  130 go to (131,132,133),i
  131 phname='1F3'
      go to 200
  132 phname='3F3'
      go to 200
  133 phname='3C3'
      go to 200
c
c
c        j = 4
c
  140 go to (141,142,143),i
  141 phname='1G4'
      go to 200
  142 phname='3G4'
      go to 200
  143 phname='3C4'
      go to 200
c
c
c        j = 5
c
  150 go to (151,152,153),i
  151 phname='1H5'
      go to 200
  152 phname='3H5'
```

```
      go to 200
  153 phname='3C5'
      go to 200
c
c
c        j = 6
c
  160 go to (161,162,163),i
  161 phname='1I6'
      go to 200
  162 phname='3I6'
      go to 200
  163 phname='3C6'
      go to 200
c
c
c        j = 7
c
  170 go to (171,172,173),i
  171 phname='1J7'
      go to 200
  172 phname='3J7'
      go to 200
  173 phname='3C7'
      go to 200
c
c
c        j = 8
c
  180 go to (181,182,183),i
  181 phname='1K8'
      go to 200
  182 phname='3K8'
      go to 200
  183 phname='3C8'
      go to 200
c
c
c        j = 9
c
  190 go to (191,192,193),i
  191 phname='1L9'
      go to 200
  192 phname='3L9'
      go to 200
  193 phname='3C9'
      go to 200
c
c
c
c
  200 call reid93p (ymev,xmev,type,vpot)
c
c
c
c
      go to (201,202,203),i
  201 vv(1)=vpot(1,1)
      go to 295
  202 vv(2)=vpot(1,1)
      go to 295
  203 vv(3)=vpot(2,2)
      vv(4)=vpot(1,1)
      vv(5)=vpot(2,1)
      vv(6)=vpot(1,2)
  295 continue
```

```
c
c
      if (j.ne.0) go to 1000
      vv(3)=vv(2)
      vv(2)=0.d0
      go to 2000
c
c
c
 1000 if (.not.heform) go to 2000
c
c
c         translation into (combination of) helicity states
c
c
      do 1005 i=1,4
 1005 vl(i)=vv(i+2)
c
      do 1020 ii=1,4
      iii=ii+2
      vv(iii)=0.d0
c
      do 1015 i=1,4
 1015 vv(iii)=vv(iii)+adminv(ii,i)*vl(i)
 1020 vv(iii)=vv(iii)*a2j1
c
c
c
c
c
c         over-all factors
c
 2000 do 2005 iv=1,6
 2005 v(iv)=vv(iv)*fa
c
c
      return
      end
```

```
      SUBROUTINE REID93P(QI,QF,TYPE,VPOT)
************************************************************************
**    Version: June 1994                                              **
**    E-mail: thefalg@sci.kun.nl                                      **
**    Reference: Stoks et al. Phys.Rev. C49 (1994) June               **
**                                                                    **
**    Updated Reid potential, regularized with a dipole form factor   **
**    of 8 pion masses, in momentum space on LSJ basis.               **
**                                                                    **
**    INPUT :  QI    center of mass momentum initial state in MeV     **
**    -----    QF    center of mass momentum final   state in MeV     **
**             TYPE  'PP', 'NN', 'NP', or 'PN' (character*2)          **
**             Name partial wave via COMMON/EMANHP/PHNAME (see below) **
**             Maximum total angular momentum J=10 !!                 **
**                                                                    **
**    OUTPUT:  This subroutine returns a 2x2 potential matrix VPOT    **
**    ------   in MeV**-2 which is the partial-wave momentum-space    **
**             potential for the partial wave PHNAME (see below)      **
**--------------------------------------------------------------------**
**    Defining the K-matrix as   2i*mu*q*K = (1-S)(1+S)^-1            **
**    (so for singlet channel   tan(delta)=-2*mu*q*K )                **
**    the partial-wave Lippmann-Schwinger equation reads              **
**                                                                    **
**       K(q'q) = V(q'q) + 2/pi int dk k^2 V(q'k) G(q,k) K(kq)        **
**    with                                                            **
**       G(q,k) = P / (E(q) - k^2/2/mu)                               **
**       V(q'k) = 1 / (4*pi) * VPOT(QI=k,QF=q')                       **
**--------------------------------------------------------------------**
**    Potential decomposition in momentum space plane-wave basis:     **
**    V(QF,QI) = VC                                                   **
**             + VS   (SIG1.SIG2)        (only in one-pion-exchnage)  **
**             + VT   [(SIG1.K)(SIG2.K)-K2/3(SIG1.SIG2)]             **
**             + VLS  (i/2)(SIG1+SIG2).N                              **
**                                                                    **
**          K = QF - QI ,   Q = (QF+QI)/2 ,   N = QI X QF = Q X K     **
**                                                                    **
**    NOTE: In the partial wave decomposition we used the             **
**          SYM-convention.                                           **
**          If you use another convention in your Lippmann-Schwinger  **
**          programm, you may need an extra minus sign for the        **
**          the off-diagonal tensor potential VPOT(1,2) and VPOT(2,1) **
**                                                                    **
**    One-pion-exchange part distinguishes between neutral and charged**
**    pion masses, and has coupling constants F0PI=FCPI=0.075         **
**    The delta-function (smeared out due to the form factor) ONLY    **
**    contributes to the S waves.                                     **
**                                                                    **
**    COMMON-block which has to be filled beforehand:                 **
**    +  COMMON/EMANHP/PHNAME                                         **
**             PHNAME is character*3 and contains the name of the     **
**             partial wave in the spectral notation.                 **
**             - singlets:           1S0  1P1  1D2  1F3  1G4 ...     **
**             - triplets uncoupled: 3P0  3P1  3D2  3F3  3G4 ...     **
**             - triplets coupled:        3C1  3C2  3C3  3C4 ...     **
**               where 3C1 denotes  3S1 -- EPS1 -- 3D1 channel       **
**                     3C2 denotes  3P2 -- EPS2 -- 3F2 channel ...   **
**                                                                    **
************************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER SPIN
      CHARACTER TYPE*2, PHNAME*3, PHNAM0*3
      REAL*8 VPOT(2,2), ELN(0:12), UL(6,-2:12),ULC(-2:12)
      REAL*8 VPIS(-1:1),VPIT(-2:2), VC(-1:1),VL(-2:2),VT(-2:2)
      REAL*8 PARSPP(5,5),PARSNP(5,5), A(5,5),B(5,5)
      COMMON/EMANHP/PHNAME
      DATA F0PI/0.075D0/, FCPI/0.075D0/, PI/3.14159265358979D0/
```

```
      DATA PIOM,PIOMC,PIOMS/134.9739D0,139.5675D0,139.5675D0/
      DATA ICAL/0/, PHNAM0/'***'/, UL/90*0D0/, ULC/15*0D0/
      DATA PARSPP/
     1  .1756084D0,-.1414234D2, .1518489D3,-.6868230D3, .1104157D4
     2,-.4224976D2, .2072246D3,-.3354364D3,-.1989250D1,-.6178469D2
     3, .2912845D2, .1511690D3, .8151964D1, .5832103D2,-.2074743D2
     4,-.5840566D0,-.1029310D2, .2263391D2, .2316915D2,-.1959172D1
     5,-.2608488D1, .1090858D2,-.4374212D0,-.2148862D2,-.6584788D0/
      DATA PARSNP/
     1 -.2234989D2, .2551761D3,-.1063549D4, .1609196D4,-.3505968D1
     2,-.4248612D1,-.5352001D1, .1827642D3,-.3927086D3, .5812273D2
     3,-.2904577D1, .3802497D2, .3395927D0, .8318097D0, .1923895D1
     4, .0913746D0,-.1274773D2, .1458600D3,-.6432461D3, .1022217D4
     5,-.0461640D0, .7950192D1,-.1925573D1, .5066234D2, .83598955D1/
      SAVE A,B, PIOMS2,PIOMM2, NCHAN,L,SPIN,J,ISO
      SAVE JMM,JM,JP,JPP,TJMM,TJM,TJ,TJP,TJPP,TJJ,JMAX

      IF(ICAL.EQ.0) THEN
        DO 1 I1=1,5
          DO 1 I2=1,5
            A(I1,I2)=PARSPP(I2,I1)
     1      B(I1,I2)=PARSNP(I2,I1)
        PIOMS2=PIOMS*PIOMS
        PIOMM=(PIOM+2D0*PIOMC)/3D0
        PIOMM2=PIOMM*PIOMM
        ICAL=1
      ENDIF

      IF(PHNAME.NE.PHNAM0) THEN
        PHNAM0=PHNAME
        NCHAN=1
        IF(PHNAME(2:2).EQ.'C') NCHAN=2
        IF(PHNAME(1:1).EQ.'1') SPIN=0
        IF(PHNAME(1:1).EQ.'3') SPIN=1
        READ(PHNAME,'(2X,I1)') J
        IF(J.GT.10) WRITE(*,*)
     .              '**** Partial wave exceeds allowable maximum J=10'
        IF(J.GT.10) STOP
        L=J
        IF(PHNAME.EQ.'3P0') L=1
        IF(NCHAN.EQ.2) L=J-1
        ISO=MOD(SPIN+L+1,2)
        JMM=J-2
        JM =J-1
        JP =J+1
        JPP=J+2
        TJMM=2D0*J-3D0
        TJM =2D0*J-1D0
        TJ  =2D0*J+1D0
        TJP =2D0*J+3D0
        TJPP=2D0*J+5D0
        TJJ =DSQRT(J*(J+1D0))
        JMAX=J+2
      ENDIF

      QI2=QI*QI
      QF2=QF*QF
      QIQF=QI*QF
      QI2F2=QI2+QF2
      S2PSI=2D0*QIQF/QI2F2
      SPSI2=QF2/QI2F2
      CPSI2=QI2/QI2F2

***   Neutral one-pion-exchange part
      AMES2=PIOM*PIOM
      ALAM2=64D0*AMES2
```

```
      X=0.5D0*(QI2F2+AMES2)/QIQF
      Y=0.5D0*(QI2F2+ALAM2)/QIQF
      CALL SDIP(X,Y,JMAX,ELN)
      DO 10 IE=0,JMAX
   10   UL(1,IE)=ELN(IE)
      FAC=2D0*PI*F0PI/PIOMS2/QIQF
      DO 11 LL=-1,1
   11   VPIS(LL)= FAC*UL(1,J+LL)*PIOM*PIOM/3D0
      VPISL0=4D0*PI*F0PI/PIOMS2/3D0 * (Y-X)*(Y-X)/(1D0-Y*Y)
      DO 12 LL=-2,2
   12   VPIT(LL)=-FAC*UL(1,J+LL)

***   Charged one-pion-exchange part
      IF(TYPE.EQ.'NP' .OR. TYPE.EQ.'PN') THEN
        AMES2=PIOMC*PIOMC
        ALAM2=64D0*AMES2
        X=0.5D0*(QI2F2+AMES2)/QIQF
        Y=0.5D0*(QI2F2+ALAM2)/QIQF
        CALL SDIP(X,Y,JMAX,ELN)
        DO 20 IE=0,JMAX
   20     ULC(IE)=ELN(IE)
        FAC=(4D0*ISO-2D0)*2D0*PI*FCPI/PIOMS2/QIQF
        DO 21 LL=-1,1
   21     VPIS(LL)= FAC*ULC(J+LL)*PIOMC*PIOMC/3D0 - VPIS(LL)
        VPISL0=4D0*PI*FCPI/PIOMS2/3D0 * (4D0*ISO-2D0)
     .         *(Y-X)*(Y-X)/(1D0-Y*Y) - VPISL0
        DO 22 LL=-2,2
   22     VPIT(LL)=-FAC*ULC(J+LL) - VPIT(LL)
      ENDIF

***   Other Yukawa's with multiples of mean pion mass
      ALAM2=64D0*PIOMM2
      Y=0.5D0*(QI2F2+ALAM2)/QIQF
      DO 30 IM=2,6
        X=0.5D0*(QI2F2+IM*IM*PIOMM2)/QIQF
        CALL SDIP(X,Y,JMAX,ELN)
        DO 35 IE=0,JMAX
   35     UL(IM,IE)=ELN(IE)
   30 CONTINUE

      FAC=2D0*PI/QIQF

***   Potential for each partial wave separately
      IF(PHNAME.EQ.'1S0') THEN
        IF(TYPE.EQ.'PP' .OR. TYPE.EQ.'NN') THEN
          VPOT(1,1)=FAC*(A(1,1)*UL(2,J)+A(1,2)*UL(3,J)+A(1,3)*UL(4,J)
     .                  +A(1,4)*UL(5,J)+A(1,5)*UL(6,J) )
        ELSEIF(TYPE.EQ.'NP' .OR. TYPE.EQ.'PN') THEN
          VPOT(1,1)=FAC*(B(1,1)*UL(3,J)+B(1,2)*UL(4,J)+B(1,3)*UL(5,J)
     .                  +B(1,4)*UL(6,J) )
        ENDIF
      ELSEIF(PHNAME.EQ.'1D2') THEN
        VPOT(1,1)=FAC*(A(2,1)*UL(4,J)+A(2,2)*UL(5,J)+A(2,3)*UL(6,J) )
      ELSEIF(PHNAME.EQ.'1G4') THEN
        VPOT(1,1)=FAC* A(2,4)*UL(3,J)
      ELSEIF(PHNAME.EQ.'3P0') THEN
        VTEN=UL(3,JP)-0.5D0*S2PSI*(TJPP*UL(3,J)+TJ*UL(3,JPP))/TJP
        VPOT(1,1)=FAC*(A(3,1)*UL(3,JP)+A(3,2)*UL(5,JP)
     .               + A(2,5)*(-VTEN/9D0/PIOMM2)*QI2F2/3D0 )
      ELSEIF(PHNAME.EQ.'3P1') THEN
        VTEN=UL(3,J)-0.5D0*S2PSI*(TJP*UL(3,JM)+TJM*UL(3,JP))/TJ
        VPOT(1,1)=FAC*(A(3,3)*UL(3,J)+A(3,4)*UL(5,J)
     .               + A(3,5)*(-VTEN/9D0/PIOMM2)*QI2F2/3D0 )
      ELSEIF(PHNAME.EQ.'3F3') THEN
        VPOT(1,1)=FAC* A(4,5)*UL(3,J)
      ELSEIF(ISO.EQ.1 .AND. SPIN.EQ.1) THEN
```

```
      DO 101 LL=-1,1
101     VC(LL)=FAC*(A(4,1)*UL(3,J+LL)+A(4,2)*UL(4,J+LL)
     .             +A(4,3)*UL(5,J+LL)+A(4,4)*UL(6,J+LL) )
        DO 102 LL=-2,2
          VT(LL)=-FAC*(A(5,1)*UL(4,J+LL)/16D0/PIOMM2
     .               +A(5,2)*UL(6,J+LL)/36D0/PIOMM2 )
          IF(PHNAME.EQ.'3C2') THEN
            VL(LL)=FAC*(A(5,3)*UL(3,J+LL)/ 9D0/PIOMM2
     .                +A(5,4)*UL(5,J+LL)/25D0/PIOMM2 )
          ELSEIF(PHNAME.EQ.'3C4') THEN
            VL(LL)=FAC* A(5,5)*UL(3,J+LL)/ 9D0/PIOMM2
          ELSE
            VL(LL)=0D0
          ENDIF
102     CONTINUE
        IF(NCHAN.EQ.1)
     .    VPOT(1,1)=VC(0) - QIQF*(VL(-1)-VL(1))/TJ + 2D0/3D0*QI2F2*
     .             (VT(0)-0.5D0*S2PSI*(TJP*VT(-1)+TJM*VT(1))/TJ)
        ELSEIF(PHNAME.EQ.'1P1') THEN
          VPOT(1,1)=FAC*(B(2,1)*UL(3,J)+B(2,2)*UL(4,J)+B(2,3)*UL(5,J)
     .                +B(2,4)*UL(6,J) )
        ELSEIF(PHNAME.EQ.'1F3') THEN
          VPOT(1,1)=FAC*(B(1,5)*UL(3,J)+B(2,5)*UL(5,J) )
        ELSEIF(PHNAME.EQ.'3D2') THEN
          VTEN=UL(3,J)-0.5D0*S2PSI*(TJP*UL(3,JM)+TJM*UL(3,JP))/TJ
          VPOT(1,1)=FAC*(B(3,1)*UL(3,J)+B(3,2)*UL(5,J)
     .              + B(3,3)*(-VTEN/9D0/PIOMM2)*QI2F2/3D0 )
        ELSEIF(PHNAME.EQ.'3G4') THEN
          VPOT(1,1)=FAC* B(3,4)*UL(3,J)
        ELSEIF(ISO.EQ.0 .AND. SPIN.EQ.1) THEN
          DO 201 LL=-1,1
201       VC(LL)=FAC*(B(4,1)*UL(2,J+LL)+B(4,2)*UL(3,J+LL)
     .               +B(4,3)*UL(4,J+LL)+B(4,4)*UL(5,J+LL)
     .               +B(4,5)*UL(6,J+LL) )
          DO 202 LL=-2,2
            VT(LL)=-FAC*(B(3,5)*UL(4,J+LL)/16D0/PIOMM2
     .                +B(5,5)*UL(6,J+LL)/36D0/PIOMM2 )
            IF(PHNAME.EQ.'3C1') THEN
              VL(LL)=FAC*(B(5,1)*UL(3,J+LL)/ 9D0/PIOMM2
     .                  +B(5,2)*UL(5,J+LL)/25D0/PIOMM2 )
            ELSEIF(PHNAME.EQ.'3C3') THEN
              VL(LL)=FAC*(B(5,3)*UL(3,J+LL)/ 9D0/PIOMM2
     .                  +B(5,4)*UL(5,J+LL)/25D0/PIOMM2 )
            ELSE
              VL(LL)=0D0
            ENDIF
202       CONTINUE
          IF(NCHAN.EQ.1)
     .      VPOT(1,1)=VC(0) - QIQF*(VL(-1)-VL(1))/TJ + 2D0/3D0*QI2F2*
     .               (VT(0)-0.5D0*S2PSI*(TJP*VT(-1)+TJM*VT(1))/TJ)
        ELSEIF(J.GE.5 .AND. SPIN.EQ.0) THEN
          IF(ISO.EQ.1) THEN
            VPOT(1,1)=FAC*(A(1,1)*UL(2,J)+A(1,2)*UL(3,J)+A(1,3)*UL(4,J)
     .                  +A(1,4)*UL(5,J)+A(1,5)*UL(6,J) )
          ELSEIF(ISO.EQ.0) THEN
            VPOT(1,1)=FAC*(B(2,1)*UL(3,J)+B(2,2)*UL(4,J)+B(2,3)*UL(5,J)
     .                  +B(2,4)*UL(6,J) )
          ENDIF
        ENDIF

        IF(NCHAN.EQ.2) THEN
          VPOT(1,1)=VC(-1) + QIQF*JM/TJM*(VL(-2)-VL(0)) +
     .              2D0/3D0*QI2F2*JM/TJ*
     .              (-VT(-1)+0.5D0*S2PSI*(TJMM*VT(0)+TJ*VT(-2))/TJM)
          VPOT(1,2)=-2D0*QI2F2*TJJ/TJ*
     .              (-S2PSI*VT(0)+CPSI2*VT(-1)+SPSI2*VT(1))
```

```
        VPOT(2,1)=-2D0*QI2F2*TJJ/TJ*
     .            (-S2PSI*VT(0)+SPSI2*VT(-1)+CPSI2*VT(1))
        VPOT(2,2)=VC(1) - QIQF*JPP/TJP*(VL(0)-VL(2)) +
     .            2D0/3D0*QI2F2*JPP/TJ*
     .            (-VT(1)+0.5D0*S2PSI*(TJPP*VT(0)+TJ*VT(2))/TJP)
      ENDIF

***   Add one-pion-exchange part
      IF(NCHAN.EQ.1) THEN
        IF(SPIN.EQ.0) THEN
          VPOT(1,1) = VPOT(1,1) - 3D0*VPIS(0)
          IF(L.EQ.0) VPOT(1,1) = VPOT(1,1) - 3D0*VPISL0
        ELSEIF(L.EQ.J) THEN
          VPOT(1,1) = VPOT(1,1) + VPIS(0) + 2D0/3D0*QI2F2*
     .         (VPIT(0)-0.5D0*S2PSI*(TJP*VPIT(-1)+TJM*VPIT(1))/TJ)
        ELSEIF(PHNAME.EQ.'3P0') THEN
          VPOT(1,1) = VPOT(1,1) + VPIS(1) + 2D0/3D0*QI2F2*JPP/TJ*
     .         (-VPIT(1)+0.5D0*S2PSI*(TJPP*VPIT(0)+TJ*VPIT(2))/TJP)
        ENDIF
      ELSE
        VPOT(1,1) = VPOT(1,1) + VPIS(-1) + 2D0/3D0*QI2F2*JM/TJ*
     .       (-VPIT(-1)+0.5D0*S2PSI*(TJMM*VPIT(0)+TJ*VPIT(-2))/TJM)
        IF(L.EQ.0) VPOT(1,1) = VPOT(1,1) + VPISL0
        VPOT(1,2) = VPOT(1,2) - 2D0*QI2F2*TJJ/TJ*
     .       (-S2PSI*VPIT(0)+CPSI2*VPIT(-1)+SPSI2*VPIT(1))
        VPOT(2,1) = VPOT(2,1) - 2D0*QI2F2*TJJ/TJ*
     .       (-S2PSI*VPIT(0)+SPSI2*VPIT(-1)+CPSI2*VPIT(1))
        VPOT(2,2) = VPOT(2,2) + VPIS(1) + 2D0/3D0*QI2F2*JPP/TJ*
     .       (-VPIT(1)+0.5D0*S2PSI*(TJPP*VPIT(0)+TJ*VPIT(2))/TJP)
      ENDIF

      RETURN
      END
*****************************************************************
      SUBROUTINE SDIP(X,Y,JMAX,ELN)
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NOGP=64)
      DIMENSION P(12), ELN(0:12), ZZ(NOGP), WZ(NOGP)
      DATA ICAL/0/
      SAVE ZZ,WZ

      QX0 = 0.5D0*DLOG((X+1D0)/(X-1D0))
      QY0 = 0.5D0*DLOG((Y+1D0)/(Y-1D0))
      ELN(0)=QX0-QY0+(Y-X)/(1D0-Y*Y)
      IF(JMAX.EQ.0) RETURN
      IF(ICAL.EQ.0) THEN
        CALL GAUS(NOGP/2,ZZ,WZ,2)
        ICAL=1
      ENDIF
      DO 1 L=1,JMAX
    1   ELN(L)=0D0
      DO 2 IZ=1,NOGP
        FORM=(Y-X)/(Y-ZZ(IZ))
        FORM=FORM*FORM/(X-ZZ(IZ))
C*      Calculate Legendre polynomials first kind and integrate
        P(1)=ZZ(IZ)
        P(2)=1.5D0*ZZ(IZ)*ZZ(IZ)-0.5D0
        ELN(1)=ELN(1)+0.5D0*WZ(IZ)*FORM*P(1)
        ELN(2)=ELN(2)+0.5D0*WZ(IZ)*FORM*P(2)
        DO 3 L=3,JMAX
          P(L)=( (2*L-1)*ZZ(IZ)*P(L-1) - (L-1)*P(L-2) ) / L
          ELN(L)=ELN(L)+0.5D0*WZ(IZ)*FORM*P(L)
    3   CONTINUE
    2 CONTINUE
      RETURN
      END
```

```
**********************************************************************
      SUBROUTINE GAUS(N,X,W,NT)
C*      The abcissas and weights for Gaussian integration,
C*      N=4,8,12,16,20,32,48 only:
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION WGAUSS(61), WEULRA(32),WEULRB(48)
      DIMENSION X(1),W(1)
      DATA WGAUSS
     1/.33998104358486,.65214515486255,.86113631159405,.34785484513745
     *,.18343464249565,.36268378337836,.52553240991633,.31370664587789
     *,.79666647741362,.22238103445337,.96028985649753,.10122853629038
     *,.12523340851147,.24914704581340,.36783149899818,.23349253653835
     *,.58731795428661,.20316742672307,.76990267419430,.16007832854335
     *,.90411725637048,.10693932599532,.98156063424672,.04717533638651
     *,.09501250983764,.18945061045507,.28160355077926,.18260341504492
     *,.45801677765722,.16915651939500,.61787624440264,.14959598881658
     *,.75540440835500,.12462897125553,.86563120238783,.09515851168249
     *,.94457502307323,.06225352393865,.98940093499165,.02715245941175
     *,.07652652113350,.15275338713072,.22778585114164,.14917298647260
     *,.37370608871542,.14209610931838,.51086700195082,.13168863844918
     *,.63605368072652,.11819453196152,.74633190646015,.10193011981724
     *,.83911697182222,.08327674157670,.91223442825132,.06267204833411
     *,.96397192727791,.04060142980039,.99312859918509,.01761400713915
     *,.00/

C  32 POINTS:
      DATA WEULRA
     1/.04830766568774,.09654008851473,.14447196158280,.09563872007927
     *,.23928736225214,.09384439908080,.33186860228213,.09117387869576
     *,.42135127613064,.08765209300440,.50689990893223,.08331192422695
     *,.58771575724076,.07819389578707,.66304426693022,.07234579410885
     *,.73218211874029,.06582222277636,.79448379596794,.05868409347854
     *,.84936761373257,.05099805926238,.89632115576605,.04283589802223
     *,.93490607593774,.03427386291302,.96476225558751,.02539206530926
     *,.98561151154527,.01627439473091,.99726386184948,.00701861000947
     */

C  48 POINTS:
      DATA WEULRB
     1/.03238017096287,.06473769681268,.09700469920946,.06446616443595
     *,.16122235606889,.06392423858465,.22476379039469,.06311419228625
     *,.28736248735546,.06203942315989,.34875588629216,.06070443916589
     *,.40868648199071,.05911483969840,.46690290475096,.05727729210040
     *,.52316097472223,.05519950369998,.57722472608397,.05289018948519
     *,.62886739677651,.05035903555385,.67787237963266,.04761665849249
     *,.72403413092381,.04467456085669,.76715903251574,.04154508294346
     *,.80706620402944,.03824135106583,.84358826162439,.03477722256477
     *,.87657202027425,.03116722783280,.90587913671557,.02742650970836
     *,.93138669070655,.02357076083932,.95298770316043,.01961616045736
     *,.97059159254625,.01557931572294,.98412458372283,.01147723457923
     *,.99353017226635,.00732755390128,.99877100725243,.00315334605231
     */

      IF(MOD(N,4).NE.0) THEN
C*      Simpson's rule: only odd N
        H=2D0/(N+1D0)
        X(1)=-1D0+H
        W23=2D0*H/3D0
        W43=2D0*W23
        W(1)=W43
        NN=(N-1)/2
        DO 10 I=1,NN
          J=2*I
          X(J)=X(J-1)+H
          X(J+1)=X(J)+H
          W(J)=W23
```

```
        W(J+1)=W43
 10   CONTINUE
      RETURN
    ENDIF

    IF(N.LE.20) THEN
      K=N/4-1
      INIT=2*K*(K+1)+1
      M=N/2
      DO 1 I=1,M
        J=INIT-2+I*2
        X(M+I)=WGAUSS(J)
        X(M+1-I)=-WGAUSS(J)
        W(M+I)=WGAUSS(J+1)
        W(M+1-I)=WGAUSS(J+1)
  1   CONTINUE
    ENDIF
    IF(N.EQ.32) THEN
      M=N/2
      DO 2 I=1,M
        J=2*I-1
        X(M+I)=WEULRA(J)
        X(M+1-I)=-WEULRA(J)
        W(M+I)=WEULRA(J+1)
        W(M+1-I)=WEULRA(J+1)
  2   CONTINUE
    ENDIF
    IF(N.EQ.48) THEN
      M=N/2
      DO 3 I=1,M
        J=2*I-1
        X(M+I)=WEULRB(J)
        X(M+1-I)=-WEULRB(J)
        W(M+I)=WEULRB(J+1)
        W(M+1-I)=WEULRB(J+1)
  3   CONTINUE
    ENDIF
    IF(NT.EQ.1) RETURN

    DO 5 I=1,N
      X(N+I)=(X(I)+1D0)/2D0
      X(I)=(X(I)-1D0)/2D0
      W(N+I)=W(I)/2D0
      W(I)=W(N+I)
  5 CONTINUE

    RETURN
    END
```

```
      subroutine nijm2
c
c**** 4/8/97
c**** UPDATED VERSION (see NOTE below) of
c**** The NIJMEGEN POTENTIALS in momentum space
c
c**** interface routine to adjust the NIJMEGEN POTENTIAL CODE
c**** (see attached) to the Bonn application routines.
c
c**** this package is selfcontained.
c
c     R. Machleidt;
c     April 8, 1997
c
c     NOTE: this package contains the UPDATED Nijm-II potential
c     in which the 1P1 is changed as compared to the version of 1994;
c     this updated Nijm-II 1P1 was received from Dirk Hueber on 4/8/97;
c     he got it from V. Stoks (who talks to Hueber but not to me).
c     otherwise this routine is identical to the earlier routine nijm.f;
c     that is, this routine also contains the Nijm93 and Nijm-I
c     potentials in their original version of 1994.
c
c
c
      implicit real*8 (a-h,o-z)
      common /crdwrt/ kread,kwrite,kpunch,kda(9)
      common/cpot/ v(6),xmev,ymev
      common/cstate/ j,heform,sing,trip,coup,endep,label
      COMMON/EMANHP/PHNAME
      COMMON/CHOICE/IDPAR
      COMMON/RELKIN/NONREL
      logical heform,sing,trip,coup,endep
      CHARACTER PHNAME*3, TYPE*2
      LOGICAL NONREL
      REAL*8 VPOT(2,2)
      dimension vv(6)
      dimension vl(4),adminv(4,4),ldminv(4),mdminv(4)
      character*4 name(3),nname(15)
      data pi/3.141592653589793d0/
      logical index
      data index/.false./
      data jj/-1/
c
c
      kread=5
      KWRITE=6

      if (index) go to 50
      index=.true.
c
c
c        read in parameters for Nijmegen potential
c
      write (kwrite,10000)
10000 format (///' Updated Nijmegen Potential',
     1' (Nijm-II 1P1 updated, 4/8/97)'/
```

```
     2           ' -------------------------')
      read (kread,10001) nname
10001 format (15a4)
      write (kwrite,10002) nname
10002 format (' ',15a4)
c        IDPAR = 0, 1, 2 for Nijm 93, I, II(local), respectively.
      read (kread,10003) name,IDPAR
10003 format (2a4,a2,i1)
      write (kwrite,10004) name,IDPAR
10004 format (' ',2a4,a2,i1)
c        TYPE = 'PP', 'NN', 'NP', or 'PN'.
      read (kread,10005) name,TYPE
10005 format (2a4,a2,a2)
      write (kwrite,10006) name,TYPE
10006 format (' ',2a4,a2,a2)
      read (kread,10007) name,NONREL
10007 format (2a4,a2,l1)
      write (kwrite,10008) name,NONREL
10008 format (' ',2a4,a2,l1)
      read (kread,10009) name,label
10009 format (2a4,a2,a4)
      write (kwrite,10010) name,label
10010 format (' ',2a4,a2,a4///)
c
c
      endep=.false.
      fa=1./(2.d0*pi*pi)
c
c
   50 if (j.eq.0.or.j.eq.jj) go to 90
      jj=j
c
c
      if (j.gt.9) write (kwrite,19001)
19001 format (///' warning. nijmegen potential for j greater 9'/
     1' not defined.'
     2' the potential is set to zero.'/
     3' execution continued.'///)
c
c
c
      aj=dfloat(j)
      aj1=dfloat(j+1)
      a2j1=dfloat(2*j+1)
      aaj6=dsqrt(aj*aj1)
c
c        coefficient matrix for the translations into lsj formalism
c
      adminv(1,1)=aj1
      adminv(1,2)=aj
      adminv(1,3)=-aaj6
      adminv(1,4)=-aaj6
      adminv(2,1)=aj
      adminv(2,2)=aj1
      adminv(2,3)=aaj6
      adminv(2,4)=aaj6
      adminv(3,1)=aaj6
      adminv(3,2)=-aaj6
      adminv(3,3)=aj1
      adminv(3,4)=-aj
      adminv(4,1)=aaj6
      adminv(4,2)=-aaj6
      adminv(4,3)=-aj
      adminv(4,4)=aj1
c
c       inversion
```

```
c
      call dminv (adminv,4,deter,ldminv,mdminv)
c
c
c
c
   90 do 95 iv=1,6
   95 vv(iv)=0.d0
c
c
      if (j.gt.9) go to 2000
c
c
      j1=j+1
      do 295 i=1,3
      if (i.eq.1.and..not.sing) go to 295
      if (i.eq.2.and..not.trip) go to 295
      if (i.eq.3.and..not.coup) go to 295
c
c
      go to (100,110,120,130,140,150,160,170,180,190),j1
c
c
c        j = 0
c
  100 go to (101,102,295),i
  101 phname='1S0'
      go to 200
  102 phname='3P0'
      go to 200
c
c
c        j = 1
c
  110 go to (111,112,113),i
  111 phname='1P1'
      go to 200
  112 phname='3P1'
      go to 200
  113 phname='3C1'
      go to 200
c
c
c        j = 2
c
  120 go to (121,122,123),i
  121 phname='1D2'
      go to 200
  122 phname='3D2'
      go to 200
  123 phname='3C2'
      go to 200
c
c
c        j = 3
c
  130 go to (131,132,133),i
  131 phname='1F3'
      go to 200
  132 phname='3F3'
      go to 200
  133 phname='3C3'
      go to 200
c
c
c        j = 4
```

```
c
  140 go to (141,142,143),i
  141 phname='1G4'
      go to 200
  142 phname='3G4'
      go to 200
  143 phname='3C4'
      go to 200
c
c
c        j = 5
c
  150 go to (151,152,153),i
  151 phname='1H5'
      go to 200
  152 phname='3H5'
      go to 200
  153 phname='3C5'
      go to 200
c
c
c        j = 6
c
  160 go to (161,162,163),i
  161 phname='1I6'
      go to 200
  162 phname='3I6'
      go to 200
  163 phname='3C6'
      go to 200
c
c
c        j = 7
c
  170 go to (171,172,173),i
  171 phname='1J7'
      go to 200
  172 phname='3J7'
      go to 200
  173 phname='3C7'
      go to 200
c
c
c        j = 8
c
  180 go to (181,182,183),i
  181 phname='1K8'
      go to 200
  182 phname='3K8'
      go to 200
  183 phname='3C8'
      go to 200
c
c
c        j = 9
c
  190 go to (191,192,193),i
  191 phname='1L9'
      go to 200
  192 phname='3L9'
      go to 200
  193 phname='3C9'
      go to 200
c
c
c
```

```
c
  200 call pnymlsj (ymev,xmev,type,vpot)
c
c
c
c
      go to (201,202,203),i
  201 vv(1)=vpot(1,1)
      go to 295
  202 vv(2)=vpot(1,1)
      go to 295
  203 vv(3)=vpot(2,2)
      vv(4)=vpot(1,1)
      vv(5)=vpot(2,1)
      vv(6)=vpot(1,2)
  295 continue
c
c
      if (j.ne.0) go to 1000
      vv(3)=vv(2)
      vv(2)=0.d0
      go to 2000
c
c
c
 1000 if (.not.heform) go to 2000
c
c
c         translation into (combination of) helicity states
c
c
      do 1005 i=1,4
 1005 vl(i)=vv(i+2)
c
      do 1020 ii=1,4
      iii=ii+2
      vv(iii)=0.d0
c
      do 1015 i=1,4
 1015 vv(iii)=vv(iii)+adminv(ii,i)*vl(i)
 1020 vv(iii)=vv(iii)*a2j1
c
c
c
c
c
c         over-all factors
c
 2000 do 2005 iv=1,6
 2005 v(iv)=vv(iv)*fa
c
c
      return
      end
```

```
      SUBROUTINE PNYMLSJ(QI,QF,TYPE,VPOT)
c
c       this is the original code as obtained by anonymous ftp
c       from nijmegen on 10/20/94
c       the only change made:
c       the write statement that writes out the parameters
c       is commented out.
c
*********************************************************************
**    NIJMEGEN NUCLEON-NUCLEON POTENTIAL PROGRAM                   **
**    ----------------------------------------                     **
**    Version: June 1994                                           **
**    E-mail: thefalg@sci.kun.nl                                   **
**    Reference: Stoks et al. Phys.Rev. C49 (1994) June            **
**                                                                 **
**    Refined and extended version of the 1978 Nijmegen NN potential  **
**    in momentum space on LSJ basis.                              **
**    Basic references for formulas and set up of this program can be **
**    found in Phys.Rev.D17 (1978) 768 and Phys.Rev.C40 (1989) 2226.  **
**    For momentum-space projection see THEF-NYM-91.05 preprint    **
**    (available via anonymous ftp from thef-nym.sci.kun.nl).      **
**                                                                 **
**                                                                 **
**    INPUT :  QI    center of mass momentum initial state in MeV  **
**    -----    QF    center of mass momentum final   state in MeV  **
**             TYPE  'PP', 'NN', 'NP', or 'PN' (character*2)      **
**             Name partial wave via COMMON/EMANHP/PHNAME (see below) **
**             Maximum total angular momentum J=9 !!               **
**                                                                 **
**    OUTPUT:  central VC, spin-spin VS, tensor VT, spin-orbit VLS,  **
**    ------   asymmetric spin-orbit VLA, quadratic spin-orbit VQ,   **
**             and extra quadratic spin-orbit part VQP.            **
**             All potentials are calculated via a partial-wave    **
**             decomposition from Jmin up to Jmax, communicated via  **
**                COMMON/POTMOM/VC(-1:1),VS(-1:1),VT(-2:2),VLS(-2:2), **
**                              VLA(-2:2),VQ(-3:3),VQP(-2:2)       **
**             The subroutine returns a 2x2 potential matrix VPOT  **
**             in MeV**-2 which is the partial-wave momentum-space  **
**             potential for the partial wave PHNAME (see below)    **
**-----------------------------------------------------------------**
**    Defining the K-matrix as   2i*mu*q*K = (1-S)(1+S)^-1         **
**    (so for singlet channel   tan(delta)=-2*mu*q*K )             **
**    the partial-wave Lippmann-Schwinger equation reads           **
```

```
**                                                                     **
**       K(q'q) = V(q'q) + 2/pi int dk k^2 V(q'k) G(q,k) K(kq)         **
**    with                                                             **
**       G(q,k) = P / (E(q) - k^2/2/mu)                                **
**       V(q'k) = 1 / (4*pi) * VPOT(QI=k,QF=q')                        **
**---------------------------------------------------------------------**
**    Potential decomposition in momentum space plane-wave basis:      **
**    V(QF,QI) = VC                                                    **
**             + VS   (SIG1.SIG2)                                      **
**             + VT   [(SIG1.K)(SIG2.K)-K2/3(SIG1.SIG2)]               **
**             + VLS  (i/2)(SIG1+SIG2).N                               **
**             + VLA  (i/2)(SIG1-SIG2).N            (NOT USED !!!)     **
**             + VQ   (SIG1.N)(SIG2.N)                                 **
**             + VQP  [(SIG1.Q)(SIG2.Q)-Q2(SIG1.SIG2)                  **
**                    -(1/4)(SIG1.K)(SIG2.K)+(1/4)K2(SIG1.SIG2)]       **
**                                                                     **
**          K = QF - QI ,   Q = (QF+QI)/2 ,   N = QI X QF = Q X K      **
**                                                                     **
**    NOTE: In the partial wave decomposition we used the              **
**          SYM-convention.                                            **
**          If you use another convention in your Lippmann-Schwinger   **
**          programm, you may need an extra minus sign for the         **
**          the off-diagonal tensor potential VPOT(1,2) and VPOT(2,1)  **
**                                                                     **
**    NQ12  integer which opts the full Fourier transform for the      **
**          quadratic spin-orbit Q12 operator                          **
**       0: Fourier from (SIG1.N)(SIG2.N) --> Q12 ==> approximation    **
**       1: exact Fourier Q12 --> (SIG1.N)(SIG2.N)+extra               **
**          so same potential in coordinate space as in momentum space**
**    ( Presently NQ12=1 is included as a DATA statement )             **
**                                                                     **
**    COMMON-blocks which have to be filled beforehand:                **
**    +  COMMON/CHOICE/IDPAR                                           **
**               IDPAR is an integer and denotes the various different **
**                     models that can be chosen.                      **
**               IDPAR=0: nijm93: potential for pp and np together.    **
**                                including a phenomenological         **
**                                parameter to give the 1S0 pp and np  **
**                                phase shift/scattering length        **
**                                difference                           **
**                     2: nijmI : Reidlike model, each partial wave has**
**                                its own parameterset                 **
**                     3: nijmII: like nijmI, but fully local          **
**    +  COMMON/EMANHP/PHNAME                                          **
**               PHNAME is character*3 and contains the name of the    **
**               partial wave in the spectral notation.                **
**               - singlets:           1S0  1P1  1D2  1F3  1G4 ...     **
**               - triplets uncoupled: 3P0  3P1  3D2  3F3  3G4 ...     **
**               - triplets coupled:        3C1  3C2  3C3  3C4 ...     **
**                 where 3C1 denotes  3S1 -- EPS1 -- 3D1 channel       **
**                       3C2 denotes  3P2 -- EPS2 -- 3F2 channel ...   **
**    +  COMMON/RELKIN/NONREL                                          **
**               NONREL is a logical used in the IDPAR=1 and 2 options.**
**               NONREL=.TRUE.  gives the deuteron binding energy of   **
**                              B=2.224575 MeV using non-relativistic  **
**                              kinematics.                            **
**               NONREL=.FALSE. gives the deuteron binding energy of   **
**                              B=2.224575 MeV using relativistic      **
**                              kinematics.                            **
**               Model IDPAR=0 gives the deuteron only using           **
**               relativistic kinematics ==> NONREL=.FALSE.            **
**                                                                     **
**    NOTE: ALL potential models use a fixed fpi**2=0.075 for the      **
**    ----- pion-nucleon coupling constant at the pion pole, which    **
**          is represented by the DATA FPPPI0/0.075D0/ statement.      **
**                                                                     **
```

```
************************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 VPOT(2,2)
      INTEGER SPIN
      CHARACTER PHNAME*3, PHNAM0*3, TYPE*2
      COMMON/EMANHP/PHNAME
      COMMON/CHOICE/IDPAR
      COMMON/JFAC/  JMMM,JMM,JM,J,JP,JPP,JPPP
      COMMON/QFAC/  QI2,QF2,QIQF,QI2F2,QI2F22,QIPF2
      COMMON/POTMOM/VC(-1:1),VS(-1:1),VT(-2:2),VLS(-2:2),VLA(-2:2),
     .              VQ(-3:3),VQP(-2:2)
      DATA HBARC/197.327053D0/
      DATA NQ12/1/, PHNAM0/'***'/,  PI/3.14159265358979323846D0/
      SAVE NCHAN,SPIN,L,ISO,  TJMM,TJM,TJ,TJP,TJPP,TJJ
      SAVE E00,F00,G00,E11,F11,G11,EMM,EPP,GMM,GPP,FMM,FPP,FPM

      IF(DABS(QI-QF).GT.30*HBARC) THEN
C*      Necessary to avoid underflows
        VPOT(1,1) = 0D0
        VPOT(1,2) = 0D0
        VPOT(2,1) = 0D0
        VPOT(2,2) = 0D0
        RETURN
      ENDIF

      NLOC=1
C*    Only IDPAR=2 is fully local model
      IF(IDPAR.EQ.2) NLOC=0

      IF(PHNAME.NE.PHNAM0) THEN
        PHNAM0=PHNAME
        NCHAN=1
        IF(PHNAME(2:2).EQ.'C') NCHAN=2
        IF(PHNAME(1:1).EQ.'1') SPIN=0
        IF(PHNAME(1:1).EQ.'3') SPIN=1
        READ(PHNAME,'(2X,I1)') J
        IF(J.GT.9) WRITE(*,*)
     .             '**** Partial wave exceeds allowable maximum J=9'
        IF(J.GT.9) STOP
        L=J
        IF(PHNAME.EQ.'3P0') L=1
        IF(NCHAN.EQ.2) L=J-1
        ISO=MOD(SPIN+L+1,2)
        JMMM=J-3
        JMM =J-2
        JM  =J-1
        JP  =J+1
        JPP =J+2
        JPPP=J+3
        TJMM=2D0*J-3D0
        TJM =2D0*J-1D0
        TJ  =2D0*J+1D0
        TJP =2D0*J+3D0
        TJPP=2D0*J+5D0
        TJJ =DSQRT(J*(J+1D0))
C**     J-dependent coefficients for quadratic spin-orbit
        E00 = J*JM/TJM/TJ
        F00 =-2D0*(J*J+JM)/TJM/TJP
        G00 = JP*JPP/TJ/TJP
        E11 = JM*JPP/TJM/TJ
        F11 =-2D0*JM*JPP/TJM/TJP
        G11 = JM*JPP/TJ/TJP
        EMM =-JM*JMM/TJM/TJMM
        EPP =-J*(2D0*J*J+7D0*JP)/TJ/TJ/TJP
        GMM =-(2D0*J*J-3D0*J+2D0)*JP/TJM/TJ/TJ
        GPP =-JPP*JPPP/TJP/TJPP
```

```
        FMM = 2D0*(2D0*J*J*J-3D0*J*J-2D0*JM)/TJ/TJ/TJMM
        FPM = 2D0*TJJ/TJ/TJ
        FPP = 2D0*(2D0*J*J*J+9D0*J*J+10D0*J+1D0)/TJ/TJ/TJPP
      ENDIF

      QI2=QI*QI
      QF2=QF*QF
      QIQF=QI*QF
      QI2F2=QI2+QF2
      QI2F22=QI2F2*QI2F2
      QIPF2=(QI+QF)*(QI+QF)
      S2PSI=2D0*QIQF/QI2F2
      SPSI2=QF2/QI2F2
      CPSI2=QI2/QI2F2

      CALL VMOM(TYPE,NLOC,NQ12,ISO)

      IF(NCHAN.EQ.1) THEN
        IF(SPIN.EQ.0) THEN
          VPOT(1,1) = VC(0) - 3D0*VS(0) +
     .                QI2*QF2*(E00*VQ(-2)+F00*VQ(0)+G00*VQ(2))
        ELSEIF(L.EQ.J) THEN
          VPOT(1,1) = VC(0) + VS(0) - QIQF*(VLS(-1)-VLS(1))/TJ +
     .                2D0/3D0*QI2F2*
     .                (VT(0)-0.5D0*S2PSI*(TJP*VT(-1)+TJM*VT(1))/TJ) +
     .                QI2*QF2*(E11*VQ(-2)+F11*VQ(0)+G11*VQ(2))
        ELSEIF(PHNAME.EQ.'3P0') THEN
          VPOT(1,1) = VC(1) + VS(1) - QIQF*JPP/TJP*(VLS(0)-VLS(2)) +
     .                2D0/3D0*QI2F2*JPP/TJ*
     .                (-VT(1)+0.5D0*S2PSI*(TJPP*VT(0)+TJ*VT(2))/TJP) +
     .                QI2*QF2*(EPP*VQ(-1) + FPP*VQ(1) + GPP*VQ(3))
        ENDIF
      ELSE
        VPOT(1,1) = VC(-1) + VS(-1) + QIQF*JM/TJM*(VLS(-2)-VLS(0)) +
     .              2D0/3D0*QI2F2*JM/TJ*
     .              (-VT(-1)+0.5D0*S2PSI*(TJMM*VT(0)+TJ*VT(-2))/TJM) +
     .              QI2*QF2*(EMM*VQ(-3) + FMM*VQ(-1) + GMM*VQ(1))
        VPOT(1,2) =-2D0*QI2F2*TJJ/TJ*
     .              (-S2PSI*VT(0)+CPSI2*VT(-1)+SPSI2*VT(1)) -
     .              QI2*QF2*FPM*(VQ(1)-VQ(-1))
        VPOT(2,1) =-2D0*QI2F2*TJJ/TJ*
     .              (-S2PSI*VT(0)+SPSI2*VT(-1)+CPSI2*VT(1)) -
     .              QI2*QF2*FPM*(VQ(1)-VQ(-1))
        VPOT(2,2) = VC(1) + VS(1) - QIQF*JPP/TJP*(VLS(0)-VLS(2)) +
     .              2D0/3D0*QI2F2*JPP/TJ*
     .              (-VT(1)+0.5D0*S2PSI*(TJPP*VT(0)+TJ*VT(2))/TJP) +
     .              QI2*QF2*(EPP*VQ(-1) + FPP*VQ(1) + GPP*VQ(3))
      ENDIF

      IF(NQ12.EQ.1) THEN
C**     Extra contribution from inverse Fourier transform of Q12
C**     so momentum space exactly equivalent to configuration space
        IF(NCHAN.EQ.1) THEN
          IF(SPIN.EQ.0) THEN
            VPOT(1,1) = VPOT(1,1) - 2D0*QIQF/TJ*(J*VQP(-1)+JP*VQP(1))
          ELSEIF(L.EQ.J) THEN
            VPOT(1,1) = VPOT(1,1) + QIQF*(-VQP(-1)+VQP(1))/TJ
          ELSEIF(PHNAME.EQ.'3P0') THEN
            VPOT(1,1) = VPOT(1,1) + QIQF*
     .                  ((2D0*J*J+5D0*J+4D0)/TJ*VQP(0)+JPP*VQP(2))/TJP
          ENDIF
        ELSE
          VPOT(1,1) = VPOT(1,1) + QIQF*
     .                (JM*VQP(-2)+(2D0*J*J-JM)/TJ*VQP(0))/TJM
          VPOT(1,2) = VPOT(1,2) + 2D0*QIQF*TJJ/TJ*VQP(0)
          VPOT(2,1) = VPOT(2,1) + 2D0*QIQF*TJJ/TJ*VQP(0)
```

```
          VPOT(2,2) = VPOT(2,2) + QIQF*
     .                ((2D0*J*J+5D0*J+4D0)/TJ*VQP(0)+JPP*VQP(2))/TJP
        ENDIF
       ENDIF

       RETURN
       END
*********************************************************************
       SUBROUTINE VMOM(TYPE,NLOC,NQ12,ISO)
       IMPLICIT REAL*8 (A-H,O-Z)
       REAL*8 NEUTM
       CHARACTER PHNAME*3,PHNAM0*3, TYPE*2,TYP0*2
       LOGICAL FIRST
       COMMON/EMANHP/PHNAME
       COMMON/CHOICE/IDPAR
       COMMON/POTMOM/VC(-1:1),VS(-1:1),VT(-2:2),VLS(-2:2),VLA(-2:2),
     .               VQ(-3:3),VQP(-2:2)
       COMMON/MESONM/AMPI,AMETA,AMETP, AMRO,AMOM,AMFI, AMA0,AMEP,AMF0,
     .               AMPIC,AMROC,AMA0C,AWPIC,AWROC,AWA0C, AVSC
       COMMON/PARAMS/PAR(6,5)
       COMMON/COPLNG/ALPV,THPV,PV1, FPI,FETA,FETP,FPI2,FETA2,FETP2,
     .               ALVE,THV ,GV1, GRO,GOM,GFI,  GRO2,GOM2,GFI2,
     .               ALVM,     FV1, FRO,FOM,FFI,  FRO2,FOM2,FFI2,
     .               ALGS,THGS,GS1, GA0,GEP,GF0,  GA02,GEP2,GF02,
     .               GFPRO,GFPOM,GFPFI, GFMRO,GFMOM,GFMFI,
     .               FPIC2,GROC2,FROC2,GFPROC,GFMROC,GA0C2
       COMMON/YUKEFF/ARO,AMR1,AROC,AMRC1,AWRC1,BRO,AMR2,BROC,AMRC2,AWRC2,
     .               AVSC1,AVSC2, AEPS,AME1,BEPS,AME2
       COMMON/BROADM/GAMRO,THRRO,GAMRC,THRRC,GAMEP,THRE0,THREC,NRO,NEP
       COMMON/SCALIN/AMT,AMPV
       COMMON/CUTOFF/ALAM,ALAMP,ALAMV,ALAMS
       COMMON/POMRON/GPOM,FPOM2,AMPOM,AMPOM2,AMPOM4,
     .               GA2D,FA2D2,AMA2D,AMA2D2,AMA2D4
       COMMON/AMCOEF/ALF,REDM, AMY,AMY2,AMYI,AMYI2, AMN,AMN2,AMNI,AMNI2,
     .               AMYPN,AMYMN, AMYPN2,AMYMN2, AY2PN2,AY2MN2,
     .               AMYN,AMYNI,AMYNI2, AYPNI2,AYMNI2
       DATA FPPPI0/0.075D0/, FIRST/.FALSE./
       DATA TYP0/'XX'/, PHNAM0/'***'/
       DATA PI/3.14159265358979323846D0/
       SAVE ISIGN,CONV,PROTM,NEUTM,HBARC

