ADAS Subroutine h9ntqd

```
       subroutine h9ntqd  ( nedim   , ntdim   , nfdim  ,
     &                      ifint   , itype   , itypt  , ilinr , iescl ,
     &                      itrn    ,
     &                      ne      , nt      ,
     &                      evt     , xa      ,
     &                      oma     , tva     ,
     &                      upsilon , dnsilon ,
     &                      kappa   , dru_val , dist   ,
     &                      nef     , en      , f      ,
     &                      ntf     , tvf     ,
     &                      lbeth   , beth    ,
     &                      nform1  , param1  , nform2 , param2
     &                     )
C-----------------------------------------------------------------------
C
C  VERSION:   1.0
C
C  PURPOSE:  Executes quadratures over collision strengths to form
C            excitation and de-excitation effective collision strengths
C            for atoms and ions with tabulated collision stregths
C            as a function of x parameter.
C
C            Quadrature can be executed over a Maxwellian, kappa
C            distribution, Druyvesteyn or numerical distribution.
C            Linear interpoation is recomended and is default (ilinr=1)
C            Quadratic interpolation is also allowed for analytic
C            distributions.
C            1/E varaible interpolation is allowed for Maxwellian only.
C
C  DATA:
C
C  PROGRAM:
C
C          (i*4)  ifint   = input     = indep. var. for interpolation
C             (1 = E)
C             (2 = 1/E)
C          (i*4)  itype   = input     = collision strength type, to give
C high energy behaviour
C             (1 = dipole --> a*log(X+b) )
C             (2 = non-dp --> a+X/b      )
C             (3 = spin ch--> a/(X+b)**2 )
C          (i*4)  itypt   = input     = threshold behaviours allowed
C             (1(ion)     = const to 1st pt.)
C             (2(neutral) = 0     to 1st pt.)
C          (i*4)  ilinr   = input     = allow linear or quadratic interp
C             (1 = linear   )
C             (2 = quadratic)
C          (i*4)  iescl   = input     = allow e**2*omega + lin. interp
C             (1 = normal use)
C             (2 = e**2*omega +lin.)
C iescl=2 not implimented
C          (i*4)  itrn    = input     = index of current transition
```

```
C          (i*4)  nedim   = input     = max no of energies in omega file
C          (i*4)  ntdim   = input     = max no of temperatures
C          (i*4)  nfdim   = input     = max no of energies in adf37 file
C          (i*4)  ne   = input     = number of energies in omega file
C          (i*4)  nef   = input     = number of energies in adf37 file
C          (i*4)  nt   = input     = number of temperatures
C          (i*4)  ntf   = input     = number of temperatures in adf37
C          (r*8)  xa()    = input     = tabul. x param. for coll. str.
C          (r*8)  evt   = input     = theshold energy (eV)
C          (r*8)  oma()   = input     = tabul. coll. str.
C          (r*8)  tva()   = input     = temperatures (eV)
C          (r*8)  tvf()   = input     = temperatures (eV) from adf37
C          (i*4)  dist    = input     = electron distribution
C             (0 = Maxwellian )
C             (1 = kappa    )
C             (2 = numerical  )
C             (3 = Druyvesteyn)
C          (r*8)  kappa   = input     = kappa value of electron dist.
C          (r*8)  dru_val = input     = x parameter from Druyvesteyn dist.
C          (r*8)  en(,)   = input     = adf37 energy (eV)
C          (r*8)  f(,)    = input     = adf37 distribution
C          (l*4)  lbeth   = input     = true if limit point exists
C          (r*8)  beth    = input     = infinite energy limit point of omega
C          (i*4)  nform1  = input     = type of threshold behaviour
C            (1 = cutoff     )
C            (2 = energy^param1)
C          (r*8)  param1  = input     = parameter of threshold form
C          (i*4)  nform2  = input     = type of high-energy behaviour
C            (1 => cutoff         )
C            (2 => energy^-param2(1)     )
C            (3 => exp(-param2(1)*energy))
C          (r*8)  param2()= input     = parameter of high-energy form
C
C          (r*8)  xf   = program   = current en(i)/evt
C          (r*8)  omega(,)= program   = oma interpolated to distribution
C         function energy grid
C          (r*8)  sumi()  = program   = gamma contrib. from i  -> i+1
C          (r*8)  sumn()  = program   = gamma contrib. from ne-1  -> ne
C          (r*8)  sumu()  = program   = gamma contrib. from ne --> inf.