       IF(FIRST) GOTO 10
       FIRST = .TRUE.
       CONV  = PI/180D0
**     Nucleon and meson masses (Particle Data Group 1990)
       HBARC = 197.327053D0
       PROTM = 938.27231D0
       NEUTM = 939.56563D0
       AMT   = 938.27231D0
       AMPV  = 139.5675D0
       AMPIC = 139.5675D0
       AMPI  = 134.9739D0
       AMETA = 548.8D0
       AMETP = 957.5D0
       AMROC = 768.3D0
       AMRO  = 768.7D0
       AMOM  = 781.95D0
       AMFI  =1019.412D0
       AMA0C = 983.3D0
       AMA0  = 983.3D0
       AMF0  = 975.6D0
       AVSC  = ((NEUTM-PROTM)/AMROC)**2
       FAC   = 2D0*DSQRT(PROTM*NEUTM)/(PROTM+NEUTM)
       AWPIC = FAC*DSQRT(AMPIC**2 - (NEUTM-PROTM)**2)
       AWROC = FAC*DSQRT(AMROC**2 - (NEUTM-PROTM)**2)
       AWA0C = FAC*DSQRT(AMA0C**2 - (NEUTM-PROTM)**2)
```

```
**    Broad rho-meson: spectral density to two effective Yukawa's
**    Yukawa's fitted to PHI0C 0.001 - 2 fm (steps 0.002, ALAMV=825 MeV)
      GAMRO = 152.4D0
      THRRO = AMPIC+AMPIC
      GAMRC = 149.1D0
      THRRC = AMPIC+AMPI
      NRO   = 1
      ARO   = 0.2655205D0
      BRO   = 0.5607493D0
      AMR1 = 645.3772D0
      AMR2 = 878.3667D0
      AROC = 0.3875515D0
      BROC = 0.4508341D0
      AMRC1= 674.1521D0
      AMRC2= 929.9742D0
      AWRC1 = FAC*DSQRT(AMRC1**2 - (NEUTM-PROTM)**2)
      AWRC2 = FAC*DSQRT(AMRC2**2 - (NEUTM-PROTM)**2)
      AVSC1 = ((NEUTM-PROTM)/AMRC1)**2
      AVSC2 = ((NEUTM-PROTM)/AMRC2)**2

   10 CONTINUE
      IF(TYPE.EQ.TYP0) THEN
        IF((IDPAR.EQ.1 .OR. IDPAR.EQ.2) .AND. PHNAME.NE.PHNAM0) GOTO 15
        GOTO 20
      ENDIF
      TYP0=TYPE
      IF(TYPE.EQ.'PP') THEN
        AMY = PROTM
        AMN = PROTM
        I3Y = 1
        I3N = 1
      ELSEIF(TYPE.EQ.'NP') THEN
        AMY = NEUTM
        AMN = PROTM
        I3Y =-1
        I3N = 1
      ELSEIF(TYPE.EQ.'PN') THEN
        AMY = PROTM
        AMN = NEUTM
        I3Y = 1
        I3N =-1
      ELSEIF(TYPE.EQ.'NN') THEN
        AMY = NEUTM
        AMN = NEUTM
        I3Y =-1
        I3N =-1
      ENDIF
      ISIGN = I3Y*I3N
      CALL AMFACS

   15 PHNAM0=PHNAME
      CALL NYMPAR(ISIGN)
C              pseudovec  vector    tensor    scalar    pomeron
C              FPI        GR0       FR0       GA0       GPOM
C              PV1        GV1       FOM       GEP       GA2D
C              ALPV       ALVE      FFI       GF0       AMPOM
C              THPV       THV       RHOC%     THGS      AMA2D
C     Cut-off  ALAMP      ALAMV     ALAM      ALAMS     AMEP
C     2 Yukawa AEPS       AME1      BEPS      AME2      GAMEP

**    (joined) cutoffs for pseudoscalar, vector, scalar
      ALAMP = PAR(5,1)
      ALAMV = PAR(5,2)
      ALAMS = PAR(5,4)
      ALAM  = PAR(5,3)
```

```
      IF(ALAMP.EQ.0D0) ALAMP=ALAM
      IF(ALAMV.EQ.0D0) ALAMV=ALAM
      IF(ALAMS.EQ.0D0) ALAMS=ALAM
**    pseudovector couplings
C     FPI   = PAR(1,1)
      FPI   = DSQRT(FPPPI0*FDEXP(-(AMPI/ALAMP)**2))
      PV1   = PAR(2,1)
      ALPV  = PAR(3,1)
      THPV  = PAR(4,1)*CONV
**    vector couplings
      GRO   = PAR(1,2)
      GV1   = PAR(2,2)
      ALVE  = PAR(3,2)
      THV   = PAR(4,2)*CONV
**    tensor couplings
      FRO   = PAR(1,3)
      FOM   = PAR(2,3)
      FFI   = PAR(3,3)
**    scalar couplings
      GA0   = PAR(1,4)
      GEP   = PAR(2,4)
      GF0   = PAR(3,4)
      THGS  = PAR(4,4)*CONV
**    diffractive contribution
      GPOM  = PAR(1,5)
      GA2D  = PAR(2,5)
      AMPOM = PAR(3,5)
      AMA2D = PAR(4,5)
      IF(AMA2D.EQ.0D0) AMA2D=AMPOM
      AMPOM2= AMPOM*AMPOM
      AMPOM4= AMPOM2*AMPOM2
      AMA2D2= AMA2D*AMA2D
      AMA2D4= AMA2D2*AMA2D2

      CALL NYMCOP(ISIGN)

**    Broad epsilon-meson: spectral density to two effective Yukawa's
**    Yukawa's fitted to PHI0C from 0.001 to 2 fm (steps 0.005)
      AMEP  = PAR(5,5)
      GAMEP = PAR(6,5)
      THRE0 = AMPI+AMPI
      THREC = AMPIC+AMPIC
      NEP   = 0
      AEPS = PAR(6,1)
      AME1 = PAR(6,2)
      BEPS = PAR(6,3)
      AME2 = PAR(6,4)

**    isospin dependence
   20 FACISO= 4D0*ISO-2D0
      GA0C2 = FACISO*GA0*GA0
      FPIC2 = FACISO*FPI*FPI/(AMPV*AMPV)
      GROC2 = FACISO*GRO*GRO
      FROC2 = FACISO*FRO*FRO/(4D0*AMT*AMT)
      GFPROC= FACISO*(GRO*FRO+FRO*GRO)/(2D0*AMT)
      GFMROC= FACISO*(GRO*FRO-FRO*GRO)/(2D0*AMT)
      IF(PHNAME.EQ.'1S0' .AND. (TYPE.EQ.'NP'.OR.TYPE.EQ.'PN')) THEN
**      Phenomenological phase-shift difference in PP and NP 1S0
        GROC2=GROC2*(1D0+PAR(4,3))
        FROC2=FROC2*(1D0+PAR(4,3))
      ENDIF

      FPOM2 = (GPOM/AMT)**2
      FA2D2 = (GA2D/AMT)**2

      DO 50 KK=-1,1
```

```
        VC(KK) = 0D0
   50   VS(KK) = 0D0
        DO 51 KK=-2,2
        VT(KK)  = 0D0
        VLS(KK) = 0D0
        VLA(KK) = 0D0
        VQ(KK)  = 0D0
   51   VQP(KK) = 0D0
        VQ(-3) = 0D0
        VQ( 3) = 0D0

        CALL PSVECT(TYPE,NLOC)
        CALL VECTOR(TYPE,NLOC,NQ12)
        CALL SCALAR(TYPE,NLOC,NQ12)
        CALL DIFRAC(TYPE,NLOC,NQ12,ISIGN,FACISO)

        RETURN
        END
************************************************************************
        SUBROUTINE AMFACS
        IMPLICIT REAL*8 (A-H,O-Z)
        COMMON/AMCOEF/ALF,REDM, AMY,AMY2,AMYI,AMYI2, AMN,AMN2,AMNI,AMNI2,
       .              AMYPN,AMYMN, AMYPN2,AMYMN2, AY2PN2,AY2MN2,
       .              AMYN,AMYNI,AMYNI2, AYPNI2,AYMNI2

        AMY2  = AMY*AMY
        AMYI  = 1D0/AMY
        AMYI2 = AMYI*AMYI
        AMN2  = AMN*AMN
        AMNI  = 1D0/AMN
        AMNI2 = AMNI*AMNI
        AMYPN = AMY+AMN
        AMYMN = AMY-AMN
        AMYPN2= AMYPN*AMYPN
        AMYMN2= AMYMN*AMYMN
        AY2PN2= AMY2+AMN2
        AY2MN2= AMY2-AMN2
        AMYN  = AMY*AMN
        AMYNI = AMYI*AMNI
        AMYNI2= AMYNI*AMYNI
        AYPNI2= AMYI2+AMNI2
        AYMNI2= AMYI2-AMNI2
        REDM = AMYN/AMYPN
        ALF  = 4D0*REDM/AMYPN
        IF(AMY.EQ.AMN) ALF = 1D0
        RETURN
        END
************************************************************************
        SUBROUTINE NYMPAR(ISIGN)
        IMPLICIT REAL*8 (A-H,O-Z)
        COMMON/PARAMS/PAR(6,5)
        COMMON/CHOICE/IDPAR
        DIMENSION PARTR(5,6)
C               pseudovec  vector    tensor    scalar    pomeron
C               FPI        GR0       FR0       GA0       GPOM
C               PV1        GV1       FOM       GEP       GA2D
C               ALPV       ALVE      FFI       GF0       AMPOM
C               THPV       THV       RHOC%     THGS      AMA2D
C     Cut-off   ALAMP      ALAMV     ALAM      ALAMS     AMEP
C     2 Yukawa  AEPS       AME1      BEPS      AME2      GAMEP
C-----------------------------------------------------------------------

**    Parameter-set with separate cutoffs, fitted to all data    Nijm93
      DATA PARTR/
     1 .2720668D+00,.9209319D+00,.3770582D+01,.1384689D+01,.5228672D+01
     2,.1595311D+00,.2594356D+01,.5816365D+00,.5310001D+01,.2204600D+00
```

```
     3,.3550000D+00,.1000000D+01,.0000000D+00,.3484505D+01,.2081618D+03
     4,-.230000D+02,.3750000D+02,.4371144D-01,.3790000D+02,.0000000D+00
     5,.1177107D+04,.9045040D+03,.0000000D+00,.5544013D+03,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/

      IF(IDPAR.EQ.0) THEN
        DO 1 I=1,6
          DO 1 J=1,5
  1         PAR(I,J)=PARTR(J,I)
      ELSEIF(IDPAR.EQ.1) THEN
        CALL PHSRDL(ISIGN)
      ELSEIF(IDPAR.EQ.2) THEN
        CALL PHSLOC(ISIGN)
      ENDIF

c**** WRITE(*,*) ' NIJMEGEN POTENTIAL PARAMETERS ARE:'
c**** DO 4 I=1,6
c*4     WRITE(*,5) (PAR(I,J),J=1,5)
  5   FORMAT(5(1X,D15.7))
      RETURN
      END
```

```
************************************************************************
C     This subroutine reads the parameters of the 0-350 MeV fitted
C     Reidlike potential, Jan 1993, chi**2/data=1.03            Nijm I
C-----------------------------------------------------------------------
      SUBROUTINE PHSRDL(ISIGN)
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER PHNAME*3
      LOGICAL NONREL
      COMMON/PARAMS/PAR(6,5)
      COMMON/EMANHP/PHNAME
      COMMON/RELKIN/NONREL
      DIMENSION APP1S0(5,6),ANP1S0(5,6),PAR1P1(5,6),PAR1D2(5,6),
     .          PAR1F3(5,6),PAR1G4(5,6),PAR3P1(5,6),PAR3D2(5,6),
     .          PAR3F3(5,6),PAR3G4(5,6),PAR3P0(5,6),PAR3C1(5,6),
     .          PAR3C2(5,6),PAR3C3(5,6),PAR3C4(5,6),PARRST(5,6),
     .          PNR3C1(5,6)
      DATA APP1S0/
     1 .2702427D+00,.6723103D+00,.3697581D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.4846532D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4744299D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA ANP1S0/
     1 .2702427D+00,.6723103D+00,.3423514D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.4942984D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR1P1/
     1 .2702427D+00,.6723103D+00,.4728635D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.3290519D+01,.5473693D-01,.4784616D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR1D2/
     1 .2702427D+00,.6723103D+00,.4728635D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.0000000D+00,.5473693D-01,.2348432D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.3639503D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR1F3/
     1 .2702427D+00,.6723103D+00,.4728635D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.4812761D+01,.5473693D-01,.6316315D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR1G4/
     1 .2702427D+00,.6723103D+00,.4728635D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.4142482D+01,.5473693D-01,.4859055D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR3P1/
     1 .2702427D+00,.6723103D+00,.0000000D+00,.8334762D+00,.2751792D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.4980991D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4923498D+03,.6130152D+00,.1021139D+04,.6400000D+03/
```

```
      DATA PAR3D2/
     1 .2702427D+00,.6723103D+00,.4728635D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.5880449D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.1693534D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR3F3/
     1 .2702427D+00,.6723103D+00,.8961465D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.5458122D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR3G4/
     1 .2702427D+00,.6723103D+00,.4728635D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.3201615D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR3P0/
     1 .2702427D+00,.6723103D+00,.3160965D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.3726932D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR3C1/
     1 .2706937D+00,.2598577D+01,.2125742D+01,.8334762D+00,.2751792D+01
     2,.0000000D+00,.2536422D+01,.5473693D-01,.4906761D+01,.44371888D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.8848614D+03,.6465221D+03,.0000000D+00,.6990612D+03,.7600000D+03
     6,.1690008D+00,.5831763D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR3C2/
     1 .2702427D+00,.6723103D+00,.5188648D+01,.8334762D+00,.3668014D+01
     2,.1341669D+00,.0000000D+00,.5473693D-01,.3995761D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.8275346D+03,.0000000D+00,.5831699D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR3C3/
     1 .2702427D+00,.6723103D+00,.3877361D+01,.8334762D+00,.4723766D+01
     2,.2811080D+00,.0000000D+00,.5473693D-01,.4818122D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR3C4/
     1 .2702427D+00,.6723103D+00,.7551377D+01,.8334762D+00,.2360920D+01
     2,.2871299D+00,.0000000D+00,.5473693D-01,.4855317D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PARRST/
     1 .2702427D+00,.6723103D+00,.4728635D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.4859055D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
C     Parameters for a nonlocal potential with non-relativistic deuteron
      DATA PNR3C1/
     1 .2706937D+00,.2588931D+01,.2120162D+01,.8334762D+00,.2751792D+01
```

```
     2,.0000000D+00,.2521404D+01,.5473693D-01,.4889314D+01,.44371806D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.8848614D+03,.6465221D+03,.0000000D+00,.6990612D+03,.7600000D+03
     6,.1690008D+00,.5831292D+03,.6130152D+00,.1021139D+04,.6400000D+03/

      DO 1 I=1,6
        DO 1 J=1,5
          IF(PHNAME.EQ.'1S0') THEN
            IF(ISIGN.EQ. 1) PAR(I,J)=APP1S0(J,I)
            IF(ISIGN.EQ.-1) PAR(I,J)=ANP1S0(J,I)
          ELSEIF(PHNAME.EQ.'1P1') THEN
            PAR(I,J)=PAR1P1(J,I)
          ELSEIF(PHNAME.EQ.'1D2') THEN
            PAR(I,J)=PAR1D2(J,I)
          ELSEIF(PHNAME.EQ.'1F3') THEN
            PAR(I,J)=PAR1F3(J,I)
          ELSEIF(PHNAME.EQ.'1G4') THEN
            PAR(I,J)=PAR1G4(J,I)
          ELSEIF(PHNAME.EQ.'3P0') THEN
            PAR(I,J)=PAR3P0(J,I)
          ELSEIF(PHNAME.EQ.'3P1') THEN
            PAR(I,J)=PAR3P1(J,I)
          ELSEIF(PHNAME.EQ.'3D2') THEN
            PAR(I,J)=PAR3D2(J,I)
          ELSEIF(PHNAME.EQ.'3F3') THEN
            PAR(I,J)=PAR3F3(J,I)
          ELSEIF(PHNAME.EQ.'3G4') THEN
            PAR(I,J)=PAR3G4(J,I)
          ELSEIF(PHNAME.EQ.'3C1') THEN
            IF(.NOT.NONREL) PAR(I,J)=PAR3C1(J,I)
            IF(NONREL) PAR(I,J)=PNR3C1(J,I)
          ELSEIF(PHNAME.EQ.'3C2') THEN
            PAR(I,J)=PAR3C2(J,I)
          ELSEIF(PHNAME.EQ.'3C3') THEN
            PAR(I,J)=PAR3C3(J,I)
          ELSEIF(PHNAME.EQ.'3C4') THEN
            PAR(I,J)=PAR3C4(J,I)
          ELSE
            PAR(I,J)=PARRST(J,I)
          ENDIF
    1 CONTINUE
      RETURN
      END
```

```
************************************************************************
C     This subroutine reads the parameters of the 0-350 MeV fitted local
C     potential (without Q**2), Jan 1993, chi**2/data=1.03       Nijm II
C-----------------------------------------------------------------------
      SUBROUTINE PHSLOC(ISIGN)
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER PHNAME*3
      LOGICAL NONREL
      COMMON/PARAMS/PAR(6,5)
      COMMON/EMANHP/PHNAME
      COMMON/RELKIN/NONREL
      DIMENSION APP1S0(5,6),ANP1S0(5,6),PAR1P1(5,6),PAR1D2(5,6),
     .          PAR1F3(5,6),PAR1G4(5,6),PAR3P1(5,6),PAR3D2(5,6),
     .          PAR3F3(5,6),PAR3G4(5,6),PAR3P0(5,6),PAR3C1(5,6),
     .          PAR3C2(5,6),PAR3C3(5,6),PAR3C4(5,6),PARRST(5,6),
     .          PNR3C1(5,6)
      DATA APP1S0/
     1 .2702427D+00,.6723103D+00,.4006116D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.4669822D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.3909783D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4692497D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA ANP1S0/
     1 .2702427D+00,.6723103D+00,.7407998D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.4626459D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2196159D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
```

```
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4722381D+03,.6130152D+00,.1021139D+04,.6400000D+03/
c
c******** below is the revised 1p1, as received from D. Hueber on 4/8/97
c******** who got it from V. Stoks.
      DATA PAR1P1/
     1 .2702427D+00,.6723103D+00,.0000000D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.3430827D+01,.5473693D-01,.2825877D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+03,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
cold  DATA PAR1P1/
cold 1 .2702427D+00,.6723103D+00,.4728635D+01,.8334762D+00,.2751792D+01
cold 2,.2871299D+00,.2849628D+01,.5473693D-01,.5411714D+01,.4437220D+00
cold 3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
cold 4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
cold 5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
cold 6,.1690008D+00,.5217623D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR1D2/
     1 .2702427D+00,.6723103D+00,.4728635D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.4138573D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2243917D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4367031D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR1F3/
     1 .2702427D+00,.6723103D+00,.4728635D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.3983000D+01,.5473693D-01,.5627977D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR1G4/
     1 .2702427D+00,.6723103D+00,.4728635D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.4859055D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2037620D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR3P1/
     1 .2702427D+00,.6723103D+00,.0000000D+00,.8334762D+00,.2751792D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.4171550D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.3368384D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4530824D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR3D2/
     1 .2702427D+00,.6723103D+00,.4728635D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.5469270D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.1847244D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR3F3/
     1 .2702427D+00,.6723103D+00,.6012926D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.5530460D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR3G4/
     1 .2702427D+00,.6723103D+00,.4728635D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.3663270D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
```

```
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR3P0/
     1 .2702427D+00,.6723103D+00,.2761025D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.3041218D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.8275346D+03,.0000000D+00,.1134832D+04,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR3C1/
     1 .2702427D+00,.1607944D+01,.1841778D+01,.5469244D+00,.3469472D+01
     2,.2871299D+00,.2240543D+01,.5473693D-01,.4035077D+01,.4437213D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.5151821D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.8275346D+03,.0000000D+00,.8044237D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR3C2/
     1 .2702427D+00,.6723103D+00,.5816373D+01,.8334762D+00,.3957678D+01
     2,.2353573D+00,.0000000D+00,.5473693D-01,.4143714D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2781205D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.8275346D+03,.0000000D+00,.6121468D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR3C3/
     1 .2702427D+00,.6723103D+00,.4050335D+01,.8334762D+00,.4316501D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.5048592D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PAR3C4/
     1 .2702427D+00,.6723103D+00,.7347855D+01,.8334762D+00,.2579081D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.5157279D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
      DATA PARRST/
     1 .2702427D+00,.6723103D+00,.4728635D+01,.8334762D+00,.2751792D+01
     2,.2871299D+00,.2849628D+01,.5473693D-01,.4859055D+01,.4437220D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.2579522D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.0000000D+00,.0000000D+00,.8275346D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/
C     Parameters for a local potential with non-relativistic deuteron
      DATA PNR3C1/
     1 .2702427D+00,.1710842D+01,.1781765D+01,.5362515D+00,.3461562D+01
     2,.2871299D+00,.2247159D+01,.5473693D-01,.4028595D+01,.44387996D+00
     3,.3550000D+00,.1000000D+01,.0000000D+00,.8389363D+00,.5184792D+03
     4,-.230000D+02,.3750000D+02,.0000000D+00,.3790000D+02,.0000000D+00
     5,.8275346D+03,.0000000D+00,.8044237D+03,.0000000D+00,.7600000D+03
     6,.1690008D+00,.4878179D+03,.6130152D+00,.1021139D+04,.6400000D+03/

      DO 1 I=1,6
        DO 1 J=1,5
          IF(PHNAME.EQ.'1S0') THEN
            IF(ISIGN.EQ. 1) PAR(I,J)=APP1S0(J,I)
            IF(ISIGN.EQ.-1) PAR(I,J)=ANP1S0(J,I)
          ELSEIF(PHNAME.EQ.'1P1') THEN
            PAR(I,J)=PAR1P1(J,I)
          ELSEIF(PHNAME.EQ.'1D2') THEN
            PAR(I,J)=PAR1D2(J,I)
          ELSEIF(PHNAME.EQ.'1F3') THEN
            PAR(I,J)=PAR1F3(J,I)
          ELSEIF(PHNAME.EQ.'1G4') THEN
            PAR(I,J)=PAR1G4(J,I)
```

```
          ELSEIF(PHNAME.EQ.'3P0') THEN
            PAR(I,J)=PAR3P0(J,I)
          ELSEIF(PHNAME.EQ.'3P1') THEN
            PAR(I,J)=PAR3P1(J,I)
          ELSEIF(PHNAME.EQ.'3D2') THEN
            PAR(I,J)=PAR3D2(J,I)
          ELSEIF(PHNAME.EQ.'3F3') THEN
            PAR(I,J)=PAR3F3(J,I)
          ELSEIF(PHNAME.EQ.'3G4') THEN
            PAR(I,J)=PAR3G4(J,I)
          ELSEIF(PHNAME.EQ.'3C1') THEN
            IF(.NOT.NONREL) PAR(I,J)=PAR3C1(J,I)
            IF(NONREL) PAR(I,J)=PNR3C1(J,I)
          ELSEIF(PHNAME.EQ.'3C2') THEN
            PAR(I,J)=PAR3C2(J,I)
          ELSEIF(PHNAME.EQ.'3C3') THEN
            PAR(I,J)=PAR3C3(J,I)
          ELSEIF(PHNAME.EQ.'3C4') THEN
            PAR(I,J)=PAR3C4(J,I)
          ELSE
            PAR(I,J)=PARRST(J,I)
          ENDIF
    1   CONTINUE
        RETURN
        END
*************************************************************************
        SUBROUTINE NYMCOP(ISIGN)
        IMPLICIT REAL*8 (A-H,O-Z)
        COMMON/COPLNG/ALPV,THPV,PV1, FPI,FETA,FETP,FPI2,FETA2,FETP2,
       .              ALVE,THV ,GV1, GRO,GOM,GFI,  GRO2,GOM2,GFI2,
       .              ALVM,     FV1, FRO,FOM,FFI,  FRO2,FOM2,FFI2,
       .              ALGS,THGS,GS1, GA0,GEP,GF0,  GA02,GEP2,GF02,
       .              GFPRO,GFPOM,GFPFI, GFMRO,GFMOM,GFMFI,
       .              FPIC2,GROC2,FROC2,GFPROC,GFMROC,GA0C2
        COMMON/SCALIN/AMT,AMPV
        DATA SR3/1.732050807568877293 5D0/

**      pseudovector coupling constants
        PV8  = FPI * (4D0*ALPV-1D0)/SR3
        COST = DCOS(THPV)
        SINT = DSIN(THPV)
        FETA = COST*PV8 - SINT*PV1
        FETP = SINT*PV8 + COST*PV1
        FPI2 = FPI*FPI/(AMPV*AMPV) * ISIGN
        FETA2= FETA*FETA/(AMPV*AMPV)
        FETP2= FETP*FETP/(AMPV*AMPV)

**      vector coupling constants
        GV8  = GRO * (4D0*ALVE-1D0)/SR3
        COST = DCOS(THV)
        SINT = DSIN(THV)
        GFI  = COST*GV8 - SINT*GV1
        GOM  = SINT*GV8 + COST*GV1
        GRO2 = GRO*GRO * ISIGN
        GOM2 = GOM*GOM
        GFI2 = GFI*GFI

**      tensor coupling constants
        COST = DCOS(THV)
        SINT = DSIN(THV)
        FV8  = COST*(GFI+FFI) + SINT*(GOM+FOM) - GV8
        FV1  =-SINT*(GFI+FFI) + COST*(GOM+FOM) - GV1
        ALVM = (SR3*(GV8+FV8)+(GRO+FRO))/(4D0*(GRO+FRO))
        FRO2 = FRO*FRO/(4D0*AMT*AMT) * ISIGN
        FOM2 = FOM*FOM/(4D0*AMT*AMT)
        FFI2 = FFI*FFI/(4D0*AMT*AMT)
```

```
      GFPRO = (GRO*FRO+FRO*GRO)/(2D0*AMT) * ISIGN
      GFPOM = (GOM*FOM+FOM*GOM)/(2D0*AMT)
      GFPFI = (GFI*FFI+FFI*GFI)/(2D0*AMT)
      GFMRO = (GRO*FRO-FRO*GRO)/(2D0*AMT) * ISIGN
      GFMOM = (GOM*FOM-FOM*GOM)/(2D0*AMT)
      GFMFI = (GFI*FFI-FFI*GFI)/(2D0*AMT)

**    scalar coupling constants
      COST = DCOS(THGS)
      SINT = DSIN(THGS)
      GS8  = COST*GEP + SINT*GF0
      GS1  =-SINT*GEP + COST*GF0
      ALGS = (SR3*GS8+GA0)/(4D0*GA0)
      GA02 = GA0*GA0 * ISIGN
      GEP2 = GEP*GEP
      GF02 = GF0*GF0

      RETURN
      END
*****************************************************************************
      SUBROUTINE PSVECT(TYPE,NLOC)
C*    Partial-wave momentum-space potentials for pseudoscalar exchange
C*       NLOC=0,1: no nonlocal (q^2+k^2/4) contributions
C*              2: nonlocal contributions in spin-spin and tensor
C-----------------------------------------------------------------------
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER TYPE*2
      COMMON/POTMOM/VC(-1:1),VS(-1:1),VT(-2:2),VLS(-2:2),VLA(-2:2),
     .              VQ(-3:3),VQP(-2:2)
      COMMON/JFAC/  JMMM,JMM,JM,J,JP,JPP,JPPP
      COMMON/QFAC/  QI2,QF2,QIQF,QI2F2,QI2F22,QIPF2
      COMMON/MESONM/AMPI,AMETA,AMETP, AMRO,AMOM,AMFI, AMA0,AMEP,AMF0,
     .              AMPIC,AMROC,AMA0C,AWPIC,AWROC,AWA0C, AVSC
      COMMON/COPLNG/ALPV,THPV,PV1, FPI,FETA,FETP,FPI2,FETA2,FETP2,
     .              ALVE,THV ,GV1, GRO,GOM,GFI,  GRO2,GOM2,GFI2,
     .              ALVM,     FV1, FRO,FOM,FFI,  FRO2,FOM2,FFI2,
     .              ALGS,THGS,GS1, GA0,GEP,GF0,  GA02,GEP2,GF02,
     .              GFPRO,GFPOM,GFPFI, GFMRO,GFMOM,GFMFI,
     .              FPIC2,GROC2,FROC2,GFPROC,GFMROC,GA0C2
      COMMON/CUTOFF/ALAM,ALAMP,ALAMV,ALAMS
      COMMON/AMCOEF/ALF,REDM, AMY,AMY2,AMYI,AMYI2, AMN,AMN2,AMNI,AMNI2,
     .              AMYPN,AMYMN, AMYPN2,AMYMN2, AY2PN2,AY2MN2,
     .              AMYN,AMYNI,AMYNI2, AYPNI2,AYMNI2
      DIMENSION U(-3:12), R(-3:12), ELN(15), ERN(15)
      DATA U/16*0D0/, R/16*0D0/
      DATA PI/3.14159265358979323846D0/

      XCOM=0.5D0*QI2F2/QIQF
      ALAM2=ALAMP*ALAMP
      Y=2D0*QIQF/ALAM2
      JMAX=J+2
      KOM=1
      FAC=2D0*PI/QIQF

C**     Pseudoscalar mesons: pi, eta, eta'
      DO 1000 IN=1,3
        IF(IN.EQ.1) THEN
          AMES=AMPI
          FP2 =FPI2
        ELSEIF(IN.EQ.2) THEN
          AMES=AMETA
          FP2 =FETA2
        ELSEIF(IN.EQ.3) THEN
          AMES=AMETP
          FP2 =FETP2
```

```
        ENDIF
        AMES2=AMES*AMES
        X=XCOM+0.5D0*AMES2/QIQF
        CALL SEX(X,Y,AMES/ALAMP,JMAX,KOM,ELN,ERN)
        XPS =-FAC*FP2*QI2F2/3D0
        YPS = FAC*FP2*2D0*QIQF/3D0
        XPT =-FAC*FP2
        XNLS= FAC*FP2*AMYNI*QI2F22/12D0
        YNLS=-FAC*FP2*AMYNI*QI2F2*QIQF/6D0
        XNLT= FAC*FP2*AMYNI*QI2F2/4D0
        DO 5 K=0,JMAX
          U(K)=ELN(K+1)
          R(K)=ERN(K+1)
    5   CONTINUE
        VS(-1) = VS(-1) + (XPS+X*YPS)*U(JM) - YPS*R(JM)
        VS( 0) = VS( 0) + (XPS+X*YPS)*U(J)  - YPS*R(J)
        VS( 1) = VS( 1) + (XPS+X*YPS)*U(JP) - YPS*R(JP)
        VT(-2) = VT(-2) +  XPT*U(JMM)
        VT(-1) = VT(-1) +  XPT*U(JM)
        VT( 0) = VT( 0) +  XPT*U(J)
        VT( 1) = VT( 1) +  XPT*U(JP)
        VT( 2) = VT( 2) +  XPT*U(JPP)
        IF(NLOC.EQ.2) THEN
          VS(-1) = VS(-1) + (XNLS+X*YNLS)*U(JM) - YNLS*R(JM)
          VS( 0) = VS( 0) + (XNLS+X*YNLS)*U(J)  - YNLS*R(J)
          VS( 1) = VS( 1) + (XNLS+X*YNLS)*U(JP) - YNLS*R(JP)
          VT(-2) = VT(-2) + XNLT*U(JMM)
          VT(-1) = VT(-1) + XNLT*U(JM)
          VT( 0) = VT( 0) + XNLT*U(J)
          VT( 1) = VT( 1) + XNLT*U(JP)
          VT( 2) = VT( 2) + XNLT*U(JPP)
        ENDIF
 1000 CONTINUE

      IF(TYPE.EQ.'PP' .OR. TYPE.EQ.'NN') RETURN

C**     Charged pseudoscalar: pi+
      AMES=AWPIC
      FP2 =FPIC2
      AMES2=AMES*AMES
      X=XCOM+0.5D0*AMES2/QIQF
      CALL SEX(X,Y,AMES/ALAMP,JMAX,KOM,ELN,ERN)
      XPS =-FAC*FP2/ALF*QI2F2/3D0
      YPS = FAC*FP2/ALF*2D0*QIQF/3D0
      XPT =-FAC*FP2/ALF
      XNLS= FAC*FP2/ALF*AYPNI2*QI2F22/24D0
      YNLS=-FAC*FP2/ALF*AYPNI2*QI2F2*QIQF/12D0
      XNLT= FAC*FP2/ALF*AYPNI2*QI2F2/8D0
      DO 6 K=0,JMAX
        U(K)=ELN(K+1)
        R(K)=ERN(K+1)
    6 CONTINUE
        VS(-1) = VS(-1) + (XPS+X*YPS)*U(JM) - YPS*R(JM)
        VS( 0) = VS( 0) + (XPS+X*YPS)*U(J)  - YPS*R(J)
        VS( 1) = VS( 1) + (XPS+X*YPS)*U(JP) - YPS*R(JP)
        VT(-2) = VT(-2) +  XPT*U(JMM)
        VT(-1) = VT(-1) +  XPT*U(JM)
        VT( 0) = VT( 0) +  XPT*U(J)
        VT( 1) = VT( 1) +  XPT*U(JP)
        VT( 2) = VT( 2) +  XPT*U(JPP)
        IF(NLOC.EQ.2) THEN
          VS(-1) = VS(-1) + (XNLS+X*YNLS)*U(JM) - YNLS*R(JM)
          VS( 0) = VS( 0) + (XNLS+X*YNLS)*U(J)  - YNLS*R(J)
          VS( 1) = VS( 1) + (XNLS+X*YNLS)*U(JP) - YNLS*R(JP)
          VT(-2) = VT(-2) + XNLT*U(JMM)
          VT(-1) = VT(-1) + XNLT*U(JM)
```

```
        VT( 0) = VT( 0) + XNLT*U(J)
        VT( 1) = VT( 1) + XNLT*U(JP)
        VT( 2) = VT( 2) + XNLT*U(JPP)
      ENDIF

      RETURN
      END
*****************************************************************************
      SUBROUTINE VECTOR(TYPE,NLOC,NQ12)
C*    Partial-wave momentum-space potentials for vector exchange
C*       NQ12=0: inexact Fourier of quadratic spin-orbit Q12 operator
C*            1: exact Fourier Q12 from coordinate to momentum
C*       NLOC=0: no nonlocal (q^2+k^2/4) contributions
C*            1: nonlocal contributions in central potential only
C*            2: nonlocal contributions also in spin-spin (only for NP)
C-----------------------------------------------------------------------
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER TYPE*2
      COMMON/POTMOM/VC(-1:1),VS(-1:1),VT(-2:2),VLS(-2:2),VLA(-2:2),
     .              VQ(-3:3),VQP(-2:2)
      COMMON/JFAC/  JMMM,JMM,JM,J,JP,JPP,JPPP
      COMMON/QFAC/  QI2,QF2,QIQF,QI2F2,QI2F22,QIPF2
      COMMON/MESONM/AMPI,AMETA,AMETP, AMRO,AMOM,AMFI, AMA0,AMEP,AMF0,
     .              AMPIC,AMROC,AMA0C,AWPIC,AWROC,AWA0C, AVSC
      COMMON/COPLNG/ALPV,THPV,PV1, FPI,FETA,FETP,FPI2,FETA2,FETP2,
     .              ALVE,THV ,GV1, GRO,GOM,GFI,  GRO2,GOM2,GFI2,
     .              ALVM,     FV1, FRO,FOM,FFI,  FRO2,FOM2,FFI2,
     .              ALGS,THGS,GS1, GA0,GEP,GF0,  GA02,GEP2,GF02,
     .              GFPRO,GFPOM,GFPFI, GFMRO,GFMOM,GFMFI,
     .              FPIC2,GROC2,FROC2,GFPROC,GFMROC,GA0C2
      COMMON/YUKEFF/ARO,AMR1,AROC,AMRC1,AWRC1,BRO,AMR2,BROC,AMRC2,AWRC2,
     .              AVSC1,AVSC2, AEPS,AME1,BEPS,AME2
      COMMON/CUTOFF/ALAM,ALAMP,ALAMV,ALAMS
      COMMON/AMCOEF/ALF,REDM, AMY,AMY2,AMYI,AMYI2, AMN,AMN2,AMNI,AMNI2,
     .              AMYPN,AMYMN, AMYPN2,AMYMN2, AY2PN2,AY2MN2,
     .              AMYN,AMYNI,AMYNI2, AYPNI2,AYMNI2
      DIMENSION U(-3:12), R(-3:12), S(-3:12), G(-3:12), ELN(15), ERN(15)
      DATA U/16*0D0/, R/16*0D0/, S/16*0D0/, G/16*0D0/
      DATA PI/3.14159265358979323846D0/

      XCOM=0.5D0*QI2F2/QIQF
      ALAM2=ALAMV*ALAMV
      Y=2D0*QIQF/ALAM2
      JMAX=J+3
      KOM=1
      FAC=2D0*PI/QIQF

C**     Vector mesons: broad rho (2 Yukawa's), omega, fi
      DO 1000 IN=1,4
        IF(IN.EQ.1) THEN
          AMES=AMR1
          GV2 =GRO2 *ARO
          FV2 =FRO2 *ARO
          GFV =GFPRO*ARO
          GFM =GFMRO*ARO
        ELSEIF(IN.EQ.2) THEN
          AMES=AMR2
          GV2 =GRO2 *BRO
          FV2 =FRO2 *BRO
          GFV =GFPRO*BRO
          GFM =GFMRO*BRO
        ELSEIF(IN.EQ.3) THEN
          AMES=AMOM
          GV2 =GOM2
          FV2 =FOM2
          GFV =GFPOM
```

```
        GFM =GFMOM
      ELSEIF(IN.EQ.4) THEN
        AMES=AMFI
        GV2 =GFI2
        FV2 =FFI2
        GFV =GFPFI
        GFM =GFMFI
      ENDIF
      AMES2=AMES*AMES
      X=XCOM+0.5D0*AMES2/QIQF
      CALL SEX(X,Y,AMES/ALAMV,JMAX+1,KOM,ELN,ERN)
      XVC = FAC * (GV2*(1D0-AMYNI/2D0/ALF*QI2F2) - GFV/4D0/REDM*QI2F2
     .            + FV2*AMYNI/4D0*QI2F22)
      YVC = FAC * (GV2*AMYNI/ALF*QIQF+GFV/2D0/REDM*QIQF
     .            - FV2*AMYNI*QIQF*QI2F2)
      ZVC = FAC * FV2*AMYNI*QI2*QF2
      XVS =-FAC * ((GV2+AMYPN*GFV+4D0*AMYN*FV2)*AMYNI/6D0*QI2F2
     .            - FV2*AMYNI/12D0*QI2F22)
      YVS = FAC * ((GV2+AMYPN*GFV+4D0*AMYN*FV2)*AMYNI/3D0*QIQF
     .            - FV2*AMYNI/3D0*QIQF*QI2F2)
      ZVS = FAC * FV2*AMYNI/3D0*QI2*QF2
      XVT = FAC * ((GV2+AMYPN*GFV+4D0*AMYN*FV2)*AMYNI/4D0
     .            - FV2*AMYNI/8D0*QI2F2)
      YVT = FAC * FV2*AMYNI/4D0*QIQF
      XVLS=-FAC * ((GV2*(0.5D0+1D0/ALF)+AMYPN*GFV)*AMYNI
     .            - FV2*(0.5D0+1D0/ALF)*AMYNI*QI2F2)
      YVLS=-FAC * FV2*(1D0+2D0/ALF)*AMYNI*QIQF
      XVQ =-FAC * (GV2+4D0*AMYPN*GFV+8D0*AMYPN2*FV2)*AMYNI2/16D0
      XVQ2= XVQ * QIQF
      XVNL= FAC * GV2*(AYPNI2+AMYNI)/4D0*QI2F2
      DO 5 K=1,JMAX
        U(K)=ELN(K+1)
        R(K)=ERN(K+1)
        S(K)=(K*ERN(K)+(K+1)*ERN(K+2))/(2*K+1)
        G(K)=(ELN(K+2)-ELN(K))/(2*K+1)
5     CONTINUE
      U(0)=ELN(1)
      R(0)=ERN(1)
      S(0)=ERN(2)
      ARG=(QIPF2+AMES2)/ALAM2
      G(0)=(ELN(2)-ELN(1))-FDEXP(-QIPF2/ALAM2)*CE1(ARG)
      VC(-1) = VC(-1) + (XVC+X*YVC+X*X*ZVC)*U(JM)
     .                - (YVC+X*ZVC)*R(JM) - ZVC*S(JM)
      VC( 0) = VC( 0) + (XVC+X*YVC+X*X*ZVC)*U(J)
     .                - (YVC+X*ZVC)*R(J)  - ZVC*S(J)
      VC( 1) = VC( 1) + (XVC+X*YVC+X*X*ZVC)*U(JP)
     .                - (YVC+X*ZVC)*R(JP) - ZVC*S(JP)
      VS(-1) = VS(-1) + (XVS+X*YVS+X*X*ZVS)*U(JM)
     .                - (YVS+X*ZVS)*R(JM) - ZVS*S(JM)
      VS( 0) = VS( 0) + (XVS+X*YVS+X*X*ZVS)*U(J)
     .                - (YVS+X*ZVS)*R(J)  - ZVS*S(J)
      VS( 1) = VS( 1) + (XVS+X*YVS+X*X*ZVS)*U(JP)
     .                - (YVS+X*ZVS)*R(JP) - ZVS*S(JP)
      VT(-2) = VT(-2) + (XVT+X*YVT)*U(JMM)- YVT*R(JMM)
      VT(-1) = VT(-1) + (XVT+X*YVT)*U(JM) - YVT*R(JM)
      VT( 0) = VT( 0) + (XVT+X*YVT)*U(J)  - YVT*R(J)
      VT( 1) = VT( 1) + (XVT+X*YVT)*U(JP) - YVT*R(JP)
      VT( 2) = VT( 2) + (XVT+X*YVT)*U(JPP)- YVT*R(JPP)
      VLS(-2)= VLS(-2)+ (XVLS+X*YVLS)*U(JMM)- YVLS*R(JMM)
      VLS(-1)= VLS(-1)+ (XVLS+X*YVLS)*U(JM) - YVLS*R(JM)
      VLS( 0)= VLS( 0)+ (XVLS+X*YVLS)*U(J)  - YVLS*R(J)
      VLS( 1)= VLS( 1)+ (XVLS+X*YVLS)*U(JP) - YVLS*R(JP)
      VLS( 2)= VLS( 2)+ (XVLS+X*YVLS)*U(JPP)- YVLS*R(JPP)
      VQ(-3) = VQ(-3) +  XVQ*U(JMMM)
      VQ(-2) = VQ(-2) +  XVQ*U(JMM)
      VQ(-1) = VQ(-1) +  XVQ*U(JM)
```

```
        VQ( 0) = VQ( 0) +  XVQ*U(J)
        VQ( 1) = VQ( 1) +  XVQ*U(JP)
        VQ( 2) = VQ( 2) +  XVQ*U(JPP)
        VQ( 3) = VQ( 3) +  XVQ*U(JPPP)
        IF(NLOC.NE.0) THEN
          VC(-1) = VC(-1) + XVNL*U(JM)
          VC( 0) = VC( 0) + XVNL*U(J)
          VC( 1) = VC( 1) + XVNL*U(JP)
        ENDIF

        IF(NQ12.EQ.1) THEN
C**       Extra contribution from inverse Fourier transform of Q12
          VQP(-2) = VQP(-2) + XVQ2*G(JMM)
          VQP(-1) = VQP(-1) + XVQ2*G(JM)
          VQP( 0) = VQP( 0) + XVQ2*G(J)
          VQP( 1) = VQP( 1) + XVQ2*G(JP)
          VQP( 2) = VQP( 2) + XVQ2*G(JPP)
        ENDIF

 1000 CONTINUE

      IF(TYPE.EQ.'PP' .OR. TYPE.EQ.'NN') RETURN

C**     Charged vector: broad rho+ (2 Yukawa's)
      DO 2000 IN=1,2
        IF(IN.EQ.1) THEN
          AMES=AWRC1
          GV2 =GROC2 *AROC
          FV2 =FROC2 *AROC
          GFV =GFPROC*AROC
          GFM =GFMROC*AROC
          FFAC=FAC*AVSC1
        ELSEIF(IN.EQ.2) THEN
          AMES=AWRC2
          GV2 =GROC2 *BROC
          FV2 =FROC2 *BROC
          GFV =GFPROC*BROC
          GFM =GFMROC*BROC
          FFAC=FAC*AVSC2
        ENDIF
        AMES2=AMES*AMES
        X=XCOM+0.5D0*AMES2/QIQF
        CALL SEX(X,Y,AMES/ALAMV,JMAX+1,KOM,ELN,ERN)
        XVC = FAC * (GV2*(ALF-AMYNI/2D0*QI2F2) - ALF*
     .         (GFV*AMYPN+FV2*AMYMN2)*AYPNI2/8D0*QI2F2 + FV2*AMYNI/4D0*
     .         (1D0+(AY2MN2**2+2D0*AMYN*AMYMN2)*AMYNI2/16D0)*QI2F22)
        YVC = FAC * (GV2*AMYNI*QIQF + ALF*(GFV*AMYPN+FV2*AMYMN2)*
     .         AYPNI2/4D0*QIQF - FV2*AMYNI*
     .         (1D0+(AY2MN2**2+2D0*AMYN*AMYMN2)*AMYNI2/16D0)*QI2F2*QIQF)
        ZVC = FAC * FV2*AMYNI*
     .         (1D0+(AY2MN2**2+2D0*AMYN*AMYMN2)*AMYNI2/16D0)*QI2*QF2
        XVS = FAC * (-(GV2+AMYPN*GFV+AMYPN2*FV2)*
     .         (AMYNI/6D0+ALF*AMYMN2*AMYNI2/16D0)*QI2F2 +
     .         FV2/ALF*AYPNI2/24D0*QI2F22)
        YVS = FAC * ((GV2+AMYPN*GFV+AMYPN2*FV2)*
     .         (AMYNI/3D0+ALF*AMYMN2*AMYNI2/8D0)*QIQF -
     .         FV2/ALF*AYPNI2/6D0*QIQF*QI2F2)
        ZVS = FAC * FV2/ALF*AYPNI2/6D0*QI2*QF2
        XVT = FAC * ((GV2+AMYPN*GFV+AMYPN2*FV2)*AMYNI/4D0 -
     .         FV2/ALF*AYPNI2/16D0*QI2F2)
        YVT = FAC* FV2/ALF*AYPNI2/8D0*QIQF
        XVLS= FAC * (-(GV2*(2D0-ALF/2D0)+2D0*AY2PN2/AMYPN*GFV +
     .                2D0*AMYMN2*FV2)*AMYNI +
     .         FV2*((2D0/ALF-0.5D0)/ALF-AMYMN2*AMYNI/8D0)*AMYNI*QI2F2)
        YVLS=-FAC* FV2*((4D0/ALF-1D0)/ALF-AMYMN2*AMYNI/4D0)*AMYNI*QIQF
        XVQ =-FAC * ALF*(GV2+4D0*AMYPN*GFV+8D0*AMYPN2*FV2)*AMYNI2/16D0
```

```
        XVNLC= FAC * (ALF*GV2*(AYPNI2+AMYNI)/2D0+GFV*AMYMN2/AMYPN*AMYNI+
     .               FV2*(4D0/ALF-2D0*ALF*AY2PN2*AMYNI))*QI2F2/2D0
        XVNLS= FAC * ALF*AMYMN2*AMYNI2/8D0*(GV2+AMYPN*GFV+AMYPN2*FV2)*
     .              QI2F2
C**       Second part of vector-vector potential (same as scalar)
        XVC  = XVC  + FFAC * ALF*GV2*(1D0+AMYNI/4D0*QI2F2)
        YVC  = YVC  - FFAC * ALF*GV2*AMYNI/2D0*QIQF
        XVLS = XVLS + FFAC * ALF*GV2*AMYNI/2D0
        XVQ  = XVQ  - FFAC * ALF*GV2*AMYNI2/16D0
        XVNLC= XVNLC+ FFAC * ALF*GV2*(AYPNI2/4D0-AMYNI)*QI2F2/2D0

        XVQ2= XVQ * QIQF

        DO 6 K=1,JMAX
          U(K)=ELN(K+1)
          R(K)=ERN(K+1)
          S(K)=(K*ERN(K)+(K+1)*ERN(K+2))/(2*K+1)
          G(K)=(ELN(K+2)-ELN(K))/(2*K+1)
    6   CONTINUE
        U(0)=ELN(1)
        R(0)=ERN(1)
        S(0)=ERN(2)
        ARG=(QIPF2+AMES2)/ALAM2
        G(0)=(ELN(2)-ELN(1))-FDEXP(-QIPF2/ALAM2)*CE1(ARG)
        VC(-1) = VC(-1) + (XVC+X*YVC+X*X*ZVC)*U(JM)
     .                  - (YVC+X*ZVC)*R(JM) - ZVC*S(JM)
        VC( 0) = VC( 0) + (XVC+X*YVC+X*X*ZVC)*U(J)
     .                  - (YVC+X*ZVC)*R(J)  - ZVC*S(J)
        VC( 1) = VC( 1) + (XVC+X*YVC+X*X*ZVC)*U(JP)
     .                  - (YVC+X*ZVC)*R(JP) - ZVC*S(JP)
        VS(-1) = VS(-1) + (XVS+X*YVS+X*X*ZVS)*U(JM)
     .                  - (YVS+X*ZVS)*R(JM) - ZVS*S(JM)
        VS( 0) = VS( 0) + (XVS+X*YVS+X*X*ZVS)*U(J)
     .                  - (YVS+X*ZVS)*R(J)  - ZVS*S(J)
        VS( 1) = VS( 1) + (XVS+X*YVS+X*X*ZVS)*U(JP)
     .                  - (YVS+X*ZVS)*R(JP) - ZVS*S(JP)
        VT(-2) = VT(-2) + (XVT+X*YVT)*U(JMM)- YVT*R(JMM)
        VT(-1) = VT(-1) + (XVT+X*YVT)*U(JM) - YVT*R(JM)
        VT( 0) = VT( 0) + (XVT+X*YVT)*U(J)  - YVT*R(J)
        VT( 1) = VT( 1) + (XVT+X*YVT)*U(JP) - YVT*R(JP)
        VT( 2) = VT( 2) + (XVT+X*YVT)*U(JPP)- YVT*R(JPP)
        VLS(-2)= VLS(-2)+ (XVLS+X*YVLS)*U(JMM)- YVLS*R(JMM)
        VLS(-1)= VLS(-1)+ (XVLS+X*YVLS)*U(JM) - YVLS*R(JM)
        VLS( 0)= VLS( 0)+ (XVLS+X*YVLS)*U(J)  - YVLS*R(J)
        VLS( 1)= VLS( 1)+ (XVLS+X*YVLS)*U(JP) - YVLS*R(JP)
        VLS( 2)= VLS( 2)+ (XVLS+X*YVLS)*U(JPP)- YVLS*R(JPP)
        VQ(-3) = VQ(-3) +  XVQ*U(JMMM)
        VQ(-2) = VQ(-2) +  XVQ*U(JMM)
        VQ(-1) = VQ(-1) +  XVQ*U(JM)
        VQ( 0) = VQ( 0) +  XVQ*U(J)
        VQ( 1) = VQ( 1) +  XVQ*U(JP)
        VQ( 2) = VQ( 2) +  XVQ*U(JPP)
        VQ( 3) = VQ( 3) +  XVQ*U(JPPP)
        IF(NLOC.NE.0) THEN
          VC(-1) = VC(-1) + XVNLC*U(JM)
          VC( 0) = VC( 0) + XVNLC*U(J)
          VC( 1) = VC( 1) + XVNLC*U(JP)
        ENDIF
        IF(NLOC.EQ.2) THEN
          VS(-1) = VS(-1) + XVNLS*U(JM)
          VS( 0) = VS( 0) + XVNLS*U(J)
          VS( 1) = VS( 1) + XVNLS*U(JP)
        ENDIF

        IF(NQ12.EQ.1) THEN
C**       Extra contribution from inverse Fourier transform of Q12
```

```
          VQP(-2) = VQP(-2) + XVQ2*G(JMM)
          VQP(-1) = VQP(-1) + XVQ2*G(JM)
          VQP( 0) = VQP( 0) + XVQ2*G(J)
          VQP( 1) = VQP( 1) + XVQ2*G(JP)
          VQP( 2) = VQP( 2) + XVQ2*G(JPP)
        ENDIF

 2000 CONTINUE

      RETURN
      END
*****************************************************************************
      SUBROUTINE SCALAR(TYPE,NLOC,NQ12)
C*    Partial-wave momentum-space potentials for scalar exchange
C*       NQ12=0: inexact Fourier of quadratic spin-orbit Q12 operator
C*            1: exact Fourier Q12 from coordinate to momentum
C*       NLOC=0: no nonlocal (q^2+k^2/4) contributions
C*            1: nonlocal contributions in central potential
C-----------------------------------------------------------------------------
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER TYPE*2
      COMMON/POTMOM/VC(-1:1),VS(-1:1),VT(-2:2),VLS(-2:2),VLA(-2:2),
     .              VQ(-3:3),VQP(-2:2)
      COMMON/JFAC/  JMMM,JMM,JM,J,JP,JPP,JPPP
      COMMON/QFAC/  QI2,QF2,QIQF,QI2F2,QI2F22,QIPF2
      COMMON/MESONM/AMPI,AMETA,AMETP, AMRO,AMOM,AMFI, AMA0,AMEP,AMF0,
     .              AMPIC,AMROC,AMA0C,AWPIC,AWROC,AWA0C, AVSC
      COMMON/COPLNG/ALPV,THPV,PV1, FPI,FETA,FETP,FPI2,FETA2,FETP2,
     .              ALVE,THV ,GV1, GRO,GOM,GFI,  GRO2,GOM2,GFI2,
     .              ALVM,     FV1, FRO,FOM,FFI,  FRO2,FOM2,FFI2,
     .              ALGS,THGS,GS1, GA0,GEP,GF0,  GA02,GEP2,GF02,
     .              GFPRO,GFPOM,GFPFI, GFMRO,GFMOM,GFMFI,
     .              FPIC2,GROC2,FROC2,GFPROC,GFMROC,GA0C2
      COMMON/YUKEFF/ARO,AMR1,AROC,AMRC1,AWRC1,BRO,AMR2,BROC,AMRC2,AWRC2,
     .              AVSC1,AVSC2, AEPS,AME1,BEPS,AME2
      COMMON/CUTOFF/ALAM,ALAMP,ALAMV,ALAMS
      COMMON/AMCOEF/ALF,REDM, AMY,AMY2,AMYI,AMYI2, AMN,AMN2,AMNI,AMNI2,
     .              AMYPN,AMYMN, AMYPN2,AMYMN2, AY2PN2,AY2MN2,
     .              AMYN,AMYNI,AMYNI2, AYPNI2,AYMNI2
      DIMENSION U(-3:12), R(-3:12), G(-3:12), ELN(15), ERN(15)
      DATA U/16*0D0/, R/16*0D0/, G/16*0D0/
      DATA PI/3.14159265358979323846D0/

      XCOM=0.5D0*QI2F2/QIQF
      ALAM2=ALAMS*ALAMS
      Y=2D0*QIQF/ALAM2
      JMAX=J+3
      KOM=1
      FAC=2D0*PI/QIQF

C**     Scalar mesons: a0, broad epsilon (2 Yukawa's), f0
      DO 1000 IN=1,4
        IF(IN.EQ.1) THEN
          AMES=AMA0
          GS2 =GA02
        ELSEIF(IN.EQ.2) THEN
          AMES=AME1
          GS2 =GEP2*AEPS
        ELSEIF(IN.EQ.3) THEN
          AMES=AME2
          GS2 =GEP2*BEPS
        ELSEIF(IN.EQ.4) THEN
          AMES=AMF0
          GS2 =GF02
        ENDIF
        AMES2=AMES*AMES
```

```
        X=XCOM+0.5D0*AMES2/QIQF
        CALL SEX(X,Y,AMES/ALAMS,JMAX+1,KOM,ELN,ERN)
        XSC =-FAC * GS2*(1D0+AYPNI2/8D0*QI2F2)
        YSC = FAC * GS2*AYPNI2/4D0*QIQF
        XSLS=-FAC * GS2*AYPNI2/4D0
        XSQ = FAC * GS2*AMYNI2/16D0
        XSNL= FAC * GS2*AMYNI/4D0*QI2F2
        XSQ2= XSQ * QIQF
        DO 5 K=1,JMAX
          U(K)=ELN(K+1)
          R(K)=ERN(K+1)
          G(K)=(ELN(K+2)-ELN(K))/(2*K+1)
    5   CONTINUE
        U(0)=ELN(1)
        R(0)=ERN(1)
        ARG=(QIPF2+AMES2)/ALAM2
        G(0)=(ELN(2)-ELN(1))-FDEXP(-QIPF2/ALAM2)*CE1(ARG)
        VC(-1) = VC(-1) + (XSC+X*YSC)*U(JM) - YSC*R(JM)
        VC( 0) = VC( 0) + (XSC+X*YSC)*U(J)  - YSC*R(J)
        VC( 1) = VC( 1) + (XSC+X*YSC)*U(JP) - YSC*R(JP)
        VLS(-2)= VLS(-2)+ XSLS*U(JMM)
        VLS(-1)= VLS(-1)+ XSLS*U(JM)
        VLS( 0)= VLS( 0)+ XSLS*U(J)
        VLS( 1)= VLS( 1)+ XSLS*U(JP)
        VLS( 2)= VLS( 2)+ XSLS*U(JPP)
        VQ(-3) = VQ(-3) + XSQ*U(JMMM)
        VQ(-2) = VQ(-2) + XSQ*U(JMM)
        VQ(-1) = VQ(-1) + XSQ*U(JM)
        VQ( 0) = VQ( 0) + XSQ*U(J)
        VQ( 1) = VQ( 1) + XSQ*U(JP)
        VQ( 2) = VQ( 2) + XSQ*U(JPP)
        VQ( 3) = VQ( 3) + XSQ*U(JPPP)
        IF(NLOC.NE.0) THEN
          VC(-1) = VC(-1) + XSNL*U(JM)
          VC( 0) = VC( 0) + XSNL*U(J)
          VC( 1) = VC( 1) + XSNL*U(JP)
        ENDIF

        IF(NQ12.EQ.1) THEN
C**       Extra contribution from inverse Fourier transform of Q12
          VQP(-2) = VQP(-2) + XSQ2*G(JMM)
          VQP(-1) = VQP(-1) + XSQ2*G(JM)
          VQP( 0) = VQP( 0) + XSQ2*G(J)
          VQP( 1) = VQP( 1) + XSQ2*G(JP)
          VQP( 2) = VQP( 2) + XSQ2*G(JPP)
        ENDIF

 1000 CONTINUE

      IF(TYPE.EQ.'PP' .OR. TYPE.EQ.'NN') RETURN

C**     Charged scalar: a0+
      AMES=AWA0C
      GS2 =GA0C2
      AMES2=AMES*AMES
      X=XCOM+0.5D0*AMES2/QIQF
      CALL SEX(X,Y,AMES/ALAMS,JMAX+1,KOM,ELN,ERN)
      XSC =-FAC * ALF*GS2*(1D0+AMYNI/4D0*QI2F2)
      YSC = FAC * ALF*GS2*AMYNI/2D0*QIQF
      XSLS=-FAC * ALF*GS2*AMYNI/2D0
      XSQ = FAC * ALF*GS2*AMYNI2/16D0
      XSNL=-FAC * ALF*GS2*(AYPNI2/4D0-AMYNI)*QI2F2/2D0
      XSQ2= XSQ * QIQF
      DO 6 K=1,JMAX
        U(K)=ELN(K+1)
        R(K)=ERN(K+1)
```

```
        G(K)=(ELN(K+2)-ELN(K))/(2*K+1)
    6 CONTINUE
      U(0)=ELN(1)
      R(0)=ERN(1)
      ARG=(QIPF2+AMES2)/ALAM2
      G(0)=(ELN(2)-ELN(1))-FDEXP(-QIPF2/ALAM2)*CE1(ARG)
      VC(-1) = VC(-1) + (XSC+X*YSC)*U(JM) - YSC*R(JM)
      VC( 0) = VC( 0) + (XSC+X*YSC)*U(J)  - YSC*R(J)
      VC( 1) = VC( 1) + (XSC+X*YSC)*U(JP) - YSC*R(JP)
      VLS(-2)= VLS(-2)+ XSLS*U(JMM)
      VLS(-1)= VLS(-1)+ XSLS*U(JM)
      VLS( 0)= VLS( 0)+ XSLS*U(J)
      VLS( 1)= VLS( 1)+ XSLS*U(JP)
      VLS( 2)= VLS( 2)+ XSLS*U(JPP)
      VQ(-3) = VQ(-3) + XSQ*U(JMMM)
      VQ(-2) = VQ(-2) + XSQ*U(JMM)
      VQ(-1) = VQ(-1) + XSQ*U(JM)
      VQ( 0) = VQ( 0) + XSQ*U(J)
      VQ( 1) = VQ( 1) + XSQ*U(JP)
      VQ( 2) = VQ( 2) + XSQ*U(JPP)
      VQ( 3) = VQ( 3) + XSQ*U(JPPP)
      IF(NLOC.NE.0) THEN
        VC(-1) = VC(-1) + XSNL*U(JM)
        VC( 0) = VC( 0) + XSNL*U(J)
        VC( 1) = VC( 1) + XSNL*U(JP)
      ENDIF

      IF(NQ12.EQ.1) THEN
C**     Extra contribution from inverse Fourier transform of Q12
        VQP(-2) = VQP(-2) + XSQ2*G(JMM)
        VQP(-1) = VQP(-1) + XSQ2*G(JM)
        VQP( 0) = VQP( 0) + XSQ2*G(J)
        VQP( 1) = VQP( 1) + XSQ2*G(JP)
        VQP( 2) = VQP( 2) + XSQ2*G(JPP)
      ENDIF

      RETURN
      END
*****************************************************************************
      SUBROUTINE DIFRAC(TYPE,NLOC,NQ12,ISIGN,FACISO)
C*    Partial-wave momentum-space potentials for diffractive part
C*       NQ12=0: inexact Fourier of quadratic spin-orbit Q12 operator
C*            1: exact Fourier Q12 from coordinate to momentum
C*       NLOC=0: no nonlocal (q^2+k^2/4) contributions
C*            1: nonlocal contributions in central potential
C-----------------------------------------------------------------------
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER TYPE*2
      COMMON/POTMOM/VC(-1:1),VS(-1:1),VT(-2:2),VLS(-2:2),VLA(-2:2),
     .              VQ(-3:3),VQP(-2:2)
      COMMON/JFAC/  JMMM,JMM,JM,J,JP,JPP,JPPP
      COMMON/QFAC/  QI2,QF2,QIQF,QI2F2,QI2F22,QIPF2
      COMMON/POMRON/GPOM,FPOM2,AMPOM,AMPOM2,AMPOM4,
     .              GA2D,FA2D2,AMA2D,AMA2D2,AMA2D4
      COMMON/AMCOEF/ALF,REDM, AMY,AMY2,AMYI,AMYI2, AMN,AMN2,AMNI,AMNI2,
     .              AMYPN,AMYMN, AMYPN2,AMYMN2, AY2PN2,AY2MN2,
     .              AMYN,AMYNI,AMYNI2, AYPNI2,AYMNI2
      DIMENSION R(-3:12), S(-3:12), G(-3:12), ELN(15), ERN(15)
      DATA R/16*0D0/, S/16*0D0/, G/16*0D0/
      DATA PI/3.14159265358979323846D0/

      XCOM=0.5D0*QI2F2/QIQF
      JMAX=J+3
      KOM=2
      FAC=4D0*PI
```

```
C**     Diffractive contribution: pomeron (f, f', A2), pomeron'
      DO 1000 IN=1,2
        IF(IN.EQ.1) THEN
          AMES2=AMPOM2
          GD2=FPOM2
        ELSEIF(IN.EQ.2) THEN
          AMES2=AMA2D2
          GD2=FA2D2*ISIGN
        ENDIF
        Y=0.5D0*QIQF/AMES2
        X=XCOM
        CALL SEX(X,Y,0D0,JMAX+1,KOM,ELN,ERN)
        XDC = FAC * GD2*(1D0+AYPNI2/8D0*QI2F2)
        YDC =-FAC * GD2*AYPNI2/4D0*QIQF
        XDLS= FAC * GD2*AYPNI2/4D0
        XDQ =-FAC * GD2*AMYNI2/16D0
        XDNL=-FAC * GD2*AMYNI/4D0*QI2F2
        XDQ2= XDQ * QIQF
        DO 5 K=1,JMAX
          R(K)=ERN(K+1)
          S(K)=(K*ERN(K)+(K+1)*ERN(K+2))/(2*K+1)
          G(K)=(ERN(K+2)-ERN(K))/(2*K+1)
    5   CONTINUE
        R(0)=ERN(1)
        S(0)=ERN(2)
        G(0)=(ERN(2)-ERN(1))-2D0*FDEXP(-QIPF2/4D0/AMES2)*AMES2/QIQF
        VC(-1) = VC(-1) + XDC*R(JM) + YDC*S(JM)
        VC( 0) = VC( 0) + XDC*R(J)  + YDC*S(J)
        VC( 1) = VC( 1) + XDC*R(JP) + YDC*S(JP)
        VLS(-2)= VLS(-2)+ XDLS*R(JMM)
        VLS(-1)= VLS(-1)+ XDLS*R(JM)
        VLS( 0)= VLS( 0)+ XDLS*R(J)
        VLS( 1)= VLS( 1)+ XDLS*R(JP)
        VLS( 2)= VLS( 2)+ XDLS*R(JPP)
        VQ(-3) = VQ(-3) + XDQ*R(JMMM)
        VQ(-2) = VQ(-2) + XDQ*R(JMM)
        VQ(-1) = VQ(-1) + XDQ*R(JM)
        VQ( 0) = VQ( 0) + XDQ*R(J)
        VQ( 1) = VQ( 1) + XDQ*R(JP)
        VQ( 2) = VQ( 2) + XDQ*R(JPP)
        VQ( 3) = VQ( 3) + XDQ*R(JPPP)
        IF(NLOC.NE.0) THEN
          VC(-1) = VC(-1) + XDNL*R(JM)
          VC( 0) = VC( 0) + XDNL*R(J)
          VC( 1) = VC( 1) + XDNL*R(JP)
        ENDIF

        IF(NQ12.EQ.1) THEN
C**       Extra contribution from inverse Fourier transform of Q12
          VQP(-2) = VQP(-2) + XDQ2*G(JMM)
          VQP(-1) = VQP(-1) + XDQ2*G(JM)
          VQP( 0) = VQP( 0) + XDQ2*G(J)
          VQP( 1) = VQP( 1) + XDQ2*G(JP)
          VQP( 2) = VQP( 2) + XDQ2*G(JPP)
        ENDIF

 1000 CONTINUE

      IF(TYPE.EQ.'PP' .OR. TYPE.EQ.'NN') RETURN

C**     Charged diffractive contribution: pomeron'
      AMES2=AMA2D2
      GD2=FA2D2*FACISO
      Y=0.5D0*QIQF/AMES2
      X=XCOM
      CALL SEX(X,Y,0D0,JMAX+1,KOM,ELN,ERN)
```

```
      XDC = FAC * ALF*GD2*(1D0+AMYNI/4D0*QI2F2)
      YDC =-FAC * ALF*GD2*AMYNI/2D0*QIQF
      XDLS= FAC * ALF*GD2*AMYNI/2D0
      XDQ =-FAC * ALF*GD2*AMYNI2/16D0
      XDNL= FAC * ALF*GD2*(AYPNI2/4D0-AMYNI)*QI2F2/2D0
      XDQ2= XDQ * QIQF
      DO 6 K=1,JMAX
        R(K)=ERN(K+1)
        S(K)=(K*ERN(K)+(K+1)*ERN(K+2))/(2*K+1)
        G(K)=(ERN(K+2)-ERN(K))/(2*K+1)
    6 CONTINUE
      R(0)=ERN(1)
      S(0)=ERN(2)
      G(0)=(ERN(2)-ERN(1))-2D0*FDEXP(-QIPF2/4D0/AMES2)*AMES2/QIQF
      VC(-1) = VC(-1) + XDC*R(JM) + YDC*S(JM)
      VC( 0) = VC( 0) + XDC*R(J)  + YDC*S(J)
      VC( 1) = VC( 1) + XDC*R(JP) + YDC*S(JP)
      VLS(-2)= VLS(-2)+ XDLS*R(JMM)
      VLS(-1)= VLS(-1)+ XDLS*R(JM)
      VLS( 0)= VLS( 0)+ XDLS*R(J)
      VLS( 1)= VLS( 1)+ XDLS*R(JP)
      VLS( 2)= VLS( 2)+ XDLS*R(JPP)
      VQ(-3) = VQ(-3) + XDQ*R(JMMM)
      VQ(-2) = VQ(-2) + XDQ*R(JMM)
      VQ(-1) = VQ(-1) + XDQ*R(JM)
      VQ( 0) = VQ( 0) + XDQ*R(J)
      VQ( 1) = VQ( 1) + XDQ*R(JP)
      VQ( 2) = VQ( 2) + XDQ*R(JPP)
      VQ( 3) = VQ( 3) + XDQ*R(JPPP)
      IF(NLOC.NE.0) THEN
        VC(-1) = VC(-1) + XDNL*R(JM)
        VC( 0) = VC( 0) + XDNL*R(J)
        VC( 1) = VC( 1) + XDNL*R(JP)
      ENDIF

      IF(NQ12.EQ.1) THEN
C**     Extra contribution from inverse Fourier transform of Q12
        VQP(-2) = VQP(-2) + XDQ2*G(JMM)
        VQP(-1) = VQP(-1) + XDQ2*G(JM)
        VQP( 0) = VQP( 0) + XDQ2*G(J)
        VQP( 1) = VQP( 1) + XDQ2*G(JP)
        VQP( 2) = VQP( 2) + XDQ2*G(JPP)
      ENDIF

      RETURN
      END
*****************************************************************************
      FUNCTION FDEXP(X)
      IMPLICIT REAL*8(A-Z)
      IF(X.LE.-100D0) THEN
        FDEXP=0D0
      ELSE
        FDEXP=DEXP(X)
      ENDIF
      RETURN
      END
*****************************************************************************
```

```
      SUBROUTINE SEX(X,Y,AMES,J,KOM,ELN,ERN)
C-----------------------------------------------------------------------
C     SEX calculates :   0.5*DEXP(AMES**2)*
C                     INTEGRAL(-1,+1) (PN(Z)/(X-Z)*F(X,Y)) DZ for LN
C                                     (PN(Z)*F(X,Y))       DZ for RN
C                     PN Legendre function
C                     F(X,Y)=DEXP(Y*(Z-X))
C     argument 1<|X|, via Pade approximant for X*LOG(X) [function GRENS]
C
C     METHOD:
C            Expand PN(Z) ( N=0,12 max) in powers of Z and calculate :
C
C            INTEGRAL(-1,+1)(DEXP(Y(Z-X)*Z**N) DZ       for TN
C            INTEGRAL(-1,+1)(DEXP(Y(Z-X)*Z**N/(X-Z)) DZ for UN
C
C            INPUT:    X : (P**2+Q**2+M**2)/(2*P*Q)
C                      Y : 2*P*Q/(CUTOFF**2)
C                      J : (total angular momentum +3)+1 for V,S,D meson
C                    KOM : 1 calculate ELN and ERN
C                          2 calculate only ERN (for "DIFRAC" routine)
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION ELN(15),ERN(15),ALN(49),BLN(7)
      DIMENSION UN(14),TN(35)
      DATA XYHH,XH,XMIN,XMED,EPS,EPS2,YH/
     .     1D20,1D20,-160D0,5D0,1D-13,1D-6,1D20/
```

```
      DATA ALN/1D0, 1D0, 3D0,-1D0, 5D0,-3D0,
     .         35D0,-30D0,3D0, 63D0,-70D0,15D0,
     .         231D0,-315D0,105D0,-5D0, 429D0,-693D0,315D0,-35D0,
     .         6435D0,-12012D0,6930D0,-1260D0,35D0,
     .         12155D0,-25740D0,18018D0,-4620D0,315D0,
     .         46189D0,-109395D0,90090D0,-30030D0,3465D0,-63D0,
     .         88179D0,-230945D0,218790D0,-90090D0,15015D0,-693D0,
     .         676039D0,-1939938D0,2078505D0,-1021020D0,225225D0,
     .                  -18018D0,231D0/
      DATA BLN/1D0,2D0,8D0,16D0,128D0,256D0,1024D0/

C-NOTE *** dimension UN=J+1, TN=J+2+20
      IF(J.GT.13) WRITE(*,*)
     .   ' *** SEX: J exceeds allowable maximum of JMAX=9 (+3+1)'
      IF(J.GT.13) STOP

C     CALL ERRSET(208,256,-1,1)
      DO 500 I=1,15
        ELN(I)=0D0
        ERN(I)=0D0
  500 CONTINUE
C**   Calculate basic quantities
      YM=Y-Y*X
      YP=Y+Y*X
      YX=Y*X
      IF((YM-AMES*AMES).LT.XMIN) RETURN
      ENU=FDEXP(AMES*AMES)
      EP=0D0
      IF(YM.GT.XMIN) EP=FDEXP(YM)*ENU/2D0
      EM=0D0
      IF(-YP.GT.XMIN) EM=FDEXP(-YP)*ENU/2D0

C**   If same momenta but different mesonmass: calculate only UN again
      XYH=YX-AMES*AMES
      IF(DABS(XYH-XYHH).LT.1D-10*XYH .AND. Y.EQ.YH) GOTO 50
    9 XYHH=XYH
      YH=Y
      SH=(EP-EM)/Y
      CH=(EP+EM)/Y
      MAX=J+2
      IF(X.GT.XMED) MAX=MAX+20
      AMAX=MAX
      CHECK=0.434*(-AMAX+(Y+AMAX+0.5)*DLOG(1+(AMAX+1)/Y))
      IF(CHECK.GT.7D0) GOTO 10

C**   If CHECK > 1D7 use recurrence relation backward

C**   Calculate TN with recurrence relation forward
      TN(1)=SH
      DO 1 I=2,MAX
        TN(I)=CH-(I-1)*TN(I-1)/Y
        DUM=CH
        CH=SH
        SH=DUM
    1 CONTINUE
      GOTO 51

C**   Calculate TN with recurrence relation backward
C*    Calculation starting point
   10 IF(Y.LT.1D-04) MMAX=AMAX+4
      IF(Y.LT.1D-04) GOTO 13
      GRENS1=9.197/Y
      GRENS2=AMAX*DLOG((AMAX+1)/2.718/Y)/2.718/Y+8.829/Y
      GRENS1=GRENS(GRENS1)*2.718*Y
      GRENS2=GRENS(GRENS2)*2.718*Y
      K1=GRENS1+1
```

```
      K2=GRENS2+1
      MMAX=MAX0(K1,K2)
C**   CHECK: on loss of more than 7 significant digits
      CHECK=0.434*(Y*DLOG(Y)+Y+1-(Y+0.5)*DLOG(Y+1))
C-V   IF(CHECK.GT.7D0) PRINT 5003,CHECK

C*    Start backward recursion
   13 T=0D0
      IF(MOD(MMAX,2).EQ.0) MMAX=MMAX+1
      SH=ENU*DSINH(Y)*FDEXP(-YX)
      CH=EP+EM
      DO 11 II=2,MMAX
        I=MMAX-II+2
        T=(SH-Y*T)/(I-1)
        IF(I.LE.(MAX+1)) TN(I-1)=T
        DUM=SH
        SH=CH
        CH=DUM
   11 CONTINUE
      GOTO 51

C**   Calculation UN functions
   50 IF(X.GT.XMED .AND. XH.LE.XMED) GOTO 9

C**   If mesonmass differs, but X has passed critical value
C**   more TN's have to be calculated

   51 XH=X
C**   For diffractive contribution calculation of RN is enough
      IF(KOM.EQ.2) GOTO 800

      IF(X.GT.XMED) GOTO 60

C**   Calculation for X < XMED using forward recurrence relation
      CE1P=EP*CE1(-YM)
      CE1M=EM*CE1(YP)
C**   CHECK: on loss of more than 7 significant digits
      IF(CE1P.NE.0D0 .OR. CE1M.NE.0D0) THEN
        IF(DABS((CE1P-CE1M)/(CE1P+CE1M)).LT.EPS2) PRINT 5001
      ENDIF
      UN(1)=CE1P-CE1M
      MAX=J+1
      DO 52 I=2,MAX
        UN(I)=X*UN(I-1)-TN(I-1)
   52 CONTINUE
      GOTO 600

C**   Calculation for X > XMED using backward recurrence relation
   60 MAX=J+1

C**   First calculate UN(MAX) with expansion in TN
      SUM=TN(MAX)/X
      IF(SUM.EQ.0D0) GOTO 62
      XI=1D0/X
      DSUM=0D0
      DO 61 I=1,20
        SUM=SUM+DSUM
        XI=XI/X
        DSUM=TN(MAX+I)*XI
        IF(DABS(DSUM/SUM).LT.EPS) GOTO 62
   61 CONTINUE
      PRINT 5002
   62 UN(MAX)=SUM
      DO 63 II=2,MAX
        I=MAX+2-II
        UN(I-1)=(UN(I)+TN(I-1))/X
```

```
   63 CONTINUE
      GOTO 600

C**   Form linear combinations to obtain LN and RN
  600 MAX=J+1
      NB=1
      DO 31 L1=1,MAX
        II=L1+MOD(L1,2)
        DO 41 N1=1,L1,2
          ELN(L1)=ELN(L1)+ALN(NB+(N1-1)/2)*UN(L1+1-N1)/BLN(II/2)
          ERN(L1)=ERN(L1)+ALN(NB+(N1-1)/2)*TN(L1+1-N1)/BLN(II/2)
   41   CONTINUE
        NB=NB+II/2
   31 CONTINUE
C     CALL ERRSET(208,256,1,1)
      RETURN
  800 MAX=J+1
      NB=1
      DO 32 L1=1,MAX
        II=L1+MOD(L1,2)
        DO 42 N1=1,L1,2
          ERN(L1)=ERN(L1)+ALN(NB+(N1-1)/2)*TN(L1+1-N1)/BLN(II/2)
   42   CONTINUE
        NB=NB+II/2
   32 CONTINUE
C     CALL ERRSET(208,256,1,1)
      RETURN
 5001 FORMAT(1X,'**** SEX **** Loss of significant digits UN(1) ')
 5002 FORMAT(1X,'**** SEX **** UN MAX not accurate  ')
 5003 FORMAT(1X,'**** SEX **** TN not accurate ; loss of :',D10.3,
     .          ' decimal digits ')
      END
*****************************************************************************
      DOUBLE PRECISION FUNCTION GRENS(X)
      IMPLICIT REAL*8 (A-H,O-Z)
      DATA A0,A1,A2,A3/-0.589654,-0.0595734,0.649227,0.1809910/

      IF(X.GT.1.2D6) THEN
        GRENS=X**0.847
      ELSEIF(X.GT.9D4) THEN
        GRENS=X**0.825
      ELSEIF(X.GT.6900D0) THEN
        GRENS=X**0.806
      ELSEIF(X.GT.460D0) THEN
        GRENS=X**0.781
      ELSEIF(X.GT.23D0) THEN
        GRENS=X**0.751
      ELSE
        GRENS=(X*A3-A1+DSQRT((A1-X*A3)*(A1-X*A3)-4D0*A2*(A0-X)))/2D0/A2
      ENDIF

      RETURN
      END
*****************************************************************************
      DOUBLE PRECISION FUNCTION CE1(X)
C**     CE1 calculates the function : DEXP(X) * E1(X)
      IMPLICIT REAL*8 (A-H,O-Z)
      DATA EPS/10D-14/
      DATA GAM/0.57721566490153286lD0/
      DATA AP0,AP1,AP2,AP3,AP4,AP5,AP6 /
     .     0.463996004278035D+01, 0.127788778637147D+03,
     .     0.735910238555843D+03, 0.139583023127254D+04,
     .     0.101614779141469D+04, 0.286647946600883D+03,
     .     0.256489038620717D+02/
      DATA AQ1,AQ2,AQ3,AQ4,AQ5,AQ6,AQ7 /
     .     0.512251050448444D+02, 0.503800829553457D+03,
```

```
     .     0.165169408854742D+04, 0.220150290642078D+04,
     .     0.127719811988873D+04, 0.312294439564262D+03,
     .     0.256489625816454D+02/
      DATA BP0,BP1,BP2,BP3,BP4,BP5     /
     .     0.335956527252693D+01, 0.204955591333077D+02,
     .     0.267757325223533D+02, 0.112883678215773D+02,
     .     0.164680678114210D+01, 0.655193572680895D-01/
      DATA BQ1,BQ2,BQ3,BQ4,BQ5,BQ6     /
     .     0.143836492361913D+02, 0.400563387674630D+02,
     .     0.366148021121537D+02, 0.128696120312766D+02,
     .     0.171232738644327D+01, 0.655193403549186D-01/
      DATA CP0,CP1,CP2,CP3,CP4         /
     .     0.298014627030798D+01, 0.113803314436134D+02,
     .     0.947288802836929D+01, 0.247747160891423D+01,
     .     0.188516317695352D+00/
      DATA CQ1,CQ2,CQ3,CQ4,CQ5         /
     .     0.988019055335016D+01, 0.189408176576544D+02,
     .     0.117618585876339D+02, 0.266598761793551D+01,
     .     0.188516320637495D+00/
      DATA DP0,DP1,DP2,DP3   /
     .     0.242331331460798D+01, 0.432777141801875D+01,
     .     0.160959648287707D+01, 0.148720388893508D+00/
      DATA DQ1,DQ2,DQ3,DQ4   /
     .     0.558734308280980D+01, 0.578865453197840D+01,
     .     0.175831677540018D+01, 0.148720389489176D+00/
      DATA EP0,EP1,EP2,EP3   /
     .     0.226526458912153D+01, 0.332000741007556D+01,
     .     0.104761178441346D+01, 0.837423061701825D-01/
      DATA EQ1,EQ2,EQ3,EQ4   /
     .     0.478887726713541D+01, 0.428387700117901D+01,
     .     0.113135408983342D+01, 0.837423061723804D-01/
      DATA FP0,FP1,FP2,FP3   /
     .     0.190053654321203D+01, 0.151285969203750D+01,
     .     0.205314346964057D+00, 0.264152351883344D-03/
      DATA FQ1,FQ2,FQ3,FQ4   /
     .     0.320887608816311D+01, 0.171790987670629D+01,
     .     0.205578499347658D+00, 0.264152351839874D-03/

      PQX=1D0
      IF(X.LT.1D0) THEN
        E1=-GAM-DLOG(X)+X
        N=1
        IX=1
        TERM=X
    1   N=N+1
        IX=-IX
        TERM=TERM*X/N
        IF(TERM.LT.EPS) GOTO 10
        E1=E1+IX*TERM/N
        GOTO 1
   10   PPX=FDEXP(X)*E1
      ELSEIF(X.LT.3D0) THEN
        PPX=AP0+X*(AP1+X*(AP2+X*(AP3+X*(AP4+X*(AP5+X*AP6)))))
        PQX=1D0+X*(AQ1+X*(AQ2+X*(AQ3+X*(AQ4+X*(AQ5+X*(AQ6+X*AQ7))))))
      ELSEIF(X.LT.6D0) THEN
        PPX=BP0+X*(BP1+X*(BP2+X*(BP3+X*(BP4+X*BP5))))
        PQX=1D0+X*(BQ1+X*(BQ2+X*(BQ3+X*(BQ4+X*(BQ5+X*BQ6)))))
      ELSEIF(X.LT.14D0) THEN
        PPX=CP0+X*(CP1+X*(CP2+X*(CP3+X*CP4)))
        PQX=1D0+X*(CQ1+X*(CQ2+X*(CQ3+X*(CQ4+X*CQ5))))
      ELSEIF(X.LT.25D0) THEN
        PPX=DP0+X*(DP1+X*(DP2+X*DP3))
        PQX=1D0+X*(DQ1+X*(DQ2+X*(DQ3+X*DQ4)))
      ELSEIF(X.LT.70D0) THEN
        PPX=EP0+X*(EP1+X*(EP2+X*EP3))
        PQX=1D0+X*(EQ1+X*(EQ2+X*(EQ3+X*EQ4)))
```

```
      ELSEIF(X.LT.165D0) THEN
        PPX=FP0+X*(FP1+X*(FP2+X*FP3))
        PQX=1D0+X*(FQ1+X*(FQ2+X*(FQ3+X*FQ4)))
      ELSEIF(X.GE.165D0) THEN
        Y=1D0/X
        N=0
        IN=1
        K=1
        CE1=Y
 2      N=N+1
        IN=-IN
        Y=Y/X
        K=K*N
        TERM=Y*K
        IF(TERM/CE1.LT.EPS) GOTO 20
        CE1=CE1+IN*TERM
        GOTO 2
20      PPX=CE1
      ENDIF
      CE1=PPX/PQX

      RETURN
      END
```

```
      subroutine idaho
c
c*****************************************************************
c
c        VERSION OF JUNE 2001
c
c*****************************************************************
c
c        This code computes the
c
c        Idaho Chiral NN Potential
c        -------------------------
c        in momentum space.
c
c        this package is self-contained and includes
c        all subroutines needed.
c        only `idaho' needs to be called by the user.
c        all codes are consistently in double precision.
c        when working on an UNIX/LINUX system, it is recommended
c        to compile this code with the  -static  option.
c        more information on the code is given below.
c
c*****************************************************************
c
c        authors:     D. R. Entem and R. Machleidt
c                     department of physics
c                     university of idaho
c                     moscow, idaho 83844
c                     u. s. a.
c                     e-mails: dentem@uidaho.edu
c                              machleid@uidaho.edu
c
c*****************************************************************
c
c
c
c
      implicit real*8 (a-h,o-z)
c
c
      common /crdwrt/ kread,kwrite,kpunch,kda(9)
c
c        arguments and values of this subroutine
c
      common /cpot/   v(6),xmev,ymev
      common /cstate/ j,heform,sing,trip,coup,endep,label
      common /cnn/ inn
c
c
c        this has been the end of the common-blocks containing
c        the arguments and values of this subroutine.
c
c        specifications for these common blocks
c
      logical heform,sing,trip,coup,endep
      character*4 label
c
c
c*****************************************************************
c        THE ABOVE FOUR COMMON BLOCKS IS ALL THE USER NEEDS
```

```
c        TO BE FAMILIAR WITH.
c*****************************************************************
c
c        here are now some explanations of what those common blocks contain:
c        -------------------------------------------------------------------
c
c        xmev and ymev are the final and initial relative momenta,
c        respectively, in units of mev/c.
c        v is the potential in units of mev**(-2).
c        concerning units and factor of pi etc.,
c        cf. with the partial-wave Lippmann-Schwinger equation, Eq. (1.32),
c        and with the phase shift relation, Eq. (1.41) of
c        R. Machleidt, in: Computational Nuclear Physics 2
c        -- Nuclear Reactions, Langanke et al., eds.
c        (Springer, New York, 1993), Chapter 1, pp. 1-29.
c
c        the partial-wave Lippmann-Schwinger equation for the
c        K-matrix reads:
c
c        K(q',q) = V(q',q) + M P \int dk k^2 V(q',k) K(k,q)/(q^2-k^2)
c
c        with M the nucleon mass in MeV and P denoting the principal value;
c        V(q',q) as provided by this code in common block /cpot/;
c        all momenta in MeV.
c
c        the phase-shift relation is:
c
c        tan \delta_L = -(pi/2) M q K_L(q,q)
c
c        with M and q in units of MeV, K_L in MeV**(-2) like V.
c
c
c        if heform=.true., v contains the 6 matrix elements
c        associated with one j in the helicity formalism
c        in the following order:
c        0v, 1v, 12v, 34v, 55v, 66v (for notation see above article).
c
c        if heform=.false., v contains the 6 matrix elements
c        associated with one j in the lsj formalism
c        in the following order:
c        0v(singlet), 1v(uncoupled triplet), v++, v--, v+-, v-+ (coupled)
c        (see above article for notation).
c
c        j is the total angular momentum. there is essentially no upper
c        limit for j.
c        sing, trip, and coup should in general be .true..
c        endep and label can be ignored.
c        it is customary, to set kread=5 and kwrite=6;
c        ignore kpunch and kda(9).
c
c        the meaning of the parameter inn in the common block
c
c                  common /cnn/ inn
c        is
c                  inn=1  means you want to use potential A (`Idaho-A')
c                  inn=2  means you want to use potential B (`Idaho-B')
c
c        the user needs to include this common block in his/her code,
c        and specify which potential he/she wants.
c        Idaho-A has a D-state probability of the deuteron P_D = 4.17%,
c        Idaho-B has P_D = 4.94%. The quality of the fit of phase shifts
c        is the same for both potentials. use Idaho-B as default;
c        if you want more binding energy, use Idaho-A.
c
c
c        THIS IS ESSENTIALLY ALL THE USER NEEDS TO KNOW.
```

```
c
c        if you have further questions, do not hesitate to contact one
c        of the authors (see e-mail addresses above).
c
c**********************************************************************
c
c
c        common block for all chi-subroutines
c
      common /cchi/ vj(32,270),c(20,270),fff,ff,f(52),aa(96),ai(19,30),
     1               wnn(3),wdd(3),x,xx,y,yy,xy2,xxpyy,ex,ey,eem12,
     2               gaa(3),fpia(3),ezz1(3),ezz2(3),ct(96),wt(96),
     3               ic(20,270),ift(3),mint(3),maxt(3),nt,
     4               mge,mgg(40,3),mggo(40,3),ima(30,40,3),
     5               imaa(3),imea(3),ime,im,mc,m,mg,inter,ide,idde,
     6               indc(2,270),indpar(3),indxy
c
c         specifications for this common block
c
      logical indc,indxy,indpar
c
      common /comlsj/ clsj(15,50),cutlsj(15,50),indlsj
      logical indlsj
c
      common /crrr/ rrr
c
c
c        further specifications
c
      dimension vl(4),adminv(4,4),ldminv(4),mdminv(4)
      dimension vv0(6),vv2(6),vv4(6)
      character*4 mesong(40)
      logical index
      logical indmg(40)
      data mesong/'0-  ','0-t ','0-st','0+  ','0+st',
     1            '1-  ','1-t ','1-tt','1-st','1-ss',
     2            'c   ','ss  ','ls  ','sq  ','sk  ',
     3            'sl  ',
     4         24*'    '/
      data index/.false./
      data indmg/40*.false./
      data jj/-1/
      data pi/3.141592653589793d0/
      data innn/-1/
      save
c
c
c
c
      inter=1
c
c
      if (inn.lt.1.or.inn.gt.2) then
c        choose Idaho-B as default:
      inn=2
      endif
      if (j.lt.0) then
      write (kwrite,19002)
19002 format (////' error in idaho: total angular momentum j',
     1' is negative.'/' execution terminated.'////)
      stop
      endif
c
c
c
c
```

```
c        call subroutine chipar
c
c
      if (inn.eq.innn) go to 30
      innn=inn
c
c
      call chipar
c
c
      if (index) go to 30
      index=.true.
c
c
      dwn=1.d0/wnn(inter)
c
c        prepare constant over-all factor
c
      fac=pi/(2.d0*pi)**3*dwn*dwn
c     ---------------------------
c
c
      ez1=ezz1(inter)
      ez2=ezz2(inter)
c
c
c        if you want the potential to be zero for very large momenta,
c        choose rrr=1000.
c        if you want no technical problems in the calculation of the deuteron
c        wave functions, choose rrr=80.
c
      rrr=80.
c
c
c
      iftgo=ift(inter)+1
c
      iman=imaa(inter)
      imen=imea(inter)
c
      imanm1=iman-1
c
      iman1=imanm1+1
      iman2=imanm1+2
      iman3=imanm1+3
      iman4=imanm1+4
      iman5=imanm1+5
      iman6=imanm1+6
      iman7=imanm1+7
      iman8=imanm1+8
      iman9=imanm1+9
      imen24=imen-24
      imen23=imen-23
      imen22=imen-22
      imen21=imen-21
      imen15=imen-15
      imen14=imen-14
c
c
   30 if (j.eq.jj) go to 50
      jj=j
      if (j.eq.0) go to 50
      aj=dble(j)
      aj1=dble(j+1)
      a2j1=dble(2*j+1)
      aaj6=dsqrt(aj*aj1)
```

```
c
c        coefficient matrix for the translations into lsj formalism
c
      adminv(1,1)=aj1
      adminv(1,2)=aj
      adminv(1,3)=-aaj6
      adminv(1,4)=-aaj6
      adminv(2,1)=aj
      adminv(2,2)=aj1
      adminv(2,3)=aaj6
      adminv(2,4)=aaj6
      adminv(3,1)=aaj6
      adminv(3,2)=-aaj6
      adminv(3,3)=aj1
      adminv(3,4)=-aj
      adminv(4,1)=aaj6
      adminv(4,2)=-aaj6
      adminv(4,3)=-aj
      adminv(4,4)=aj1
c
c       inversion
c
      call dminv (adminv,4,deter,ldminv,mdminv)
c
c
c
c
c        prepare expressions depending on x and y
c        ----------------------------------------
c        ----------------------------------------
c
c
c
c
   50 xa=xmev*dwn
      ya=ymev*dwn
      indxy=.false.
      x=xa
      xx=x*x
      y=ya
      yy=y*y
      xy2=x*y*2.d0
      xxpyy=xx+yy
      ex=dsqrt(1.d0+xx)
      ey=dsqrt(1.d0+yy)
      eem12=(ex*ey-1.d0)*2.d0
c
c
      xy=xy2*0.5d0
      ee=ex*ey
      ree=dsqrt(ee)
      eem1=ee-1.d0
      eme=ex-ey
      emeh=eme*0.5d0
      emehq=emeh*emeh
      eep1=ee+1.d0
       epe=ex+ey
      xxyy=xx*yy
c
c
      xxpyyh=xxpyy*0.5d0
      xy3=xy*3.d0
      xy4=xy*4.d0
c
c
c
```

```
c
      do 63 iv=1,6
      vv0(iv)=0.d0
      vv2(iv)=0.d0
      vv4(iv)=0.d0
   63 v(iv)=0.d0
      do 65 il=iman,imen
      do 65 iv=1,32
   65 vj(iv,il)=0.d0
c
c
c
c
c        prepare over-all factor
c
c
      go to (70,71,72,71,72,75,76),iftgo
c
c        no additional factor
c
   70 fff=fac
      go to 80
c
c        minimal relativity
c
   71 fff=fac/ree
      go to 80
c
c        factor m/e*m/e
c
   72 fff=fac/ee
      go to 80
c
c        sharp cutoff
c
   75 if (xmev.gt.ez1.or.ymev.gt.ez1) then
      return
      else
      fff=fac
      end if
      go to 80
c
c        exponential cutoff
c
   76 expo=(xmev/ez1)**(2.d0*ez2)+(ymev/ez1)**(2.d0*ez2)
      if (expo.gt.rrr) then
      expo=rrr
      end if
      fff=fac*dexp(-expo)
c
c
   80 continue
c
c
c
c
c        contributions
c        -------------
c        -------------
c
c
c
c
      do 5995 img=1,mge
      mg=mggo(img,inter)
      if (mg.gt.16) go to 9000
```

```
      if (mg.eq.0) go to 8000
      me=mgg(mg,inter)
      go to (9000,9000,9000,9000,9000,9000,9000,9000,9000,9000,
     1       1100,1200,1300,1400,1500,1600),mg
c
c
c
c
c
c        c   , central force
c        -------------------
c
c
c
c
 1100 mc=1
c
      ff=1.d0
      f(1)=2.d0
      f(2)=0.d0
      f(3)=f(1)
      f(4)=f(2)
      f(5)=f(2)
      f(6)=f(1)
      f(7)=-f(1)
      f(8)=f(7)
c
      call chistr(1,1,me)
      go to 5995
c
c
c
c
c        ss  , spin-spin force
c        ---------------------
c
c
c
c
 1200 mc=1
c
      ff=1.d0
      f(1)=-6.d0
      f(2)=0.d0
      f(3)=2.d0
      f(4)=0.d0
      f(5)=0.d0
      f(6)=f(3)
      f(7)=-f(3)
      f(8)=f(7)
c
      call chistr(1,1,me)
      go to 5995
c
c
c
c
c        ls  , spin-orbit force
c        ----------------------
c
c
c
c
 1300 mc=1
c
      ff=1.d0
```

```
      f(1)=0.d0
      f(2)=0.d0
      f(3)=0.d0
      f(4)=-xy2
      f(5)=-xy2
      f(6)=0.d0
      f(7)=0.d0
      f(8)=0.d0
      f(9)=0.d0
      f(10)=+xy2
      f(11)=-xy2
c
      call chistr(2,1,me)
      go to 5995
c
c
c
c
c        sq  , sq tensor force (where q denotes the momentum transfer)
c        ---------------------
c
c
c
c
 1400 mc=1
c
      ff=1.d0
      f(1)=-xxpyy*2.0d0
      f(2)=xy*4.d0
      f(3)=-f(1)
      f(4)=-f(2)
      f(5)=f(2)
      f(6)=f(1)
      f(7)=(xx-yy)*2.0d0
      f(8)=-f(7)
c
      call chistr(1,1,me)
      go to 5995
c
c
c
c
c        sk  , sk tensor force (where k denotes the average momentum)
c        ---------------------
c
c
c
c
 1500 mc=1
c
      ff=0.25d0
      f(1)=-xxpyy*2.0d0
      f(2)=-xy*4.d0
      f(3)=-f(1)
      f(4)=-f(2)
      f(5)=f(2)
      f(6)=f(1)
      f(7)=(xx-yy)*2.0d0
      f(8)=-f(7)
c
      call chistr(1,1,me)
      go to 5995
c
c
c
c
```

```
c        sl  , "quadratic spin-orbit force"
c               or sigma-l operator
c        ---------------------------------
c
c
c
c
 1600 mc=1
c
      ff=1.d0
      f(1)=-xxyy*2.d0
      f(2)=0.d0
      f(3)=f(1)
      f(4)=f(2)
      f(5)=f(2)
      f(6)=-f(1)
      f(7)=f(1)
      f(8)=f(7)
      f(9)=f(6)*2.d0
c
      call chistr(4,1,me)
      go to 5995
c
c
c
c
c
c        this has been the end of the contributions
c        ------------------------------------------
c
c
c
c
c        errors and warnings
c        -------------------
c
c
c
c
 9000 if (indmg(mg)) go to 5995
      write (kwrite,19000) mesong(mg)
19000 format(1h ////' warning in idaho: contribution ',a4,' does not exi
     1st in this program.'/' contribution ignored. execution continued.'
     2////)
      indmg(mg)=.true.
c
c
c
c
 5995 continue
c
c
c
c
c        add up contributions
c        --------------------
c
c
c
c
 8000 continue
c
c
c        charge-dependent OPE contribution
c        ---------------------------------
c
```

```
      if (mod(j,2).eq.1) go to 8020
c
c        j even
c
      v(1)=-vj(1,iman1)+2.d0*vj(1,iman5)
      v(1)=v(1)-vj(1,iman2)+2.d0*vj(1,iman6)
      v(1)=v(1)-vj(1,iman3)+2.d0*vj(1,iman7)
      v(1)=v(1)-vj(1,iman4)+2.d0*vj(1,iman8)
c
      v(2)=-vj(2,iman1)-2.d0*vj(2,iman5)
      v(2)=v(2)-vj(2,iman2)-2.d0*vj(2,iman6)
      v(2)=v(2)-vj(2,iman3)-2.d0*vj(2,iman7)
      v(2)=v(2)-vj(2,iman4)-2.d0*vj(2,iman8)
c
      do 8015 iv=3,6
      v(iv)=-vj(iv,iman1)+2.d0*vj(iv,iman5)
      v(iv)=v(iv)-vj(iv,iman2)+2.d0*vj(iv,iman6)
      v(iv)=v(iv)-vj(iv,iman3)+2.d0*vj(iv,iman7)
      v(iv)=v(iv)-vj(iv,iman4)+2.d0*vj(iv,iman8)
 8015 continue
      go to 8030
c
c        j odd
c
 8020 continue
      v(1)=-vj(1,iman1)-2.d0*vj(1,iman5)
      v(1)=v(1)-vj(1,iman2)-2.d0*vj(1,iman6)
      v(1)=v(1)-vj(1,iman3)-2.d0*vj(1,iman7)
      v(1)=v(1)-vj(1,iman4)-2.d0*vj(1,iman8)
c
      v(2)=-vj(2,iman1)+2.d0*vj(2,iman5)
      v(2)=v(2)-vj(2,iman2)+2.d0*vj(2,iman6)
      v(2)=v(2)-vj(2,iman3)+2.d0*vj(2,iman7)
      v(2)=v(2)-vj(2,iman4)+2.d0*vj(2,iman8)
c
      do 8025 iv=3,6
      v(iv)=-vj(iv,iman1)-2.d0*vj(iv,iman5)
      v(iv)=v(iv)-vj(iv,iman2)-2.d0*vj(iv,iman6)
      v(iv)=v(iv)-vj(iv,iman3)-2.d0*vj(iv,iman7)
      v(iv)=v(iv)-vj(iv,iman4)-2.d0*vj(iv,iman8)
 8025 continue
c
c
 8030 continue
c
c
      if (iman9.gt.imen) go to 8500
c
c
      if (.not.indlsj) then
      do 8105 il=iman9,imen
      do 8105 iv=1,6
 8105 v(iv)=v(iv)+vj(iv,il)
      else
c
c
c        there are contact terms
c        -----------------------
c
      if (iman9.gt.imen24) go to 8200
c
c        the non-contact terms
c
      do 8155 il=iman9,imen24
      do 8155 iv=1,6
 8155 v(iv)=v(iv)+vj(iv,il)
```