C          (r*8)  suml()  = program   = gamma contrib. from thres. -> 1
C          (r*8)  en()    = program   = tabul. ener. for coll. str. (ev)
C          (r*8)  fva()   = program   = indep. var. for interpolation
C          (r*8)  expi()  = program   = current exp(-(ui-ut))
C          (r*8)  expi1() = program   = current exp(-(ui1-ut))
C          (r*8)  exp1()  = program   = exp(-(u1-ut))
C          (r*8)  ui()    = program   = current eva(i)/kte
C          (r*8)  ui1()   = program   = current eva(i+1)/kte
C          (r*8)  u1()    = program   = eva(1)/kte
C          (r*8)  ut   = program   = evt/kte
C          (r*8)  uj()    = program   = ui-ut
C          (r*8)  uj1()   = program   = ui1-ut
C          (r*8)  w0   = program   = interpolation working variable
C          (r*8)  w1   = program   = interpolation working variable
```

```
C          (r*8)  w2   = program   = interpolation working variable
C          (r*8)  v0   = program   = interpolation working variable
C          (r*8)  v1   = program   = interpolation working variable
C          (r*8)  v2   = program   = interpolation working variable
C          (r*8)  y1   = program   = interpolation working variable
C          (r*8)  y2   = program   = interpolation working variable
C          (r*8)  c0   = program   = interpolation working variable
C          (r*8)  c1   = program   = interpolation working variable
C          (r*8)  c2   = program   = interpolation working variable
C          (r*8)  cc0   = program   = interpolation working variable
C          (r*8)  cc1   = program   = interpolation working variable
C          (r*8)  ww0   = program   = interpolation working variable
C          (r*8)  ww1   = program   = interpolation working variable
C          (r*8)  ww2   = program   = interpolation working variable
C          (r*8)  a1   = program   = interpolation working variable
C          (r*8)  a2   = program   = interpolation working variable
C          (r*8)  b1   = program   = interpolation working variable
C          (r*8)  b2   = program   = interpolation working variable
C
C          (r*8)  upsilon(,)  = output    = upsilon values
C          (r*8)  dnsilon(,)  = output    = downsilon values
C
C  routines:
C     routine    source brief description
C     ------------------------------------------------------------
C     eei        copase evaluates exp(x)*E1(x)
C     ee2        copase evaluates exp(x)*E2(x)
C     lngama evaluates ln(gamma(a))
c     ingama evaluates incomplete gamma P(a,x)
c     ingamq evaluates incomplete gamma 1-P(a,x)
C
C author:  H P Summers
C     K1/1/57
C     JET ext. 4941
C
C date:    26/05/93
C
C update:  30/11/01  HP Summers - altered input to use x parameter
C
C update:  23/11/04  P Bryans - altered to evaluate non-maxwellian
C  electron distributions
C
C update:  20/07/07  A Whiteford - Modified comments slightly to allow
C                                  for automatic generation of
C                                  documentation.
C
C-----------------------------------------------------------------------
      INTEGER             DIST,        IESCL,       IFINT,       ILINR
      INTEGER             ITRN,        ITYPE,       ITYPT,       NE
      INTEGER             NEDIM,       NEF,         NFDIM,       NFORM1
      INTEGER             NFORM2,      NT,          NTDIM,       NTF
      LOGICAL             LBETH
      REAL*8              BETH,        DNSILON(NTDIM),           DRU_VAL
```

```
REAL*8              EN(NTDIM,NFDIM),          EVT
REAL*8              F(NTDIM,NFDIM),           KAPPA
REAL*8              OMA(NEDIM),  PARAM1,      PARAM2(2)
REAL*8              TVA(NTDIM),  TVF(NTDIM),  UPSILON(NTDIM)
REAL*8              XA(NEDIM)
```