```
c
c        contact contributions
c        ---------------------
c
 8200 continue
c
c        Q^0 contacts
      do 8205 il=imen23,imen22
      do 8205 iv=1,6
 8205 vv0(iv)=vv0(iv)+vj(iv,il)
c
c        Q^2 contacts
      do 8215 il=imen21,imen15
      do 8215 iv=1,6
 8215 vv2(iv)=vv2(iv)+vj(iv,il)
c
c        Q^4 contacts
      do 8225 il=imen14,imen
      do 8225 iv=1,6
 8225 vv4(iv)=vv4(iv)+vj(iv,il)
c
c
c        ------------------------------------------------------
c        NOTE: partial-wave potentials that add-up to zero need
c        to be cutoff, because they diverge for large momenta.
c        ------------------------------------------------------
c
c        use 3d3 cutoff as default for all j.gt.3 partial waves
c
      if (j.gt.3) then
      if (cutlsj(1,15).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,15))**(2.d0*cutlsj(1,15))
     1    +(ymev/cutlsj(2,15))**(2.d0*cutlsj(1,15))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
c
      do 8275 iv=1,6
      vv0(iv)=vv0(iv)*expexp
      vv2(iv)=vv2(iv)*expexp
 8275 vv4(iv)=vv4(iv)*expexp
      go to 8400
      end if
c
c
c        look into individual partial waves and
c        multiply with partial-wave dependent cutoffs
c        --------------------------------------------
c
      j1=j+1
      go to (8310,8320,8330,8340),j1
c
c
c        j=0
c        ---
c        ---
c
 8310 continue
c
c        1s0
c        ---
c        Q^0 term
c
      if (cutlsj(1,1).eq.0.d0) then
```

```
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,1))**(2.d0*cutlsj(1,1))
     1    +(ymev/cutlsj(2,1))**(2.d0*cutlsj(1,1))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv0(1)=vv0(1)*expexp
c
c        Q^2 terms
c
      if (cutlsj(3,1).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(4,1))**(2.d0*cutlsj(3,1))
     1    +(ymev/cutlsj(4,1))**(2.d0*cutlsj(3,1))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv2(1)=vv2(1)*expexp
c
c        Q^4 terms
c
      if (cutlsj(5,1).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(6,1))**(2.d0*cutlsj(5,1))
     1    +(ymev/cutlsj(6,1))**(2.d0*cutlsj(5,1))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(1)=vv4(1)*expexp
c
c        3p0
c        ---
c        Q^2 term
c
      if (cutlsj(1,2).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,2))**(2.d0*cutlsj(1,2))
     1    +(ymev/cutlsj(2,2))**(2.d0*cutlsj(1,2))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv2(3)=vv2(3)*expexp
      vv0(3)=vv0(3)*expexp
c
c        Q^4 term
c
      if (cutlsj(3,2).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(4,2))**(2.d0*cutlsj(3,2))
     1    +(ymev/cutlsj(4,2))**(2.d0*cutlsj(3,2))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(3)=vv4(3)*expexp
c
      go to 8400
c
c
c        j=1
c        ---
c        ---
```

```
c
 8320 continue
c
c        1p1
c        ---
c        Q^2 term
c
      if (cutlsj(1,3).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,3))**(2.d0*cutlsj(1,3))
     1    +(ymev/cutlsj(2,3))**(2.d0*cutlsj(1,3))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv2(1)=vv2(1)*expexp
      vv0(1)=vv0(1)*expexp
c
c        Q^4 term
c
      if (cutlsj(3,3).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(4,3))**(2.d0*cutlsj(3,3))
     1    +(ymev/cutlsj(4,3))**(2.d0*cutlsj(3,3))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(1)=vv4(1)*expexp
c
c        3p1
c        ---
c        Q^2 term
c
      if (cutlsj(1,4).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,4))**(2.d0*cutlsj(1,4))
     1    +(ymev/cutlsj(2,4))**(2.d0*cutlsj(1,4))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv2(2)=vv2(2)*expexp
      vv0(2)=vv0(2)*expexp
c
c        Q^4 term
c
      if (cutlsj(3,4).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(4,4))**(2.d0*cutlsj(3,4))
     1    +(ymev/cutlsj(4,4))**(2.d0*cutlsj(3,4))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(2)=vv4(2)*expexp
c
c        3s1
c        ---
c        Q^0 term
c
      if (cutlsj(1,5).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,5))**(2.d0*cutlsj(1,5))
     1    +(ymev/cutlsj(2,5))**(2.d0*cutlsj(1,5))
```

```
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv0(4)=vv0(4)*expexp
c
c        Q^2 terms
c
      if (cutlsj(3,5).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(4,5))**(2.d0*cutlsj(3,5))
     1    +(ymev/cutlsj(4,5))**(2.d0*cutlsj(3,5))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv2(4)=vv2(4)*expexp
c
c        Q^4 terms
c
      if (cutlsj(5,5).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(6,5))**(2.d0*cutlsj(5,5))
     1    +(ymev/cutlsj(6,5))**(2.d0*cutlsj(5,5))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(4)=vv4(4)*expexp
c
c        3d1
c        ---
c        Q^4 term
c
      if (cutlsj(1,6).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,6))**(2.d0*cutlsj(1,6))
     1    +(ymev/cutlsj(2,6))**(2.d0*cutlsj(1,6))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(3)=vv4(3)*expexp
      vv2(3)=vv2(3)*expexp
      vv0(3)=vv0(3)*expexp
c
c        3s/d1
c        -----
c        Q^2 term
c
      if (cutlsj(1,7).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,7))**(2.d0*cutlsj(1,7))
     1    +(ymev/cutlsj(2,7))**(2.d0*cutlsj(1,7))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv2(5)=vv2(5)*expexp
      vv2(6)=vv2(6)*expexp
      vv0(5)=vv0(5)*expexp
      vv0(6)=vv0(6)*expexp
c
c        Q^4 term
c
      if (cutlsj(3,7).eq.0.d0) then
      expexp=1.d0
```

```
      else
      expo=(xmev/cutlsj(4,7))**(2.d0*cutlsj(3,7))
     1    +(ymev/cutlsj(4,7))**(2.d0*cutlsj(3,7))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(5)=vv4(5)*expexp
      vv4(6)=vv4(6)*expexp
c
      go to 8400
c
c
c        j=2
c        ---
c        ---
c
 8330 continue
c
c        1d2
c        ---
c        Q^4 term
c
      if (cutlsj(1,8).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,8))**(2.d0*cutlsj(1,8))
     1    +(ymev/cutlsj(2,8))**(2.d0*cutlsj(1,8))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(1)=vv4(1)*expexp
      vv2(1)=vv2(1)*expexp
      vv0(1)=vv0(1)*expexp
c
c        3d2
c        ---
c        Q^4 term
c
      if (cutlsj(1,9).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,9))**(2.d0*cutlsj(1,9))
     1    +(ymev/cutlsj(2,9))**(2.d0*cutlsj(1,9))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(2)=vv4(2)*expexp
      vv2(2)=vv2(2)*expexp
      vv0(2)=vv0(2)*expexp
c
c        3p2
c        ---
c
c        Q^2 term
c
      if (cutlsj(1,10).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,10))**(2.d0*cutlsj(1,10))
     1    +(ymev/cutlsj(2,10))**(2.d0*cutlsj(1,10))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv2(4)=vv2(4)*expexp
      vv0(4)=vv0(4)*expexp
      vv2(3)=vv2(3)*expexp
```

```
      vv0(3)=vv0(3)*expexp
c
c        Q^4 terms
c
      if (cutlsj(3,10).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(4,10))**(2.d0*cutlsj(3,10))
     1    +(ymev/cutlsj(4,10))**(2.d0*cutlsj(3,10))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(4)=vv4(4)*expexp
      vv4(3)=vv4(3)*expexp
c
c        3p/f2
c        -----
c        Q^4 term
c
      if (cutlsj(1,12).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,12))**(2.d0*cutlsj(1,12))
     1    +(ymev/cutlsj(2,12))**(2.d0*cutlsj(1,12))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(5)=vv4(5)*expexp
      vv4(6)=vv4(6)*expexp
      vv2(5)=vv2(5)*expexp
      vv2(6)=vv2(6)*expexp
      vv0(5)=vv0(5)*expexp
      vv0(6)=vv0(6)*expexp
c
      go to 8400
c
c
c        j=3
c        ---
c        ---
c
 8340 continue
c
c        3d3
c        ---
c        Q^4 term
c
      if (cutlsj(1,15).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,15))**(2.d0*cutlsj(1,15))
     1    +(ymev/cutlsj(2,15))**(2.d0*cutlsj(1,15))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
c
c        use 3d3 cutoff for all j.eq.3 partial waves
c
      do 8345 iv=1,6
      vv0(iv)=vv0(iv)*expexp
      vv2(iv)=vv2(iv)*expexp
 8345 vv4(iv)=vv4(iv)*expexp
c
c
c
c
```

```
c
c
c        final add up
c        ------------
c
 8400 do 8405 iv=1,6
 8405 v(iv)=v(iv)+vv0(iv)+vv2(iv)+vv4(iv)
c
      end if
c
c
c
c
 8500 if (j.eq.0.or..not.heform) go to 8900
c
c
c         translation into (combinations of) helicity states
c
c
      do 8505 i=1,4
 8505 vl(i)=v(i+2)
c
      do 8520 ii=1,4
      iii=ii+2
      v(iii)=0.d0
c
      do 8515 i=1,4
 8515 v(iii)=v(iii)+adminv(ii,i)*vl(i)
 8520 v(iii)=v(iii)*a2j1
c
c
c
c
 8900 return
      end
      subroutine chipar
c
c        chipar provides the parameters for all chi-subroutines.
c
c
      implicit real*8 (a-h,o-z)
c
c
      common /crdwrt/ kread,kwrite,kpunch,kda(9)
c
      common /cstate/ j,heform,sing,trip,coup,endep,label
      common /cnn/ inn
      logical heform,sing,trip,coup,endep
      character*4 label
c
c
c        common block for all chi-subroutines
c
      common /cchi/ vj(32,270),c(20,270),fff,ff,f(52),aa(96),ai(19,30),
     1               wnn(3),wdd(3),x,xx,y,yy,xy2,xxpyy,ex,ey,eem12,
     2               gaa(3),fpia(3),ezz1(3),ezz2(3),ct(96),wt(96),
     3               ic(20,270),ift(3),mint(3),maxt(3),nt,
     4               mge,mgg(40,3),mggo(40,3),ima(30,40,3),
     5               imaa(3),imea(3),ime,im,mc,m,mg,inter,ide,idde,
     6               indc(2,270),indpar(3),indxy
c
c         specifications for this common block
c
      logical indc,indxy,indpar
c
      common /comlsj/ clsj(15,50),cutlsj(15,50),indlsj
```

```
      logical indlsj
c
c
c        further specifications
c
      dimension cc(5),cca(5)
      dimension clec(15,50)
      dimension a(1024),b(32)
      dimension ttab(5,69),tab(5,69)
      dimension taba(5,8)
      real*4 eps
      character*4 name(3)
      character*4 ntab(3,69)
      integer imga(3)
      character*4 nucnuc(2)
      character*4 cut,cuta,fun,lsj,lec,end
      character*4 mesong(40)
      logical index
      logical zerocp
      logical indlec
      logical indca,indlca
      data mesong/'0-  ','0-t ','0-st','0+  ','0+st',
     1            '1-  ','1-t ','1-tt','1-st','1-ss',
     2            'c   ','ss  ','ls  ','sq  ','sk  ',
     3            'sl  ',
     4         24*'    '/
      data index/.false./
      data zerocp/.true./
      data pi/3.141592653589793d0/
      data eps/1.e-15/
      data cut/'cut '/,cuta/'cuta'/
      data fun/'fun '/,lsj/'lsj '/,lec/'lec '/,end/'end '/
      data nucnuc/'ID-A','ID-B'/
c
c
c
c
c        parameter tables
c        ----------------
c        ----------------
c
c
c        identification table
c        --------------------
c
      data ntab/
     1 'cuta','ll  ','  ',
     2 'sq  ','opep','  ',
     3 'fun ','    ','  ',
     4 'sk  ','opep','  ',
     5 'fun ','    ','  ',
     6 'sl  ','opep','  ',
     7 'fun ','    ','  ',
     8 'ss  ','opep','  ',
     9 'fun ','    ','  ',
     * 'sq  ','opep','  ',
     1 'fun ','    ','  ',
     2 'sk  ','opep','  ',
     3 'fun ','    ','  ',
     4 'sl  ','opep','  ',
     5 'fun ','    ','  ',
     6 'ss  ','opep','  ',
     7 'fun ','    ','  ',
     8 'cuta','ll  ','  ',
     9 'c   ','tpn1','  ',
     * 'fun ','    ','  ',
```

```
     1 'sq  ','tpn1','   ',
     2 'fun ','    ','   ',
     3 'ss  ','tpn1','   ',
     4 'fun ','    ','   ',
     5 'fun ','    ','   ',
     6 'c   ','tpn2','   ',
     7 'fun ','    ','   ',
     8 'c   ','tpn2','   ',
     9 'fun ','    ','   ',
     * 'sq  ','tpn2','   ',
     1 'fun ','    ','   ',
     2 'ss  ','    ','   ',
     3 'fun ','    ','   ',
     4 'fun ','    ','   ',
     5 'sq  ','tpn2','   ',
     6 'fun ','    ','   ',
     7 'ss  ','tpn2','   ',
     8 'fun ','    ','   ',
     9 'fun ','    ','   ',
     * 'ls  ','tpn2','   ',
     1 'fun ','    ','   ',
     2 'ls  ','tpn2','   ',
     3 'fun ','    ','   ',
     4 'cuta','ll  ','   ',
     5 'lsj ',' 1S0','   ',
     6 'lsj ',' 1S0','   ',
     7 'lsj ',' 1S0','   ',
     8 'lsj ',' 1S0','   ',
     9 'lsj ',' 3P0','   ',
     * 'lsj ',' 3P0','   ',
     1 'lsj ',' 1P1','   ',
     2 'lsj ',' 1P1','   ',
     3 'lsj ',' 3P1','   ',
     4 'lsj ',' 3P1','   ',
     5 'lsj ',' 3S1','   ',
     6 'lsj ',' 3S1','   ',
     7 'lsj ',' 3S1','   ',
     8 'lsj ',' 3S1','   ',
     9 'lsj ',' 3D1','   ',
     * 'lsj ','3S-D','   ',
     1 'lsj ','3S-D','   ',
     2 'lsj ','3S-D','   ',
     3 'lsj ',' 1D2','   ',
     4 'lsj ',' 3D2','   ',
     5 'lsj ',' 3P2','   ',
     6 'lsj ',' 3P2','   ',
     7 'lsj ','3P-F','   ',
     8 'lsj ',' 3D3','   ',
     9 'end ','para','   '/
c
c
c        parameters
c        ----------
c
c        parameters for the Idaho-B potential
c        (these parameters are identical to the ones for the
c        Idaho-A potential except for the 3S1/3D1 parameters,
c        s. below)
      data tab/
     1      6.    ,  0.0  ,   4.       , 600.   ,   0.,
     2     -1.29  , 92.4  ,   134.9766, 0.     ,   0.,
     3      34.   ,  0.0  ,     0.     , 0.     ,   0.,
     4     -1.29  , 92.4  ,   134.9766, 0.     ,   0.,
     5      11.   , 0.5   ,     0.     , 0.     ,   0.,
     6     -1.29  , 92.4  ,   134.9766, 0.     ,   0.,
     7      32.   , 0.5   ,     0.     , 0.     ,   0.,
```

```
     8     -1.29  , 92.4  ,   134.9766, 0.    ,   0.,
     9      24.   , -0.5  ,     0.    , 0.    ,   0.,
     *     -1.29  , 92.4  ,   139.5702, 0.    ,   0.,
     1      34.   ,  0.0  ,     0.    , 0.    ,   0.,
     2     -1.29  , 92.4  ,   139.5702, 0.    ,   0.,
     3      11.   , 0.5   ,     0.    , 0.    ,   0.,
     4     -1.29  , 92.4  ,   139.5702, 0.    ,   0.,
     5      32.   , 0.5   ,     0.    , 0.    ,   0.,
     6     -1.29  , 92.4  ,   139.5702, 0.    ,   0.,
     7      24.   , -0.5  ,     0.    , 0.    ,   0.,
     8      6.    , 0.    ,   2.      , 460.  ,   0.,
     9      1.    , 0.    ,   138.039 , 1.    ,   -1.,
     *      13.   , 0.    ,     0.    , 0.    ,   0.,
     1      1.    , 0.    ,   138.039 , 0.    ,   -1.,
     2      14.   , 0.    ,     0.    , 0.    ,   0.,
     3     -1.    , 0.    ,   138.039 , 0.    ,   -1.,
     4      11.   , 0.    ,     0.    , 0.    ,   0.,
     5      14.   , 0.    ,     0.    , 0.    ,   0.,
     6      1.    , 0.    ,   138.039 , 0.    ,   -1.,
     7      15.   , -0.81 ,   -4.70   , 3.40  ,   0.,
     8      1.    , 0.    ,   138.039 , 1.    ,   -1.,
     9      16.   , 0.    ,     0.    , 0.    ,   0.,
     *      1.    , 0.    ,   138.039 , 0.    ,   -1.,
     1      17.   , 0.    ,     0.    , 0.    ,   0.,
     2     -1.    , 0.    ,   138.039 , 0.    ,   -1.,
     3      11.   , 0.    ,     0.    , 0.    ,   0.,
     4      17.   , 0.    ,     0.    , 0.    ,   0.,
     5      1.    , 0.    ,   138.039 , 1.    ,   -1.,
     6      18.   , -0.81 ,   -4.70   , 3.40  ,   0.,
     7     -1.    , 0.    ,   138.039 , 1.    ,   -1.,
     8      11.   , 0.    ,     0.    , 0.    ,   0.,
     9      18.   , -0.81 ,   -4.70   , 3.40  ,   0.,
     *      1.    , 0.    ,   138.039 , 0.    ,   -1.,
     1      19.   , 0.    ,     0.    , 0.    ,   0.,
     2      1.    , 0.    ,   138.039 , 1.    ,   -1.,
     3      20.   , 0.    ,     0.    , 0.    ,   0.,
     4      0.    , 0.    ,   2.0     , 460.  ,   0.,
     5     -0.118522d0,3.   ,   412.    , 0.    ,   0.,
     6      2.890   ,2.    ,   443.    , 0.    ,   0.,
     7      9.84    ,2.    ,   448.    , 0.    ,   0.,
     8     12.43    ,2.    ,   448.    , 0.    ,   0.,
     9      1.07    ,2.    ,   483.    , 0.    ,   0.,
     *     -5.28    ,2.    ,   416.    , 0.    ,   0.,
     1      0.140   ,2.    ,   430.    , 0.    ,   0.,
     2     -2.01    ,2.    ,   430.    , 0.    ,   0.,
     3     -1.03    ,2.    ,   407.    , 0.    ,   0.,
     4     -9.06    ,2.    ,   451.    , 0.    ,   0.,
     5     -0.11091316d0,3.   ,   463.    , 0.    ,   0.,
     6      1.098   ,2.    ,   495.    , 0.    ,   0.,
     7      8.00    ,2.    ,   428.    , 0.    ,   0.,
     8     14.20    ,2.    ,   428.    , 0.    ,   0.,
     9      2.58    ,2.    ,   425.    , 0.    ,   0.,
     *      0.615   ,2.    ,   472.    , 0.    ,   0.,
     1      2.08    ,2.    ,   480.    , 0.    ,   0.,
     2      2.66    ,2.    ,   480.    , 0.    ,   0.,
     3      2.983   ,2.    ,   422.    , 0.    ,   0.,
     4      2.07    ,2.    ,   452.    , 0.    ,   0.,
     5     -0.968   ,2.    ,   430.    , 0.    ,   0.,
     6     -6.679   ,2.    ,   458.    , 0.    ,   0.,
     7      0.258   ,2.    ,   366.    , 0.    ,   0.,
     8      3.26    ,2.    ,   430.    , 0.    ,   0.,
     9      0.      ,0.    ,     0.    , 0.    ,   0./
c
c
c        3S1/3D1 parameters for the Idaho-A potential:
      data taba/
```

```
     1     -0.129299457d0,3.    ,   463. , 0. , 0.,
     2      1.228    ,2.    ,   495. , 0. , 0.,
     3      8.00     ,2.    ,   428. , 0. , 0.,
     4     12.95     ,2.    ,   428. , 0. , 0.,
     5      2.88     ,2.    ,   430. , 0. , 0.,
     6      0.618    ,2.    ,   465. , 0. , 0.,
     7      3.00     ,2.    ,   480. , 0. , 0.,
     8      1.99     ,2.    ,   480. , 0. , 0./
c
c
c        this has been the end of all tables
c
c
      save
c
c
c
c
10004 format (1h ,2a4,a2,f12.6,f10.6,1x,f10.5,2(f7.1,3x))
10005 format (1h ,2a4,a2,f4.1,1x,2f10.5,f13.5,f10.5)
10007 format (1h ,2a4,a2,3i3)
10008 format (1h ,57(1h-))
10010 format (1h ,2a4,a2,i3,2f10.2)
10011 format
     1 (//' IDAHO: NN potential based upon chiral perturbation theory')
10020 format (1h ,2a4,a2,f12.6,4f9.2)
10030 format (//)
10031 format (' using Idaho-A')
10032 format (' using Idaho-B')
10033 format (' -------------')
c
c
c
c
      if (index) go to 50
c**** index=.true.
c
      x=-1.d0
      y=-1.d0
c
c
c
c
c        maxima of certain indices related to the dimension as follows:
c        dimension c(mme,imee),ic(mice,imee),indc(mindce,imee),
c                  mgg(mge,3),mggo(mge,3),mesong(mge),vj(32,imee),
c                  ima(mee,mge,3)
c
      mge=40
      mee=30
      mme=20
      mice=20
      mindce=2
      imb=1
      ime=0
      imee=270
c        mme always ge mice, mindce
c
c        set all parameters and indices to zero or .false.
c
      do 1 int=1,3
      imga(int)=0
      indpar(int)=.false.
      do 1 mgx=1,mge
      mgg(mgx,int)=0
    1 mggo(mgx,int)=0
```

```
c
c
      do 2 il=1,imee
      do 2 mm=1,mme
      if (mm.le.mindce) indc(mm,il)=.false.
      if (mm.le.mice) ic(mm,il)=0
    2 c(mm,il)=0.d0
      endep=.false.
c
c
      pi2=pi*pi
c
c
c        prepare tables
c
c        the default: Idaho-B
      do 3 line=1,69
      do 3 i=1,5
    3 ttab(i,line)=tab(i,line)
c
      if (inn.eq.1) then
c        insert the 3S1/3D1 parameters for Idaho-A
      do 4 line=55,62
      do 4 i=1,5
    4 ttab(i,line)=taba(i,line-54)
      end if
c
c
c
c
c        start
c        -----
c        -----
c
c
c
c        write title
c
   50 continue
      write (kwrite,10011)
      write (kwrite,10008)
      write (kwrite,10008)
      go to (51,52),inn
   51 write (kwrite,10031)
      go to 53
   52 write (kwrite,10032)
   53 write (kwrite,10033)
      write (kwrite,10030)
c
      label=nucnuc(inn)
      indpar(inter)=.true.
c
      indca=.false.
      indlca=.false.
      indlsj=.false.
      indlec=.false.
      ilsj=0
      ilec=0
      line=0
      do 55 ii=1,50
      do 55 i=1,15
      clsj(i,ii)=0.d0
      cutlsj(i,ii)=0.d0
   55 clec(i,ii)=0.d0
c
c        fix index-parameter concerning the factor for the
```

```
c        whole potential and cutoff mass
c
      ift(inter)=0
      ezz1(inter)=0.d0
      ezz2(inter)=0.d0
c
c**** write (kwrite,10010) name,ift(inter),ezz1(inter),ezz2(inter)
      iftyp=ift(inter)
      if (iftyp.lt.0.or.iftyp.gt.6) go to 9003
c
c        fix parameters for numerical integration
c
      mint(inter)=4
      maxt(inter)=48
c
c**** write (kwrite,10007) name,mint(inter),maxt(inter)
c
c        use average nucleon mass
c
      wn=938.9187d0
c
c**** write (kwrite,10004) name,wn
      wnq=wn*wn
      dwn=1.d0/wn
      dwnq=dwn*dwn
      wnn(inter)=wn
c
c        ga and fpi
c
      ga=1.29d0
      fpi=92.4d0
c
c**** write (kwrite,10004) name,ga,fpi
      ga2=ga*ga
      ga4=ga2*ga2
      fpi2=fpi*fpi*dwnq
      fpi4=fpi2*fpi2
      gaa(inter)=ga2
      fpia(inter)=fpi2
c
c
c
c
c        get parameters from tables, line by line
c        ----------------------------------------
c        ----------------------------------------
c
c
c
   61 line=line+1
      do i=1,5
      if (i.le.3) then
      name(i)=ntab(i,line)
      end if
      cc(i)=ttab(i,line)
      end do
c
c        check if end of input
c
      if (name(1).eq.end) go to 7000
c
c        check if lsj or lec
c
      if (name(1).eq.lsj) go to 6000
      if (name(1).eq.lec) go to 6500
c
```

```
c        check if data-card just read contains cut-off or
c        function parameters
c
      if (name(1).eq.cut.or.name(1).eq.fun) go to 70
c
      if (name(1).eq.cuta) then
c**** write (kwrite,10005) name,cc
      indca=.true.
      do i=1,5
      cca(i)=cc(i)
      end do
      go to 61
      end if

c
c
c
c
c        write parameters which are no cut-off or function parameters
c        -----------------------------------------------------------
c
c
c
c
c**** write (kwrite,10004) name,cc
c
c        check if coupling constant is zero
c
c****    do not use zerocp anymore
c****    because the first two input lines are always pions.
c****    these lines must never be skipped even when g_pi zero.
c**** if (cc(1).ne.0.d0) go to 62
c**** zerocp=.true.
c**** go to 61
c
   62 zerocp=.false.
c
c        find out number of contribution mg
c
      do 63 mg=1,mge
      if (name(1).eq.mesong(mg)) go to 64
   63 continue
      go to 9000
c
c
c
c
c        store parameters which are no cut-off or function parameters
c        -----------------------------------------------------------
c
c
c
c
   64 ime=ime+1
      if (ime.gt.imee) go to 9011
      mgg(mg,inter)=mgg(mg,inter)+1
      m=mgg(mg,inter)
      if (m.gt.mee) go to 9001
      ima(m,mg,inter)=ime
      if (m.ne.1) go to 65
      imga(inter)=imga(inter)+1
      mggo(imga(inter),inter)=mg
   65 continue
c
c
      c(1,ime)=cc(1)
```

```
c
c
      if (mg.le.10) then
      c(1,ime)=c(1,ime)*4.d0*pi
      end if

      if (mg.le.3.and.cc(2).ne.0.d0) then
      c(1,ime)=(cc(1)/cc(2)*wn)**2
      if (cc(1).lt.0.d0) c(1,ime)=-c(1,ime)
      end if
c
c
      if (mg.ge.6.and.mg.le.10) then
c        store coupling constant f*g
      c(3,ime)=cc(2)*c(1,ime)
c        store coupling constant f**2
      c(2,ime)=cc(2)*c(3,ime)
      if (mg.eq.10)
     1  c(1,ime)=c(1,ime)+c(3,ime)*2.d0+c(2,ime)
      end if
c
c
      if (mg.ge.11.and.cc(2).ne.0.d0) then
      c(1,ime)=(cc(1)/(2.d0*cc(2))*wn)**2
      if (cc(1).lt.0.d0) c(1,ime)=-c(1,ime)
      end if
c
c
c        store meson mass square in units of nucleon mass square
      c(4,ime)=cc(3)*cc(3)*dwnq
c
c        test iso-spin
      icc=cc(4)
      if (icc.ne.0.and.icc.ne.1) go to 9004
c         store isospin as logical constant
      if (icc.eq.1) indc(1,ime)=.true.
c        store and test iprsp
      icc=cc(5)
      ic(1,ime)=icc
      if (iabs(ic(1,ime)).gt.1) go to 9005
c
c        index values for further storing
      mi=4
      mm=5
c
c
c        check if there is a `cutall' cutoff
c
      if (indca) then
      name(1)=cut
      do i=1,5
      cc(i)=cca(i)
      end do
      go to 72
      else
      go to 61
      end if
c
c
c
c
c        write cut-off or function parameters
c        -----------------------------------
c
c
c
```

```
c
   70 continue
c**** write (kwrite,10005) name,cc
c
      if (zerocp) go to 61
c
   72 continue
c
c
c
c
c        store parameters
c        ----------------
c
c
c
      ityp=cc(1)
c
      if (ityp.eq.0) go to 5995
      if (ityp.lt.1.or.ityp.gt.35) go to 9002
c
      im=ime
c
c        store typ of cut-off or function
      ic(mi,im)=ityp
c
      if (ityp.le.10) then
c        store and test typ of propagator of cut-off
      ic(mi+1,im)=cc(2)
      if (ic(mi+1,im).lt.0.or.ic(mi+1,im).gt.1) go to 9006
      end if
c
      go to (100,100,300,9002,500,600,9002,9002,9002,1000,
     1 1100,1200,1300,1400,1500,1600,1700,1800,1900,2000,
     2 2100,2200,2300,2400,2500,2600,2700,2800,2900,3000,
     3 3100,3200,3300,3400,3500),ityp
c
c
c
c
c        cut-off of dipole type
c        **********************
c
c
c        store and test exponent of cut-off
  100 ic(mi+2,im)=cc(3)
      if (ic(mi+2,im).lt.0) go to 9009
      if (ic(mi+2,im).gt.0) go to 101
c        exponent is zero, omit cut-off
      ic(mi,im)=0
      ic(mi+1,im)=0
      go to 5995
c        store cut-off mass for denominator
  101 c(mm+1,im)=cc(4)*cc(4)*dwnq
c        store numerator of cut-off
      c(mm,im)=c(mm+1,im)
      if (ityp.eq.2)     c(mm,im)=c(mm,im)-c(4,im)
      mi=mi+3
      mm=mm+2
      go to 5995
c
c
c
c
c        exponential form factor of momentum transfer
c        ********************************************
```

```
c
c
c        check exponent
  300 if (cc(3).lt.0.d0) go to 9009
      if (cc(3).gt.0.d0) go to 301
c        exponent is zero, omit cutoff
      ic (mi,im)=0
      ic (mi+1,im)=0
      go to 5995
c        store exponent
  301 c(mm+1,im)=cc(3)
c        compute constant factor for argument of exponential function
      c(mm,im)=wnq/(cc(4)*cc(4))
      mi=mi+2
      mm=mm+2
      go to 5995
c
c
c
c
c        sharp cutoff in x and y
c        ***********************
c
c
  500 c(mm,im)=cc(4)*dwn
      mi=mi+2
      mm=mm+1
      go to 5995
c
c
c
c
c        exponential form factor of xx and yy
c        ************************************
c
c
c        check exponent
  600 if (cc(3).lt.0.d0) go to 9009
      if (cc(3).gt.0.d0) go to 601
c        exponent is zero, omit cutoff
      ic (mi,im)=0
      ic (mi+1,im)=0
      go to 5995
c        store exponent
  601 c(mm+1,im)=cc(3)
c        compute constant factor for argument of exponential function
      c(mm,im)=wnq/(cc(4)*cc(4))
      mi=mi+2
      mm=mm+2
      go to 5995
c
c
c
c
c        pi-gamma potential
c        ******************
c
c
 1000 c(mm,im)=cc(3)
      mi=mi+2
      mm=mm+1
      go to 5995
c
c
c
c
```

```
c        function q^2 (momentum-transfer squared)
c        ************
c
c
 1100 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        function k^2 (average-momentum squared)
c        ************
c
c
 1200 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        function 1 for tpn1
c        ********************
c
c
 1300 c(mm,im)=-1.d0/(384.d0*pi2*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        function 2 for tpn1
c        ********************
c
c
 1400 c(mm,im)=-3.d0*ga4/(64.d0*pi2*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn2, function 1
c        ****************
c
c
 1500 c(mm,im)=-3.d0*ga2/(16.d0*pi*fpi4)
      c(mm+1,im)=cc(2)*wn*1.d-3
      c(mm+2,im)=cc(3)*wn*1.d-3
      c(mm+3,im)=cc(4)*wn*1.d-3
      mi=mi+1
      mm=mm+4
      go to 5995
c
c
c
c
c        tpn2, function 2
```

```
c        ****************
c
c
 1600 c(mm,im)=-ga2/(128.d0*pi*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn2, function 3
c        ****************
c
c
 1700 c(mm,im)=9.d0*ga4/(512.d0*pi*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn2, function 4
c        ****************
c
c
 1800 c(mm,im)=-ga2/(32.d0*pi*fpi4)
      c(mm+1,im)=cc(2)*wn*1.d-3
      c(mm+2,im)=cc(3)*wn*1.d-3
      c(mm+3,im)=cc(4)*wn*1.d-3
      mi=mi+1
      mm=mm+4
      go to 5995
c
c
c
c
c
c        tpn2, function 5
c        ****************
c
c
 1900 c(mm,im)=6.d0*ga4/(64.d0*pi*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c        tpn2, function 6
c        ****************
c
c
 2000 c(mm,im)=2.d0*ga2*(1.d0-ga2)/(64.d0*pi*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        function q^4 (momentum-transfer to the power of 4)
c        ************
c
 2100 continue
```

```
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function k^4 (average-momentum to the power of 4)
c        ************
c
 2200 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function +q^2*k^2
c        *****************
c
 2300 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function  (\vec q x \vec k)^2
c        ****************************
c
 2400 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function xy
c        ***********
c
 2500 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function xx+yy
c        **************
c
 2600 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function xx*xx+yy*yy
c        ********************
c
 2700 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
```

```
c        function xx
c        ***********
c
 2800 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function yy
c        ***********
c
 2900 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn3, function 1
c        ****************
c
c
 3000 c(mm,im)=3.d0/(16.d0*pi2*fpi4)
      c(mm+1,im)=cc(2)*wn*1.d-3
      c(mm+2,im)=cc(3)*wn*1.d-3
      c(mm+3,im)=cc(4)*wn*1.d-3
      c(mm+4,im)=cc(5)*wn*1.d-3
      mi=mi+1
      mm=mm+5
      go to 5995
c
c
c
c
c        tpn3, function 2
c        ****************
c
c
 3100 continue
      cb4=cc(5)*wn*1.d-3
      c(mm,im)=cb4*cb4/(96.d0*pi2*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function 1.d0
c        *************
c
 3200 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function 1-q^2/8-k^2/2
c        **************************
c
 3300 continue
      c(mm,im)=cc(2)
      mi=mi+1
```

```
      mm=mm+1
      go to 5995
c
c
c        function 1-q^2/8
c        ****************
c
 3400 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function 1+k^2/2
c        ****************
c
 3500 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        end cut-offs and functions
c        **************************
c
c        test dimensions
 5995 if (mi.gt.mice.or.mm.gt.mme) go to 9010
c
      if (indlca) go to 7800
c
      go to 61
c
c
c
c
c        partial wave LEC's
c        ------------------
c
c
 6000 continue
c**** write (kwrite,10020) name,cc
      indlsj=.true.
      ilsj=ilsj+1
      if (ilsj.le.4) iilsj=1
      if (ilsj.ge.5.and.ilsj.le.6) iilsj=2
      if (ilsj.ge.7.and.ilsj.le.8) iilsj=3
      if (ilsj.ge.9.and.ilsj.le.10) iilsj=4
      if (ilsj.ge.11.and.ilsj.le.14) iilsj=5
      if (ilsj.ge.15.and.ilsj.le.15) iilsj=6
      if (ilsj.ge.16.and.ilsj.le.18) iilsj=7
      if (ilsj.ge.19.and.ilsj.le.19) iilsj=8
      if (ilsj.ge.20.and.ilsj.le.20) iilsj=9
      if (ilsj.ge.21.and.ilsj.le.22) iilsj=10
      if (ilsj.ge.23.and.ilsj.le.23) iilsj=12
      if (ilsj.ge.24.and.ilsj.le.24) iilsj=15
      if (ilsj.eq.1) iord=0
      if (ilsj.eq.5) iord=0
      if (ilsj.eq.7) iord=0
      if (ilsj.eq.9) iord=0
      if (ilsj.eq.11) iord=0
      if (ilsj.eq.15) iord=0
      if (ilsj.eq.16) iord=0
```

```
      if (ilsj.eq.19) iord=0
      if (ilsj.eq.20) iord=0
      if (ilsj.eq.21) iord=0
      if (ilsj.eq.23) iord=0
      if (ilsj.eq.24) iord=0
      iord=iord+1
      clsj(iord,iilsj)=cc(1)
      cutlsj(2*iord-1,iilsj)=cc(2)
      cutlsj(2*iord,iilsj)=cc(3)
      go to 61
c
c
c
c
c        lec LEC's
c        ---------
c
c
 6500 continue
c**** write (kwrite,10020) name,cc
      indlec=.true.
      ilec=ilec+1
      go to (6510,6510,6522,6540,6542,6544),ilec
 6510 do 6515 i=1,5
 6515 clec(ilec,i)=cc(i)
      go to 61
 6522 do 6523 i=1,2
 6523 clec(2,i+5)=cc(i)
      go to 61
 6540 do 6541 i=1,5
 6541 clec(3,i)=cc(i)
      go to 61
 6542 do 6543 i=1,5
 6543 clec(3,i+5)=cc(i)
      go to 61
 6544 do 6545 i=1,5
 6545 clec(3,i+10)=cc(i)
      go to 61
c
c
c
c
c        conclusions
c        -----------
c        -----------
c
c
c        write end
 7000 continue
c**** write (kwrite,10004) name
c**** write (kwrite,10008)
c**** write (kwrite,10008)
c
      if (indlsj) go to 7100
      if (indlec) go to 7500
      go to 8995
c
c
c        determine the low-energy constants (clec)
c        from the partial wave constants (clsj)
c
c
c        LEC's for Q^0 (LO)
c        ------------------
c
 7100 clec(1,1)=(clsj(1,1)+3.d0*clsj(1,5))*0.25d0/(4.d0*pi)
```

```
      clec(1,2)=(clsj(1,5)  -   clsj(1,1))*0.25d0/(4.d0*pi)
c
c
c        LEC's for Q^2 (NLO)
c        -------------------
c
c        vector b
c
c        1s0
      b(1)=clsj(2,1)
c        3p0
      b(2)=clsj(1,2)
c        1p1
      b(3)=clsj(1,3)
c        3p1
      b(4)=clsj(1,4)
c        3s1
      b(5)=clsj(2,5)
c        3s-d1
      b(6)=clsj(1,7)
c        3p2
      b(7)=clsj(1,10)
c
c
      do 7205 i=1,7
 7205 b(i)=b(i)/(4.d0*pi)
c
c
c        matrix a for the C parameters
c
c        1. column
      a(1)=1.d0
      a(2)=-2.d0/3.d0
      a(3)=a(2)
      a(4)=a(2)
      a(5)=1.d0
      a(6)=0.d0
      a(7)=a(2)
c
c        2. column
      a(8)=0.25d0
      a(9)=1.d0/6.d0
      a(10)=a(9)
      a(11)=a(9)
      a(12)=0.25d0
      a(13)=0.d0
      a(14)=a(9)
c
c        3. column
      a(15)=-3.d0
      a(16)=-2.d0/3.d0
      a(17)=2.d0
      a(18)=a(16)
      a(19)=1.d0
      a(20)=0.d0
      a(21)=a(16)
c
c        4. column
      a(22)=-0.75d0
      a(23)=1.d0/6.d0
      a(24)=-0.5d0
      a(25)=a(23)
      a(26)=0.25d0
      a(27)=0.d0
      a(28)=a(23)
c
```

```
c        5. column
      a(29)=0.d0
      a(30)=-2.d0/3.d0
      a(31)=0.d0
      a(32)=-1.d0/3.d0
      a(33)=0.d0
      a(34)=0.d0
      a(35)=1.d0/3.d0
c
c        6. column
      a(36)=-1.d0
      a(37)=2.d0
      a(38)=2.d0/3.d0
      a(39)=-4.d0/3.d0
      a(40)=1.d0/3.d0
      a(41)=-2.d0*dsqrt(2.d0)/3.d0
      a(42)=0.d0
c
c        7. column
      a(43)=-0.25d0
      a(44)=-0.5d0
      a(45)=-1.d0/6.d0
      a(46)=1.d0/3.d0
      a(47)=1.d0/12.d0
      a(48)=-dsqrt(2.d0)/6.d0
      a(49)=0.d0
c
c
c
c
      call dgelg (b,a,7,1,eps,ier)
c
      if (ier.ne.o) write (kwrite,19500) ier
19500 format (///' warning in chipar. the error index of dgelg is',
     1 ' ier =',i5/' for the calculation of the C parameters.'///)
c

      do 7255 i=1,7
 7255 clec(2,i)=b(i)
c
c
c        LEC's for Q^4 (N^3LO)
c        ---------------------
c
c        vector b
c
c        1s0
      b(1)=clsj(3,1)
c        1s0
      b(2)=clsj(4,1)
c        3p0
      b(3)=clsj(2,2)
c        1p1
      b(4)=clsj(2,3)
c        3p1
      b(5)=clsj(2,4)
c        3s1
      b(6)=clsj(3,5)
c        3s1
      b(7)=clsj(4,5)
c        3d1
      b(8)=clsj(1,6)
c        3s-d1
      b(9)=clsj(2,7)
c        3s-d1
      b(10)=clsj(3,7)
```

```
c        1d2
      b(11)=clsj(1,8)
c        3d2
      b(12)=clsj(1,9)
c        3p2
      b(13)=clsj(2,10)
c        3p-f2
      b(14)=clsj(1,12)
c        3d3
      b(15)=clsj(1,15)
c
c
      do 7305 i=1,15
 7305 b(i)=b(i)/(4.d0*pi)
c
c
c        matrix a for the D parameters
c
c        1. column
      a(1)=1.d0
      a(2)=10.d0/3.d0
      a(3)=-4.d0/3.d0
      a(4)=a(3)
      a(5)=a(3)
      a(6)=1.d0
      a(7)=a(2)
      a(8)=8.d0/15.d0
      a(9)=0.d0
      a(10)=0.d0
      a(11)=a(8)
      a(12)=a(8)
      a(13)=a(3)
      a(14)=0.d0
      a(15)=a(8)
c
c        2. column
      a(16)=1.d0/16.d0
      a(17)=5.d0/24.d0
      a(18)=1.d0/12.d0
      a(19)=a(18)
      a(20)=a(18)
      a(21)=a(16)
      a(22)=a(17)
      a(23)=1.d0/30.d0
      a(24)=0.d0
      a(25)=0.d0
      a(26)=a(23)
      a(27)=a(23)
      a(28)=a(18)
      a(29)=0.d0
      a(30)=a(23)
c
c        3. column
      a(31)=1.d0/4.d0
      a(32)=1.d0/6.d0
      a(33)=0.d0
      a(34)=0.d0
      a(35)=0.d0
      a(36)=a(31)
      a(37)=a(32)
      a(38)=-2.d0/15.d0
      a(39)=0.d0
      a(40)=0.d0
      a(41)=a(38)
      a(42)=a(38)
      a(43)=0.d0
```

```
      a(44)=0.d0
      a(45)=a(38)
c
c        4. column
      a(46)=0.d0
      a(47)=2.d0/3.d0
      a(48)=0.d0
      a(49)=0.d0
      a(50)=0.d0
      a(51)=0.d0
      a(52)=a(47)
      a(53)=-2.d0/15.d0
      a(54)=0.d0
      a(55)=0.d0
      a(56)=a(53)
      a(57)=a(53)
      a(58)=0.d0
      a(59)=0.d0
      a(60)=a(53)
c
c        5. column
      a(61)=-3.d0
      a(62)=-10.d0
      a(63)=-4.d0/3.d0
      a(64)=4.d0
      a(65)=a(63)
      a(66)=1.d0
      a(67)=10.d0/3.d0
      a(68)=8.d0/15.d0
      a(69)=0.d0
      a(70)=0.d0
      a(71)=-8.d0/5.d0
      a(72)=a(68)
      a(73)=a(63)
      a(74)=0.d0
      a(75)=a(68)
c
c        6. column
      a(76)=-3.d0/16.d0
      a(77)=-5.d0/8.d0
      a(78)=1.d0/12.d0
      a(79)=-1.d0/4.d0
      a(80)=a(78)
      a(81)=1.d0/16.d0
      a(82)=5.d0/24.d0
      a(83)=1.d0/30.d0
      a(84)=0.d0
      a(85)=0.d0
      a(86)=-1.d0/10.d0
      a(87)=a(83)
      a(88)=a(78)
      a(89)=0.d0
      a(90)=a(83)
c
c        7. column
      a(91)=-3.d0/4.d0
      a(92)=-1.d0/2.d0
      a(93)=0.d0
      a(94)=0.d0
      a(95)=0.d0
      a(96)=1.d0/4.d0
      a(97)=1.d0/6.d0
      a(98)=-2.d0/15.d0
      a(99)=0.d0
      a(100)=0.d0
      a(101)=2.d0/5.d0
```

```
      a(102)=a(98)
      a(103)=0.d0
      a(104)=0.d0
      a(105)=a(98)
c
c        8. column
      a(106)=0.d0
      a(107)=-2.d0
      a(108)=0.d0
      a(109)=0.d0
      a(110)=0.d0
      a(111)=0.d0
      a(112)=2.d0/3.d0
      a(113)=-2.d0/15.d0
      a(114)=0.d0
      a(115)=0.d0
      a(116)=2.d0/5.d0
      a(117)=a(113)
      a(118)=0.d0
      a(119)=0.d0
      a(120)=a(113)
c
c        9. column
      a(121)=0.d0
      a(122)=0.d0
      a(123)=-2.d0/3.d0
      a(124)=0.d0
      a(125)=-1.d0/3.d0
      a(126)=0.d0
      a(127)=0.d0
      a(128)=2.d0/5.d0
      a(129)=0.d0
      a(130)=0.d0
      a(131)=0.d0
      a(132)=2.d0/15.d0
      a(133)=1.d0/3.d0
      a(134)=0.d0
      a(135)=-4.d0/15.d0
c
c        10. column
      a(136)=0.d0
      a(137)=0.d0
      a(138)=-1.d0/6.d0
      a(139)=0.d0
      a(140)=-1.d0/12.d0
      a(141)=0.d0
      a(142)=0.d0
      a(143)=-1.d0/10.d0
      a(144)=0.d0
      a(145)=0.d0
      a(146)=0.d0
      a(147)=-1.d0/30.d0
      a(148)=1.d0/12.d0
      a(149)=0.d0
      a(150)=1.d0/15.d0
c
c        11. column
      a(151)=-1.d0
      a(152)=-10.d0/3.d0
      a(153)=8.d0/3.d0
      a(154)=4.d0/3.d0
      a(155)=-2.d0
      a(156)=1.d0/3.d0
      a(157)=10.d0/9.d0
      a(158)=-4.d0/9.d0
      a(159)=-2.d0*dsqrt(2.d0)/3.d0
```

```
      a(160)=-14.d0*dsqrt(2.d0)/9.d0
      a(161)=-8.d0/15.d0
      a(162)=4.d0/5.d0
      a(163)=-2.d0/15.d0
      a(164)=4.d0*dsqrt(6.d0)/15.d0
      a(165)=0.d0
c
c        12. column
      a(166)=-1.d0/4.d0
      a(167)=-1.d0/6.d0
      a(168)=1.d0/3.d0
      a(169)=0.d0
      a(170)=-1.d0/6.d0
      a(171)=1.d0/12.d0
      a(172)=1.d0/18.d0
      a(173)=1.d0/9.d0
      a(174)=-dsqrt(2.d0)/6.d0
      a(175)=dsqrt(2.d0)/18.d0
      a(176)=2.d0/15.d0
      a(177)=-1.d0/5.d0
      a(178)=1.d0/30.d0
      a(179)=-dsqrt(6.d0)/15.d0
      a(180)=0.d0
c
c        13. column
      a(181)=-1.d0/4.d0
      a(182)=-1.d0/6.d0
      a(183)=-1.d0/3.d0
      a(184)=0.d0
      a(185)=1.d0/6.d0
      a(186)=1.d0/12.d0
      a(187)=1.d0/18.d0
      a(188)=1.d0/9.d0
      a(189)=-dsqrt(2.d0)/6.d0
      a(190)=dsqrt(2.d0)/18.d0
      a(191)=2.d0/15.d0
      a(192)=-1.d0/5.d0
      a(193)=-1.d0/30.d0
      a(194)=dsqrt(6.d0)/15.d0
      a(195)=0.d0
c
c        14. column
      a(196)=-1.d0/16.d0
      a(197)=-5.d0/24.d0
      a(198)=-1.d0/6.d0
      a(199)=-1.d0/12.d0
      a(200)=1.d0/8.d0
      a(201)=1.d0/48.d0
      a(202)=5.d0/72.d0
      a(203)=-1.d0/36.d0
      a(204)=-dsqrt(2.d0)/24.d0
      a(205)=-7.d0*dsqrt(2.d0)/72.d0
      a(206)=-1.d0/30.d0
      a(207)=1.d0/20.d0
      a(208)=1.d0/120.d0
      a(209)=-dsqrt(6.d0)/60.d0
      a(210)=0.d0
c
c        15. column
      a(211)=0.d0
      a(212)=-2.d0/3.d0
      a(213)=0.d0
      a(214)=0.d0
      a(215)=0.d0
      a(216)=0.d0
      a(217)=2.d0/9.d0
```

```
      a(218)=-16.d0/45.d0
      a(219)=0.d0
      a(220)=2.d0*dsqrt(2.d0)/9.d0
      a(221)=2.d0/15.d0
      a(222)=4.d0/15.d0
      a(223)=0.d0
      a(224)=0.d0
      a(225)=-2.d0/15.d0
c
c
c
c
      call dgelg (b,a,15,1,eps,ier)
c
      if (ier.ne.o) write (kwrite,19501) ier
19501 format (///' warning in chipar. the error index of dgelg is',
     1 ' ier =',i5/' for the calculation of the D parameters.'///)
c
c
      do 7355 i=1,15
 7355 clec(3,i)=b(i)
c
c
c
c        write LEC's
c        -----------
c
c
 7500 continue
c**** write (kwrite,10100)
10100 format (//' Low energy parameters (LEC):'/
     1          ' ---------------------------'/)
c
c        Q^0 (LO)
c**** write (kwrite,10101) (clec(1,i),i=1,2)
10101 format ('lec  CS,CT',2f10.6)
c
c        Q^2 (NLO)
c**** write (kwrite,10102) (clec(2,i),i=1,7)
10102 format ('lec  C_i  ',5f10.6)
c
c        Q^4 (N^3LO)
c**** write (kwrite,10103) (clec(3,i),i=1,15)
10103 format ('lec  D_i  ',5f10.6)
c
c
c
c
c        store LEC's appropriately
c        -------------------------
c
c
      iorder=0
 7600 iorder=iorder+1
c
c
      mg=10
      iterm=0
 7700 iterm=iterm+1
c
c
      if (iorder.eq.1.and.iterm.gt.2) go to 7600
      if (iorder.eq.2.and.iterm.gt.7) go to 7600
c
c
      mg=mg+1
```

```
c
      if (iorder.eq.2) then
      if (iterm.eq.2) mg=mg-1
      if (iterm.eq.4) mg=mg-1
      end if
c
      if (iorder.eq.3) then
      if (iterm.eq.2) mg=mg-1
      if (iterm.eq.3) mg=mg-1
      if (iterm.eq.4) mg=mg-1
      if (iterm.eq.6) mg=mg-1
      if (iterm.eq.7) mg=mg-1
      if (iterm.eq.8) mg=mg-1
      if (iterm.eq.10) mg=mg-1
      if (iterm.eq.12) mg=mg-1
      if (iterm.eq.14) mg=mg-1
      end if
c
c
      ime=ime+1
      if (ime.gt.imee) go to 9011
      mgg(mg,inter)=mgg(mg,inter)+1
      m=mgg(mg,inter)
      if (m.gt.mee) go to 9001
      ima(m,mg,inter)=ime
      if (m.eq.1) then
      imga(inter)=imga(inter)+1
      mggo(imga(inter),inter)=mg
      end if
c
c
      c(1,ime)=clec(iorder,iterm)*wnq*1.d-2
      ic(1,ime)=-1
c
c
      mi=4
      mm=5
c
c
      if (indca) then
      indlca=.true.
      name(1)=cut
      do i=1,5
      cc(i)=cca(i)
      end do
      go to 72
      end if
c
c
 7800 indlca=.false.
c
c
      if (iorder.eq.2) then
      c(1,ime)=c(1,ime)*wnq*1.d-6
      if (iterm.le.4) then
      imod=mod(iterm,2)
      if (imod.eq.0) imod=2
      ic(mi,ime)=10+imod
      end if
      end if
c
c
      if (iorder.eq.3) then
      c(1,ime)=c(1,ime)*(wnq*1.d-6)**2
      if (iterm.le.8) then
      imod=mod(iterm,4)
```

```
      if (imod.eq.0) imod=4
      ic(mi,ime)=20+imod
      end if
      if (iterm.ge.9.and.iterm.le.14) then
      imod=mod(iterm,2)
      if (imod.eq.0) imod=2
      ic(mi,ime)=10+imod
      end if
      end if
c
c
 7900 if (iterm.lt.15) go to 7700
      if (iorder.lt.3) go to 7600
c
c
 8995 imaa(inter)=imb
      imea(inter)=ime
      imb=ime+1
c
      return
c
c
c
c        errors
c        ------
c        ------
c
c
c
c
 9000 write (kwrite,19000) name(1)
19000 format (1h ////' error in chipar:  contribution  ',a4,'   does not
     1 exist in this program.'/' execution terminated.'////)
      go to 9999
c
c
 9001 write (kwrite,19001)
19001 format (1h ////' error in chipar:too many contributions within a g
     1roup with respect to'/' the given dimensions. execution terminated
     2.'////)
      go to 9999
c
c
 9002 write (kwrite,19002) cc(1)
19002 format (1h ////' error in chipar: cut/fun typ',f10.4,'  does not e
     1xist in this program.'/' execution terminated.'////)
      go to 9999
c
c
 9003 write (kwrite,19003) iftyp
19003 format (1h ////' error in chipar: factor typ has the non-permissib
     1le value',i4,' .'/' execution terminated.'////)
      go to 9999
c
c
 9004 write (kwrite,19004) cc(4)
19004 format (1h ////' error in chipar: isospin has the non-permissible
     1value',f10.4,'  .'/' execution terminated.'////)
      go to 9999
c
c
 9005 write (kwrite,19005) cc(5)
19005 format (1h ////' error in chipar: iprop/spe has the non-permissibl
     1e value',f10.4,'  .'/' execution terminated.'////)
      go to 9999
c
```

```
c
 9006 write (kwrite,19006) cc(2)
19006 format (1h ////' error in chipar: the index for the propagator of
     1the cut-off has the'/' non-permissible value',f10.4,'  . execution
     2 terminated.'////)
      go to 9999
c
c
 9009 write (kwrite,19009)
19009 format (1h ////' error in chipar: the exponent of the cut-off is l
     1ess than zero.'/' execution terminated.'////)
      go to 9999
c
c
 9010 write (kwrite,19010)
19010 format (1h ////' error in chipar: too many cut/fun parameters with
     1 respect to the given'/' dimensions. execution terminated.'////)
      go to 9999
c
c
 9011 write (kwrite,19011)
19011 format (1h ////' error in chipar:  too many contr. with respect to
     1 the dimensions given'/' to this program. execution terminated.'
     2////)
      go to 9999
c
c
 9999 stop
      end
      subroutine chistr (icase,max,mex)
c
c        chistr computes the structure of one-boson-exchanges
c
c
      implicit real*8 (a-h,o-z)
c
c
c        common blocks
c
      common /crdwrt/ kread,kwrite,kpunch,kda(9)
c
      common /cstate/ j,heform,sing,trip,coup,endep,label
      logical heform,sing,trip,coup,endep
      character*4 label
c
c
c        common block for all chi-subroutines
c
      common /cchi/ vj(32,270),c(20,270),fff,ff,f(52),aa(96),ai(19,30),
     1               wnn(3),wdd(3),x,xx,y,yy,xy2,xxpyy,ex,ey,eem12,
     2               gaa(3),fpia(3),ezz1(3),ezz2(3),ct(96),wt(96),
     3               ic(20,270),ift(3),mint(3),maxt(3),nt,
     4               mge,mgg(40,3),mggo(40,3),ima(30,40,3),
     5               imaa(3),imea(3),ime,im,mc,m,mg,inter,ide,idde,
     6               indc(2,270),indpar(3),indxy
c
c         specifications for this common block
c
      logical indc,indxy,indpar
c
c     further specifications
c
      dimension vv(32)
      dimension tt(2,3)
      logical index
      logical indiso
```

```
      data jj/-1/
      data index/.false./
      save
c
c
c
c
      if (index) go to 50
      index=.true.
c
c
      tt(1,1)=1.d0
      tt(2,1)=-3.d0
c
      do 1 ii=2,3
      do 1 i=1,2
    1 tt(i,ii)=1.d0
c
c
c
c
c
   50 do 1095 m=max,mex
      im=ima(m,mg,inter)
c
c
      if (mc.ne.1) go to 60
c
c
c
c
c        call integrals
c        --------------
c
c
c
c
      call chiai
c
c
c
c
   60 if (mc.lt.1) mc=1
c
      if (c(mc,im).eq.0.d0) go to 1095
c
c
c
c
c        nn-nn helicity amplitudes /combinations/
c        ----------------------------------------
c
c
c
c
c        basic structure (a factor of 2 is included in v5 and v6)
c
c
      ive=6
c
      vv(1)=f(1)*ai(1,m)+f(2)*ai(2,m)
      vv(2)=f(3)*ai(1,m)+f(4)*ai(3,m)
      vv(3)=f(5)*ai(1,m)+f(6)*ai(2,m)
      vv(4)=f(4)*ai(1,m)+f(3)*ai(3,m)
      vv(5)=f(7)*ai(4,m)
      vv(6)=f(8)*ai(4,m)
```

```
c
c
      go to (1000,120,130,140),icase
c
c
c        additional terms required for the tensor coupling
c        of the rho-meson or for certain operators,
c        like, the spin-orbit operator (`ls  ')
c
c
  120 vv(1)=vv(1)+f(9)*ai(5,m)
      vv(2)=vv(2)+f(10)*ai(2,m)+f(9)*ai(6,m)
      vv(3)=vv(3)+f(10)*ai(5,m)
      vv(4)=vv(4)+f(9)*ai(2,m)+f(10)*ai(6,m)
         e1=f(11)*ai(7,m)
      vv(5)=vv(5)+e1
      vv(6)=vv(6)+e1
      go to 1000
c
c
c        additional terms in case of 2+ mesons
c        (not needed)
c
c
  130 continue
      go to 1000
c
c
c        additional terms needed for the sigma-l operator (`sl  ')
c
c
  140 vv(1)=vv(1)+f(6)*ai(5,m)
      vv(2)=vv(2)+f(1)*ai(5,m)+f(9)*ai(6,m)
      vv(3)=vv(3)+f(1)*ai(11,m)
      vv(4)=vv(4)+f(9)*ai(2,m)+f(1)*ai(12,m)
      vv(5)=vv(5)+f(6)*ai(13,m)
      vv(6)=vv(6)+f(6)*ai(13,m)
c
c
c
c
 1000 continue
c
c
c
c
c        set certain cases to zero in case of inter=1
c
      if (j.ne.0) go to 1021
      vv(2)=0.d0
      vv(4)=0.d0
      vv(5)=0.d0
      vv(6)=0.d0
c
 1021 if (.not.sing) vv(1)=0.d0
      if (.not.trip) vv(2)=0.d0
      if (coup) go to 1030
      do 1025 iv=3,6
 1025 vv(iv)=0.d0
c
 1030 continue
c
c
c        transformation into lsj-formalism
c
      if (j.eq.jj) go to 1035
```

```
      jj=j
      aj=dfloat(j)
      aj1=dfloat(j+1)
      d2j1=1.d0/dfloat(2*j+1)
      arjj1=dsqrt(aj*aj1)
c
 1035 v3=vv(3)
      v4=vv(4)
      v5=vv(5)
      v6=vv(6)
      v34=-arjj1*(v3-v4)
      v56=arjj1*(v5+v6)
      vv(3)=d2j1*(aj1*v3+aj*v4-v56)
      vv(4)=d2j1*(aj*v3+aj1*v4+v56)
      vv(5)=d2j1*(v34-aj1*v5+aj*v6)
      vv(6)=d2j1*(v34+aj*v5-aj1*v6)
c
c
c        possible different sign depending on the convention used
      vv(5)=-vv(5)
      vv(6)=-vv(6)
c
c
c
c
c        multiply with factors
c        ---------------------
c
c
c
c
 1040 is=mod(j,2)+1
      it=mod(is,2)+1
      indiso=indc(1,im)
      cmc=c(mc,im)
      fc=fff*ff*cmc
      do 1045 iv=1,ive
c
c        multiply with coupling-constant and factors fff and ff
c
      vv(iv)=vv(iv)*fc
c
c        multiply with isospin factor
c
      if (.not.indiso) go to 1045
      if (iv.eq.2) go to 1043
      vv(iv)=vv(iv)*tt(is,inter)
      go to 1045
 1043 vv(iv)=vv(iv)*tt(it,inter)
c
c     add up in case of several couplings for one meson-exchange
c     and store
 1045 vj(iv,im)=vj(iv,im)+vv(iv)
c
c
 1095 continue
c
c
      return
      end
      subroutine chiai
c
c        chiai integrates over theta
c
c
      implicit real*8 (a-h,o-z)
```

```
c
      common /cpot/   v(6),xmev,ymev
      common /cstate/ j,heform,sing,trip,coup,endep,label
      logical heform,sing,trip,coup,endep
      character*4 label
c
c
c        common block for all chi-subroutines
c
      common /cchi/ vj(32,270),c(20,270),fff,ff,f(52),aa(96),ai(19,30),
     1               wnn(3),wdd(3),x,xx,y,yy,xy2,xxpyy,ex,ey,eem12,
     2               gaa(3),fpia(3),ezz1(3),ezz2(3),ct(96),wt(96),
     3               ic(20,270),ift(3),mint(3),maxt(3),nt,
     4               mge,mgg(40,3),mggo(40,3),ima(30,40,3),
     5               imaa(3),imea(3),ime,im,mc,m,mg,inter,ide,idde,
     6               indc(2,270),indpar(3),indxy
c
c         specifications for this common block
c
      logical indc,indxy,indpar
c
c
c        further specifications
      dimension gi(7)
c
      dimension pj(7,96)
      real*4 axy2,aomq,am
      logical indj
      data ige/7/
      data nnt/-1/,iinter/-1/,jj/-1/
      save
c
c
c
c
      if (inter.eq.iinter) go to 60
      iinter=inter
      min=mint(inter)
      max=maxt(inter)
c
      igeint=7
c
      wn=wnn(inter)
      dwn=1.d0/wn
      wnq=wn*wn
c
c
c
c
   60 if (j.eq.jj) go to 70
      jj=j
      indj=.false.
c
c
      aj=dfloat(j)
      aj1=dfloat(j+1)
      dj1=1.d0/aj1
      ajdj1=aj*dj1
      aaj=dsqrt(ajdj1)
c
c
      aj2=dfloat(j+2)
      ajm1=dfloat(j-1)
c
c
      ajj1=aj*aj1
```

```
      ajj2=ajm1*aj2
      ajjb=aj*ajm1
c
      aajj=0.d0
      if (j.gt.1)
     1aajj=aj/dsqrt(ajj1*ajj2)
c
      aaj1=aajj*ajm1
      aaj2=aajj*aj1
      aaj3=aajj*2.d0
c
      if (j.gt.1) go to 62
      aajj=0.d0
      go to 63
   62 aajj=1.d0/(aj1*dsqrt(ajj2))
c
   63 aaj4=aajj*ajjb
      aaj5=aajj*aj1*2.d0
      aaj6=aajj*(ajj1+2.d0)
      aaj7=aajj*ajj2
c
c
c
c
c        find out appropriate number of gauss-points
c        -------------------------------------------
c
c
   70 c4=c(4,im)
      if (c4.eq.0.d0) then
      c4=(2.*138.*dwn)**2
      end if
      iprsp=ic(1,im)
c
c
c        compute am
c
      axy2=xy2
      if (iprsp.ne.1) go to 91
      aomq=eem12+c4
      go to 92
   91 aomq=xxpyy+c4
c
   92 am=axy2/aomq
c
c
c        compute number of gausspoints (nt)
c
c
      if (am.gt.0.999) go to 94
c
c
      if (am.gt.0.85) am=am**(-alog(1.-am)-0.9)
c
c
      nt=float(min)/(1.-am)+0.9
c
c
      if (nt.gt.max) nt=max
      go to 95
c
c
   94 nt=max
c
c
   95 nt=nt+j
```

```
c
c        compute nt, which is suitable for gset
c
      if (nt.le.16) go to 98
      if (nt.gt.24) go to 96
      nt=4*(nt/4)
      go to 98
   96 if (nt.gt.48) go to 97
      nt=8*(nt/8)
      go to 98
   97 nt=16*(nt/16)
      if (nt.gt.96) nt=96
c
   98 if (nt.eq.nnt.and.indj) go to 100
c
c
c
c
c        call gauss-points
c        -----------------
c
c
c
c
      call gset (-1.d0,1.d0,nt,ct,wt)
      nnt=nt
c
c
c
c
c        call legendre-polynoms if necessary
c        -----------------------------------
c
c
c
c
      indxy=.false.
      indj=.true.
      do 99 i=1,nt
      t=ct(i)
      call legp (pj(1,i),pj(3,i),t,j)
      pj(2,i)=pj(1,i)*t
      pj(4,i)=pj(2,i)*t
      pj(6,i)=pj(4,i)*t
      pj(5,i)=pj(3,i)*t
   99 pj(7,i)=pj(5,i)*t
c
c
c
c
c        call integrand
c        --------------
c
c
c
c
  100 call chiaa
c
c
c
c
c        prepare for integration
c
c
c
c
```

```
      do 2001 ig=1,igeint
 2001 gi(ig)=0.d0
c
c
c
c
c        integration-loop of theta
c        -------------------------
c
c
c
c
      do 2005 i=1,nt
      do 2005 ig=1,igeint
 2005 gi(ig)=gi(ig)+pj(ig,i)*aa(i)
c
c
c
      if (j.ne.0) go to 2010
      gi(3)=0.d0
      gi(5)=0.d0
      gi(7)=0.d0
c
c
c
c
c        combinations of integrals
c        -------------------------
c
c
c
c
 2010 ai(1,m)=gi(1)
c
      ai(2,m)=gi(2)
      ai(3,m)= ajdj1*gi(2)+dj1*gi(3)
      gi23m  =gi(2)-gi(3)
      ai(4,m)=aaj*gi23m
c
c
      ai(5,m)=gi(4)
      ai(6,m)= ajdj1*gi(4)+dj1*gi(5)
      gi45m  =gi(4)-gi(5)
      ai(7,m)=aaj*gi45m
c
c
      ai( 8,m)= aaj1*gi(4)-aaj2*gi(1)+aaj3*gi(5)
      aai1    = aaj4*gi(4)+aaj5*gi(1)-aaj6*gi(5)
      aai2    = aaj7*gi23m
      ai( 9,m)= aai2+aai1
      ai(10,m)= aai2-aai1
c
c
      ai(11,m)=gi(6)
      ai(12,m)=ajdj1*gi(6)+dj1*gi(7)
      ai(13,m)=aaj*(gi(6)-gi(7))
c
c
      return
      end
      subroutine chiaa
c
c        chiaa computes propagators, cutoffs, and functions
c
c
      implicit real*8 (a-h,o-z)
```

```
c
      common /cstate/ j,heform,sing,trip,coup,endep,label
      logical heform,sing,trip,coup,endep
      character*4 label
c
c
c        common block for all chi-subroutines
c
      common /cchi/ vj(32,270),c(20,270),fff,ff,f(52),aa(96),ai(19,30),
     1               wnn(3),wdd(3),x,xx,y,yy,xy2,xxpyy,ex,ey,eem12,
     2               gaa(3),fpia(3),ezz1(3),ezz2(3),ct(96),wt(96),
     3               ic(20,270),ift(3),mint(3),maxt(3),nt,
     4               mge,mgg(40,3),mggo(40,3),ima(30,40,3),
     5               imaa(3),imea(3),ime,im,mc,m,mg,inter,ide,idde,
     6               indc(2,270),indpar(3),indxy
c
c         specifications for this common block
c
      logical indc,indxy,indpar
c
c
      common /crrr/ rrr
c
c
c
c        further specifications
      dimension deltaq(96,7)
      dimension ell(96),cpa(96),cpaa(96)
      logical indla
      data iinter/-1/
      data cc4/-1.d0/
      save
c
c
c
c
      if (inter.eq.iinter) go to 10
      iinter=inter
      ga2=gaa(inter)
      fpi2=fpia(inter)
   10 continue
c
c
c
c
c        delta square
c        ------------
c
c
c
c
      if (indxy) go to 50
      indxy=.true.
      indla=.false.
      do 15 i=1,nt
      xy2t=xy2*ct(i)
c
c
c        function  -q^2 (- momentum-transfer-squared)
c        --------------
c
c        retardation ignored
c
      deltaq(i,1)=xy2t-xxpyy
c
c        retardation incorporated
```

```
c
      deltaq(i,2)=xy2t-eem12
c
c
c        function  +k^2 (average-momentum squared)
c        --------------
c
      deltaq(i,3)=(xy2t+xxpyy)*0.25d0
c
c        function  q^4 (momentum-transfer to the power of 4)
c        -------------
c
      deltaq(i,4)=deltaq(i,1)*deltaq(i,1)
c
c        function  k^4 (average-momentum to the power of 4)
c        -------------
c
      deltaq(i,5)=deltaq(i,3)*deltaq(i,3)
c
c        function  +q^2*k^2
c        -----------------
c
      deltaq(i,6)=-deltaq(i,1)*deltaq(i,3)
c
c        function  (\vec q x \vec k)^2
c        ----------------------------
c
      deltaq(i,7)=xx*yy*(1.d0-ct(i)*ct(i))
c
   15 continue
      go to 50
c
c
c
c     calculate ell, cpa, and cpaa
c
   20 indla=.true.
      cc4=c4
      do 25 i=1,nt
      akk=-deltaq(i,1)
      ak=dsqrt(akk)
      radi=4.d0*c4+akk
      root=dsqrt(radi)
      deno=2.d0*dsqrt(c4)
      ell(i)=root*dlog((root+ak)/deno)/ak
      cpa(i)=datan(ak/deno)/(2.d0*ak)
      cpaa(i)=(2.d0*c4+akk)*cpa(i)
   25 continue
      go to 6000
c
c
c
c
c        propagator
c        ----------
c        ----------
c
c
c
c
   50 c4=c(4,im)
      iprsp=ic(1,im)
      if (iprsp.lt.0) go to 60
      iret=iprsp+1
c
c         propagator for the nn case
```

```
      do 55 i=1,nt
   55 aa(i)=wt(i)/(c4-deltaq(i,iret))
      go to 80
c
c
c        "no propagator"
c
   60 do 65 i=1,nt
   65 aa(i)=wt(i)
c
c
   80 continue
c
c
c
c
c
c        cut-offs and functions
c        ----------------------
c        ----------------------
c
c
c
c
      mi=4
      mm=5
c
c
 5999 ityp=ic(mi,im)
      if (ityp.eq.0) go to 8000
      if (ityp.le.10) then
      iprspc=ic(mi+1,im)
      iret=iprspc+1
      end if
 6000 go to (100,100,300,100,500,600,100,100,100,1000,
     1 1100,1200,1300,1400,1500,1600,1700,1800,1900,2000,
     2 2100,2200,2300,2400,2500,2600,2700,2800,2900,3000,
     3 3100,3200,3300,3400,3500),ityp
c
c
c
c
c        cut-off of dipole type
c        **********************
c
c
  100 c5=c(mm,im)
      c6=c(mm+1,im)
      nexp=ic(mi+2,im)
c
      do 105 i=1,nt
c
      aaa=c5/(c6-deltaq(i,iret))
c     -------------------------
c
      do 105 ii=1,nexp
  105 aa(i)=aa(i)*aaa
c
c
      mi=mi+3
      mm=mm+2
      go to 5999
c
c
c
c
```

```
c        exponential form factor of momentum transfer
c        ********************************************
c
c
  300 c5=c(mm,im)
      c6=c(mm+1,im)
      do 305 i=1,nt
c
      expo=(c5*dabs(deltaq(i,iret)))**c6
c     ----------------------------
c
      if (expo.gt.rrr) expo=rrr
c
      aa(i)=aa(i)*dexp(-expo)
c     ----------------------
c
  305 continue
      mi=mi+2
      mm=mm+2
      go to 5999
c
c
c
c
c        sharp cutoff of x and y
c        ***********************
c
c
  500 c5=c(mm,im)
c
      if (x.gt.c5.or.y.gt.c5) then
c     ----------------------------
      do 505 i=1,nt
  505 aa(i)=0.d0
      end if
c
      mi=mi+2
      mm=mm+1
      go to 5999
c
c
c
c
c        exponential form factor of xx and yy
c        ************************************
c
c
  600 c5=c(mm,im)
      c6=c(mm+1,im)
c
      expo=(c5*xx)**c6+(c5*yy)**c6
c     ----------------------------
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
c     ------------------
c
      do 605 i=1,nt
  605 aa(i)=aa(i)*expexp
      mi=mi+2
      mm=mm+2
      go to 5999
c
c
c
c
c
```

```
c        pi-gamma potential
c        ******************
c
c
 1000 c5=c(mm,im)
      do 1055 i=1,nt
      betaq=-deltaq(i,1)/c4
      betaq1=betaq+1.d0
      aaa=-(1.d0-betaq)**2/(2.d0*betaq*betaq)*dlog(betaq1)
     1    +betaq1/(2.d0*betaq)
     2    -2.d0*c5
 1055 aa(i)=aa(i)*aaa
      mi=mi+2
      mm=mm+1
      go to 5999
c
c
c
c
c        function +q^2 (momentum-transfer squared)
c        *************
c
c
 1100 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 1105 i=1,nt
 1105 aa(i)=-aa(i)*deltaq(i,1)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        function k^2 (average-momentum squared)
c        ************
c
c
 1200 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 1205 i=1,nt
 1205 aa(i)=aa(i)*deltaq(i,3)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        function 1 for tpn1
c        *******************
c
c
 1300 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 1305 i=1,nt
      ga4=ga2*ga2
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      brak=4.d0*c4*(5.d0*ga4-4.d0*ga2 -1.d0)
     1    +akk*(23.d0*ga4-10.d0*ga2-1.d0)
     2    +48.d0*ga4*c4*c4/radi
 1305 aa(i)=aa(i)*c5*ell(i)*brak
      mi=mi+1
      mm=mm+1
      go to 5999
```

```
c
c
c
c
c        function 2 for tpn1
c        ********************
c
c
 1400 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 1405 i=1,nt
 1405 aa(i)=aa(i)*c5*ell(i)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn2, function 1
c        ****************
c
c
 1500 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      cb1=c(mm+1,im)
      cb3=c(mm+2,im)
      cb4=c(mm+3,im)
      do 1505 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      term1=-ga2*c4**(2.5d0)/(16.d0*radi)
      term2=(2.d0*c4*(2.d0*cb1-cb3)-akk*(cb3+3.d0/16.d0*ga2))
     1     *cpaa(i)
 1505 aa(i)=aa(i)*c5*(term1+term2)
      mi=mi+1
      mm=mm+4
      go to 5999
c
c
c
c
c        tpn2, function 2
c        ****************
c
c
 1600 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 1605 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      term1=-3.d0*ga2*c4**(2.5d0)/radi
      term2=(4.d0*c4+2.d0*akk-ga2*(4.d0*c4+3.d0*akk))
     1     *cpaa(i)
 1605 aa(i)=aa(i)*c5*(term1+term2)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn2, function 3
c        ****************
c
c
```

```
 1700 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 1705 i=1,nt
 1705 aa(i)=aa(i)*c5*cpaa(i)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn2, function 4
c        ****************
c
c
 1800 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      cb1=c(mm+1,im)
      cb3=c(mm+2,im)
      cb4=c(mm+3,im)
      do 1805 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      term1=(cb4+0.25d0)*radi
      term2=-ga2/8.d0*(10.d0*c4+3.d0*akk)
 1805 aa(i)=aa(i)*c5*(term1+term2)*cpa(i)
      mi=mi+1
      mm=mm+4
      go to 5999
c
c
c
c
c        tpn2, function 5
c        ****************
c
c
 1900 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 1905 i=1,nt
 1905 aa(i)=aa(i)*c5*cpaa(i)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn2, function 6
c        ****************
c
c
 2000 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 2005 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
 2005 aa(i)=aa(i)*c5*radi*cpa(i)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        function q^4 (momentum-transfer to the power of 4)
```

```
c        ************
c
c
 2100 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 2105 i=1,nt
 2105 aa(i)=aa(i)*deltaq(i,4)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        function k^4 (average-momentum to the power of 4)
c        ************
c
c
 2200 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 2205 i=1,nt
 2205 aa(i)=aa(i)*deltaq(i,5)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        function +q^2*k^2
c        ******************
c
c
 2300 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 2305 i=1,nt
 2305 aa(i)=aa(i)*deltaq(i,6)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        function  (\vec q x \vec k)^2
c        ****************************
c
c
 2400 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 2405 i=1,nt
 2405 aa(i)=aa(i)*deltaq(i,7)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c        function xy
c        ***********
c
 2500 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      aaxy=xy2*0.5d0*c5
      do 2505 i=1,nt
 2505 aa(i)=aa(i)*aaxy
      mi=mi+1
```

```
      mm=mm+1
      go to 5999
c
c
c        function xx+yy
c        **************
c
 2600 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 2605 i=1,nt
 2605 aa(i)=aa(i)*xxpyy*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c        function xx*xx+yy*yy
c        ********************
c
 2700 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      aaxy=(xx*xx+yy*yy)*c5
      do 2705 i=1,nt
 2705 aa(i)=aa(i)*aaxy
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c        function xx
c        ***********
c
 2800 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 2805 i=1,nt
 2805 aa(i)=aa(i)*xx*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c        function yy
c        ***********
c
 2900 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 2905 i=1,nt
 2905 aa(i)=aa(i)*yy*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn3, function 1
c        ****************
c
c
 3000 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      cb1=c(mm+1,im)
      cb2=c(mm+2,im)
      cb3=c(mm+3,im)
      cb4=c(mm+4,im)
      do 3005 i=1,nt
```

```
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      brak=(cb2*radi/6.d0+cb3*(2.d0*c4+akk)-4.d0*cb1*c4)**2
     1    +(cb2*radi)**2/45.d0
 3005 aa(i)=aa(i)*c5*ell(i)*brak
      mi=mi+1
      mm=mm+5
      go to 5999
c
c
c
c
c        tpn3, function 2
c        ****************
c
c
 3100 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 3105 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
 3105 aa(i)=aa(i)*c5*ell(i)*radi
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c        function 1.d0
c        *************
c
 3200 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 3205 i=1,nt
 3205 aa(i)=aa(i)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c        function 1-q^2/8-k^/2
c        *********************
c
 3300 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 3305 i=1,nt
 3305 aa(i)=aa(i)*(1.d0+deltaq(i,1)/8.d0-deltaq(i,3)/2.d0)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c        function 1-q^2/8
c        ****************
c
 3400 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 3405 i=1,nt
 3405 aa(i)=aa(i)*(1.d0+deltaq(i,1)/8.d0)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c        function 1+k^/2
c        ***********************
c
```

```
 3500 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 3505 i=1,nt
 3505 aa(i)=aa(i)*(1.d0+deltaq(i,3)/2.d0)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c
 8000 return
      end
      subroutine legp (pj,pjm1,x,j)
c
c
c        subroutine legp   computes the legendre polynominals
c
      real*8 pj,pjm1,x,a,b
c
c
c
c        compute legendre polynom for j equals zero
c
c
      if (j.gt.0) go to 1
      pj=1.d0
      pjm1=0.d0
      if (j.lt.0) pj=0.d0
      return
c
c
c
c        compute legendre polynoms for j equals one
c
c
c
    1 pj=x
      pjm1=1.d0
      if (j.eq.1) return
c
c
c
c        compute legendre polynom for j greater or equal two
c
c
c
      do 2 i=2,j
      a=x*pj
      b=a-pjm1
      pjm1=pj
    2 pj=-b/dfloat(i)+b+a
c
c
      return
      end
      subroutine gset(ax,bx,n,z,w)
c
c
c        this code has been obtained by R. M. from the CERN computer library
c        in the year of The Lord 1972. (God rest its soul!)
c
c
      implicit real*8 (a-h,o-z)
c
```

```
c     n-point gauss zeros and weights for the interval (ax,bx) are
c           stored in  arrays z and w respectively.
c
      dimension     a(273),x(273),ktab(96)
      dimension z(2),w(2)
c
c-----table of initial subscripts for n=2(1)16(4)96
      data ktab(2)/1/
      data ktab(3)/2/
      data ktab(4)/4/
      data ktab(5)/6/
      data ktab(6)/9/
      data ktab(7)/12/
      data ktab(8)/16/
      data ktab(9)/20/
      data ktab(10)/25/
      data ktab(11)/30/
      data ktab(12)/36/
      data ktab(13)/42/
      data ktab(14)/49/
      data ktab(15)/56/
      data ktab(16)/64/
      data ktab(20)/72/
      data ktab(24)/82/
      data ktab(28)/82/
      data ktab(32)/94/
      data ktab(36)/94/
      data ktab(40)/110/
      data ktab(44)/110/
      data ktab(48)/130/
      data ktab(52)/130/
      data ktab(56)/130/
      data ktab(60)/130/
      data ktab(64)/154/
      data ktab(68)/154/
      data ktab(72)/154/
      data ktab(76)/154/
      data ktab(80)/186/
      data ktab(84)/186/
      data ktab(88)/186/
      data ktab(92)/186/
      data ktab(96)/226/
c
c-----table of abscissae (x) and weights (a) for interval (-1,+1).
c
c**** n=2
      data x(1)/0.577350269189626  d0/, a(1)/1.000000000000000  d0/
c**** n=3
      data x(2)/0.774596669241483  d0/, a(2)/0.555555555555556  d0/
      data x(3)/0.000000000000000  d0/, a(3)/0.888888888888889  d0/
c**** n=4
      data x(4)/0.861136311594053  d0/, a(4)/0.347854845137454  d0/
      data x(5)/0.339981043584856  d0/, a(5)/0.652145154862546  d0/
c**** n=5
      data x(6)/0.906179845938664  d0/, a(6)/0.236926885056189  d0/
      data x(7)/0.538469310105683  d0/, a(7)/0.478628670499366  d0/
      data x(8)/0.000000000000000  d0/, a(8)/0.568888888888889  d0/
c**** n=6
      data x(9)/0.932469514203152  d0/, a(9)/0.171324492379170  d0/
      data x(10)/0.661209386466265 d0/, a(10)/0.360761573048139 d0/
      data x(11)/0.238619186083197 d0/, a(11)/0.467913934572691 d0/
c**** n=7
      data x(12)/0.949107912342759 d0/, a(12)/0.129484966168870 d0/
      data x(13)/0.741531185599394 d0/, a(13)/0.279705391489277 d0/
      data x(14)/0.405845151377397 d0/, a(14)/0.381830050505119 d0/
      data x(15)/0.000000000000000 d0/, a(15)/0.417959183673469 d0/
```

```
c**** n=8
      data x(16)/0.960289856497536 d0/, a(16)/0.101228536290376 d0/
      data x(17)/0.796666477413627 d0/, a(17)/0.222381034453374 d0/
      data x(18)/0.525532409916329 d0/, a(18)/0.313706645877887 d0/
      data x(19)/0.183434642495650 d0/, a(19)/0.362683783378362 d0/
c**** n=9
      data x(20)/0.968160239507626 d0/, a(20)/0.081274388361574 d0/
      data x(21)/0.836031107326636 d0/, a(21)/0.180648160694857 d0/
      data x(22)/0.613371432700590 d0/, a(22)/0.260610696402935 d0/
      data x(23)/0.324253423403809 d0/, a(23)/0.312347077040003 d0/
      data x(24)/0.000000000000000 d0/, a(24)/0.330239355001260 d0/
c**** n=10
      data x(25)/0.973906528517172 d0/, a(25)/0.066671344308688 d0/
      data x(26)/0.865063366688985 d0/, a(26)/0.149451349150581 d0/
      data x(27)/0.679409568299024 d0/, a(27)/0.219086362515982 d0/
      data x(28)/0.433395394129247 d0/, a(28)/0.269266719309996 d0/
      data x(29)/0.148874338981631 d0/, a(29)/0.295524224714753 d0/
c**** n=11
      data x(30)/0.978228658146057 d0/, a(30)/0.055668567116174 d0/
      data x(31)/0.887062599768095 d0/, a(31)/0.125580369464905 d0/
      data x(32)/0.730152005574049 d0/, a(32)/0.186290210927734 d0/
      data x(33)/0.519096129206812 d0/, a(33)/0.233193764591990 d0/
      data x(34)/0.269543155952345 d0/, a(34)/0.262804544510247 d0/
      data x(35)/0.000000000000000 d0/, a(35)/0.272925086777901 d0/
c**** n=12
      data x(36)/0.981560634246719 d0/, a(36)/0.047175336386512 d0/
      data x(37)/0.904117256370475 d0/, a(37)/0.106939325995318 d0/
      data x(38)/0.769902674194305 d0/, a(38)/0.160078328543346 d0/
      data x(39)/0.587317954286617 d0/, a(39)/0.203167426723066 d0/
      data x(40)/0.367831498998180 d0/, a(40)/0.233492536538355 d0/
      data x(41)/0.125233408511469 d0/, a(41)/0.249147045813403 d0/
c**** n=13
      data x(42)/0.984183054718588 d0/, a(42)/0.040484004765316 d0/
      data x(43)/0.917598399222978 d0/, a(43)/0.092121499837728 d0/
      data x(44)/0.801578090733310 d0/, a(44)/0.138873510219787 d0/
      data x(45)/0.642349339440340 d0/, a(45)/0.178145980761946 d0/
      data x(46)/0.448492751036447 d0/, a(46)/0.207816047536889 d0/
      data x(47)/0.230458315955135 d0/, a(47)/0.226283180262897 d0/
      data x(48)/0.000000000000000 d0/, a(48)/0.232551553230874 d0/
c**** n=14
      data x(49)/0.986283808696812 d0/, a(49)/0.035119460331752 d0/
      data x(50)/0.928434883663574 d0/, a(50)/0.080158087159760 d0/
      data x(51)/0.827201315069765 d0/, a(51)/0.121518570687903 d0/
      data x(52)/0.687292904811685 d0/, a(52)/0.157203167158194 d0/
      data x(53)/0.515248636358154 d0/, a(53)/0.185538397477938 d0/
      data x(54)/0.319112368927890 d0/, a(54)/0.205198463721296 d0/
      data x(55)/0.108054948707344 d0/, a(55)/0.215263853463158 d0/
c**** n=15
      data x(56)/0.987992518020485 d0/, a(56)/0.030753241996117 d0/
      data x(57)/0.937273392400706 d0/, a(57)/0.070366047488108 d0/
      data x(58)/0.848206583410427 d0/, a(58)/0.107159220467172 d0/
      data x(59)/0.724417731360170 d0/, a(59)/0.139570677926154 d0/
      data x(60)/0.570972172608539 d0/, a(60)/0.166269205816994 d0/
      data x(61)/0.394151347077563 d0/, a(61)/0.186161000015562 d0/
      data x(62)/0.201194093997435 d0/, a(62)/0.198431485327111 d0/
      data x(63)/0.000000000000000 d0/, a(63)/0.202578241925561 d0/
c**** n=16
      data x(64)/0.989400934991650 d0/, a(64)/0.027152459411754 d0/
      data x(65)/0.944575023073233 d0/, a(65)/0.062253523938648 d0/
      data x(66)/0.865631202387832 d0/, a(66)/0.095158511682493 d0/
      data x(67)/0.755404408355003 d0/, a(67)/0.124628971255534 d0/
      data x(68)/0.617876244402644 d0/, a(68)/0.149595988816577 d0/
      data x(69)/0.458016777657227 d0/, a(69)/0.169156519395003 d0/
      data x(70)/0.281603550779259 d0/, a(70)/0.182603415044924 d0/
      data x(71)/0.095012509837637 d0/, a(71)/0.189450610455069 d0/
c**** n=20
```

```
      data x(72)/0.993128599185094 d0/, a(72)/0.017614007139152 d0/
      data x(73)/0.963971927277913 d0/, a(73)/0.040601429800386 d0/
      data x(74)/0.912234428251325 d0/, a(74)/0.062672048334109 d0/
      data x(75)/0.839116971822218 d0/, a(75)/0.083276741576704 d0/
      data x(76)/0.746331906460150 d0/, a(76)/0.101930119817240 d0/
      data x(77)/0.636053680726515 d0/, a(77)/0.118194531961518 d0/
      data x(78)/0.510867001950827 d0/, a(78)/0.131688638449176 d0/
      data x(79)/0.373706088715419 d0/, a(79)/0.142096109318382 d0/
      data x(80)/0.227785851141645 d0/, a(80)/0.149172986472603 d0/
      data x(81)/0.076526521133497 d0/, a(81)/0.152753387130725 d0/
c**** n=24
      data x(82)/0.995187219997021 d0/, a(82)/0.012341229799987 d0/
      data x(83)/0.974728555971309 d0/, a(83)/0.028531388628933 d0/
      data x(84)/0.938274552002732 d0/, a(84)/0.044277438817419 d0/
      data x(85)/0.886415527004401 d0/, a(85)/0.059298584915436 d0/
      data x(86)/0.820001985973902 d0/, a(86)/0.073346481411080 d0/
      data x(87)/0.740124191578554 d0/, a(87)/0.086190161531953 d0/
      data x(88)/0.648093651936975 d0/, a(88)/0.097618652104113 d0/
      data x(89)/0.545421471388839 d0/, a(89)/0.107444270115965 d0/
      data x(90)/0.433793507626045 d0/, a(90)/0.115505668053725 d0/
      data x(91)/0.315042679696163 d0/, a(91)/0.121670472927803 d0/
      data x(92)/0.191118867473616 d0/, a(92)/0.125837456346828 d0/
      data x(93)/0.064056892862605 d0/, a(93)/0.127938195346752 d0/
c**** n=32
      data x(94)/0.997263861849481 d0/, a(94)/0.007018610009470 d0/
      data x(95)/0.985611511545268 d0/, a(95)/0.016274394730905 d0/
      data x(96)/0.964762255587506 d0/, a(96)/0.025392065309262 d0/
      data x(97)/0.934906075937739 d0/, a(97)/0.034273862913021 d0/
      data x(98)/0.896321155766052 d0/, a(98)/0.042835898022226 d0/
      data x(99)/0.849367613732569 d0/, a(99)/0.050998059262376 d0/
      data x(100)/0.794483795967942d0/, a(100)/0.058684093478535d0/
      data x(101)/0.732182118740289d0/, a(101)/0.065822222776361d0/
      data x(102)/0.663044266930215d0/, a(102)/0.072345794108848d0/
      data x(103)/0.587715757240762d0/, a(103)/0.078193895787070d0/
      data x(104)/0.506899908932229d0/, a(104)/0.083311924226946d0/
      data x(105)/0.421351276130635d0/, a(105)/0.087652093004403d0/
      data x(106)/0.331868602282127d0/, a(106)/0.091173878695763d0/
      data x(107)/0.239287362252137d0/, a(107)/0.093844399080804d0/
      data x(108)/0.144471961582796d0/, a(108)/0.095638720079274d0/
      data x(109)/0.048307665687738d0/, a(109)/0.096540088514727d0/
c**** n=40
      data x(110)/0.998237709710559d0/, a(110)/0.004521277098533d0/
      data x(111)/0.990726238699457d0/, a(111)/0.010498284531152d0/
      data x(112)/0.977259949983774d0/, a(112)/0.016421058381907d0/
      data x(113)/0.957916819213791d0/, a(113)/0.022245849194166d0/
      data x(114)/0.932812808278676d0/, a(114)/0.027937006980023d0/
      data x(115)/0.902098806968874d0/, a(115)/0.033460195282547d0/
      data x(116)/0.865959503212259d0/, a(116)/0.038782167974472d0/
      data x(117)/0.824612230833311d0/, a(117)/0.043870908185673d0/
      data x(118)/0.778305651426519d0/, a(118)/0.048695807635072d0/
      data x(119)/0.727318255189927d0/, a(119)/0.053227846983936d0/
      data x(120)/0.671956684614179d0/, a(120)/0.057439769099391d0/
      data x(121)/0.612553889667980d0/, a(121)/0.061306242492928d0/
      data x(122)/0.549467125095128d0/, a(122)/0.064804013456601d0/
      data x(123)/0.483075801686178d0/, a(123)/0.067912045815233d0/
      data x(124)/0.413779204371605d0/, a(124)/0.070611647391286d0/
      data x(125)/0.341994090825758d0/, a(125)/0.072886582395804d0/
      data x(126)/0.268152185007253d0/, a(126)/0.074723169057968d0/
      data x(127)/0.192697580701371d0/, a(127)/0.076110361900626d0/
      data x(128)/0.116084070675255d0/, a(128)/0.077039818164247d0/
      data x(129)/0.038772417506050d0/, a(129)/0.077505947978424d0/
c**** n=48
      data x(130)/0.998771007252426d0/, a(130)/0.003153346052305d0/
      data x(131)/0.993530172266350d0/, a(131)/0.007327553901276d0/
      data x(132)/0.984124583722826d0/, a(132)/0.011477234579234d0/
      data x(133)/0.970591592546247d0/, a(133)/0.015579315722943d0/
```

```
      data x(134)/0.952987703160430d0/, a(134)/0.019616160457355d0/
      data x(135)/0.931386690706554d0/, a(135)/0.023570760839324d0/
      data x(136)/0.905879136715569d0/, a(136)/0.027426509708356d0/
      data x(137)/0.876572020274247d0/, a(137)/0.031167227832798d0/
      data x(138)/0.843588261624393d0/, a(138)/0.034777222564770d0/
      data x(139)/0.807066204029442d0/, a(139)/0.038241351065830d0/
      data x(140)/0.767159032515740d0/, a(140)/0.041545082943464d0/
      data x(141)/0.724034130923814d0/, a(141)/0.044674560856694d0/
      data x(142)/0.677872379632663d0/, a(142)/0.047616658492490d0/
      data x(143)/0.628867396776513d0/, a(143)/0.050359035553854d0/
      data x(144)/0.577224726083972d0/, a(144)/0.052890189485193d0/
      data x(145)/0.523160974722233d0/, a(145)/0.055199503699984d0/
      data x(146)/0.466902904750958d0/, a(146)/0.057277292100403d0/
      data x(147)/0.408686481990716d0/, a(147)/0.059114839698395d0/
      data x(148)/0.348755886292160d0/, a(148)/0.060704439165893d0/
      data x(149)/0.287362487355455d0/, a(149)/0.062039423159892d0/
      data x(150)/0.224763790394689d0/, a(150)/0.063114192286254d0/
      data x(151)/0.161222356068891d0/, a(151)/0.063924238584648d0/
      data x(152)/0.097004699209462d0/, a(152)/0.064466164435950d0/
      data x(153)/0.032380170962869d0/, a(153)/0.064737696812683d0/
c**** n=64
      data x(154)/0.999305041735772d0/, a(154)/0.001783280721696d0/
      data x(155)/0.996340116771955d0/, a(155)/0.004147033260562d0/
      data x(156)/0.991013371476744d0/, a(156)/0.006504457968978d0/
      data x(157)/0.983336253884625d0/, a(157)/0.008846759826363d0/
      data x(158)/0.973326827789910d0/, a(158)/0.011168139460131d0/
      data x(159)/0.961008799652053d0/, a(159)/0.013463047896718d0/
      data x(160)/0.946411374858402d0/, a(160)/0.015726030476024d0/
      data x(161)/0.929569172131939d0/, a(161)/0.017951715775697d0/
      data x(162)/0.910522137078502d0/, a(162)/0.020134823153530d0/
      data x(163)/0.889315445995114d0/, a(163)/0.022270173808383d0/
      data x(164)/0.865999398154092d0/, a(164)/0.024352702568710d0/
      data x(165)/0.840629296252580d0/, a(165)/0.026377469715054d0/
      data x(166)/0.813265315122797d0/, a(166)/0.028339672614259d0/
      data x(167)/0.783972358943341d0/, a(167)/0.030234657072402d0/
      data x(168)/0.752819907260531d0/, a(168)/0.032057928354851d0/
      data x(169)/0.719881850171610d0/, a(169)/0.033805161837141d0/
      data x(170)/0.685236313054233d0/, a(170)/0.035472213256882d0/
      data x(171)/0.648965471254657d0/, a(171)/0.037055128540240d0/
      data x(172)/0.611155355172393d0/, a(172)/0.038550153178615d0/
      data x(173)/0.571895646202634d0/, a(173)/0.039953741132720d0/
      data x(174)/0.531279464019894d0/, a(174)/0.041262563242623d0/
      data x(175)/0.489403145707052d0/, a(175)/0.042473515123653d0/
      data x(176)/0.446366017253464d0/, a(176)/0.043583724529323d0/
      data x(177)/0.402270157963991d0/, a(177)/0.044590558163756d0/
      data x(178)/0.357220158337668d0/, a(178)/0.045491627927418d0/
      data x(179)/0.311322871990210d0/, a(179)/0.046284796581314d0/
      data x(180)/0.264687162208767d0/, a(180)/0.046968182816210d0/
      data x(181)/0.217423643740007d0/, a(181)/0.047540165714830d0/
      data x(182)/0.169644420423992d0/, a(182)/0.047999388596458d0/
      data x(183)/0.121462819296120d0/, a(183)/0.048344762234802d0/
      data x(184)/0.072993121787799d0/, a(184)/0.048575467441503d0/
      data x(185)/0.024350292663424d0/, a(185)/0.048690957009139d0/
c**** n=80
      data x(186)/0.999553822651630d0/, a(186)/0.001144950003186d0/
      data x(187)/0.997649864398237d0/, a(187)/0.002663533589512d0/
      data x(188)/0.994227540965688d0/, a(188)/0.004180313124694d0/
      data x(189)/0.989291302499755d0/, a(189)/0.005690922451403d0/
      data x(190)/0.982848572738629d0/, a(190)/0.007192904768117d0/
      data x(191)/0.974909140585727d0/, a(191)/0.008683945269260d0/
      data x(192)/0.965485089043799d0/, a(192)/0.010161766041103d0/
      data x(193)/0.954590766343634d0/, a(193)/0.011624114120797d0/
      data x(194)/0.942242761309872d0/, a(194)/0.013068761592401d0/
      data x(195)/0.928459877172445d0/, a(195)/0.014493508040509d0/
      data x(196)/0.913263102571757d0/, a(196)/0.015896183583725d0/
      data x(197)/0.896675579438770d0/, a(197)/0.017274652056269d0/
```

```
      data x(198)/0.878722567678213d0/, a(198)/0.018626814208299d0/
      data x(199)/0.859431406663111d0/, a(199)/0.019950610878141d0/
      data x(200)/0.838831473580255d0/, a(200)/0.021244026115782d0/
      data x(201)/0.816954138681463d0/, a(201)/0.022505090246332d0/
      data x(202)/0.793832717504605d0/, a(202)/0.023731882865930d0/
      data x(203)/0.769502420135041d0/, a(203)/0.024922535764115d0/
      data x(204)/0.744000297583597d0/, a(204)/0.026075235767565d0/
      data x(205)/0.717365185362099d0/, a(205)/0.027188227500486d0/
      data x(206)/0.689637644342027d0/, a(206)/0.028259816057276d0/
      data x(207)/0.660859898986119d0/, a(207)/0.029288369583267d0/
      data x(208)/0.631075773046871d0/, a(208)/0.030272321759557d0/
      data x(209)/0.600330622829751d0/, a(209)/0.031210174188114d0/
      data x(210)/0.568671268122709d0/, a(210)/0.032100498673487d0/
      data x(211)/0.536145920897131d0/, a(211)/0.032941939397645d0/
      data x(212)/0.502804111888784d0/, a(212)/0.033733214984611d0/
      data x(213)/0.468696615170544d0/, a(213)/0.034473120451753d0/
      data x(214)/0.433875370831756d0/, a(214)/0.035160529044747d0/
      data x(215)/0.398393405881969d0/, a(215)/0.035794393953416d0/
      data x(216)/0.362304753499487d0/, a(216)/0.036373749905835d0/
      data x(217)/0.325664370747701d0/, a(217)/0.036897714638276d0/
      data x(218)/0.288528054884511d0/, a(218)/0.037365490238730d0/
      data x(219)/0.250952358392272d0/, a(219)/0.037776364362001d0/
      data x(220)/0.212994502857666d0/, a(220)/0.038129711314477d0/
      data x(221)/0.174712291832646d0/, a(221)/0.038424993006959d0/
      data x(222)/0.136164022809143d0/, a(222)/0.038661759774076d0/
      data x(223)/0.097408398441584d0/, a(223)/0.038839651059051d0/
      data x(224)/0.058504437152420d0/, a(224)/0.038958395962769d0/
      data x(225)/0.019511383256793d0/, a(225)/0.039017813656306d0/
c**** n=96
      data x(226)/0.999689503883230d0/, a(226)/0.000796792065552d0/
      data x(227)/0.998364375863181d0/, a(227)/0.001853960788946d0/
      data x(228)/0.995981842987209d0/, a(228)/0.002910731817934d0/
      data x(229)/0.992543900323762d0/, a(229)/0.003964554338444d0/
      data x(230)/0.988054126329623d0/, a(230)/0.005014202742927d0/
      data x(231)/0.982517263563014d0/, a(231)/0.006058545504235d0/
      data x(232)/0.975939174585136d0/, a(232)/0.007096470791153d0/
      data x(233)/0.968326828463264d0/, a(233)/0.008126876925698d0/
      data x(234)/0.959688291448742d0/, a(234)/0.009148671230783d0/
      data x(235)/0.950032717784437d0/, a(235)/0.010160770535008d0/
      data x(236)/0.939370339752755d0/, a(236)/0.011162102099838d0/
      data x(237)/0.927712456722308d0/, a(237)/0.012151604671088d0/
      data x(238)/0.915071423120898d0/, a(238)/0.013128229566961d0/
      data x(239)/0.901460635315852d0/, a(239)/0.014090941772314d0/
      data x(240)/0.886894517402420d0/, a(240)/0.015038721026994d0/
      data x(241)/0.871388505909296d0/, a(241)/0.015970562902562d0/
      data x(242)/0.854959033434601d0/, a(242)/0.016885479864245d0/
      data x(243)/0.837623511228187d0/, a(243)/0.017782502316045d0/
      data x(244)/0.819400310737931d0/, a(244)/0.018660679627411d0/
      data x(245)/0.800308744139140d0/, a(245)/0.019519081140145d0/
      data x(246)/0.780369043867433d0/, a(246)/0.020356797154333d0/
      data x(247)/0.759602341176647d0/, a(247)/0.021172939892191d0/
      data x(248)/0.738030643744400d0/, a(248)/0.021966644438744d0/
      data x(249)/0.715676812348967d0/, a(249)/0.022737069658329d0/
      data x(250)/0.692564536642171d0/, a(250)/0.023483399085926d0/
      data x(251)/0.668718310043916d0/, a(251)/0.024204841792364d0/
      data x(252)/0.644163403784967d0/, a(252)/0.024900633222483d0/
      data x(253)/0.618925840125468d0/, a(253)/0.025570036005349d0/
      data x(254)/0.593032364777572d0/, a(254)/0.026212340735672d0/
      data x(255)/0.566510418561397d0/, a(255)/0.026826866725591d0/
      data x(256)/0.539388108324357d0/, a(256)/0.027412962726029d0/
      data x(257)/0.511694177154667d0/, a(257)/0.027970007616848d0/
      data x(258)/0.483457973920596d0/, a(258)/0.028497411065085d0/
      data x(259)/0.454709422167743d0/, a(259)/0.028994614150555d0/
      data x(260)/0.425478988407300d0/, a(260)/0.029461089958167d0/
      data x(261)/0.395797649828908d0/, a(261)/0.029896344136328d0/
      data x(262)/0.365696861472313d0/, a(262)/0.030299915420827d0/
```

```
      data x(263)/0.335208522892625d0/, a(263)/0.030671376123669d0/
      data x(264)/0.304364944354496d0/, a(264)/0.031010332586313d0/
      data x(265)/0.273198812591049d0/, a(265)/0.031316425596861d0/
      data x(266)/0.241743156163840d0/, a(266)/0.031589330770727d0/
      data x(267)/0.210031310460567d0/, a(267)/0.031828758894411d0/
      data x(268)/0.178096882367618d0/, a(268)/0.032034456231992d0/
      data x(269)/0.145973714654896d0/, a(269)/0.032206204794030d0/
      data x(270)/0.113695850110665d0/, a(270)/0.032343822568575d0/
      data x(271)/0.081297495464425d0/, a(271)/0.032447163714064d0/
      data x(272)/0.048812985136049d0/, a(272)/0.032516118713868d0/
      data x(273)/0.016276744849602d0/, a(273)/0.032550614492363d0/
c
c
c-----test n
      alpha=0.5d0*(ax+bx)
      beta=0.5d0*(bx-ax)
      if( n.lt.1 .or. n.gt.96 ) go to 100
      if(n.ne.1) go to 1
      z(1)=alpha
      w(1)=bx-ax
      return
c
    1 if (n.le.16) go to 3
      if (n.gt.24) go to 4
      n=4*(n/4)
      go to 3
    4 if (n.gt.48) go to 5
      n=8*(n/8)
      go to 3
    5 n=16*(n/16)
c
c----- set k equal to initial subscript and store results
    3 k=ktab(n)
      m=n/2
      do 2 j=1,m
      jtab=k-1+j
      wtemp=beta*a(jtab)
      delta=beta*x(jtab)
      z(j)=alpha-delta
      w(j)=wtemp
      jp=n+1-j
      z(jp)=alpha+delta
      w(jp)=wtemp
    2 continue
      if((n-m-m).eq.0) return
      z(m+1)=alpha
      jmid=k+m
      w(m+1)=beta*a(jmid)
      return
c
  100 zn=n
      write(6,200) zn
  200 format(/////' error in gset. n has the non-permissible value',
     1e11.3/' execution terminated.')
      stop
      end
```

```
c name:    dgelg
c        programmbibliothek rhrz bonn        02/02/81       dgelg
c                                            fortran iv     ibm 370/168
c
c purpose:
c
c to solve a general system of simultaneous linear equations.
c
c usage:   call dgelg(r,a,m,n,eps,ier)
c
c parameters:
c
c r:       double precision m by n right hand side matrix
c          (destroyed). on return r contains the solutions
c          of the equations.
c
c a:       double precision m by m coefficient matrix
c          (destroyed).
c
c m:       the number of equations in the system.
c
c n:       the number of right hand side vectors.
c
c eps:     single precision input constant which is used as
c          relative tolerance for test on loss of
c          significance.
c
c ier:     resulting error parameter coded as follows
c           ier=0  - no error,
c           ier=-1 - no result because of m less than 1 or
c                    pivot element at any elimination step
c                    equal to 0,
c           ier=k  - warning due to possible loss of signifi-
c                    cance indicated at elimination step k+1,
c                    where pivot element was less than or
```

```
c                   equal to the internal tolerance eps times
c                   absolutely greatest element of matrix a.
c
c remarks: (1) input matrices r and a are assumed to be stored
c              columnwise in m*n resp. m*m successive storage
c              locations. on return solution matrix r is stored
c              columnwise too.
c          (2) the procedure gives results if the number of equations m
c              is greater than 0 and pivot elements at all elimination
c              steps are different from 0. however warning ier=k - if
c              given indicates possible loss of significance. in case
c              of a well scaled matrix a and appropriate tolerance eps,
c              ier=k may be interpreted that matrix a has the rank k.
c              no warning is given in case m=1.
c
c method:
c
c solution is done by means of gauss-elimination with
c complete pivoting.
c
c programs required:
c          none
c
c access:
c
c load module:    sys3.fortlib(dgelg)
c source module:  sys3.symlib.fortran(dgelg)
c description:    sys3.infolib(dgelg)
c
c author:         ibm, ssp iii
c installation:   ibm 370/168, mvs-jes2, fortran iv (h ext. enh.)
c
c***********************************************************************
      subroutine dgelg(r,a,m,n,eps,ier)
c
c
      implicit real*8 (a-h,o-z)
      dimension a(1),r(1)
      real*4 eps
c
c
c
c
      if(m)23,23,1
c
c     search for greatest element in matrix a
    1 ier=0
      piv=0.d0
      mm=m*m
      nm=n*m
      do 3 l=1,mm
      tb=dabs(a(l))
      if(tb-piv)3,3,2
    2 piv=tb
      i=l
    3 continue
      tol=eps*piv
c     a(i) is pivot element. piv contains the absolute value of a(i).
c
c
c     start elimination loop
      lst=1
      do 17 k=1,m
c
c     test on singularity
      if(piv)23,23,4
```

```
    4 if(ier)7,5,7
    5 if(piv-tol)6,6,7
    6 ier=k-1
    7 pivi=1.d0/a(i)
      j=(i-1)/m
      i=i-j*m-k
      j=j+1-k
c     i+k is row-index, j+k column-index of pivot element
c
c     pivot row reduction and row interchange in right hand side r
      do 8 l=k,nm,m
      ll=l+i
      tb=pivi*r(ll)
      r(ll)=r(l)
    8 r(l)=tb
c
c     is elimination terminated
      if(k-m)9,18,18
c
c     column interchange in matrix a
    9 lend=lst+m-k
      if(j)12,12,10
   10 ii=j*m
      do 11 l=lst,lend
      tb=a(l)
      ll=l+ii
      a(l)=a(ll)
   11 a(ll)=tb
c
c     row interchange and pivot row reduction in matrix a
   12 do 13 l=lst,mm,m
      ll=l+i
      tb=pivi*a(ll)
      a(ll)=a(l)
   13 a(l)=tb
c
c     save column interchange information
      a(lst)=j
c
c     element reduction and next pivot search
      piv=0.d0
      lst=lst+1
      j=0
      do 16 ii=lst,lend
      pivi=-a(ii)
      ist=ii+m
      j=j+1
      do 15 l=ist,mm,m
      ll=l-j
      a(l)=a(l)+pivi*a(ll)
      tb=dabs(a(l))
      if(tb-piv)15,15,14
   14 piv=tb
      i=l
   15 continue
      do 16 l=k,nm,m
      ll=l+j
   16 r(ll)=r(ll)+pivi*r(l)
   17 lst=lst+m
c     end of elimination loop
c
c
c     back substitution and back interchange
   18 if(m-1)23,22,19
   19 ist=mm+m
      lst=m+1
```

```
      do 21 i=2,m
      ii=lst-i
      ist=ist-lst
      l=ist-m
      l=a(l)+.5d0
      do 21 j=ii,nm,m
      tb=r(j)
      ll=j
      do 20 k=ist,mm,m
      ll=ll+1
   20 tb=tb-a(k)*r(ll)
      k=j+l
      r(j)=r(k)
   21 r(k)=tb
   22 return
c
c
c     error return
   23 ier=-1
      return
      end
```

```
c      program coul
c      call fafita(6,1,0,0.,0.,1.0)
c      osla=1.72110
c      x=cmagjj(0,2,3,0,2,3,0,2,3,0,2,3,0,1,1)/osla
c      write(*,*) osla, x
c      end
cDear Morton,c
c
cbelow you will find the programms to calculate the matrix-elements for
cthe coulomb interaction. Note the following points:
c- in the present form all programs are in single precision
```

```
c- the matrix elements are calculated in an oscillator basis, in which
c  the single-particle states are defined by integers n,l,j with
c  n=0,1... the radial quantum number
c  l=0,1.... the orbital angular momentum
c  j=1,3....  two times the total single-particle angular momentum
c- the matrix element <a,b/V/c,d> is calculated calling the function
c   cmaggj (na,la,ja,nb,lb,jb,nc,ld,jd,jtot,isosp,isopro)
c   and dividing the result by the oscillator length of the harmonic oscillator
c   functions in the lab coordinates (given in fm).
c   jtot is the total angular momentum of the two-particle states,
c   isosp the total isospin
c   and isopro the projection of isospin, i.e. =1 for proton proton
c   so a typical example for 16 O:
c     osla=1.72
c     x=cmagjj(0,0,1,0,0,1,0,0,1,0,0,1,0,1,1)/oslac
c
c- the resulting matrix elements are antisymmetrized, however, they
c   do not include the normalisation factor
((1+delta_{a,b})*(1+delta_{c,d}))**c(-1/2)

c- before you calculate a coulomb matrix element you should call once
c    the subroutine FAFITA with the following arguments:

c    call fafita(6,1,0,0.,0.,1.0)

      FUNCTION CMAGJJ(NODEIK,LBIK,JJKM2,NODEIL,LBIL,JJLM2,
     1                NODEIM,LBIM,JJMM2,NODEIN,LBIN,JJNM2,
     2                JTOT,ISOSP,ISOPRO)
C  ******************************************************************
C
C  UP ZUR BERECHNUNG EINES J-J GEKOPPELTEN MATRIXELEMENTES
C
C                 ( K L /V/ M N )
C
C  DER COULOMBKRAFT IN DER NORMIERUNG
C
C                 (LADUNG**2)*(OSZILLATORLAENGE/R).
C
C  DAS MATRIXELEMENT HAENGT IN DIESER NORMIERUNG NICHT VON DER
C  OSZILLATORLAENGE AB, WENN DIE ZUSTAENDE |K>, |L>, |M>, |N> EIGEN-
C  ZUSTAENDE EINES HARMONISCHEN OSZILLATORPOTENTIALS SIND.
C  DIE GUTEN QUANTENZAHLEN DES ZUSTANDES |K> SIND:
C
C                 NODEIK  RADIALQUANTENZAHL N
C                 LBIK    BAHNDREHIMPULS L
C                 JJKM2   TOTALER EINTEILCHENDREHIMPULS J * 2
C
C  DIE EINTEILCHENDREHIMPULSE J SIND ZUM TOTALEN DREHIMPULS JTOT
C  GEKOPPELT, DIE ISOSPINS ZUM TOTALEN ISISPIN ISOSP.  DAS MATRIXELEMENT
C  HAENGT NICHT VON DER PROJEKTION DES GESAMTDREHIMPULSES JTOT AB.
C  DAS MATRIXELEMENT IST 0 , WENN DER GESAMTISOSPIN 0 ODER DIE
C  ISOSPINPROJEKTION ISOPRO NICHT +1 IST.                              #
C
C     ZUR RICHTIGEN NORMIERUNG DES ANTISYMMETRSIERTEN MATRIXEL. FEHLT
C     DER FAKTOR
C             ( (1+DELTA(K,L)) * (1+DELTA(M,N) ) ** (-1/2)
C     DAZU C AUS DEN KARTEN DER AM ENDE GEKENNZ. STELLE ENTFERNEN.
C
C  DAS PROGRAMM RECHNET NACH DER HORIE-SASAKI-METHODE.
C  ES BENOETIGT DIE UNTERPROGRAMME FAFITA,SLAT,HISLAT,ELKE1,TENSOR,
C  VORZ,WFU,CFU,STIRL,ONEP,IDR. IM HAUPTPROGRAMM IST VOR BENUTZUNG DES
C  PROGRAMMS CMAGJJ DAS UP FAFITA IN DER FORM
C
C        CALL FAFITA(6,1,0,0,0,0)
C
C  AUFZURUFEN.
```

```
C
C  *********************************************************************
C
      COMMON/ERROR/IERR
C  WENN
C     IERR = 0 , KEINE AUSWAHLREGEL DER QUANTENZAHLEN VERLETZT,
C     IERR = 1 , AUSWAHLREGEL DER QUANTENZAHLEN VERLETZT.
C
C
      IF((ISOSP.NE.1).OR .(ISOPRO.NE.1)) GOTO 63
C
      AY1 = FLOAT(JJKM2)/2.0
      BY1 = FLOAT(JJLM2)/2.0
      AY2 = FLOAT(JJMM2)/2.0
      BY2 = FLOAT(JJNM2)/2.0
C
      ATRIX=0.0
C
      K5 = MAX0(IABS(JJKM2 - JJLM2),IABS(JJMM2 - JJNM2))/2
      K6 = MIN0(JJKM2 + JJLM2,JJMM2 + JJNM2)/2
      W = JTOT                                                          COUL1140
      K7=1
      IF(((JJKM2.EQ.JJLM2).AND.( NODEIK.EQ.NODEIL ).AND.( LBIK.EQ.LBIL
     1)).OR.(( JJMM2.EQ.JJNM2 ).AND.(NODEIM.EQ.NODEIN ).AND.(LBIM.EQ.LBI
     2N))) GOTO 15
      GO TO 16                                                                 #
   15 K7=2
      IF(MOD(K5,2).EQ.1) K5=K5+1
   16 IF((JTOT.LT.K5).OR.(JTOT.GT.K6)) GOTO 63                          COUL1220
      IF(MOD(JTOT-K5,K7).NE.0) GOTO 63                                  COUL1230
      K1 = MAX0(IABS( JJKM2 - JJMM2), IABS(JJLM2 - JJNM2))/2
      AS1=0.0
      IF(MOD(LBIK + LBIM + K1,2).EQ.1) K1 = K1 + 1
      IF(MOD(LBIL + LBIN + K1,2).EQ.1) GOTO 1
      K1=K1+1
      K2 = MIN0(JJKM2 + JJMM2,JJLM2 + JJNM2)/2 + 1
      IF(K1.GT.K2) GOTO 1
      DO61 K=K1,K2,2
      SK1=K-1
   61 AS1=AS1+  CFU(AY1,AY2,SK1,0.5,-0.5,0.0)*  CFU(BY1,BY2,SK1,0.5,-0.5
     1,0.0)*  WFU(AY1,BY1,AY2,BY2,W,SK1)*
     2SLAT(K - 1,NODEIK + 1,LBIK, NODEIL + 1, LBIL,NODEIM + 1,
     3LBIM,NODEIN + 1, LBIN) / (2.0 * SK1 + 1.0)
      P1=1.
      IF(MOD( (JJLM2 + JJNM2)/2 + JTOT,2).EQ.1) P1 = -1.0
    1 K1 = MAX0(IABS( JJKM2 - JJNM2), IABS(JJLM2 - JJMM2))/2
      AS2=0.0
      IF(MOD(LBIK + LBIN + K1,2).EQ.1) K1 = K1 + 1
      IF(MOD(LBIL + LBIM + K1,2).EQ.1) GOTO 2
      K1=K1+1
      K2 = - MAX0 ( - JJKM2 - JJNM2, - JJLM2 - JJMM2)/2 + 1
      IF(K1.GT.K2) GOTO 2
      DO 62 K=K1,K2,2
      SK1=K-1
   62 AS2=AS2+  CFU(AY1,BY2,SK1,0.5,-0.5,0.0)*  CFU(BY1,AY2,SK1,0.5,-0.5
     1,0.0)*  WFU(AY1,BY1,BY2,AY2,W,SK1)*
     2SLAT(K - 1,NODEIK + 1,LBIK, NODEIL + 1, LBIL,NODEIN + 1,
     3LBIN,NODEIM + 1, LBIM) / (2.0 * SK1 + 1.0)
    2 P2=-1.
      IF(MOD( (JJLM2 + JJNM2)/2 +ISOSP,2).EQ.1) P2 = 1.0
      P3=0.5
      P4=0.5
CCCCCCC  ZUR RICHTIGEN NORMIERUNG KOMMENTAR-C ENTFERNEN
C     IF(( JJKM2.EQ.JJLM2).AND.( NODEIK.EQ.NODEIL ).AND.( LBIK.EQ.LBIL))
C    1P3 = 0.25
C     IF    (( JJMM2.EQ.JJNM2 ).AND.(NODEIM.EQ.NODEIN ).AND.(LBIM.EQ.LBI       #
```

```
C    1N)) P4 = 0.25                                                    COUL1590
CCCCCCC
      ATRIX=2.*SQRT(P3*P4*(2.*AY1+1.)*(2.*BY1+1.)*(2.*AY2+1.)*(2.*BY2+1.
     1))*(AS1*P1+AS2*P2)
      CMAGJJ = ATRIX * 7.29719E-3 * 0.210309 * 938.211
      IERR = 0
      RETURN
C
   63 CMAGJJ = 0.0
      IERR = 1
      RETURN
      END
```

```
      FUNCTION WFU (X1,X2,X3,X4,X5,X6)
C  ***********************************************************************
C  PROGRAMM BERECHNET DEN RACAH KOEFFIZIENTEN WFU(X1,X2,X3,X4,X5,X6), DER ALS
C  QUANTUM MECHANICS, P. 97, MIT 6-J-SYMBOL VERKNUEPFT IST. PROGRAMM TESTET
C  W(J1,J2,L2,L1.,J3,L3) IN GLEICHUNG (6.2.13) VON EDMONDS, ANGULAR MOMENTUM IN
C  AUSWAHLREGELN UND BENOETIGT DIE UP STIRL,IDR UND ONEP.
C  ***********************************************************************
      WFU = 0.0
      Y1=X1
      Y2=X2
      Y3=X3
      Y4=X4
      Y5=X5
      Y6=X6
      IF ( IDR(Y1,Y2,Y5) + IDR(Y3,Y4,Y5) + IDR(Y1,Y3,Y6) + IDR(Y4,Y2,Y6)
     1 )  25, 26, 25
   25 GO TO 23
   26 AA = Y1 + Y2 + Y3 + Y4
      B=Y1+Y4+Y5+Y6
      E=Y2+Y3+Y5+Y6
      E1=AA
      IF(AA-B-0.1)  80,80,81
   81 E1=B
   80 IF(E1-E-0.1)  82,82,83
   83 E1=E
   82 AA=Y1+Y2+Y5
      B=Y3+Y4+Y5
      E=Y1+Y3+Y6
      D=Y4+Y2+Y6
      E2=AA
      IF(AA-B+0.1)  84,85,85
   84 E2=B
   85 IF(E2-E+0.1)  86,87,87
   86 E2=E
   87 IF(E2-D+0.1)  88,55,55
   88 E2=D
   55 WFU=0
      AA=0.5*(STIRL(X1+X2-X5 )+STIRL( X1+X5-X2)+STIRL(X2+X5-X1) -STIRL(X
     11+X2+X5+1.0)+STIRL(X3+X4-X5 )+STIRL(X3+X5-X4 )+STIRL(X4+ X5-X3)-ST
```

```
     2IRL(X3+X4+X5+1.0)+STIRL(X1+X3-X6 )+STIRL(X1+X6-X3)+STIRL (X3+X6-X1
     3)-STIRL(X1+X3+X6+1.0)+STIRL(X2+X4-X6 )+STIRL(X2+X6-X4 ) +STIRL(X4+
     4X6-X2)-STIRL(X2+X4+X6+1.0))
  302 E=E2
      Y=ONEP (E+X1+X2+X3+X4)*EXP(AA+STIRL(E+1.0)-(STIRL(E-X1-X2-X5)+ STI
     1RL(E-X3-X4-X5) +STIRL(E-X1-X3-X6) +STIRL(E-X4-X2-X6))- (STIRL(X1+X
     22+X3+X4-E)+STIRL(X1+X4+X5+X6-E)+STIRL(X2+X3+X5+X6-E)))
      WFU=WFU+Y
      E2=E2+1.0
      IF(E2-E1-0.1)  302,302,23
   23 RETURN
      END
```

```
      SUBROUTINE FAFITA(KRAFT,IPOT,MFACH,POTPAR,POTSTA,OSLA)
C
C  ********************************************************************
C  UP SPEICHERT DIE BENOETIGTEN FAKULTAETEN, DOPPELFAKULTAETEN UND DIE
C  HALBZAHLIGE GAMMAFUNKTION MIT EINFACHER UND DOPPELTER GENAUIGKEIT.
C  UP BERECHNET UND SPEICHERT DIE INTEGRALE JM, DEFINIERT BEI
C  HORIE SASAKI PROG.THEOR.PHYS. 25,487 (1961).
C  DER POTENTIALPARAMETER POTPAR SPEZIFIZIERT  DIE BENUTZTE KRAFT VOLL-
C  STAENDIG. OSLA GIBT DIE OSZILLATORLAENGE AN, FUER DIE DIE MATRIX-
C  ELEMENTE BERECHNET WERDEN SOLLEN.
C  *******
C  OPTIONS
C  *******
C  OPTION KRAFT WAEHLT KRAFTGESETZ FUER NUKLEON-NUKLEON-WECHSELWIRKUNG
C  AUS.
C  KRAFT = 1   GAUSSPOTENTIAL
C  KRAFT = 2   YUKAWAPOTENTIAL
C  KRAFT = 3   DELTAPOTENTIAL
C  KRAFT = 4   POTENTIAL WIRD DEFINIERT DURCH DIE INTEGRALE JM, DIE
C              NACH TJ(M+1) EINGELESEN WERDEN
C  KRAFT = 5   KONSTANTES POTENTIAL ENDLICHER REICHWEITE
C  KRAFT = 6   COULOMBPOTENTIAL                                          #
C
C  MFACH = 0   DIE INTEGRALE JM WERDEN AUFGEBAUT MIT DEN VORGEGEBENEN
C              POTENTIALPARAMETERN
C  MFACH = 1   DIE DURCH DIE EINGEGEBENEN POTENTIALPARAMETER BESTIMMTEN
C              GROESSEN XJ(J) WERDEN AUF DIE BEREITS BERECHNETEN TJ(J)
C              AUFADDIERT. DIE HILFSGROESSEN FUER DIE SLATERINTEGRALE
C              WERDEN NICHT AUFGEBAUT
C  MFACH = 9   DIE DURCH DIE EINGEGEBENEN POTENTIALPARAMETER BESTIMMTEN
C              GROESSEN XJ(J) WERDEN AUF DIE BEREITS BERECHNETEN TJ(J)
C              AUFADDIERT. DIE HILFSGROESSEN FUER DIE SLATERINTEGRALE
C              WERDEN AUFGEBAUT
C
C  IPOT  = 1   ZENTRALKRAFT, DEFINIERT NACH OPTION KRAFT
C  IPOT  = 3   TENSORKRAFT, DEFINIERT NACH OPTION KRAFT
C
C  PROGRAMM FAFITA BENOETIGT DIE  UP HISLAT, TENSOR UND VORZ.
C
      COMMON /FAKFID/ FAK(40),FID(40)
      COMMON /TIN/ TJ(20)
      DIMENSION XJ(20)
      DOUBLEPRECISION A1,A2,PD,GD(20),TJD(20),TA(20),DFAK(40),DFID(40),
     1DF
      K1K = IPOT
```

```
C
C  ***************************************************************
C  IM 1.RECHENSCHRITT WERDEN DIE BENOETIGTEN FAKULTAETEN,DOPPELFAKULTAE-
C  TEN UND HALBZAHLIGEN GAMMAFUNKTIONEN BERECHNET.
C  ***************************************************************
C  1111111111111111111111111111111111111111111111111111111111111111
C
C  FAKULTAETSSPEICHERUNG MIT EINFACHER GENAUIGKEIT
      IF(K1K-1) 20,20,21
   20 FAK(1) =1.
      FAK(2)=1.
      DO 2 J=2,39
    2 FAK(J+1)=FLOAT(J)*FAK(J)
      IF(KRAFT.NE.6) GOTO 1010
C
C  FAKULTAETENSPEICHERUNG MIT DOPPELTER GENAUIGKEIT
      DFAK(1) = 1.0
      DFAK(2) = 1.0
      DO 1000 J = 2,39
      DF = FLOAT(J)
      DFAK(J+1) = DF * DFAK(J)
 1000 CONTINUE
C
C  DOPPELFAKULTAETSSPEICHERUNG MIT EINFACHER GENAUIGKEIT
 1010 FID(1)=1.
      FID(2)=1.
      DO 3 J=2,39
    3 FID(J+1)=FID(J-1)*FLOAT(J)
      IF(KRAFT.NE.6) GOTO 1030
C
C  DOPPELFAKULTAETSSPEICHERUNG MIT DOPPELTER GENAUIGKEIT
      DFID(1) = 1.0
      DFID(2) = 1.0
      DO 1020 J= 2,39
      DF = FLOAT(J)
      DFID(J+1) = DF * DFID(J-1)
 1020 CONTINUE
C
C  HALBZAHLIGE GAMMAFUNKTIONEN GD(N) = GAMMA(N+1/2)
 1030 GD(1)=0.886226925452755
      DO 1 J=1,19
      DJ=J
    1 GD(J+1)=GD(J)*(DJ+0.5)
   21 GOTO (4,5,6,7,17,100),KRAFT
C
C  1111111111111111111111111111111111111111111111111111111111111111
C  ***************************************************************
C  IM ZWEITEN RECHENSCHRITT WERDEN DIE INTEGRALE JM JE NACH OPTION
C  KRAFT BERECHNET.
C  ***************************************************************
C  2222222222222222222222222222222222222222222222222222222222222222
C
C  GAUSSPOTENTIAL V(R) = EXP(-R**2/R0**2)
C  HORIE SASAKI PROG.THEOR.PHYS. 25,FORMEL 52 (1961)
C  EINZULESENDER POTENTIALPARAMETER IST POTPAR = R0.
C  DAS PROGRAMM KANN POTENTIALE DER FORM  POTSTA * V(R) AUFADDIEREN.
C
    4 P = POTPAR/(OSLA*SQRT(2.0))
      IF(MFACH.GT.1) GOTO 31
      DO 30 J = 1,20
   30 TJ(J) = 0.0
   31 DO 8 J=1,20
    8 XJ(J)=FID(2*J)/2.**(J-1)*P**3/(1.+P*P)**(FLOAT(J)+0.5)
      DO 32 J=1,20
   32 TJ(J) = TJ(J) + XJ(J) * POTSTA
      IF((MFACH.EQ.0).OR.(MFACH.EQ.9)) CALL HISLAT
```

```
      IF((K1K.EQ.3).OR.(K1K.EQ.6)) CALL TENSOR
      RETURN
C
C  YUKAWAPOTENTIAL V(R) = EXP(-R/R0)/(R/R0)
C  HORIE SASAKI PROG.THEOR.PHYS. 25,FORMEL 54 (1961)
C  EINZULESEENDER POTENTIALPARAMETER IST
C
C                 POTPAR = R0.
C
C  FUER DAS YUKAWAPOTENTIAL WERDEN AUSSERDEM DIE PARAMETER
C
C                 A1     = ERF(P)
C                 A2     = (D/DP)(ERF(P))
C                        = (2.0/SQRT(PI)) * EXP(-P**2)
C                 P      = OSLA/(POTPAR*SQRT(2.0))
C
C  BENOETIGT. ERF(X) IST DAS FEHLERINTEGRAL. GENAUE WERTE VON A1 UND A2
C  FINDET MAN IN  'TABLES OF THE ERRORFUNCTION' ,NATIONAL BUREAU OF
C  STANDARDS, APPLIED MATHEMATICAL SERIES, NO. 41.
C  DAS PROGRAMM KANN POTENTIALE DER FORM  POTSTA * V(R) AUFADDIEREN.
C
    5 READ(5,9) A1,A2
    9 FORMAT(2D21.15)
      P = OSLA/(POTPAR*SQRT(2.0))
      PD=P
      IF(MFACH.NE.0) GOTO 41
      DO 40 J = 1,20
   40 TJ(J) = 0.0
   41 TJD(1)=0.564189583547756/PD-(1.-A1)*1.1283791670954/A2
      XJ(1)=TJD(1)
      DO 10 J=2,20                                                          #
      TJD(J)=0.31830988618379*GD(J-1)/PD-PD*PD*TJD(J-1)
   10 XJ(J)=TJD(J)
      DO 42 J=1,20
   42 TJ(J) = TJ(J) + XJ(J) * POTSTA
      IF((MFACH.EQ.0).OR.(MFACH.EQ.9)) CALL HISLAT
      IF((K1K.EQ.3).OR.(K1K.EQ.6)) CALL TENSOR
      RETURN
C
C  DELTAPOTENTIAL V(R) = V0 * DELTA(R)/R**2 , WOBEI V0
C  OSZILLATORLAENGE**3/SQRT(0.5*PI)  IST. DIESER FAKTOR GARANTIERT
C  EIN DIMENSIONSLOSES ERGEBNIS FUER DIE INTEGRALE JM.
C  HORIE SASAKI PROG.THEOR.PHYS. 25,489 (1961)
C
    6 DO 11 J=1,20
   11 TJ(J)=1.1283791670954*GD(J)
      CALL HISLAT
      IF((K1K.EQ.3).OR.(K1K.EQ.6)) CALL TENSOR
      RETURN
C
C EINLESEMOEGLICHKEIT FUER DIE INTEGRALE JM
C
    7 READ(5,12)(TJ(J),J=1,20)
   12 FORMAT(5E14.7)
      CALL HISLAT
      IF((K1K.EQ.3).OR.(K1K.EQ.6)) CALL TENSOR
      RETURN
C
C  SOFT CORE POTENTIAL , DAS DURCH EIN KONSTANTES POTENTIAL ENDLICHER
C  REICHWEITE SIMULIERT WIRD. V(R) = 1.0 (0.0), WENN R.LE.R0 (R.GT.R0).
C  EINZULESENDER POTENTIALPARAMETER POTPAR IST R0. ZUR
C  BERECHNUNG DER TALMIINTEGRALE WIRD DIE  CHI-SQARE PRO-
C  BABILITY FUNCTION NACH  DEN GLN (6.5.5) UND (26.4.7) DES 'HANDBOOK
C  OF MATHEM. FUNCTIONS WITH FORMULAS, GRAPHS AND MATHEM. TABLES' ,
C  NATIONAL BUREAU OF STANDARDS, APPLIED MATHEM. SERIES, NO. 55 (1964)
C  BENUTZT.  DA DIE REIHENENTWICKLUNG DER PROBABILITY FUNCTION SCHNELL
```

```
C  KONVERGIEREN SOLL, MUSS DAS VERHAELTNIS R0/OSCILLATORLAENGE KLEIN
C  SEIN.
C
   17 P = POTPAR/(OSLA*SQRT(2.0))
      PD = P*P
      DO 13 L=1,20
      TA(L)=0.0
      DO 13 J=1,10
      P1=-1.
      IF(MOD(J,2).EQ.1) P1=1.
   13 TA(L)=TA(L)+P1*(PD**(J+L)/P)/(FAK(J)*(FLOAT(J+L)-0.5))
      DO 14 J=1,20
      TJD(J)=0.0
      DO 15 K=1,J
      P1=-1.
      IF(MOD(K,2).EQ.1) P1=1.
      M=J-K+1
   15 TJD(J)=TJD(J)+P1*TA(K)*FAK(J)/(FAK(K)*FAK(M)*GD(K))
   14 TJ(J)=TJD(J)*GD(J)*0.56418958*2.
      CALL HISLAT
      IF((K1K.EQ.3).OR.(K1K.EQ.6)) CALL TENSOR
      RETURN
C
C  COULOMBPOTENTIAL V(R) = OSZILLATORLAENGE/R
C  IN TJ(M) STEHEN DIE INTEGRALE J(0/M-1) VON H-S GLN(46). DIE INTEGRALE
C  I(L) WERDEN NACH H-S GLN(44)' BERECHNET.
C
C  I(L) = FAK(L+1)/FID(2L+2) * 1.0/(SQRT(2.0*PI)*0.5**(L+1))
C
C  J(0/M) = 1.0/SQRT(2.0*PI) * FID(2*M+2)/2.0**M * SUMME( FAK(M+1)/
C           (FAK(M-L+1)*FID(2*L+2)) * (-1)**L/0.5**(L+1))
C
  100 DO 300 J = 1,17
      TJD(J) = 0.0
      DO 200 L = 1,J
      JJJ = J - L +1
     0TJD(J)=TJD(J)+ VORZ(L-1)*(DFID(2*J)/DFID(2*L))*(DFAK(J)/DFAK(JJJ))
     1 *(2.0**(L-J+1))
  200 CONTINUE
      TJD(J) = TJD(J) * DSQRT(2.D0) * 0.564189583547756 /2.0
      TJ(J) = TJD(J)
  300 CONTINUE
      CALL HISLAT
      RETURN
C  22222222222222222222222222222222222222222222222222222222222222222222
      END
C     TENSOR-DUMMY VERSION WOLTER JULY 1969
```

```
      SUBROUTINE TENSOR
C  ********************************************************************
C  DUMMY PROGRAMM FUER PROGRAMM GLEICHER KENNUNG. PROGRAMM HAELT
C  AUFRUFMOEGLICHKEIT VON TENSORWECHSELWIRKUNGEN OFFEN.
C  ********************************************************************
      RETURN
```

```
      END
C     SLAT         2.9.1969
      FUNCTION SLAT(K,N11,L11,N12,L12,N21,L21,N22,L22)
C
C ********************************************************************
C FUNCTION SLAT BERECHNET DAS INTEGRAL
C
C        R(K1,K2,KAPPA/N11-1,L11,N12-1,L12/N21-1,L21,N22-1,L22)
C
C NACH H-S GLN(32) FUER KAPPA=0, DH. FUER K1=K2=K, UND MULTIPLIZIERT
C ES MIT DEM FAKTOR (2*K+1).
C SLATERINTEGRALE IN TERMEN HARMONISCHER OSZILLATORWELLENFUNKTIONEN
C HORIE SASAKI 25,481 (24-32), (1961)
C DIE PHASEN DER OSZILLATORFUNKTIONEN SIND GEGENUEBER HORIE SASAKI
C AUF DIE NILSSONNORMIERUNG GEAENDERT.
C DAS PROGRAMM BENOETIGT DIE UP HISLAT,ELKE1 UND VORZ UND SETZT UP
C FAFITA VORAUS.
C ********************************************************************
C
      COMMON /FAKFID/ FAK(40),FID(40)
      COMMON /FK/F(14,14,14)
      S1=0.0
      S=0.0
      M11=L11+L21+1
      M21=M11+2*(N11+N21)-4
      M12=L12+L22+1
      M22=M12+2*(N12+N22)-4
      IF ((K+1) .GT. 14 .OR. M21 .GT. 14 .OR. M22 .GT. 14)
     * WRITE (6,100) K,M21,M22
  100 FORMAT (/18H0*** FK ZU KLEIN   ,3I10/)
      DO 1 M1=M11,M21,2
      S=0.0                                                              #
      DO 2 M2=M12,M22,2
    2 S=S+ELKE1(M2-1,N12-1,L12,N22-1,L22)*F(K+1,M1,M2)
    1 S1=S*ELKE1(M1-1,N11-1,L11,N21-1,L21)+S1
      K1=2*(L11+N11)
      K2=2*(L12+N12)
      K3=2*(L21+N21)
      K4=2*(L22+N22)
      AK=2*K+1
      SLAT=S1/SQRT(FAK(N11)*FAK(N12)*FID(K1)*FID(K2)*FAK(N21)*FAK(N22)*
     1FID(K3)*FID(K4)*2.**(N11+N12+N21+N22-4))*AK*VORZ(N11+N12+N21+N22)
      RETURN
      END
C     ELKE1        2.9.1969
      FUNCTION ELKE1(M,N1,L1,N2,L2)
C
C ********************************************************************
C UP ELKE1(7,N1,L1,N2,L2) BERECHNET DIE GROESSEN A(M/N1,L1/N2,L2) MIT
C M = L1 + L2 + 2*S VON H-S GLN(36)*.
C ENTWICKLUNGSKOEFFIZIENTEN FUER DAS PRODUKT LAGUERRESCHER POLYNOME
C ARGUMENTE INTEGER
C HORIE SASAKI PROG.THEOR.PHYS. 25,483,(1961.)
C UP SETZT UP FAFITA MIT UP'S VORAUS.
C ********************************************************************
C
      COMMON /FAKFID/ FAK(40),FID(40)
      J=(M-L1-L2)/2
      KA1=MIN0(J,N1)+1
      KA2=MAX0(0,J-N2)+1
      E=1.
      IF(MOD(J,2).EQ.1) E=-1.
      J=J+1
      J1=2*(L1+N1+1)
      J2=2*L1
      J3=2*(L2+N2+1)
```

```
      J4=2*(L2+J+1)
      J5=N1+2
      J6=N2-J+1
      J7=N1+1
      J8=N2+1
      J9=J+1
      A=FAK(J7)*FAK(J8)
      B=FID(J1)*FID(J3)
      ELKE1=0.0
      DO 1 K=KA2,KA1
      K2=J2+2*K
      K4=J4-2*K
      K5=J5-K
      K6=J6+K
      K9=J9-K
    1 ELKE1=ELKE1+A/(FAK(K)*FAK(K5)*FAK(K9)*FAK(K6))*B/(FID(K2)*FID(K4))
      ELKE1=E*ELKE1
      RETURN
      END
```

```
C     HISLAT       2.9.1969
      SUBROUTINE HISLAT
C
C ***************************************************************
C ZWISCHENERGEBNISSE ZU DEN SLATERINTEGRALEN  FUER SKALARE WECHSEL-
C WIRKUNG.
C DIE GROESSEN F(K1,K2,KAPPA/M1,M2) VON H-S GLN(33) WERDEN FUER
C KAPPA = 0,D.H. K1 = K2 = K IN F(K+1,M1+1,M2+1) GESPEICHERT.
C HORIE SASAKI PROG.THEOR.PHYS. 25,483,(33),(1961)
C PROGRAMM BENOETIGT UP ELKE1 UND SETZT UP FAFITA MIT UP'S VORAUS.
C ***************************************************************
C
      COMMON/TIN/TJ(20)/FK/F(14,14,14)
      DO 1 M1=1,14,2
      DO 1 M2=1,M1,2
      DO 1 K=1,M2,2
      LR1=(M1-K)/2
      LR2=(M2-K)/2
      LK=K-1
      K1=(M1+M2)/2
      F1=0.0
      DO 2 M=K,K1
      LM=2*M-2
```

```
    2 F1=F1+ELKE1(LM,LR1,LK,LR2,LK)*TJ(M)
      F(K,M1,M2)=F1
    1 F(K,M2,M1)=F(K,M1,M2)
      DO 3 M1=2,14,2
      DO 3 M2=2,M1,2
      DO 3 K=2,M2,2
      LR1=(M1-K)/2
      LR2=(M2-K)/2
      LK=K-1
      K1=(M1+M2)/2
      F1=0.0
      DO 4 M=K,K1
      LM = 2*M - 2
    4 F1 = F1 + ELKE1(LM,LR1,LK,LR2,LK)*TJ(M)
      F(K,M1,M2) = F1
    3 F(K,M2,M1) = F(K,M1,M2)
      RETURN
      END
C     CFU          2.9.1969
      FUNCTION CFU(Y1,Y2,Y3,Y4,Y5,Y6)
C  ***************************************************************************
C  PROGRAMM BERECHNET CLEBSCH-GORDAN-KOEFFIZIENTEN IN DER CONDON-SHORTLY-
C  PHASENKONVENTION. CFU(Y1,Y2,Y3,Y4,Y5,Y6) IST GLEICH DEM V-C-KOEFFIZIENTEN
C  (Y1,Y4,Y2,Y5/Y1,Y2,Y3,Y6) = (J1,M1,J2,M2/J1,J2,J3,M3) DER GLEICHUNG (3.6.10)
C  AUF P. 44 VON EDMONDS, ANGULAR MOMENTUM IN QUANTUM MECHANICS. PROGRAMM TESTET
C  AUSWAHLREGELN UND BENOETIGT DIE UP STIRL, IDR UND ONEP.
C  ***************************************************************************
      DIMENSION C(10)
      X1=Y3
      X2=Y1
      X3=Y2
      X4=Y6
      X5=Y4
      X6=Y5
      CFU=0
      IF (IDR(X1,X2,X3))14,15,14
   15 IF (ABS(X5+X6-X4)-0.1)17,14,14
   17 IF (X1-ABS(X4))14,19,19
   19 IF (X2-ABS(X5))14,20,20
   20 IF (X3-ABS(X6))14,16,16
   16 IF (X4)600,13,600
   13 IF (X5) 600,18,600
   18 L = IFIX( X1 + X2 + X3 + 0.1)
      IF (2*(L/2)-L)60,114,60
  114 X=L
      Y=X/2.0
      CFU=ONEP(Y+X2-X3)*SQRT(2.0*X1+1.0)*EXP(0.5*(STIRL(X-2.0*X1)+STIRL
     1(X-2.0*X2)+STIRL(X-2.0*X3)-STIRL(X+1.0))+STIRL(Y)-STIRL(Y-X1)-STI
     2RL(Y-X2)-STIRL(Y-X3))
      GO TO 51
  600 C(1)=X2+X3-X1
      C(2)=X2-X5
      C(3)=X3+X6
      C(4)=-(X1-X3+X5)
      C(5)=-(X1-X2-X6)
      DO 41 J=1,2
      DO 41 K=2,3
      IF (C(J)-C(K))42,41,41
   42 TEMP1=C(J)
      C(J)=C(K)
      C(K)=TEMP1
   41 CONTINUE
      K1=C(3)
      IF (C(4)-C(5))46,46,47
   46 K2=C(5)
      GO TO 48
```

```
   47 K2=C(4)
   48 IF (K2) 44,45,45
   44 K2=0.0
   45 IF (K1-K2)60,300,300
  300 Z=0
      BB=SQRT(2.0*X1+1.0)
      AA=0.5*(STIRL(X2+X3-X1)+STIRL(X1+X2-X3)+STIRL(X1+X3-X2)
     1-STIRL(X1+X2+X3+1.0)+STIRL(X2+X5)+STIRL(X2-X5)+STIRL(X3+X6)
     2+STIRL(X1+X4)+STIRL(X1-X4)+STIRL(X3-X6))
  301 E=K2
      X=BB*ONEP(E)*EXP(AA-STIRL(E)-STIRL(X2+X3-X1-E)-STIRL(X2-X5-E)
     1-STIRL(X3+X6-E)-STIRL(X1-X3+X5+E)-STIRL(X1-X2-X6+E))
   49 Z=Z+X
      K2=K2+1
      IF (K2-K1)301,301,50
   50 CFU=Z
      GO TO 51
   60 CFU=0.0
   51 CONTINUE
   14 RETURN
      END
C     VORZ         2.9.1969
       FUNCTION VORZ(J)
C  UP BERECHNET (-I)**J ALS REAL-GROESSE
      JD2= J/ 2
      IF(J - JD2 - JD2) 10,20,10
   10 VORZ = -1.0
      RETURN
   20 VORZ = 1.0
      RETURN
      END
C     ONEP         2.9.1969
      FUNCTION ONEP(X)
C  ********************************************************************************
C  UP BERECHNET PHASE (-1)**X.
C  ********************************************************************************
      L =  ABS(X) + 0.1
  101 IF(L-2*(L/2)) 3,4,3
    3 ONEP=-1.0
      RETURN
    4 ONEP=1.0
    5 RETURN
      END
C     IDR          2.9.1969
      FUNCTION  IDR(X,Y,Z)
C  ********************************************************************
C  UP PRUEFT, OB DREIECKSRELATION FUER DIE DREHIMPULSE X,Y,Z ERFUELLT
C  IST.BEIM TEST WIRD X ALS GESAMTDREHIMPULS BETRACHTET. IST DREIECKS-
C  UNGLEICHUNG ERFUELLT (IDR = 0, SONST IDR = 1), DANN IST SIE AUCH FUER
C  JEDE PERMUTATION DER X,Y,Z ERFUELLT.
C  ********************************************************************
      IDR = 0
      IF ( ABS(Y-Z) -X -.1) 1, 1, 2
    1 IF    (Y+Z-X+ 0.1 )  2, 3, 3
    2 IDR = 1
    3 RETURN
      END
C     STIRL        2.9.1969                                                #
      FUNCTION STIRL(X)
  101 IF(X-2.0)3,3,4
    3 IF(X-1.0)5,5,6
    5 STIRL=0.0
      RETURN
    6 STIRL=0.69314718
      RETURN
    4 Y=X+1.0
```

```
      STIRL=(Y-0.5)*ALOG(Y)-Y+0.91893853+(.83333333E-01-(.27777778E-02-.
     179365079E-03*(1.0/(Y*Y)))*(1.0/(Y*Y)))/Y
    8 RETURN
      END
```

```
      subroutine n3lo
c
c*****************************************************************
c
c       FEBRUARY 2003
c
c*****************************************************************
c
c       This code computes the
c
c       Charge-Dependent Chiral NN Potential at Order Four (N3LO)
c       ---------------------------------------------------------
c       in momentum space.
c
c       this package is self-contained and includes
c       all subroutines needed.
c       only `n3lo' needs to be called by the user.
c       all codes are consistently in double precision.
c       when working on an UNIX/LINUX system, it is recommended
c       to compile this code with the  -static  option.
c       more information on the code is given below.
c
c*****************************************************************
c
c       authors:     D. R. Entem and R. Machleidt
c                    department of physics
c                    university of idaho
c                    moscow, idaho 83844
c                    u. s. a.
```

```
c                     e-mails: dentem@uidaho.edu
c                              machleid@uidaho.edu
c
c        preprint:    nucl-th/0304018
c
c*********************************************************************
c
c
c
c
      implicit real*8 (a-h,o-z)
c
c
      common /crdwrt/ kread,kwrite,kpunch,kda(9)
c
c        arguments and values of this subroutine
c
      common /cpot/   v(6),xmev,ymev
      common /cstate/ j,heform,sing,trip,coup,endep,label
      common /cnn/ inn
c
c
c        this has been the end of the common-blocks containing
c        the arguments and values of this subroutine.
c
c        specifications for these common blocks
c
      logical heform,sing,trip,coup,endep
      character*4 label
c
c
c*********************************************************************
c        THE ABOVE FOUR COMMON BLOCKS IS ALL THE USER NEEDS
c        TO BE FAMILIAR WITH.
c*********************************************************************
c
c        here are now some explanations of what those common blocks contain:
c        ---------------------------------------------------------------
c
c        xmev and ymev are the final and initial relative momenta,
c        respectively, in units of mev/c.
c        v is the potential in units of mev**(-2).
c        concerning units, factors of pi, etc.,
c        cf. with the partial-wave Lippmann-Schwinger equation, Eq. (A25),
c        and with the phase shift relation, Eq. (A33), given in Appendix A
c        of the article: R. Machleidt, PRC 63, 024001 (2001).
c
c        the partial-wave Lippmann-Schwinger equation for the
c        K-matrix reads:
c
c        K(q',q) = V(q',q) + M P \int dk k^2 V(q',k) K(k,q)/(q^2-k^2)
c
c        with M the nucleon mass in MeV and P denoting the principal value;
c        V(q',q) as provided by this code in common block /cpot/;
c        all momenta in MeV.
c
c        the phase-shift relation is:
c
c        tan \delta_L = -(pi/2) M q K_L(q,q)
c
c        with M and q in units of MeV, K_L in MeV**(-2) like V.
c
c
c        if heform=.true., v contains the 6 matrix elements
c        associated with one j in the helicity formalism
c        in the following order:
```

```
c        0v, 1v, 12v, 34v, 55v, 66v
c        (for notation see Appendix A of above article).
c
c        if heform=.false., v contains the 6 matrix elements
c        associated with one j in the lsj formalism
c        in the following order:
c        0v(singlet), 1v(uncoupled triplet), v++, v--, v+-, v-+ (coupled)
c        (for notation, see explanations given in the above article
c        below Eq. (A31)).
c
c        j is the total angular momentum. there is essentially no upper
c        limit for j.
c        sing, trip, and coup should in general be .true..
c        endep and label can be ignored.
c        it is customary, to set kread=5 and kwrite=6;
c        ignore kpunch and kda(9).
c
c        the meaning of the parameter inn in the common block
c
c                  common /cnn/ inn
c        is
c                  inn=1  means pp potential,
c                  inn=2  means np potential, and
c                  inn=3  means nn potential.
c
c        the user needs to include this common block in his/her code,
c        and specify which potential he/she wants to use.
c
c
c        THIS IS ESSENTIALLY ALL THE USER NEEDS TO KNOW.
c
c        if you have further questions, do not hesitate to contact one
c        of the authors (see e-mail addresses above).
c
c*======================================================================*
c
c
c        common block for all chi-subroutines
c
      common /cchi/ vj(32,270),c(20,270),fff,ff,f(52),aa(96),ai(19,30),
     1               wnn(3),wdd(3),x,xx,y,yy,xy2,xxpyy,ex,ey,eem12,
     2               gaa(3),fpia(3),ezz1(3),ezz2(3),ct(96),wt(96),
     3               ic(20,270),ift(3),mint(3),maxt(3),nt,
     4               mge,mgg(40,3),mggo(40,3),ima(30,40,3),
     5               imaa(3),imea(3),ime,im,mc,m,mg,inter,ide,idde,
     6               indc(2,270),indpar(3),indxy
c
c         specifications for this common block
c
      logical indc,indxy,indpar
c
      common /comlsj/ clsj(15,50),cutlsj(15,50),indlsj
      logical indlsj
c
      common /crrr/ rrr
c
c
c        further specifications
c
      dimension vl(4),adminv(4,4),ldminv(4),mdminv(4)
      dimension vv0(6),vv2(6),vv4(6)
      character*4 nucnuc(3)
      character*4 mesong(40)
      logical index
      logical indmg(40)
      data mesong/'0-  ','0-t ','0-st','0+  ','0+st',
```

```
     1            '1-  ','1-t ','1-tt','1-st','1-ss',
     2            'c   ','ss  ','ls  ','sq  ','sk  ',
     3            'sl  ',
     4         24*'    '/
      data index/.false./
      data indmg/40*.false./
      data jj/-1/
      data pi/3.141592653589793d0/
      data innn/-1/
      data nucnuc/'N3pp','N3np','N3nn'/
      save
c
c
c
c
      if (index) go to 10
      index=.true.
c
c
c        call subroutine chipar_n3lo once and only once
c
c
      call chipar_n3lo
c     -----------
c
c
c        if you want the potential to be zero for very large momenta,
c        choose rrr=1000.
c        if you want no technical problems in the calculation of the deuteron
c        wave functions, choose rrr=80.
c
      rrr=80.
c
   10 continue
c
c
c
c
      if (inn.lt.1.or.inn.gt.3) then
c        choose the np potential as the default:
      inn=2
      endif
      if (j.lt.0) then
      write (kwrite,19002)
19002 format (////' error in n3lo: total angular momentum j',
     1' is negative.'/' execution terminated.'////)
      stop
      endif
c
c
c
c
c        set the inn dependent parameters
c
      if (inn.eq.innn) go to 30
      innn=inn
      inter=inn
      label=nucnuc(inter)
c
      go to (21,22,23), inter
   21 write (kwrite,10001)
10001 format (' The pp potential is used.')
      go to 24
   22 write (kwrite,10002)
10002 format (' The np potential is used.')
      go to 24
```

```
   23 write (kwrite,10003)
10003 format (' The nn potential is used.')
   24 write (kwrite,10004)
10004 format (' -------------------------'/)
c
c
      iftgo=ift(inter)+1
      dwn=1.d0/wnn(inter)
c
c
c        prepare constant over-all factor
c
      fac=pi/(2.d0*pi)**3*dwn*dwn
c     ---------------------------
c
c
c
      iman=imaa(inter)
      imen=imea(inter)
c
      imanm1=iman-1
c
      iman1=imanm1+1
      iman2=imanm1+2
      iman3=imanm1+3
      iman4=imanm1+4
      iman5=imanm1+5
      iman6=imanm1+6
      iman7=imanm1+7
      iman8=imanm1+8
      iman9=imanm1+9
      imen24=imen-24
      imen23=imen-23
      imen22=imen-22
      imen21=imen-21
      imen15=imen-15
      imen14=imen-14
c
c
      ez1=ezz1(inter)
      ez2=ezz2(inter)
c
c
c
   30 if (j.eq.jj) go to 50
      jj=j
      if (j.eq.0) go to 50
      aj=dble(j)
      aj1=dble(j+1)
      a2j1=dble(2*j+1)
      aaj6=dsqrt(aj*aj1)
c
c        coefficient matrix for the translations into lsj formalism
c
      adminv(1,1)=aj1
      adminv(1,2)=aj
      adminv(1,3)=-aaj6
      adminv(1,4)=-aaj6
      adminv(2,1)=aj
      adminv(2,2)=aj1
      adminv(2,3)=aaj6
      adminv(2,4)=aaj6
      adminv(3,1)=aaj6
      adminv(3,2)=-aaj6
      adminv(3,3)=aj1
      adminv(3,4)=-aj
```

```
      adminv(4,1)=aaj6
      adminv(4,2)=-aaj6
      adminv(4,3)=-aj
      adminv(4,4)=aj1
c
c       inversion
c
      call dminv (adminv,4,deter,ldminv,mdminv)
c
c
c
c
c        prepare expressions depending on x and y
c        ----------------------------------------
c        ----------------------------------------
c
c
c
c
   50 xa=xmev*dwn
      ya=ymev*dwn
      indxy=.false.
      x=xa
      xx=x*x
      y=ya
      yy=y*y
      xy2=x*y*2.d0
      xxpyy=xx+yy
      ex=dsqrt(1.d0+xx)
      ey=dsqrt(1.d0+yy)
      eem12=(ex*ey-1.d0)*2.d0
c
c
      xy=xy2*0.5d0
      ee=ex*ey
      ree=dsqrt(ee)
      eem1=ee-1.d0
      eme=ex-ey
      emeh=eme*0.5d0
      emehq=emeh*emeh
      eep1=ee+1.d0
       epe=ex+ey
      xxyy=xx*yy
c
c
      xxpyyh=xxpyy*0.5d0
      xy3=xy*3.d0
      xy4=xy*4.d0
c
c
c
c
      do 63 iv=1,6
      vv0(iv)=0.d0
      vv2(iv)=0.d0
      vv4(iv)=0.d0
   63 v(iv)=0.d0
      do 65 il=iman,imen
      do 65 iv=1,32
   65 vj(iv,il)=0.d0
c
c
c
c
c        prepare over-all factor
c
```

```
c
      go to (70,71,72,71,72,75,76),iftgo
c
c        no additional factor
c
   70 fff=fac
      go to 80
c
c        minimal relativity
c
   71 fff=fac/ree
      go to 80
c
c        factor m/e*m/e
c
   72 fff=fac/ee
      go to 80
c
c        sharp cutoff
c
   75 if (xmev.gt.ez1.or.ymev.gt.ez1) then
      return
      else
      fff=fac
      end if
      go to 80
c
c        exponential cutoff
c
   76 expo=(xmev/ez1)**(2.d0*ez2)+(ymev/ez1)**(2.d0*ez2)
      if (expo.gt.rrr) then
      expo=rrr
      end if
      fff=fac*dexp(-expo)
c
c
   80 continue
c
c
c
c
c        contributions
c        -------------
c        -------------
c
c
c
c
      do 5995 img=1,mge
      mg=mggo(img,inter)
      if (mg.gt.16) go to 9000
      if (mg.eq.0) go to 8000
      me=mgg(mg,inter)
      go to (9000,9000,9000,9000,9000,9000,9000,9000,9000,9000,
     1       1100,1200,1300,1400,1500,1600),mg
c
c
c
c
c        c   , central force
c        -------------------
c
c
c
c
 1100 mc=1
```

```
c
      ff=1.d0
      f(1)=2.d0
      f(2)=0.d0
      f(3)=f(1)
      f(4)=f(2)
      f(5)=f(2)
      f(6)=f(1)
      f(7)=-f(1)
      f(8)=f(7)
c
      call chistr_n3lo(1,1,me)
      go to 5995
c
c
c
c
c        ss  , spin-spin force
c        ---------------------
c
c
c
c
 1200 mc=1
c
      ff=1.d0
      f(1)=-6.d0
      f(2)=0.d0
      f(3)=2.d0
      f(4)=0.d0
      f(5)=0.d0
      f(6)=f(3)
      f(7)=-f(3)
      f(8)=f(7)
c
      call chistr_n3lo(1,1,me)
      go to 5995
c
c
c
c
c        ls  , spin-orbit force
c        ----------------------
c
c
c
c
 1300 mc=1
c
      ff=1.d0
      f(1)=0.d0
      f(2)=0.d0
      f(3)=0.d0
      f(4)=-xy2
      f(5)=-xy2
      f(6)=0.d0
      f(7)=0.d0
      f(8)=0.d0
      f(9)=0.d0
      f(10)=+xy2
      f(11)=-xy2
c
      call chistr_n3lo(2,1,me)
      go to 5995
c
c
```

```
c
c
c        sq  , sq tensor force (where q denotes the momentum transfer)
c        ---------------------
c
c
c
c
 1400 mc=1
c
      ff=1.d0
      f(1)=-xxpyy*2.0d0
      f(2)=xy*4.d0
      f(3)=-f(1)
      f(4)=-f(2)
      f(5)=f(2)
      f(6)=f(1)
      f(7)=(xx-yy)*2.0d0
      f(8)=-f(7)
c
      call chistr_n3lo(1,1,me)
      go to 5995
c
c
c
c
c        sk  , sk tensor force (where k denotes the average momentum)
c        ---------------------
c
c
c
c
 1500 mc=1
c
      ff=0.25d0
      f(1)=-xxpyy*2.0d0
      f(2)=-xy*4.d0
      f(3)=-f(1)
      f(4)=-f(2)
      f(5)=f(2)
      f(6)=f(1)
      f(7)=(xx-yy)*2.0d0
      f(8)=-f(7)
c
      call chistr_n3lo(1,1,me)
      go to 5995
c
c
c
c
c        sl  , "quadratic spin-orbit force"
c               or sigma-l operator
c        ----------------------------------
c
c
c
c
 1600 mc=1
c
      ff=1.d0
      f(1)=-xxyy*2.d0
      f(2)=0.d0
      f(3)=f(1)
      f(4)=f(2)
      f(5)=f(2)
      f(6)=-f(1)
```

```
      f(7)=f(1)
      f(8)=f(7)
      f(9)=f(6)*2.d0
c
      call chistr_n3lo(4,1,me)
      go to 5995
c
c
c
c
c
c        this has been the end of the contributions of mesons
c        ---------------------------------------------------
c
c
c
c
c        errors and warnings
c        -------------------
c
c
c
c
 9000 if (indmg(mg)) go to 5995
c**** write (kwrite,19000) mesong(mg)
19000 format(1h ////' warning in chinn: contribution ',a4,' does not exi
     1st in this program.'/' contribution ignored. execution continued.'
     2////)
      indmg(mg)=.true.
c
c
c
c
 5995 continue
c
c
c
c
c        add up contributions
c        --------------------
c
c
c
c
 8000 continue
c
c
c        charge-dependent OPE contribution
c        ---------------------------------
c
      if (mod(j,2).eq.1) go to 8020
c
c        j even
c
      v(1)=-vj(1,iman1)+2.d0*vj(1,iman5)
      v(1)=v(1)-vj(1,iman2)+2.d0*vj(1,iman6)
      v(1)=v(1)-vj(1,iman3)+2.d0*vj(1,iman7)
      v(1)=v(1)-vj(1,iman4)+2.d0*vj(1,iman8)
c
      v(2)=-vj(2,iman1)-2.d0*vj(2,iman5)
      v(2)=v(2)-vj(2,iman2)-2.d0*vj(2,iman6)
      v(2)=v(2)-vj(2,iman3)-2.d0*vj(2,iman7)
      v(2)=v(2)-vj(2,iman4)-2.d0*vj(2,iman8)
c
      do 8015 iv=3,6
      v(iv)=-vj(iv,iman1)+2.d0*vj(iv,iman5)
```

```
      v(iv)=v(iv)-vj(iv,iman2)+2.d0*vj(iv,iman6)
      v(iv)=v(iv)-vj(iv,iman3)+2.d0*vj(iv,iman7)
      v(iv)=v(iv)-vj(iv,iman4)+2.d0*vj(iv,iman8)
 8015 continue
      go to 8030
c
c        j odd
c
 8020 continue
      v(1)=-vj(1,iman1)-2.d0*vj(1,iman5)
      v(1)=v(1)-vj(1,iman2)-2.d0*vj(1,iman6)
      v(1)=v(1)-vj(1,iman3)-2.d0*vj(1,iman7)
      v(1)=v(1)-vj(1,iman4)-2.d0*vj(1,iman8)
c
      v(2)=-vj(2,iman1)+2.d0*vj(2,iman5)
      v(2)=v(2)-vj(2,iman2)+2.d0*vj(2,iman6)
      v(2)=v(2)-vj(2,iman3)+2.d0*vj(2,iman7)
      v(2)=v(2)-vj(2,iman4)+2.d0*vj(2,iman8)
c
      do 8025 iv=3,6
      v(iv)=-vj(iv,iman1)-2.d0*vj(iv,iman5)
      v(iv)=v(iv)-vj(iv,iman2)-2.d0*vj(iv,iman6)
      v(iv)=v(iv)-vj(iv,iman3)-2.d0*vj(iv,iman7)
      v(iv)=v(iv)-vj(iv,iman4)-2.d0*vj(iv,iman8)
 8025 continue
c
c
 8030 continue
c
c
      if (iman9.gt.imen) go to 8500
c
c
      if (.not.indlsj) then
      do 8105 il=iman9,imen
      do 8105 iv=1,6
 8105 v(iv)=v(iv)+vj(iv,il)
      else
c
c
c        there are contact terms
c        -----------------------
c
      if (iman9.gt.imen24) go to 8200
c
c        the non-contact terms
c
      do 8155 il=iman9,imen24
      do 8155 iv=1,6
 8155 v(iv)=v(iv)+vj(iv,il)
c
c        contact contributions
c        ---------------------
c
 8200 continue
c
c        Q^0 contacts
      do 8205 il=imen23,imen22
      do 8205 iv=1,6
 8205 vv0(iv)=vv0(iv)+vj(iv,il)
c
c        Q^2 contacts
      do 8215 il=imen21,imen15
      do 8215 iv=1,6
 8215 vv2(iv)=vv2(iv)+vj(iv,il)
c
```

```
c        Q^4 contacts
      do 8225 il=imen14,imen
      do 8225 iv=1,6
 8225 vv4(iv)=vv4(iv)+vj(iv,il)
c
c
c        ------------------------------------------------------
c        NOTE: partial-wave potentials that add-up to zero need
c        to be cutoff, because they diverge for large momenta.
c        ------------------------------------------------------
c
c        use 3d3 cutoff as default for all j.gt.3 partial waves
c
      if (j.gt.3) then
      if (cutlsj(1,15).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,15))**(2.d0*cutlsj(1,15))
     1    +(ymev/cutlsj(2,15))**(2.d0*cutlsj(1,15))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
c
      do 8275 iv=1,6
      vv0(iv)=vv0(iv)*expexp
      vv2(iv)=vv2(iv)*expexp
 8275 vv4(iv)=vv4(iv)*expexp
      go to 8400
      end if
c
c
c        look into individual partial waves and
c        multiply with partial-wave dependent cutoffs
c        --------------------------------------------
c
      j1=j+1
      go to (8310,8320,8330,8340),j1
c
c
c        j=0
c        ---
c        ---
c
 8310 continue
c
c        1s0
c        ---
c        Q^0 term
c
      if (cutlsj(1,1).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,1))**(2.d0*cutlsj(1,1))
     1    +(ymev/cutlsj(2,1))**(2.d0*cutlsj(1,1))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv0(1)=vv0(1)*expexp
c
c        Q^2 terms
c
      if (cutlsj(3,1).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(4,1))**(2.d0*cutlsj(3,1))
     1    +(ymev/cutlsj(4,1))**(2.d0*cutlsj(3,1))
```

```
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv2(1)=vv2(1)*expexp
c
c        Q^4 terms
c
      if (cutlsj(5,1).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(6,1))**(2.d0*cutlsj(5,1))
     1    +(ymev/cutlsj(6,1))**(2.d0*cutlsj(5,1))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(1)=vv4(1)*expexp
c
c        3p0
c        ---
c        Q^2 term
c
      if (cutlsj(1,2).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,2))**(2.d0*cutlsj(1,2))
     1    +(ymev/cutlsj(2,2))**(2.d0*cutlsj(1,2))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv2(3)=vv2(3)*expexp
      vv0(3)=vv0(3)*expexp
c
c        Q^4 term
c
      if (cutlsj(3,2).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(4,2))**(2.d0*cutlsj(3,2))
     1    +(ymev/cutlsj(4,2))**(2.d0*cutlsj(3,2))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(3)=vv4(3)*expexp
c
      go to 8400
c
c
c        j=1
c        ---
c        ---
c
 8320 continue
c
c        1p1
c        ---
c        Q^2 term
c
      if (cutlsj(1,3).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,3))**(2.d0*cutlsj(1,3))
     1    +(ymev/cutlsj(2,3))**(2.d0*cutlsj(1,3))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv2(1)=vv2(1)*expexp
```

```
      vv0(1)=vv0(1)*expexp
c
c        Q^4 term
c
      if (cutlsj(3,3).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(4,3))**(2.d0*cutlsj(3,3))
     1    +(ymev/cutlsj(4,3))**(2.d0*cutlsj(3,3))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(1)=vv4(1)*expexp
c
c        3p1
c        ---
c        Q^2 term
c
      if (cutlsj(1,4).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,4))**(2.d0*cutlsj(1,4))
     1    +(ymev/cutlsj(2,4))**(2.d0*cutlsj(1,4))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv2(2)=vv2(2)*expexp
      vv0(2)=vv0(2)*expexp
c
c        Q^4 term
c
      if (cutlsj(3,4).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(4,4))**(2.d0*cutlsj(3,4))
     1    +(ymev/cutlsj(4,4))**(2.d0*cutlsj(3,4))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(2)=vv4(2)*expexp
c
c        3s1
c        ---
c        Q^0 term
c
      if (cutlsj(1,5).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,5))**(2.d0*cutlsj(1,5))
     1    +(ymev/cutlsj(2,5))**(2.d0*cutlsj(1,5))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv0(4)=vv0(4)*expexp
c
c        Q^2 terms
c
      if (cutlsj(3,5).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(4,5))**(2.d0*cutlsj(3,5))
     1    +(ymev/cutlsj(4,5))**(2.d0*cutlsj(3,5))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv2(4)=vv2(4)*expexp
```

```
c
c        Q^4 terms
c
      if (cutlsj(5,5).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(6,5))**(2.d0*cutlsj(5,5))
     1    +(ymev/cutlsj(6,5))**(2.d0*cutlsj(5,5))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(4)=vv4(4)*expexp
c
c        3d1
c        ---
c        Q^4 term
c
      if (cutlsj(1,6).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,6))**(2.d0*cutlsj(1,6))
     1    +(ymev/cutlsj(2,6))**(2.d0*cutlsj(1,6))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(3)=vv4(3)*expexp
      vv2(3)=vv2(3)*expexp
      vv0(3)=vv0(3)*expexp
c
c        3s/d1
c        -----
c        Q^2 term
c
      if (cutlsj(1,7).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,7))**(2.d0*cutlsj(1,7))
     1    +(ymev/cutlsj(2,7))**(2.d0*cutlsj(1,7))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv2(5)=vv2(5)*expexp
      vv2(6)=vv2(6)*expexp
      vv0(5)=vv0(5)*expexp
      vv0(6)=vv0(6)*expexp
c
c        Q^4 term
c
      if (cutlsj(3,7).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(4,7))**(2.d0*cutlsj(3,7))
     1    +(ymev/cutlsj(4,7))**(2.d0*cutlsj(3,7))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(5)=vv4(5)*expexp
      vv4(6)=vv4(6)*expexp
c
      go to 8400
c
c
c        j=2
c        ---
c        ---
c
```

```
 8330 continue
c
c        1d2
c        ---
c        Q^4 term
c
      if (cutlsj(1,8).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,8))**(2.d0*cutlsj(1,8))
     1    +(ymev/cutlsj(2,8))**(2.d0*cutlsj(1,8))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(1)=vv4(1)*expexp
      vv2(1)=vv2(1)*expexp
      vv0(1)=vv0(1)*expexp
c
c        3d2
c        ---
c        Q^4 term
c
      if (cutlsj(1,9).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,9))**(2.d0*cutlsj(1,9))
     1    +(ymev/cutlsj(2,9))**(2.d0*cutlsj(1,9))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(2)=vv4(2)*expexp
      vv2(2)=vv2(2)*expexp
      vv0(2)=vv0(2)*expexp
c
c        3p2
c        ---
c        Q^2 term
c
      if (cutlsj(1,10).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,10))**(2.d0*cutlsj(1,10))
     1    +(ymev/cutlsj(2,10))**(2.d0*cutlsj(1,10))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv2(4)=vv2(4)*expexp
      vv0(4)=vv0(4)*expexp
      vv2(3)=vv2(3)*expexp
      vv0(3)=vv0(3)*expexp
c
c        Q^4 terms
c
      if (cutlsj(3,10).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(4,10))**(2.d0*cutlsj(3,10))
     1    +(ymev/cutlsj(4,10))**(2.d0*cutlsj(3,10))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(4)=vv4(4)*expexp
      vv4(3)=vv4(3)*expexp
c
c        3p/f2
c        -----
```

```
c        Q^4 term
c
      if (cutlsj(1,12).eq.0.d0) then
      expexp=1.d0
      else
      expo=(xmev/cutlsj(2,12))**(2.d0*cutlsj(1,12))
     1    +(ymev/cutlsj(2,12))**(2.d0*cutlsj(1,12))
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
      end if
      vv4(5)=vv4(5)*expexp
      vv4(6)=vv4(6)*expexp
      vv2(5)=vv2(5)*expexp
      vv2(6)=vv2(6)*expexp
      vv0(5)=vv0(5)*expexp
      vv0(6)=vv0(6)*expexp
c
      go to 8400
c
c
c        j=3
c        ---
c        ---
c
 8340 continue
c
c        3d3
c        ---
c        special cutoff: for the exponent, the parameter
c        from the list is not used. Instead it is used
c        what you see below.
c        Q^4 term
c
      if (cutlsj(1,15).eq.0.d0) then
      expexp=1.d0
      else
      expo1=(xmev/cutlsj(2,15))**(2.d0*2.0d0)
     1    +(ymev/cutlsj(2,15))**(2.d0*2.0d0)
      if (expo1.gt.rrr) expo1=rrr
      expexp1=dexp(-expo1)
      expo2=(xmev/cutlsj(2,15))**(2.d0*3.0d0)
     1    +(ymev/cutlsj(2,15))**(2.d0*3.0d0)
      if (expo2.gt.rrr) expo2=rrr
      expexp2=dexp(-expo2)
      expexp=0.5d0*(expexp1+expexp2)
      end if
c
c        use 3d3 cutoff for all j.eq.3 partial waves
c
      do 8345 iv=1,6
      vv0(iv)=vv0(iv)*expexp
      vv2(iv)=vv2(iv)*expexp
 8345 vv4(iv)=vv4(iv)*expexp
c
c
c
c
c
c
c        final add up
c        ------------
c
 8400 do 8405 iv=1,6
 8405 v(iv)=v(iv)+vv0(iv)+vv2(iv)+vv4(iv)
c
      end if
```

```
c
c
c
c
 8500 if (j.eq.0.or..not.heform) go to 8900
c
c
c         translation into (combinations of) helicity states
c
c
      do 8505 i=1,4
 8505 vl(i)=v(i+2)
c
      do 8520 ii=1,4
      iii=ii+2
      v(iii)=0.d0
c
      do 8515 i=1,4
 8515 v(iii)=v(iii)+adminv(ii,i)*vl(i)
 8520 v(iii)=v(iii)*a2j1
c
c
c
c
 8900 return
      end









      subroutine chipar_n3lo
c
c        chipar_n3lo reads, writes, and stores the parameter for all
c        chi-subroutines.
c
c
      implicit real*8 (a-h,o-z)
c
c
      common /crdwrt/ kread,kwrite,kpunch,kda(9)
c
      common /cstate/ j,heform,sing,trip,coup,endep,label
      common /cnn/ inn
      logical heform,sing,trip,coup,endep
c
c
c        common block for all chi-subroutines
c
      common /cchi/ vj(32,270),c(20,270),fff,ff,f(52),aa(96),ai(19,30),
     1               wnn(3),wdd(3),x,xx,y,yy,xy2,xxpyy,ex,ey,eem12,
```

```
     2               gaa(3),fpia(3),ezz1(3),ezz2(3),ct(96),wt(96),
     3               ic(20,270),ift(3),mint(3),maxt(3),nt,
     4               mge,mgg(40,3),mggo(40,3),ima(30,40,3),
     5               imaa(3),imea(3),ime,im,mc,m,mg,inter,ide,idde,
     6               indc(2,270),indpar(3),indxy
c
c         specifications for this common block
c
      logical indc,indxy,indpar
c
      common /compar/ cb1a(3),cb2a(3),cb3a(3),cb4a(3),
     1               cd12a(3),cd3a(3),cd5a(3),cd145a(3)
c
      common /comlsj/ clsj(15,50),cutlsj(15,50),indlsj
      logical indlsj
c
c
c        further specifications
c
      dimension cc(5),cca(5)
      dimension clec(15,50)
      dimension a(1024),b(32)
      dimension ttab(5,131),tab(5,131)
      dimension topepp(5,2),topenp(5,2),topenn(5,2)
      dimension t1s0pp(5),t1s0np(5),t1s0nn(5)
      real*4 eps
      character*4 name(3)
      character*4 ntab(3,131)
      integer imga(3)
      character*4 cut,cuta,fun,lsj,lec,end
      character*4 mesong(40)
      logical index
      logical zerocp
      logical indlec
      logical indca,indlca
      data mesong/'0-  ','0-t ','0-st','0+  ','0+st',
     1            '1-  ','1-t ','1-tt','1-st','1-ss',
     2            'c   ','ss  ','ls  ','sq  ','sk  ',
     3            'sl  ',
     4         24*'    '/
      data index/.false./
      data zerocp/.true./
      data pi/3.141592653589793d0/
      data eps/1.e-15/
      data cut/'cut '/,cuta/'cuta'/
      data fun/'fun '/,lsj/'lsj '/,lec/'lec '/,end/'end '/
c
c
c
c
c        parameter tables
c        ----------------
c        ----------------
c
c
c        identification table
c        --------------------
c
c
      data ntab/
     1 'cuta','ll  ','  ',
     2 'sq  ',' ope','p ',
     3 '    ','    ','  ',
     4 '    ','    ','  ',
     5 '    ','    ','  ',
     6 'sq  ',' ope','p ',
```

```
      7 'sq  ',' pi-','g ',
      8 'fun ','    ','  ',
      9 '    ','    ','  ',
      * '    ','    ','  ',
      1 'cuta','ll  ','  ',
      2 'sq  ',' tpn','1 ',
      3 'fun ','    ','  ',
      4 'ss  ',' tpn','1 ',
      5 'fun ','    ','  ',
      6 'fun ','    ','  ',
      7 'c   ',' tpn','2 ',
      8 'fun ','    ','  ',
      9 'sq  ',' tpn','2 ',
      * 'fun ','    ','  ',
      1 'ss  ',' tpn','2 ',
      2 'fun ','    ','  ',
      3 'fun ','    ','  ',
      4 'ls  ',' tpn','2 ',
      5 'fun ','    ','  ',
      6 'c   ',' tpn','3 ',
      7 'fun ','    ','  ',
      8 'c   ',' tpn','3 ',
      9 'fun ','    ','  ',
      * 'ls  ',' tpn','3 ',
      1 'fun ','    ','  ',
      2 'c   ',' tpn','32',
      3 'fun ','    ','  ',
      4 'ss  ',' tpn','32',
      5 'fun ','    ','  ',
      6 'fun ','    ','  ',
      7 'sq  ',' tpn','32',
      8 'fun ','    ','  ',
      9 'c   ',' tpn','3m',
      * 'fun ','    ','  ',
      1 'ss  ',' tpn','3m',
      2 'fun ','    ','  ',
      3 'fun ','    ','  ',
      4 'sq  ',' tpn','3m',
      5 'fun ','    ','  ',
      6 'ls  ',' tpn','3m',
      7 'fun ','    ','  ',
      8 'sl  ',' tpn','3m',
      9 'fun ','    ','  ',
      * 'c   ',' tpn','2c',
      1 'fun ','    ','  ',
      2 'sq  ',' tpn','2c',
      3 'fun ','    ','  ',
      4 'ss  ',' tpn','2c',
      5 'fun ','    ','  ',
      6 'fun ','    ','  ',
      7 'c   ',' tpn','1 ',
      8 'fun ','    ','  ',
      9 'c   ',' tpn','2 ',
      * 'fun ','    ','  ',
      1 'sq  ',' tpn','2 ',
      2 'fun ','    ','  ',
      3 'ss  ',' tpn','2 ',
      4 'fun ','    ','  ',
      5 'fun ','    ','  ',
      6 'ls  ',' tpn','2 ',
      7 'fun ','    ','  ',
      8 'sq  ',' tpn','3 ',
      9 'fun ','    ','  ',
      * 'ss  ',' tpn','3 ',
      1 'fun ','    ','  ',
      2 'fun ','    ','  ',
```

```
     3 'c   ',' tpn','3 ',
     4 'fun ','    ','  ',
     5 'fun ','    ','  ',
     6 'ss  ',' tpn','3 ',
     7 'fun ','    ','  ',
     8 'fun ','    ','  ',
     9 'sq  ',' tpn','3 ',
     * 'fun ','    ','  ',
     1 'ls  ',' tpn','3 ',
     2 'fun ','    ','  ',
     3 'ss  ',' tpn','32',
     4 'fun ','    ','  ',
     5 'fun ','    ','  ',
     6 'sq  ',' tpn','32',
     7 'fun ','    ','  ',
     8 'c   ',' tpn','32',
     9 'fun ','    ','  ',
     * 'c   ',' tpn','3m',
     1 'fun ','    ','  ',
     2 'sq  ',' tpn','3m',
     3 'fun ','    ','  ',
     4 'ss  ',' tpn','3m',
     5 'fun ','    ','  ',
     6 'fun ','    ','  ',
     7 'ls  ',' tpn','3m',
     8 'fun ','    ','  ',
     9 'c   ',' tpn','2c',
     * 'fun ','    ','  ',
     1 'sq  ',' tpn','2c',
     2 'fun ','    ','  ',
     3 'ss  ',' tpn','2c',
     4 'fun ','    ','  ',
     5 'fun ','    ','  ',
     6 'cuta','ll  ','  ',
     7 'lsj ',' 1S0','  ',
     8 'lsj ',' 1S0','  ',
     9 'lsj ',' 1S0','  ',
     * 'lsj ',' 1S0','  ',
     1 'lsj ',' 3P0','  ',
     2 'lsj ',' 3P0','  ',
     3 'lsj ',' 1P1','  ',
     4 'lsj ',' 1P1','  ',
     5 'lsj ',' 3P1','  ',
     6 'lsj ',' 3P1','  ',
     7 'lsj ',' 3S1','  ',
     8 'lsj ',' 3S1','  ',
     9 'lsj ',' 3S1','  ',
     * 'lsj ',' 3S1','  ',
     1 'lsj ',' 3D1','  ',
     2 'lsj ',' 3S-','D1',
     3 'lsj ',' 3S-','D1',
     4 'lsj ',' 3S-','D1',
     5 'lsj ',' 1D2','  ',
     6 'lsj ',' 3D2','  ',
     7 'lsj ',' 3P2','  ',
     8 'lsj ',' 3P2','  ',
     9 'lsj ',' 3P-','F2',
     * 'lsj ',' 3D3','  ',
     1 'end ','para','m.'/
c
c
c        parameters
c        ----------
c
c
      data tab/
```

```
     1    6.000000d0,   0.0d0,    4.0000d0,  500.0d0,   0.0d0,
     2   -1.290000d0,  92.4d0,  134.9766d0,    0.0d0,   0.0d0,
     3    0.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     4    0.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     5    0.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     6   -1.290000d0,  92.4d0,  139.5702d0,    0.0d0,   0.0d0,
     7   -0.062170d0,  92.4d0,  139.5702d0,    0.0d0,   0.0d0,
     8   10.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     9    0.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     *    0.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     1    6.000000d0,   0.0d0,    2.0000d0,  500.0d0,   0.0d0,
     2    1.000000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     3   14.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     4   -1.000000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     5   11.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     6   14.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     7    1.000000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     8   15.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     9    1.000000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     *   17.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     1   -1.000000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     2   11.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     3   17.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     4    1.000000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     5   19.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     6    1.000000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     7   30.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     8    1.000000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     9   39.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     *    1.000000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     1   38.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     2    1.000000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     3   40.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     4    1.000000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     5   42.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     6   11.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     7   -1.000000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     8   42.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     9    1.000000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     *   48.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     1   -1.000000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     2   50.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     3   11.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     4    1.000000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     5   50.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     6    1.000000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     7   53.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     8    1.000000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     9   54.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     *   -1.500000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     1   55.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     2   -1.500000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     3   56.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     4    1.500000d0,   0.0d0,  138.0390d0,    0.0d0,  -1.0d0,
     5   11.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     6   56.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     7    1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     8   13.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     9    1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     *   16.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     1    1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     2   18.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     3   -1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     4   11.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     5   18.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     6    1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
```

```
     7   20.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     8    1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     9   31.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     *   -1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     1   11.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     2   31.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     3    1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     4   36.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     5   11.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     6   -1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     7   37.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     8   11.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     9    1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     *   37.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     1    4.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     2   36.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     3    1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     4   41.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     5   11.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     6   -1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     7   41.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     8    1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     9   43.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     *    1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     1   49.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     2    1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     3   51.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     4   -1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     5   51.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     6   11.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     7    1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     8   52.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     9    1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     *   55.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     1    1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     2   56.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     3   -1.000000d0,   0.0d0,  138.0390d0,    1.0d0,  -1.0d0,
     4   11.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     5   56.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0,
     6    0.000000d0,   0.0d0,    2.0000d0,  500.0d0,   0.0d0,
     7   -0.147167d0,   3.0d0,  500.0000d0,    0.0d0,   0.0d0,
     8    2.380000d0,   2.0d0,  500.0000d0,    0.0d0,   0.0d0,
     9   -2.545000d0,   2.0d0,  500.0000d0,    0.0d0,   0.0d0,
     *  -16.000000d0,   2.0d0,  500.0000d0,    0.0d0,   0.0d0,
     1    1.487000d0,   2.0d0,  500.0000d0,    0.0d0,   0.0d0,
     2    0.245000d0,   3.0d0,  500.0000d0,    0.0d0,   0.0d0,
     3    0.656000d0,   2.0d0,  500.0000d0,    0.0d0,   0.0d0,
     4    5.250000d0,   2.0d0,  500.0000d0,    0.0d0,   0.0d0,
     5   -0.630000d0,   2.0d0,  500.0000d0,    0.0d0,   0.0d0,
     6    2.350000d0,   4.0d0,  500.0000d0,    0.0d0,   0.0d0,
     7   -0.118972496d0,3.0d0,  500.0000d0,    0.0d0,   0.0d0,
     8    0.760000d0,   2.0d0,  500.0000d0,    0.0d0,   0.0d0,
     9    7.000000d0,   2.0d0,  500.0000d0,    0.0d0,   0.0d0,
     *    6.550000d0,   2.0d0,  500.0000d0,    0.0d0,   0.0d0,
     1   -2.800000d0,   2.0d0,  500.0000d0,    0.0d0,   0.0d0,
     2    0.826000d0,   2.0d0,  500.0000d0,    0.0d0,   0.0d0,
     3    2.250000d0,   2.0d0,  500.0000d0,    0.0d0,   0.0d0,
     4    6.610000d0,   2.0d0,  500.0000d0,    0.0d0,   0.0d0,
     5   -1.770000d0,   4.0d0,  500.0000d0,    0.0d0,   0.0d0,
     6   -1.460000d0,   2.0d0,  500.0000d0,    0.0d0,   0.0d0,
     7   -0.538000d0,   2.0d0,  500.0000d0,    0.0d0,   0.0d0,
     8    2.295000d0,   2.0d0,  500.0000d0,    0.0d0,   0.0d0,
     9   -0.465000d0,   4.0d0,  500.0000d0,    0.0d0,   0.0d0,
     *    5.660000d0,   2.5d0,  500.0000d0,    0.0d0,   0.0d0,
     1    0.000000d0,   0.0d0,    0.0000d0,    0.0d0,   0.0d0/
c
```

```
c
      data topepp/
     6   -1.290000d0,  92.4d0,  134.9766d0,    0.0d0,   0.0d0,
     7    0.000000d0,   0.0d0,  139.5702d0,    0.0d0,   0.0d0/
c
c
      data topenp/
     6   -1.290000d0,  92.4d0,  139.5702d0,    0.0d0,   0.0d0,
     7   -0.062170d0,  92.4d0,  139.5702d0,    0.0d0,   0.0d0/
c
c
      data topenn/
     6   -1.290000d0,  92.4d0,  134.9766d0,    0.0d0,   0.0d0,
     7    0.000000d0,   0.0d0,  139.5702d0,    0.0d0,   0.0d0/
c
c
      data t1s0pp/
     7   -0.145286d0,   3.0d0,  500.0000d0,    0.0d0,   0.0d0/
c
c
      data t1s0np/
     7   -0.147167d0,   3.0d0,  500.0000d0,    0.0d0,   0.0d0/
c
c
      data t1s0nn/
     7   -0.146285d0,   3.0d0,  500.0000d0,    0.0d0,   0.0d0/
c
c
c        this has been the end of tables
c
c
      save
c
c
c
c
c
10004 format (1h ,2a4,a2,f12.6,f10.6,1x,f10.5,2(f7.1,3x))
10005 format (1h ,2a4,a2,f4.1,1x,2f10.5,f13.5,f10.5)
10007 format (1h ,2a4,a2,3i3)
10008 format (1h ,63(1h-))
10010 format (1h ,2a4,a2,i3,2f10.2)
10011 format (//' n3lo: Charge-Dependent Chiral NN Potential',
     1 ' at Order Four (N3LO)')
10020 format (1h ,2a4,a2,f12.6,4f9.2)
10021 format (1h ,2a4,a2,5f10.6)
c
c
c
c
      if (index) go to 50
      index=.true.
c
      x=-1.d0
      y=-1.d0
c
c
c
c
c        maxima of certain indices related to the dimension as follows:
c        dimension c(mme,imee),ic(mice,imee),indc(mindce,imee),
c                  mgg(mge,3),mggo(mge,3),mesong(mge),vj(32,imee),
c                  ima(mee,mge,3)
c
      mge=40
      mee=30
```

```
      mme=20
      mice=20
      mindce=2
      imb=1
      ime=0
      imee=270
c        mme always ge mice, mindce
c
c        set all parameters and indices to zero or .false.
c
      do 1 int=1,3
      imga(int)=0
      indpar(int)=.false.
      do 1 mgx=1,mge
      mgg(mgx,int)=0
    1 mggo(mgx,int)=0
c
c
      do 2 il=1,imee
      do 2 mm=1,mme
      if (mm.le.mindce) indc(mm,il)=.false.
      if (mm.le.mice) ic(mm,il)=0
    2 c(mm,il)=0.d0
      endep=.false.
c
c
      pi2=pi*pi
      pi4=pi2*pi2
c
c
c
c
c        start
c        -----
c        -----
c
c
c
c        write title
c
   50 continue
      write (kwrite,10011)
      write (kwrite,10008)
      write (kwrite,10008)
c
c
c        store systematically the parameter sets
c        for pp, np, nn.
c
c
      do 8999 inter=1,3
c
c
      indca=.false.
      indlca=.false.
      indlsj=.false.
      indlec=.false.
      ilsj=0
      ilec=0
      do 55 ii=1,50
      do 55 i=1,15
      clsj(i,ii)=0.d0
      cutlsj(i,ii)=0.d0
   55 clec(i,ii)=0.d0
c
c
```

```
c        fix index-parameters concerning the factor
c        and the cutoff for the potential as a whole
c
      ift(inter)=1
      ezz1(inter)=0.d0
      ezz2(inter)=0.d0
c
c**** write (kwrite,10010) name,ift(inter),ezz1(inter),ezz2(inter)
      iftyp=ift(inter)
      if (iftyp.lt.0.or.iftyp.gt.6) go to 9003
c
c
c        fix parameters for numerical integration
c
      mint(inter)=4
      maxt(inter)=48
c
c**** write (kwrite,10007) name,mint(inter),maxt(inter)
c
c        nucleon mass
c
      go to (51,52,53), inter
c        mass used for pp
   51 wn=938.272d0
      go to 54
c        mass used for np
   52 wn=938.9182d0
      go to 54
c        mass used for nn
   53 wn=939.5653d0
   54 continue
c**** write (kwrite,10004) name,wn
      wnq=wn*wn
      dwn=1.d0/wn
      dwnq=dwn*dwn
      wnn(inter)=wn
c
c
c        ga and fpi
c
      ga=1.29d0
      fpi=92.4d0
c
c**** write (kwrite,10004) name,ga,fpi
      ga2=ga*ga
      ga4=ga2*ga2
      ga6=ga4*ga2
      fpi=fpi*dwn
      fpi2=fpi*fpi
      fpi4=fpi2*fpi2
      fpi6=fpi4*fpi2
      gaa(inter)=ga2
      fpia(inter)=fpi2
c
c        fix the LECs of the pi-N Lagrangian
c
c        the c_i LECs
      cc(1)=-0.81d0
      cc(2)=2.8d0
      cc(3)=-3.2d0
      cc(4)=5.4d0
c**** write (kwrite,10021) name,cc
      cb1a(inter)=cc(1)*wn*1.d-3
      cb2a(inter)=cc(2)*wn*1.d-3
      cb3a(inter)=cc(3)*wn*1.d-3
      cb4a(inter)=cc(4)*wn*1.d-3
```

```
c
c        the d_i LECs
      cc(1)=3.06d0
      cc(2)=-3.27d0
      cc(3)=0.45d0
      cc(4)=-5.65d0
c**** write (kwrite,10021) name,cc
      cd12a(inter)=cc(1)*wnq*1.d-6
      cd3a(inter)=cc(2)*wnq*1.d-6
      cd5a(inter)=cc(3)*wnq*1.d-6
      cd145a(inter)=cc(4)*wnq*1.d-6
c
      cb1=cb1a(inter)
      cb2=cb2a(inter)
      cb3=cb3a(inter)
      cb4=cb4a(inter)
      cd12=cd12a(inter)
      cd3=cd3a(inter)
      cd5=cd5a(inter)
      cd145=cd145a(inter)
c
c
c
c        prepare table
c
      do 56 ll=1,131
      do 56 i=1,5
   56 ttab(i,ll)=tab(i,ll)
c
c
c        charge-dependent modifications for pp
c
      if (inter.eq.1) then
      do 57 i=1,5
      ttab(i,6)=topepp(i,1)
      ttab(i,7)=topepp(i,2)
   57 ttab(i,107)=t1s0pp(i)
      end if
c
c
c        charge-dependent modifications for np
c
      if (inter.eq.2) then
      do 58 i=1,5
      ttab(i,6)=topenp(i,1)
      ttab(i,7)=topenp(i,2)
   58 ttab(i,107)=t1s0np(i)
      end if
c
c
c        charge-dependent modifications for nn
c
      if (inter.eq.3) then
      do 59 i=1,5
      ttab(i,6)=topenn(i,1)
      ttab(i,7)=topenn(i,2)
   59 ttab(i,107)=t1s0nn(i)
      end if
c
c
c
c
c        get parameters from tables, line by line
c        ----------------------------------------
c        ----------------------------------------
c
```

```
c
c
      line=0
c
   61 line=line+1
      do i=1,5
      if (i.le.3) then
      name(i)=ntab(i,line)
      end if
      cc(i)=ttab(i,line)
      end do
c
c        check if end of input
c
      if (name(1).eq.end) go to 7000
c
c        check if lsj or lec
c
      if (name(1).eq.lsj) go to 6000
      if (name(1).eq.lec) go to 6500
c
c        check if data-card just read contains cut-off or
c        function parameters
c
      if (name(1).eq.cut.or.name(1).eq.fun) go to 70
c
      if (name(1).eq.cuta) then
c**** write (kwrite,10005) name,cc
      indca=.true.
      do i=1,5
      cca(i)=cc(i)
      end do
      go to 61
      end if

c
c
c
c
c        write parameters which are no cut-off or function parameters
c        ------------------------------------------------------------
c
c
c
c
c**** write (kwrite,10004) name,cc
c
c        check if coupling constant is zero
c
c****    do not use zerocp anymore
c****    because the first eight input lines are always pions.
c****    these lines must never be skipped even when g_pi zero.
c**** if (cc(1).ne.0.d0) go to 62
c**** zerocp=.true.
c**** go to 61
c
   62 zerocp=.false.
c
c        find out number of contribution mg
c
      do 63 mg=1,mge
      if (name(1).eq.mesong(mg)) go to 64
   63 continue
      go to 9000
c
c
```

```
c
c
c        store parameters which are no cut-off or function parameters
c        ------------------------------------------------------------
c
c
c
c
c
   64 ime=ime+1
      if (ime.gt.imee) go to 9011
      mgg(mg,inter)=mgg(mg,inter)+1
      m=mgg(mg,inter)
      if (m.gt.mee) go to 9001
      ima(m,mg,inter)=ime
      if (m.ne.1) go to 65
      imga(inter)=imga(inter)+1
      mggo(imga(inter),inter)=mg
   65 continue
c
c
      c(1,ime)=cc(1)
c
c
      if (mg.le.10) then
      c(1,ime)=c(1,ime)*4.d0*pi
      end if

      if (mg.le.3.and.cc(2).ne.0.d0) then
      c(1,ime)=(cc(1)/cc(2)*wn)**2
      if (cc(1).lt.0.d0) c(1,ime)=-c(1,ime)
      end if
c
c
      if (mg.ge.6.and.mg.le.10) then
c        store coupling constant f*g
      c(3,ime)=cc(2)*c(1,ime)
c        store coupling constant f**2
      c(2,ime)=cc(2)*c(3,ime)
      if (mg.eq.10)
     1  c(1,ime)=c(1,ime)+c(3,ime)*2.d0+c(2,ime)
      end if
c
c
      if (mg.ge.11.and.cc(2).ne.0.d0) then
      c(1,ime)=(cc(1)/(2.d0*cc(2))*wn)**2
      if (cc(1).lt.0.d0) c(1,ime)=-c(1,ime)
      end if
c
c
c        store meson mass square in units of nucleon mass square
      c(4,ime)=cc(3)*cc(3)*dwnq
c
c        test iso-spin
      icc=cc(4)
      if (icc.ne.0.and.icc.ne.1) go to 9004
c         store isospin as logical constant
      if (icc.eq.1) indc(1,ime)=.true.
c        store and test iprsp
      icc=cc(5)
      ic(1,ime)=icc
      if (iabs(ic(1,ime)).gt.1) go to 9005
c
c        index values for further storing
      mi=4
      mm=5
c
```

```
c
c        check if there is a `cutall' cutoff
c
      if (indca) then
      name(1)=cut
      do i=1,5
      cc(i)=cca(i)
      end do
      go to 72
      else
      go to 61
      end if
c
c
c
c
c        write cut-off or function parameters
c        ------------------------------------
c
c
c
c
   70 continue
c**** write (kwrite,10005) name,cc
c
      if (zerocp) go to 61
c
   72 continue
c
c
c
c
c        store parameters
c        ----------------
c
c
c
      ityp=cc(1)
c
      if (ityp.eq.0) go to 5995
      if (ityp.lt.1.or.ityp.gt.56) go to 9002
c
      im=ime
c
c        store typ of cut-off or function
      ic(mi,im)=ityp
c
      if (ityp.le.10) then
c        store and test typ of propagator of cut-off
      ic(mi+1,im)=cc(2)
      if (ic(mi+1,im).lt.0.or.ic(mi+1,im).gt.1) go to 9006
      end if
c
      go to (100,100,300,9002,500,600,9002,9002,9002,1000,
     1 1100,1200,1300,1400,1500,1600,1700,1800,1900,2000,
     2 2100,2200,2300,2400,2500,2600,2700,2800,2900,3000,
     3 3100,3200,3300,3400,3500,3600,3700,3800,3900,4000,
     4 4100,4200,4300,9002,9002,9002,9002,4800,4900,5000,
     5 5100,5200,5300,5400,5500,5600),ityp
c
c
c
c
c        cut-off of dipole type
c        **********************
c
```

```
c
c        store and test exponent of cut-off
  100 ic(mi+2,im)=cc(3)
      if (ic(mi+2,im).lt.0) go to 9009
      if (ic(mi+2,im).gt.0) go to 101
c        exponent is zero, omit cut-off
      ic(mi,im)=0
      ic(mi+1,im)=0
      go to 5995
c        store cut-off mass for denominator
  101 c(mm+1,im)=cc(4)*cc(4)*dwnq
c        store numerator of cut-off
      c(mm,im)=c(mm+1,im)
      if (ityp.eq.2)     c(mm,im)=c(mm,im)-c(4,im)
      mi=mi+3
      mm=mm+2
      go to 5995
c
c
c
c
c        exponential form factor of momentum transfer
c        ****************************************
c
c
c        check exponent
  300 if (cc(3).lt.0.d0) go to 9009
      if (cc(3).gt.0.d0) go to 301
c        exponent is zero, omit cutoff
      ic (mi,im)=0
      ic (mi+1,im)=0
      go to 5995
c        store exponent
  301 c(mm+1,im)=cc(3)
c        compute constant factor for argument of exponential function
      c(mm,im)=wnq/(cc(4)*cc(4))
      mi=mi+2
      mm=mm+2
      go to 5995
c
c
c
c
c        sharp cutoff in x and y
c        ***********************
c
c
  500 c(mm,im)=cc(4)*dwn
      mi=mi+2
      mm=mm+1
      go to 5995
c
c
c
c
c        exponential form factor of xx and yy
c        ************************************
c
c
c        check exponent
  600 if (cc(3).lt.0.d0) go to 9009
      if (cc(3).gt.0.d0) go to 601
c        exponent is zero, omit cutoff
      ic (mi,im)=0
      ic (mi+1,im)=0
      go to 5995
```

```
c        store exponent
  601 c(mm+1,im)=cc(3)
c        compute constant factor for argument of exponential function
      c(mm,im)=wnq/(cc(4)*cc(4))
      mi=mi+2
      mm=mm+2
      go to 5995
c
c
c
c
c        pi-gamma potential
c        ******************
c
c
 1000 c(mm,im)=cc(3)
      mi=mi+2
      mm=mm+1
      go to 5995
c
c
c
c
c        function q^2 (momentum-transfer squared)
c        ************
c
c
 1100 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        function k^2 (average-momentum squared)
c        ************
c
c
 1200 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        function 1 for tpn1 (=NLO)
c        **************************
c
c
 1300 c(mm,im)=-1.d0/(384.d0*pi2*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        function 2 for tpn1
c        *******************
c
c
 1400 c(mm,im)=-3.d0*ga4/(64.d0*pi2*fpi4)
```

```
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn2 (=N^2LO), function 1
c        *************************
c
c
 1500 c(mm,im)=-3.d0*ga2/(16.d0*pi*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn2, function 2
c        ****************
c
c
 1600 c(mm,im)=-ga2/(128.d0*pi*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn2, function 3
c        ****************
c
c
 1700 c(mm,im)=9.d0*ga4/(512.d0*pi*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn2, function 4
c        ****************
c
c
 1800 c(mm,im)=-ga2/(32.d0*pi*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c
c        tpn2, function 5
c        ****************
c
c
 1900 c(mm,im)=6.d0*ga4/(64.d0*pi*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
```

```
c
c        tpn2, function 6
c        ****************
c
c
 2000 c(mm,im)=2.d0*ga2*(1.d0-ga2)/(64.d0*pi*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        function q^4 (momentum-transfer to the power of 4)
c        ************
c
 2100 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function k^4 (average-momentum to the power of 4)
c        ************
c
 2200 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function +q^2*k^2
c        *****************
c
 2300 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function  (\vec q x \vec k)^2
c        ****************************
c
 2400 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function xy
c        ***********
c
 2500 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function xx+yy
c        **************
c
```

```
 2600 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function xx*xx+yy*yy
c        ********************
c
 2700 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function xx
c        ***********
c
 2800 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function yy
c        ***********
c
 2900 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn3 (= N^3LO with one loop), function 1
c        ****************************************
c
c
 3000 c(mm,im)=3.d0/(16.d0*pi2*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn3, function 2
c        ****************
c
c
 3100 continue
      c(mm,im)=cb4*cb4/(96.d0*pi2*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function 1.d0
c        *************
c
 3200 continue
```

```
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function 1-q^2/8-k^2/2
c        ************************
c
 3300 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function 1-q^2/8
c        ****************
c
 3400 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c        function 1+k^2/2
c        ****************
c
 3500 continue
      c(mm,im)=cc(2)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn3, function 3
c        ****************
c
c
 3600 c(mm,im)=-cb4/(192.d0*pi2*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn3, function 4
c        ****************
c
c
 3700 c(mm,im)=cb4/(192.d0*pi2*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn3, function 5
c        ****************
c
c
```

```
 3800 continue
      c(mm,im)=cb2*ga2/(8.d0*pi2*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn3, function 6
c        ****************
c
c
 3900 c(mm,im)=-ga2/(32.d0*pi2*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn32 (=N^3LO with 2 loops), function 1
c        ***************************************
c
c
 4000 c(mm,im)=3.d0*ga4/(1024.d0*pi2*fpi6)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn32, function 2
c        *****************
c
c
 4100 c(mm,im)=-ga4/(2048.d0*pi2*fpi6)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn32, function 3
c        *****************
c
c
 4200 c(mm,im)=ga2*cd145/(32.d0*pi2*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn32, function 4
c        *****************
c
c
 4300 c(mm,im)=1.d0/(18432.d0*pi4*fpi6)
      mi=mi+1
      mm=mm+1
      go to 5995
c
```

```
c
c
c
c        tpn3m (= N^3LO, 1/M^2 terms), function 1
c        ****************************************
c
c
 4800 c(mm,im)=-ga4/(32.d0*pi2*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn3m, function 2
c        *****************
c
c
 4900 c(mm,im)=-1.d0/(768.d0*pi2*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn3m, function 3
c        *****************
c
c
 5000 c(mm,im)=ga4/(32.d0*pi2*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn3m, function 4
c        *****************
c
c
 5100 c(mm,im)=1.d0/(1536.d0*pi2*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn3m, function 5
c        *****************
c
c
 5200 c(mm,im)=1.d0/(256.d0*pi2*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn3m, function 6
c        *****************
c
```

```
c
 5300 c(mm,im)=ga4/(4.d0*pi2*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn3m, function 7
c        *****************
c
c
 5400 c(mm,im)=ga4/(32.d0*pi2*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn2c (correction for our it 2pi), function 1
c        *********************************************
c
c
 5500 c(mm,im)=ga4/(128.d0*pi*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        tpn2c (correction for our it 2pi), function 2
c        *********************************************
c
c
 5600 c(mm,im)=-ga4/(256.d0*pi*fpi4)
      mi=mi+1
      mm=mm+1
      go to 5995
c
c
c
c
c        end cut-offs and functions
c        **************************
c
c        test dimensions
 5995 if (mi.gt.mice.or.mm.gt.mme) go to 9010
c
      if (indlca) go to 7800
c
      go to 61
c
c
c
c
c        partial wave LEC's
c        ------------------
c
c
 6000 continue
c**** write (kwrite,10020) name,cc
      indlsj=.true.
      ilsj=ilsj+1
```

```
      if (ilsj.le.4) iilsj=1
      if (ilsj.ge.5.and.ilsj.le.6) iilsj=2
      if (ilsj.ge.7.and.ilsj.le.8) iilsj=3
      if (ilsj.ge.9.and.ilsj.le.10) iilsj=4
      if (ilsj.ge.11.and.ilsj.le.14) iilsj=5
      if (ilsj.ge.15.and.ilsj.le.15) iilsj=6
      if (ilsj.ge.16.and.ilsj.le.18) iilsj=7
      if (ilsj.ge.19.and.ilsj.le.19) iilsj=8
      if (ilsj.ge.20.and.ilsj.le.20) iilsj=9
      if (ilsj.ge.21.and.ilsj.le.22) iilsj=10
      if (ilsj.ge.23.and.ilsj.le.23) iilsj=12
      if (ilsj.ge.24.and.ilsj.le.24) iilsj=15
      if (ilsj.eq.1) iord=0
      if (ilsj.eq.5) iord=0
      if (ilsj.eq.7) iord=0
      if (ilsj.eq.9) iord=0
      if (ilsj.eq.11) iord=0
      if (ilsj.eq.15) iord=0
      if (ilsj.eq.16) iord=0
      if (ilsj.eq.19) iord=0
      if (ilsj.eq.20) iord=0
      if (ilsj.eq.21) iord=0
      if (ilsj.eq.23) iord=0
      if (ilsj.eq.24) iord=0
      iord=iord+1
      clsj(iord,iilsj)=cc(1)
      cutlsj(2*iord-1,iilsj)=cc(2)
      cutlsj(2*iord,iilsj)=cc(3)
      go to 61
c
c
c
c
c        lec LEC's
c        ---------
c
c
 6500 continue
c**** write (kwrite,10020) name,cc
      indlec=.true.
      ilec=ilec+1
      go to (6510,6510,6522,6540,6542,6544),ilec
 6510 do 6515 i=1,5
 6515 clec(ilec,i)=cc(i)
      go to 61
 6522 do 6523 i=1,2
 6523 clec(2,i+5)=cc(i)
      go to 61
 6540 do 6541 i=1,5
 6541 clec(3,i)=cc(i)
      go to 61
 6542 do 6543 i=1,5
 6543 clec(3,i+5)=cc(i)
      go to 61
 6544 do 6545 i=1,5
 6545 clec(3,i+10)=cc(i)
      go to 61
c
c
c
c
c        conclusions
c        -----------
c        -----------
c
c
```

```
c        write end
 7000 continue
c**** write (kwrite,10004) name
c**** write (kwrite,10008)
c**** write (kwrite,10008)
c
      if (indlsj) go to 7100
      if (indlec) go to 7500
      go to 8995
c
c
c        determine the low-energy constants (clec)
c        from the partial wave constants (clsj)
c
c
c        LEC's for Q^0 (LO)
c        ------------------
c
 7100 clec(1,1)=(clsj(1,1)+3.d0*clsj(1,5))*0.25d0/(4.d0*pi)
      clec(1,2)=(clsj(1,5)  -   clsj(1,1))*0.25d0/(4.d0*pi)
c
c
c        LEC's for Q^2 (NLO)
c        -------------------
c
c        vector b
c
c        1s0
      b(1)=clsj(2,1)
c        3p0
      b(2)=clsj(1,2)
c        1p1
      b(3)=clsj(1,3)
c        3p1
      b(4)=clsj(1,4)
c        3s1
      b(5)=clsj(2,5)
c        3s-d1
      b(6)=clsj(1,7)
c        3p2
      b(7)=clsj(1,10)
c
c
      do 7205 i=1,7
 7205 b(i)=b(i)/(4.d0*pi)
c
c
c        matrix a for the C parameters
c
c        1. column
      a(1)=1.d0
      a(2)=-2.d0/3.d0
      a(3)=a(2)
      a(4)=a(2)
      a(5)=1.d0
      a(6)=0.d0
      a(7)=a(2)
c
c        2. column
      a(8)=0.25d0
      a(9)=1.d0/6.d0
      a(10)=a(9)
      a(11)=a(9)
      a(12)=0.25d0
      a(13)=0.d0
      a(14)=a(9)
```

```
c
c        3. column
      a(15)=-3.d0
      a(16)=-2.d0/3.d0
      a(17)=2.d0
      a(18)=a(16)
      a(19)=1.d0
      a(20)=0.d0
      a(21)=a(16)
c
c        4. column
      a(22)=-0.75d0
      a(23)=1.d0/6.d0
      a(24)=-0.5d0
      a(25)=a(23)
      a(26)=0.25d0
      a(27)=0.d0
      a(28)=a(23)
c
c        5. column
      a(29)=0.d0
      a(30)=-2.d0/3.d0
      a(31)=0.d0
      a(32)=-1.d0/3.d0
      a(33)=0.d0
      a(34)=0.d0
      a(35)=1.d0/3.d0
c
c        6. column
      a(36)=-1.d0
      a(37)=2.d0
      a(38)=2.d0/3.d0
      a(39)=-4.d0/3.d0
      a(40)=1.d0/3.d0
      a(41)=-2.d0*dsqrt(2.d0)/3.d0
      a(42)=0.d0
c
c        7. column
      a(43)=-0.25d0
      a(44)=-0.5d0
      a(45)=-1.d0/6.d0
      a(46)=1.d0/3.d0
      a(47)=1.d0/12.d0
      a(48)=-dsqrt(2.d0)/6.d0
      a(49)=0.d0
c
c
c
c
      call dgelg (b,a,7,1,eps,ier)
c
      if (ier.ne.0) write (kwrite,19500) ier
19500 format (///' warning in chipar_n3lo. the error index of dgelg is',
     1 ' ier =',i5/' for the calculation of the C parameters.'///)
c

      do 7255 i=1,7
 7255 clec(2,i)=b(i)
c
c
c        LEC's for Q^4 (N^3LO)
c        ---------------------
c
c        vector b
c
c        1s0
```

```
      b(1)=clsj(3,1)
c        1s0
      b(2)=clsj(4,1)
c        3p0
      b(3)=clsj(2,2)
c        1p1
      b(4)=clsj(2,3)
c        3p1
      b(5)=clsj(2,4)
c        3s1
      b(6)=clsj(3,5)
c        3s1
      b(7)=clsj(4,5)
c        3d1
      b(8)=clsj(1,6)
c        3s-d1
      b(9)=clsj(2,7)
c        3s-d1
      b(10)=clsj(3,7)
c        1d2
      b(11)=clsj(1,8)
c        3d2
      b(12)=clsj(1,9)
c        3p2
      b(13)=clsj(2,10)
c        3p-f2
      b(14)=clsj(1,12)
c        3d3
      b(15)=clsj(1,15)
c
c
      do 7305 i=1,15
 7305 b(i)=b(i)/(4.d0*pi)
c
c
c        matrix a for the D parameters
c
c        1. column
      a(1)=1.d0
      a(2)=10.d0/3.d0
      a(3)=-4.d0/3.d0
      a(4)=a(3)
      a(5)=a(3)
      a(6)=1.d0
      a(7)=a(2)
      a(8)=8.d0/15.d0
      a(9)=0.d0
      a(10)=0.d0
      a(11)=a(8)
      a(12)=a(8)
      a(13)=a(3)
      a(14)=0.d0
      a(15)=a(8)
c
c        2. column
      a(16)=1.d0/16.d0
      a(17)=5.d0/24.d0
      a(18)=1.d0/12.d0
      a(19)=a(18)
      a(20)=a(18)
      a(21)=a(16)
      a(22)=a(17)
      a(23)=1.d0/30.d0
      a(24)=0.d0
      a(25)=0.d0
      a(26)=a(23)
```

```
      a(27)=a(23)
      a(28)=a(18)
      a(29)=0.d0
      a(30)=a(23)
c
c        3. column
      a(31)=1.d0/4.d0
      a(32)=1.d0/6.d0
      a(33)=0.d0
      a(34)=0.d0
      a(35)=0.d0
      a(36)=a(31)
      a(37)=a(32)
      a(38)=-2.d0/15.d0
      a(39)=0.d0
      a(40)=0.d0
      a(41)=a(38)
      a(42)=a(38)
      a(43)=0.d0
      a(44)=0.d0
      a(45)=a(38)
c
c        4. column
      a(46)=0.d0
      a(47)=2.d0/3.d0
      a(48)=0.d0
      a(49)=0.d0
      a(50)=0.d0
      a(51)=0.d0
      a(52)=a(47)
      a(53)=-2.d0/15.d0
      a(54)=0.d0
      a(55)=0.d0
      a(56)=a(53)
      a(57)=a(53)
      a(58)=0.d0
      a(59)=0.d0
      a(60)=a(53)
c
c        5. column
      a(61)=-3.d0
      a(62)=-10.d0
      a(63)=-4.d0/3.d0
      a(64)=4.d0
      a(65)=a(63)
      a(66)=1.d0
      a(67)=10.d0/3.d0
      a(68)=8.d0/15.d0
      a(69)=0.d0
      a(70)=0.d0
      a(71)=-8.d0/5.d0
      a(72)=a(68)
      a(73)=a(63)
      a(74)=0.d0
      a(75)=a(68)
c
c        6. column
      a(76)=-3.d0/16.d0
      a(77)=-5.d0/8.d0
      a(78)=1.d0/12.d0
      a(79)=-1.d0/4.d0
      a(80)=a(78)
      a(81)=1.d0/16.d0
      a(82)=5.d0/24.d0
      a(83)=1.d0/30.d0
      a(84)=0.d0
```

```
      a(85)=0.d0
      a(86)=-1.d0/10.d0
      a(87)=a(83)
      a(88)=a(78)
      a(89)=0.d0
      a(90)=a(83)
c
c        7. column
      a(91)=-3.d0/4.d0
      a(92)=-1.d0/2.d0
      a(93)=0.d0
      a(94)=0.d0
      a(95)=0.d0
      a(96)=1.d0/4.d0
      a(97)=1.d0/6.d0
      a(98)=-2.d0/15.d0
      a(99)=0.d0
      a(100)=0.d0
      a(101)=2.d0/5.d0
      a(102)=a(98)
      a(103)=0.d0
      a(104)=0.d0
      a(105)=a(98)
c
c        8. column
      a(106)=0.d0
      a(107)=-2.d0
      a(108)=0.d0
      a(109)=0.d0
      a(110)=0.d0
      a(111)=0.d0
      a(112)=2.d0/3.d0
      a(113)=-2.d0/15.d0
      a(114)=0.d0
      a(115)=0.d0
      a(116)=2.d0/5.d0
      a(117)=a(113)
      a(118)=0.d0
      a(119)=0.d0
      a(120)=a(113)
c
c        9. column
      a(121)=0.d0
      a(122)=0.d0
      a(123)=-2.d0/3.d0
      a(124)=0.d0
      a(125)=-1.d0/3.d0
      a(126)=0.d0
      a(127)=0.d0
      a(128)=2.d0/5.d0
      a(129)=0.d0
      a(130)=0.d0
      a(131)=0.d0
      a(132)=2.d0/15.d0
      a(133)=1.d0/3.d0
      a(134)=0.d0
      a(135)=-4.d0/15.d0
c
c        10. column
      a(136)=0.d0
      a(137)=0.d0
      a(138)=-1.d0/6.d0
      a(139)=0.d0
      a(140)=-1.d0/12.d0
      a(141)=0.d0
      a(142)=0.d0
```

```
      a(143)=-1.d0/10.d0
      a(144)=0.d0
      a(145)=0.d0
      a(146)=0.d0
      a(147)=-1.d0/30.d0
      a(148)=1.d0/12.d0
      a(149)=0.d0
      a(150)=1.d0/15.d0
c
c        11. column
      a(151)=-1.d0
      a(152)=-10.d0/3.d0
      a(153)=8.d0/3.d0
      a(154)=4.d0/3.d0
      a(155)=-2.d0
      a(156)=1.d0/3.d0
      a(157)=10.d0/9.d0
      a(158)=-4.d0/9.d0
      a(159)=-2.d0*dsqrt(2.d0)/3.d0
      a(160)=-14.d0*dsqrt(2.d0)/9.d0
      a(161)=-8.d0/15.d0
      a(162)=4.d0/5.d0
      a(163)=-2.d0/15.d0
      a(164)=4.d0*dsqrt(6.d0)/15.d0
      a(165)=0.d0
c
c        12. column
      a(166)=-1.d0/4.d0
      a(167)=-1.d0/6.d0
      a(168)=1.d0/3.d0
      a(169)=0.d0
      a(170)=-1.d0/6.d0
      a(171)=1.d0/12.d0
      a(172)=1.d0/18.d0
      a(173)=1.d0/9.d0
      a(174)=-dsqrt(2.d0)/6.d0
      a(175)=dsqrt(2.d0)/18.d0
      a(176)=2.d0/15.d0
      a(177)=-1.d0/5.d0
      a(178)=1.d0/30.d0
      a(179)=-dsqrt(6.d0)/15.d0
      a(180)=0.d0
c
c        13. column
      a(181)=-1.d0/4.d0
      a(182)=-1.d0/6.d0
      a(183)=-1.d0/3.d0
      a(184)=0.d0
      a(185)=1.d0/6.d0
      a(186)=1.d0/12.d0
      a(187)=1.d0/18.d0
      a(188)=1.d0/9.d0
      a(189)=-dsqrt(2.d0)/6.d0
      a(190)=dsqrt(2.d0)/18.d0
      a(191)=2.d0/15.d0
      a(192)=-1.d0/5.d0
      a(193)=-1.d0/30.d0
      a(194)=dsqrt(6.d0)/15.d0
      a(195)=0.d0
c
c        14. column
      a(196)=-1.d0/16.d0
      a(197)=-5.d0/24.d0
      a(198)=-1.d0/6.d0
      a(199)=-1.d0/12.d0
      a(200)=1.d0/8.d0
```

```
      a(201)=1.d0/48.d0
      a(202)=5.d0/72.d0
      a(203)=-1.d0/36.d0
      a(204)=-dsqrt(2.d0)/24.d0
      a(205)=-7.d0*dsqrt(2.d0)/72.d0
      a(206)=-1.d0/30.d0
      a(207)=1.d0/20.d0
      a(208)=1.d0/120.d0
      a(209)=-dsqrt(6.d0)/60.d0
      a(210)=0.d0
c
c        15. column
      a(211)=0.d0
      a(212)=-2.d0/3.d0
      a(213)=0.d0
      a(214)=0.d0
      a(215)=0.d0
      a(216)=0.d0
      a(217)=2.d0/9.d0
      a(218)=-16.d0/45.d0
      a(219)=0.d0
      a(220)=2.d0*dsqrt(2.d0)/9.d0
      a(221)=2.d0/15.d0
      a(222)=4.d0/15.d0
      a(223)=0.d0
      a(224)=0.d0
      a(225)=-2.d0/15.d0
c
c
c
c
      call dgelg (b,a,15,1,eps,ier)
c
      if (ier.ne.0) write (kwrite,19501) ier
19501 format (///' warning in chipar_n3lo. the error index of dgelg is',
     1 ' ier =',i5/' for the calculation of the D parameters.'///)
c
c
      do 7355 i=1,15
 7355 clec(3,i)=b(i)
c
c
c
c        write LEC's
c        -----------
c
c
 7500 continue
c**** write (kwrite,10100)
10100 format (//' Low energy parameters (LEC):'/
     1          ' ---------------------------'/)
c
c        Q^0 (LO)
c**** write (kwrite,10101) (clec(1,i),i=1,2)
10101 format ('lec  CS,CT',2f10.6)
c
c        Q^2 (NLO)
c**** write (kwrite,10102) (clec(2,i),i=1,7)
10102 format ('lec  C_i  ',5f10.6)
c
c        Q^4 (N^3LO)
c**** write (kwrite,10103) (clec(3,i),i=1,15)
10103 format ('lec  D_i  ',5f10.6)
c
c
c
```

```
c
c        store LEC's appropriately
c        -------------------------
c
c
      iorder=0
 7600 iorder=iorder+1
c
c
      mg=10
      iterm=0
 7700 iterm=iterm+1
c
c
      if (iorder.eq.1.and.iterm.gt.2) go to 7600
      if (iorder.eq.2.and.iterm.gt.7) go to 7600
c
c
      mg=mg+1
c
      if (iorder.eq.2) then
      if (iterm.eq.2) mg=mg-1
      if (iterm.eq.4) mg=mg-1
      end if
c
      if (iorder.eq.3) then
      if (iterm.eq.2) mg=mg-1
      if (iterm.eq.3) mg=mg-1
      if (iterm.eq.4) mg=mg-1
      if (iterm.eq.6) mg=mg-1
      if (iterm.eq.7) mg=mg-1
      if (iterm.eq.8) mg=mg-1
      if (iterm.eq.10) mg=mg-1
      if (iterm.eq.12) mg=mg-1
      if (iterm.eq.14) mg=mg-1
      end if
c
c
      ime=ime+1
      if (ime.gt.imee) go to 9011
      mgg(mg,inter)=mgg(mg,inter)+1
      m=mgg(mg,inter)
      if (m.gt.mee) go to 9001
      ima(m,mg,inter)=ime
      if (m.eq.1) then
      imga(inter)=imga(inter)+1
      mggo(imga(inter),inter)=mg
      end if
c
c
      c(1,ime)=clec(iorder,iterm)*wnq*1.d-2
      ic(1,ime)=-1
c
c
      mi=4
      mm=5
c
c
      if (indca) then
      indlca=.true.
      name(1)=cut
      do i=1,5
      cc(i)=cca(i)
      end do
      go to 72
      end if
```

```
c
c
 7800 indlca=.false.
c
c
      if (iorder.eq.2) then
      c(1,ime)=c(1,ime)*wnq*1.d-6
      if (iterm.le.4) then
      imod=mod(iterm,2)
      if (imod.eq.0) imod=2
      ic(mi,ime)=10+imod
      end if
      end if
c
c
      if (iorder.eq.3) then
      c(1,ime)=c(1,ime)*(wnq*1.d-6)**2
      if (iterm.le.8) then
      imod=mod(iterm,4)
      if (imod.eq.0) imod=4
      ic(mi,ime)=20+imod
      end if
      if (iterm.ge.9.and.iterm.le.14) then
      imod=mod(iterm,2)
      if (imod.eq.0) imod=2
      ic(mi,ime)=10+imod
      end if
      end if
c
c
 7900 if (iterm.lt.15) go to 7700
      if (iorder.lt.3) go to 7600
c
c
 8995 imaa(inter)=imb
      imea(inter)=ime
      imb=ime+1
c
 8999 continue
c        this has been the end of the inter loop
c
      return
c
c
c
c        errors
c        ------
c        ------
c
c
c
c
 9000 write (kwrite,19000) name(1)
19000 format (1h ////' error in printcpar  contribution  ',a4,'   does not
     1 exist in this program.'/' execution terminated.'////)
      go to 9999
c
c
 9001 write (kwrite,19001)
19001 format (1h ////' error in printcpartoo many contributions within a g
     1roup with respect to'/' the given dimensions. execution terminated
     2.'////)
      go to 9999
c
c
 9002 write (kwrite,19002) cc(1)
```

```
19002 format (1h ////' error in printcpar cut/fun typ',f10.4,'  does not e
     1xist in this program.'/' execution terminated.'////)
      go to 9999
c
c
 9003 write (kwrite,19003) iftyp
19003 format (1h ////' error in printcpar factor typ has the non-permissib
     1le value',i4,' .'/' execution terminated.'////)
      go to 9999
c
c
 9004 write (kwrite,19004) cc(4)
19004 format (1h ////' error in printcpar isospin has the non-permissible
     1value',f10.4,'  .'/' execution terminated.'////)
      go to 9999
c
c
 9005 write (kwrite,19005) cc(5)
19005 format (1h ////' error in printcpar iprop/spe has the non-permissibl
     1e value',f10.4,'  .'/' execution terminated.'////)
      go to 9999
c
c
 9006 write (kwrite,19006) cc(2)
19006 format (1h ////' error in printcpar the index for the propagator of
     1the cut-off has the'/' non-permissible value',f10.4,'  . execution
     2 terminated.'////)
      go to 9999
c
c
 9009 write (kwrite,19009)
19009 format (1h ////' error in printcpar the exponent of the cut-off is l
     1ess than zero.'/' execution terminated.'////)
      go to 9999
c
c
 9010 write (kwrite,19010)
19010 format (1h ////' error in printcpar too many cut/fun parameters with
     1 respect to the given'/' dimensions. execution terminated.'////)
      go to 9999
c
c
 9011 write (kwrite,19011)
19011 format (1h ////' error in printcpar  too many contr. with respect to
     1 the dimensions given'/' to this program. execution terminated.'
     2////)
      go to 9999
c
c
 9999 stop
      end
```

```
      subroutine chistr_n3lo (icase,max,mex)
c
c        chistr_n3lo computes the structure of one-boson-exchanges
c
c
      implicit real*8 (a-h,o-z)
c
c
c        common blocks
c
      common /crdwrt/ kread,kwrite,kpunch,kda(9)
c
      common /cstate/ j,heform,sing,trip,coup,endep,label
      logical heform,sing,trip,coup,endep
      common /cnn/ inn
c
c
c        common block for all chi-subroutines
c
      common /cchi/ vj(32,270),c(20,270),fff,ff,f(52),aa(96),ai(19,30),
     1               wnn(3),wdd(3),x,xx,y,yy,xy2,xxpyy,ex,ey,eem12,
     2               gaa(3),fpia(3),ezz1(3),ezz2(3),ct(96),wt(96),
     3               ic(20,270),ift(3),mint(3),maxt(3),nt,
     4               mge,mgg(40,3),mggo(40,3),ima(30,40,3),
     5               imaa(3),imea(3),ime,im,mc,m,mg,inter,ide,idde,
     6               indc(2,270),indpar(3),indxy
c
c         specifications for this common block
c
      logical indc,indxy,indpar
c
c     further specifications
c
      dimension vv(32)
      dimension tt(2,3)
      logical index
      logical indiso
      data jj/-1/
      data index/.false./
      save
c
c
c
```

```
c
      if (index) go to 50
      index=.true.
c
c
      do 1 ii=1,3
      tt(1,ii)=1.d0
    1 tt(2,ii)=-3.d0
c
c
c
c
c
   50 do 1095 m=max,mex
      im=ima(m,mg,inter)
c
c
      if (mc.ne.1) go to 60
c
c
c
c
c        call integrals
c        --------------
c
c
c
c
      call chiai_n3lo
c
c
c
c
   60 if (mc.lt.1) mc=1
c
      if (c(mc,im).eq.0.d0) go to 1095
c
c
c
c
c        nn-nn helicity amplitudes /combinations/
c        ----------------------------------------
c
c
c
c
c        basic structure (a factor of 2 is included in v5 and v6)
c
c
      ive=6
c
      vv(1)=f(1)*ai(1,m)+f(2)*ai(2,m)
      vv(2)=f(3)*ai(1,m)+f(4)*ai(3,m)
      vv(3)=f(5)*ai(1,m)+f(6)*ai(2,m)
      vv(4)=f(4)*ai(1,m)+f(3)*ai(3,m)
      vv(5)=f(7)*ai(4,m)
      vv(6)=f(8)*ai(4,m)
c
c
      go to (1000,120,130,140),icase
c
c
c        additional terms required for the tensor coupling
c        of the rho-meson or for certain operators,
c        like, the spin-orbit operator (`ls  ')
c
```

```
c
  120 vv(1)=vv(1)+f(9)*ai(5,m)
      vv(2)=vv(2)+f(10)*ai(2,m)+f(9)*ai(6,m)
      vv(3)=vv(3)+f(10)*ai(5,m)
      vv(4)=vv(4)+f(9)*ai(2,m)+f(10)*ai(6,m)
         e1=f(11)*ai(7,m)
      vv(5)=vv(5)+e1
      vv(6)=vv(6)+e1
      go to 1000
c
c
c        additional terms in case of 2+ mesons
c        not needed here
c
c
  130 continue
      go to 1000
c
c
c        additional terms needed for the sigma-l operator (`sl  ')
c
c
  140 vv(1)=vv(1)+f(6)*ai(5,m)
      vv(2)=vv(2)+f(1)*ai(5,m)+f(9)*ai(6,m)
      vv(3)=vv(3)+f(1)*ai(11,m)
      vv(4)=vv(4)+f(9)*ai(2,m)+f(1)*ai(12,m)
      vv(5)=vv(5)+f(6)*ai(13,m)
      vv(6)=vv(6)+f(6)*ai(13,m)
c
c
c
c
 1000 continue
c
c
c
c
c        set certain cases to zero
c
      if (j.ne.0) go to 1021
      vv(2)=0.d0
      vv(4)=0.d0
      vv(5)=0.d0
      vv(6)=0.d0
c
 1021 mmod=mod(j,2)
      if (.not.sing.or.(mmod.eq.1.and.inn.ne.2)) vv(1)=0.d0
      if (.not.trip.or.(mmod.eq.0.and.inn.ne.2)) vv(2)=0.d0
      if (coup.and.(mmod.eq.0.or.inn.eq.2)) go to 1030
      do 1025 iv=3,6
 1025 vv(iv)=0.d0
c
 1030 continue
c
c
c
c
c        transformation into lsj-formalism
c
      if (j.eq.jj) go to 1035
      jj=j
      aj=dfloat(j)
      aj1=dfloat(j+1)
      d2j1=1.d0/dfloat(2*j+1)
      arjj1=dsqrt(aj*aj1)
c
```

```
 1035 v3=vv(3)
      v4=vv(4)
      v5=vv(5)
      v6=vv(6)
      v34=-arjj1*(v3-v4)
      v56=arjj1*(v5+v6)
      vv(3)=d2j1*(aj1*v3+aj*v4-v56)
      vv(4)=d2j1*(aj*v3+aj1*v4+v56)
      vv(5)=d2j1*(v34-aj1*v5+aj*v6)
      vv(6)=d2j1*(v34+aj*v5-aj1*v6)
c
c
c        possible different sign depending on the convention used
      vv(5)=-vv(5)
      vv(6)=-vv(6)
c
c
c
c
c        multiply with factors
c        ---------------------
c
c
c
c
 1040 is=mod(j,2)+1
      it=mod(is,2)+1
      indiso=indc(1,im)
      cmc=c(mc,im)
      fc=fff*ff*cmc
      do 1045 iv=1,ive
c
c        multiply with coupling-constant and factors fff and ff
c
      vv(iv)=vv(iv)*fc
c
c        multiply with isospin factor
c
      if (.not.indiso) go to 1045
      if (iv.eq.2) go to 1043
      vv(iv)=vv(iv)*tt(is,inter)
      go to 1045
 1043 vv(iv)=vv(iv)*tt(it,inter)
c
c     add up in case of several couplings for one meson-exchange
c     and store
 1045 vj(iv,im)=vj(iv,im)+vv(iv)
c
c
 1095 continue
c
c
      return
      end
```

```
      subroutine chiai_n3lo
c
c        chiai_n3lo integrates over theta
c
c
      implicit real*8 (a-h,o-z)
c
      common /cpot/   v(6),xmev,ymev
      common /cstate/ j,heform,sing,trip,coup,endep,label
      logical heform,sing,trip,coup,endep
c
c
c        common block for all chi-subroutines
c
      common /cchi/ vj(32,270),c(20,270),fff,ff,f(52),aa(96),ai(19,30),
     1               wnn(3),wdd(3),x,xx,y,yy,xy2,xxpyy,ex,ey,eem12,
     2               gaa(3),fpia(3),ezz1(3),ezz2(3),ct(96),wt(96),
     3               ic(20,270),ift(3),mint(3),maxt(3),nt,
     4               mge,mgg(40,3),mggo(40,3),ima(30,40,3),
     5               imaa(3),imea(3),ime,im,mc,m,mg,inter,ide,idde,
     6               indc(2,270),indpar(3),indxy
c
c         specifications for this common block
c
      logical indc,indxy,indpar
c
c
c        further specifications
      dimension gi(7)
c
      dimension pj(7,96)
      real*4 axy2,aomq,am
      logical indj
      data ige/7/
      data nnt/-1/,iinter/-1/,jj/-1/
      save
c
c
```

```
c
c
      if (inter.eq.iinter) go to 60
      iinter=inter
      min=mint(inter)
      max=maxt(inter)
c
      igeint=7
c
      wn=wnn(inter)
      dwn=1.d0/wn
      wnq=wn*wn
c
c
c
c
   60 if (j.eq.jj) go to 70
      jj=j
      indj=.false.
c
c
      aj=dfloat(j)
      aj1=dfloat(j+1)
      dj1=1.d0/aj1
      ajdj1=aj*dj1
      aaj=dsqrt(ajdj1)
c
c
      aj2=dfloat(j+2)
      ajm1=dfloat(j-1)
c
c
      ajj1=aj*aj1
      ajj2=ajm1*aj2
      ajjb=aj*ajm1
c
      aajj=0.d0
      if (j.gt.1)
     1aajj=aj/dsqrt(ajj1*ajj2)
c
      aaj1=aajj*ajm1
      aaj2=aajj*aj1
      aaj3=aajj*2.d0
c
      if (j.gt.1) go to 62
      aajj=0.d0
      go to 63
   62 aajj=1.d0/(aj1*dsqrt(ajj2))
c
   63 aaj4=aajj*ajjb
      aaj5=aajj*aj1*2.d0
      aaj6=aajj*(ajj1+2.d0)
      aaj7=aajj*ajj2
c
c
c
c
c        find out appropriate number of gauss-points
c        -------------------------------------------
c
c
   70 c4=c(4,im)
      if (c4.eq.0.d0) then
      c4=(138.*dwn)**2
      end if
      iprsp=ic(1,im)
```

```
c
c
c        compute am
c
      axy2=xy2
      if (iprsp.ne.1) go to 91
      aomq=eem12+c4
      go to 92
   91 aomq=xxpyy+c4
c
   92 am=axy2/aomq
c
c
c        compute number of gausspoints (nt)
c
c
      if (am.gt.0.999) go to 94
c
c
      if (am.gt.0.85) am=am**(-alog(1.-am)-0.9)
c
c
      nt=float(min)/(1.-am)+0.9
c
c
      if (nt.gt.max) nt=max
      go to 95
c
c
   94 nt=max
c
c
   95 nt=nt+j
c
c        compute nt, which is suitable for gset
c
      if (nt.le.16) go to 98
      if (nt.gt.24) go to 96
      nt=4*(nt/4)
      go to 98
   96 if (nt.gt.48) go to 97
      nt=8*(nt/8)
      go to 98
   97 nt=16*(nt/16)
      if (nt.gt.96) nt=96
c
   98 if (nt.eq.nnt.and.indj) go to 100
c
c
c
c
c        call gauss-points
c        -----------------
c
c
c
c
      call gset (-1.d0,1.d0,nt,ct,wt)
      nnt=nt
c
c
c
c
c        call legendre-polynoms if necessary
c        -----------------------------------
c
```

```
c
c
c
      indxy=.false.
      indj=.true.
      do 99 i=1,nt
      t=ct(i)
      call legp (pj(1,i),pj(3,i),t,j)
      pj(2,i)=pj(1,i)*t
      pj(4,i)=pj(2,i)*t
      pj(6,i)=pj(4,i)*t
      pj(5,i)=pj(3,i)*t
   99 pj(7,i)=pj(5,i)*t
c
c
c
c
c        call integrand
c        --------------
c
c
c
c
  100 call chiaa_n3lo
c
c
c
c
c        prepare for integration
c
c
c
c
      do 2001 ig=1,igeint
 2001 gi(ig)=0.d0
c
c
c
c
c        integration-loop of theta
c        -------------------------
c
c
c
c
      do 2005 i=1,nt
      do 2005 ig=1,igeint
 2005 gi(ig)=gi(ig)+pj(ig,i)*aa(i)
c
c
c
      if (j.ne.0) go to 2010
      gi(3)=0.d0
      gi(5)=0.d0
      gi(7)=0.d0
c
c
c
c
c        combinations of integrals
c        -------------------------
c
c
c
c
 2010 ai(1,m)=gi(1)
```

```
c
      ai(2,m)=gi(2)
      ai(3,m)= ajdj1*gi(2)+dj1*gi(3)
      gi23m  =gi(2)-gi(3)
      ai(4,m)=aaj*gi23m
c
c
      ai(5,m)=gi(4)
      ai(6,m)= ajdj1*gi(4)+dj1*gi(5)
      gi45m  =gi(4)-gi(5)
      ai(7,m)=aaj*gi45m
c
c
      ai( 8,m)= aaj1*gi(4)-aaj2*gi(1)+aaj3*gi(5)
      aai1    = aaj4*gi(4)+aaj5*gi(1)-aaj6*gi(5)
      aai2    = aaj7*gi23m
      ai( 9,m)= aai2+aai1
      ai(10,m)= aai2-aai1
c
c
      ai(11,m)=gi(6)
      ai(12,m)=ajdj1*gi(6)+dj1*gi(7)
      ai(13,m)=aaj*(gi(6)-gi(7))
c
c
      return
      end
      subroutine chiaa_n3lo
```

```
c
c        chiaa_n3lo computes propagators, cutoffs, and functions
c
c
      implicit real*8 (a-h,o-z)
c
      common /crdwrt/ kread,kwrite,kpunch,kda(9)
c
      common /cstate/ j,heform,sing,trip,coup,endep,label
      logical heform,sing,trip,coup,endep
c
c
c        common block for all chi-subroutines
```

```
c
      common /cchi/ vj(32,270),c(20,270),fff,ff,f(52),aa(96),ai(19,30),
     1               wnn(3),wdd(3),x,xx,y,yy,xy2,xxpyy,ex,ey,eem12,
     2               gaa(3),fpia(3),ezz1(3),ezz2(3),ct(96),wt(96),
     3               ic(20,270),ift(3),mint(3),maxt(3),nt,
     4               mge,mgg(40,3),mggo(40,3),ima(30,40,3),
     5               imaa(3),imea(3),ime,im,mc,m,mg,inter,ide,idde,
     6               indc(2,270),indpar(3),indxy
c
c         specifications for this common block
c
      logical indc,indxy,indpar
c
      common /compar/ cb1a(3),cb2a(3),cb3a(3),cb4a(3),
     1               cd12a(3),cd3a(3),cd5a(3),cd145a(3)
c
c
      common /crrr/ rrr
c
c
c
c        further specifications
      dimension deltaq(96,7)
      dimension ell(96),cpa(96),cpaa(96)
      logical indla
      data iinter/-1/
      data cc4/-1.d0/
      data pi/3.141592653589793d0/
      save
c
c
c
c
      if (inter.eq.iinter) go to 10
      iinter=inter
      ga2=gaa(inter)
      fpi2=fpia(inter)
c
      cb1=cb1a(inter)
      cb2=cb2a(inter)
      cb3=cb3a(inter)
      cb4=cb4a(inter)
      cd12=cd12a(inter)
      cd3=cd3a(inter)
      cd5=cd5a(inter)
      cd145=cd145a(inter)
c
      pi2=pi*pi
   10 continue
c
c
c
c
c        delta square
c        ------------
c
c
c
c
      if (indxy) go to 50
      indxy=.true.
      indla=.false.
      do 15 i=1,nt
      xy2t=xy2*ct(i)
c
c
```

```
c        function  -q^2 (- momentum-transfer-squared)
c        --------------
c
c        retardation ignored
c
      deltaq(i,1)=xy2t-xxpyy
c
c        retardation incorporated
c
      deltaq(i,2)=xy2t-eem12
c
c
c        function  +k^2 (average-momentum squared)
c        --------------
c
      deltaq(i,3)=(xy2t+xxpyy)*0.25d0
c
c        function  q^4 (momentum-transfer to the power of 4)
c        -------------
c
      deltaq(i,4)=deltaq(i,1)*deltaq(i,1)
c
c        function  k^4 (average-momentum to the power of 4)
c        -------------
c
      deltaq(i,5)=deltaq(i,3)*deltaq(i,3)
c
c        function  +q^2*k^2
c        -----------------
c
      deltaq(i,6)=-deltaq(i,1)*deltaq(i,3)
c
c        function  (\vec q x \vec k)^2
c        ----------------------------
c
      deltaq(i,7)=xx*yy*(1.d0-ct(i)*ct(i))
c
   15 continue
      go to 50
c
c
c
c     calculate ell, cpa, and cpaa
c
   20 indla=.true.
      cc4=c4
      do 25 i=1,nt
      akk=-deltaq(i,1)
      ak=dsqrt(akk)
      radi=4.d0*c4+akk
      root=dsqrt(radi)
      deno=2.d0*dsqrt(c4)
      ell(i)=root*dlog((root+ak)/deno)/ak
      cpa(i)=datan(ak/deno)/(2.d0*ak)
      cpaa(i)=(2.d0*c4+akk)*cpa(i)
   25 continue
      go to 6000
c
c
c
c
c        propagator
c        ----------
c        ----------
c
c
```

```
c
c
   50 c4=c(4,im)
      iprsp=ic(1,im)
      if (iprsp.lt.0) go to 60
      iret=iprsp+1
c
c         propagator for the nn case
      do 55 i=1,nt
   55 aa(i)=wt(i)/(c4-deltaq(i,iret))
      go to 80
c
c
c        "no propagator"
c
   60 do 65 i=1,nt
   65 aa(i)=wt(i)
c
c
   80 continue
c
c
c
c
c
c        cut-offs and functions
c        ----------------------
c        ----------------------
c
c
c
c
      mi=4
      mm=5
c
c
 5999 ityp=ic(mi,im)
      if (ityp.eq.0) go to 8000
      if (ityp.le.10) then
      iprspc=ic(mi+1,im)
      iret=iprspc+1
      end if
 6000 go to (100,100,300,9002,500,600,9002,9002,9002,1000,
     1 1100,1200,1300,1400,1500,1600,1700,1800,1900,2000,
     2 2100,2200,2300,2400,2500,2600,2700,2800,2900,3000,
     3 3100,3200,3300,3400,3500,3600,3700,3800,3900,4000,
     4 4100,4200,4300,9002,9002,9002,9002,4800,4900,5000,
     5 5100,5200,5300,5400,5500,5600),ityp
c
c
c
c
c        cut-off of dipole type
c        **********************
c
c
  100 c5=c(mm,im)
      c6=c(mm+1,im)
      nexp=ic(mi+2,im)
c
      do 105 i=1,nt
c
      aaa=c5/(c6-deltaq(i,iret))
c     -------------------------
c
      do 105 ii=1,nexp
```

```
  105 aa(i)=aa(i)*aaa
c
c
      mi=mi+3
      mm=mm+2
      go to 5999
c
c
c
c
c        exponential form factor of momentum transfer
c        ********************************************
c
c
  300 c5=c(mm,im)
      c6=c(mm+1,im)
      do 305 i=1,nt
c
      expo=(c5*dabs(deltaq(i,iret)))**c6
c     ----------------------------
c
      if (expo.gt.rrr) expo=rrr
c
      aa(i)=aa(i)*dexp(-expo)
c     ----------------------
c
  305 continue
      mi=mi+2
      mm=mm+2
      go to 5999
c
c
c
c
c        sharp cutoff of x and y
c        ***********************
c
c
  500 c5=c(mm,im)
c
      if (x.gt.c5.or.y.gt.c5) then
c     ----------------------------
      do 505 i=1,nt
  505 aa(i)=0.d0
      end if
c
      mi=mi+2
      mm=mm+1
      go to 5999
c
c
c
c
c        exponential form factor of xx and yy
c        ************************************
c
c
  600 c5=c(mm,im)
      c6=c(mm+1,im)
c
      expo=(c5*xx)**c6+(c5*yy)**c6
c     ----------------------------
      if (expo.gt.rrr) expo=rrr
      expexp=dexp(-expo)
c     ------------------
c
```

```
      do 605 i=1,nt
  605 aa(i)=aa(i)*expexp
      mi=mi+2
      mm=mm+2
      go to 5999
c
c
c
c
c
c        pi-gamma potential
c        ******************
c
c
 1000 c5=c(mm,im)
      do 1055 i=1,nt
      betaq=-deltaq(i,1)/c4
      betaq1=betaq+1.d0
      aaa=-(1.d0-betaq)**2/(2.d0*betaq*betaq)*dlog(betaq1)
     1    +betaq1/(2.d0*betaq)
     2    -2.d0*c5
 1055 aa(i)=aa(i)*aaa
      mi=mi+2
      mm=mm+1
      go to 5999
c
c
c
c
c        function +q^2 (momentum-transfer squared)
c        *************
c
c
 1100 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 1105 i=1,nt
 1105 aa(i)=-aa(i)*deltaq(i,1)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        function k^2 (average-momentum squared)
c        ************
c
c
 1200 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 1205 i=1,nt
 1205 aa(i)=aa(i)*deltaq(i,3)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        function 1 for tpn1
c        *******************
c
c
 1300 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 1305 i=1,nt
```

```
      ga4=ga2*ga2
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      brak=4.d0*c4*(5.d0*ga4-4.d0*ga2 -1.d0)
     1    +akk*(23.d0*ga4-10.d0*ga2-1.d0)
     2    +48.d0*ga4*c4*c4/radi
 1305 aa(i)=aa(i)*c5*ell(i)*brak
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        function 2 for tpn1
c        ********************
c
c
 1400 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 1405 i=1,nt
 1405 aa(i)=aa(i)*c5*ell(i)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn2, function 1
c        ****************
c
c
 1500 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 1505 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      term1=-ga2*c4**(2.5d0)/(16.d0*radi)
      term2=(2.d0*c4*(2.d0*cb1-cb3)-akk*(cb3+3.d0/16.d0*ga2))
     1     *cpaa(i)
 1505 aa(i)=aa(i)*c5*(term1+term2)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn2, function 2
c        ****************
c
c
 1600 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 1605 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      term1=-3.d0*ga2*c4**(2.5d0)/radi
      term2=(4.d0*c4+2.d0*akk-ga2*(4.d0*c4+3.d0*akk))
     1     *cpaa(i)
 1605 aa(i)=aa(i)*c5*(term1+term2)
      mi=mi+1
      mm=mm+1
      go to 5999
c
```

```
c
c
c
c        tpn2, function 3
c        ****************
c
c
 1700 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 1705 i=1,nt
 1705 aa(i)=aa(i)*c5*cpaa(i)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn2, function 4
c        ****************
c
c
 1800 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 1805 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      term1=(cb4+0.25d0)*radi
      term2=-ga2/8.d0*(10.d0*c4+3.d0*akk)
 1805 aa(i)=aa(i)*c5*(term1+term2)*cpa(i)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn2, function 5
c        ****************
c
c
 1900 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 1905 i=1,nt
 1905 aa(i)=aa(i)*c5*cpaa(i)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn2, function 6
c        ****************
c
c
 2000 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 2005 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
 2005 aa(i)=aa(i)*c5*radi*cpa(i)
      mi=mi+1
      mm=mm+1
      go to 5999
c
```

```
c
c
c
c        function q^4 (momentum-transfer to the power of 4)
c        ************
c
c
 2100 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 2105 i=1,nt
 2105 aa(i)=aa(i)*deltaq(i,4)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        function k^4 (average-momentum to the power of 4)
c        ************
c
c
 2200 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 2205 i=1,nt
 2205 aa(i)=aa(i)*deltaq(i,5)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        function +q^2*k^2
c        ******************
c
c
 2300 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 2305 i=1,nt
 2305 aa(i)=aa(i)*deltaq(i,6)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        function  (\vec q x \vec k)^2
c        ****************************
c
c
 2400 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 2405 i=1,nt
 2405 aa(i)=aa(i)*deltaq(i,7)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c        function xy
c        ***********
c
 2500 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
```

```
      aaxy=xy2*0.5d0*c5
      do 2505 i=1,nt
 2505 aa(i)=aa(i)*aaxy
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c        function xx+yy
c        **************
c
 2600 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 2605 i=1,nt
 2605 aa(i)=aa(i)*xxpyy*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c        function xx*xx+yy*yy
c        ********************
c
 2700 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      aaxy=(xx*xx+yy*yy)*c5
      do 2705 i=1,nt
 2705 aa(i)=aa(i)*aaxy
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c        function xx
c        ***********
c
 2800 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 2805 i=1,nt
 2805 aa(i)=aa(i)*xx*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c        function yy
c        ***********
c
 2900 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 2905 i=1,nt
 2905 aa(i)=aa(i)*yy*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn3, function 1
c        ****************
c
c
 3000 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 3005 i=1,nt
```

```
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      brak=(cb2*radi/6.d0+cb3*(2.d0*c4+akk)-4.d0*cb1*c4)**2
     1    +(cb2*radi)**2/45.d0
 3005 aa(i)=aa(i)*c5*ell(i)*brak
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn3, function 2
c        ****************
c
c
 3100 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 3105 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
 3105 aa(i)=aa(i)*c5*ell(i)*radi
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c        function 1.d0
c        *************
c
 3200 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 3205 i=1,nt
 3205 aa(i)=aa(i)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c        function 1-q^2/8-k^/2
c        *********************
c
 3300 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 3305 i=1,nt
 3305 aa(i)=aa(i)*(1.d0+deltaq(i,1)/8.d0-deltaq(i,3)/2.d0)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c        function 1-q^2/8
c        ****************
c
 3400 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 3405 i=1,nt
 3405 aa(i)=aa(i)*(1.d0+deltaq(i,1)/8.d0)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c        function 1+k^/2
c        ***************
c
```

```
 3500 c5=c(mm,im)
      if (c5.eq.0.d0) c5=1.d0
      do 3505 i=1,nt
 3505 aa(i)=aa(i)*(1.d0+deltaq(i,3)/2.d0)*c5
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn3, function 3
c        ****************
c
c
 3600 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 3605 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      brak=radi+ga2*(8.d0*c4+5.d0*akk)
 3605 aa(i)=aa(i)*c5*ell(i)*brak
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn3, function 4
c        ****************
c
c
 3700 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 3705 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      brak=radi-ga2*(16.d0*c4+7.d0*akk)
 3705 aa(i)=aa(i)*c5*ell(i)*brak
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn3, function 5
c        ****************
c
c
 3800 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 3805 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
 3805 aa(i)=aa(i)*c5*ell(i)*radi
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn3, function 6
c        ****************
```

```
c
c
 3900 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 3905 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      brak=(cb2-6.d0*cb3)*akk*akk
     1     +4.d0*(6.d0*cb1+cb2-3.d0*cb3)*akk*c4
     2     +6.d0*(cb2-2.d0*cb3)*c4*c4
     3     +24.d0*(2.d0*cb1+cb3)*c4*c4*c4/radi
 3905 aa(i)=aa(i)*c5*ell(i)*brak
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn32, function 1
c        *****************
c
c
 4000 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 4005 i=1,nt
      akk=-deltaq(i,1)
      wpi=dsqrt(c4)
      brak=(c4+2.d0*akk)*(2.d0*wpi+cpaa(i))
     1     +4.d0*ga2*wpi*(2.d0*c4+akk)
 4005 aa(i)=aa(i)*c5*brak*cpaa(i)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn32, function 2
c        *****************
c
c
 4100 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 4105 i=1,nt
      akk=-deltaq(i,1)
      wpi=dsqrt(c4)
      radi=4.d0*c4+akk
      brak=radi*cpa(i)+2.d0*wpi
     1     +4.d0*ga2*wpi
 4105 aa(i)=aa(i)*c5*brak*radi*cpa(i)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn32, function 3
c        *****************
c
c
 4200 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 4205 i=1,nt
      akk=-deltaq(i,1)
```

```
      radi=4.d0*c4+akk
 4205 aa(i)=aa(i)*c5*ell(i)*radi
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn32, function 4
c        ****************
c
c
 4300 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 4305 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      brak1=4.d0*c4*(2.d0*ga2+1.d0)+akk*(5.d0*ga2+1.d0)
      brak2=akk*(5.d0+13.d0*ga2)/3.d0+8.d0*c4*(1.d0+2.d0*ga2)
      brak=brak1*(brak1*ell(i)-brak2)
c
      brak3=384.d0*pi2*fpi2*(cd12*(2.d0*c4+akk)+4.d0*c4*cd5)
      brak4=2.d0*ga2*(2.d0*c4+akk)-3.d0/5.d0*(ga2-1.d0)*radi
      brak5=192.d0*pi2*fpi2*radi*cd3
      brakbrak=brak1*brak3+brak4*brak5
c
 4305 aa(i)=aa(i)*c5*(brak+brakbrak)*ell(i)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn3m (= N^3LO, 1/M^2 terms), function 1
c        ****************************************
c
c
 4800 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      c44=c4*c4
      c46=c44*c4
      c48=c46*c4
      do 4805 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      radiq=radi*radi
      brak1=c46/(2.d0*radi)
      brak2=(2.d0*c48/radiq+8.d0*c46/radi-akk*akk-2.d0*c44)*ell(i)
 4805 aa(i)=aa(i)*c5*(brak1+brak2)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn3m, function 2
c        *****************
c
c
 4900 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      c44=c4*c4
      c46=c44*c4
```

```
      c48=c46*c4
      yyoff=0.5d0*(xx+yy)
      do 4905 i=1,nt
      akk=-deltaq(i,1)
      akkq=akk*akk
      radi=4.d0*c4+akk
      radiq=radi*radi
      brak1=(radi*(akk-4.d0*yyoff)
     1      +8.d0*ga2*(11.d0/4.d0*akkq+5.d0*c4*akk+3.d0*c44
     2       -6.d0*c46/radi-yyoff*(8.d0*c4+5.d0*akk))
     3      +4.d0*ga4*(yyoff*(20.d0*c4+7.d0*akk-16.d0*c44/radi)
     5       +16.d0*c48/radiq+12.d0*c46/radi-27.d0/4.d0*akkq
     6       -11.d0*c4*akk-6.d0*c44))*ell(i)
      brak2=16.d0*ga4*c46/radi
 4905 aa(i)=aa(i)*c5*(brak1+brak2)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn3m, function 3
c        *****************
c
c
 5000 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      c44=c4*c4
      yyoff=0.5d0*(xx+yy)
      do 5005 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      brak=yyoff+3.d0/8.d0*akk+c44/radi
 5005 aa(i)=aa(i)*c5*brak*ell(i)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn3m, function 4
c        *****************
c
c
 5100 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      c44=c4*c4
      do 5105 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      brak=radi-32.d0*ga2*(c4+7.d0/16.d0*akk)
     1     +4.d0*ga4*(7.d0*c4+17.d0/4.d0*akk+4.d0*c44/radi)
 5105 aa(i)=aa(i)*c5*brak*ell(i)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn3m, function 5
c        *****************
c
c
```

```
 5200 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      c44=c4*c4
      do 5205 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      brak=-radi+16.d0*ga2*(c4+3.d0/8.d0*akk)
     1     +4.d0/3.d0*ga4*(-9.d0*c4-11.d0/4.d0*akk+4.d0*c44/radi)
 5205 aa(i)=aa(i)*c5*brak*ell(i)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn3m, function 6
c        *****************
c
c
 5300 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      c44=c4*c4
      do 5305 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      brak=11.d0/32.d0*akk+c44/radi
 5305 aa(i)=aa(i)*c5*brak*ell(i)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn3m, function 7
c        *****************
c
c
 5400 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 5405 i=1,nt
 5405 aa(i)=aa(i)*c5*ell(i)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
c        tpn2c (correction for our it 2pi), function 1
c        *********************************************
c
c
 5500 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 5505 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      term1=dsqrt(c4)*radi
      term2=(2.d0*c4+akk)*cpaa(i)
 5505 aa(i)=aa(i)*c5*(term1+term2)
      mi=mi+1
      mm=mm+1
      go to 5999
c
```

```
c
c
c
c        tpn2c (correction for our it 2pi), function 2
c        ********************************************
c
c
 5600 if (.not.indla.or.cc4.ne.c4) go to 20
      c5=c(mm,im)
      do 5605 i=1,nt
      akk=-deltaq(i,1)
      radi=4.d0*c4+akk
      term1=dsqrt(c4)
      term2=radi*cpa(i)
 5605 aa(i)=aa(i)*c5*(term1+term2)
      mi=mi+1
      mm=mm+1
      go to 5999
c
c
c
c
 9002 write (kwrite,19002) ityp
19002 format (1h ////' error in chiaa_n3lo:  cut/fun typ',i10  ,'  does not e
     1xist in this program.'/' execution terminated.'////)
      stop
c
c
c
c
c
 8000 return
      end

c**************** this is the end of the program n3lo **********************
```