mp2grad_uhf_sv1

```
                            SIAL MBPT2_GRAD_AO2
#
# This SIAL program computes the second-order correction to the density
# and the second-order correction to the energy-weighted density matrix.
# They are written to file to be used in the final gradient construction.
# Author: Victor Lotrich
# Copyright: University of Florida
# This program is licensed under the GPLv2
#
#
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
#
# Declare indeces
# ---------------
#
      index kiter    = 1, 100
#
      aoindex mu     = 1, norb
      aoindex nu     = 1, norb
      aoindex lambda = 1, norb
      aoindex sigma  = 1, norb
#
      moaindex i = baocc, eaocc
      moaindex i1= baocc, eaocc
      moaindex i2= baocc, eaocc
      moaindex i3= baocc, eaocc
#
      moaindex a = bavirt, eavirt
      moaindex a1= bavirt, eavirt
      moaindex a2= bavirt, eavirt
      moaindex a3= bavirt, eavirt
#
      mobindex j = bbocc, ebocc
      mobindex j1= bbocc, ebocc
      mobindex j2= bbocc, ebocc
      mobindex j3= bbocc, ebocc
#
      mobindex b = bbvirt, ebvirt
      mobindex b1= bbvirt, ebvirt
      mobindex b2= bbvirt, ebvirt
      mobindex b3= bbvirt, ebvirt
#
      moaindex p = baocc, eavirt
      moaindex p1= baocc, eavirt
      moaindex p2= baocc, eavirt
      moaindex p3= baocc, eavirt
#
      mobindex q = bbocc, ebvirt
      mobindex q1= bbocc, ebvirt
      mobindex q2= bbocc, ebvirt
      mobindex q3= bbocc, ebvirt
#
# Arrays used in transformation for AO2 algorithm
```

```
# ------------------------------------------------
#
      temp Txixi(mu,i1,lambda,i)
      temp T1xixi(mu,i1,lambda,i)
      temp Txxii(mu,nu,i1,i)
      temp Tixxi(i1,nu,lambda,i)
      temp Txipi(mu,i,p,i1)
      temp Tpipi(p1,i,p,i1)
      temp T1pipi(p1,i,p,i1)
      temp T2pipi(p1,i,p,i1)
      temp Tixai(i,mu,a,i1)
      temp Txaii(mu,a,i,i1)
      temp Tiaai(i,a1,a,i1)
      temp Taaii(a,a1,i,i1)
      temp Txaai(mu,a1,a,i)
      temp Taaai(a2,a,a1,i)
      temp T1aaai(a2,a,a1,i)
      temp Txxai(mu,nu,a,i)
#
      served Vxixi(mu,i1,lambda,i)
      served Vxxii(mu,nu,i1,i)
      served Viixx(i1,i,mu,nu)
      served Vixxi(i1,nu,lambda,i)
      served Vxipi(mu,i,p,i1)
      served Vixai(i,mu,a,i1)
      served Vxaii(mu,a,i,i1)
#
#     Permanent
#     ---------
      served Viaai(i,a1,a,i1)
      served Vaaii(a,a1,i,i1)
      served VSpipi(p1,i,p,i1)
      served ASpipi(p1,i,p,i1)
#
      temp Txjxj(mu,j1,lambda,j)
      temp T1xjxj(mu,j1,lambda,j)
      temp Txxjj(mu,nu,j1,j)
      temp Tjjxx(j1,nu,lambda,j)
      temp Tjxxj(j1,nu,lambda,j)
      temp Txjqj(mu,j,q,j1)
      temp Tqjqj(q1,j,q,j1)
      temp T1qjqj(q1,j,q,j1)
      temp T2qjqj(q1,j,q,j1)
      temp Tjxbj(j,mu,b,j1)
      temp Txbjj(mu,b,j,j1)
      temp Tjbbj(j,b1,b,j1)
      temp Tbbjj(b,b1,j,j1)
      temp Txbbj(mu,b1,b,j)
      temp Tbbbj(b2,b,b1,j)
      temp T1bbbj(b2,b,b1,j)
      temp Txxbj(mu,nu,b,j)
#
      served Vxjxj(mu,j1,lambda,j)
      served Vxxjj(mu,nu,j1,j)
      served Vjjxx(j1,nu,lambda,j)
      served Vjxxj(j1,nu,lambda,j)
      served Vxjqj(mu,j,q,j1)
```

```
      served Vjxbj(j,mu,b,j1)
      served Vxbjj(mu,b,j,j1)
#
#     Permanent
#     ---------
      served Vjbbj(j,b1,b,j1)
      served Vbbjj(b,b1,j,j1)
      served VSqjqj(q1,j,q,j1)
      served ASqjqj(q1,j,q,j1)
#
      temp Txixj(mu,i,nu,j)
      temp Txiqj(mu,i,q,j)
      temp Tpiqj(p,i,q,j)
      temp T1piqj(p,i,q,j)
      temp T2piqj(p,i,q,j)
      temp Tiixx(i,i1,mu,nu)
      temp Tiixb(i,i1,mu,b)
      temp Tiibb(i,i1,b1,b)
      temp Txajj(mu,a,j,j1)
      temp Taajj(a,a1,j,j1)
      temp Txabj(mu,a,b,j)
      temp Tixxj(i,mu,nu,j)
      temp Tixbj(i,mu,b,j)
      temp Tiabj(i,a,b,j)
      temp Taabj(a,a1,b,j)
#
      served Vxixj(mu,i,nu,j)
      served Vxiqj(mu,i,q,j)
      served Vpiqj(p,i,q,j)
      served Apiqj(p,i,q,j)
      served Viixb(i,i1,mu,b)
      served Viibb(i,i1,b1,b)
      served Vxajj(mu,a,j,j1)
      served Vixxj(i,mu,nu,j)
      served Vixbj(i,mu,b,j)
#
#     Permanent
#     ---------
      served Viabj(i,a,b,j)
      served Vaajj(a,a1,j,j1)
#
      temp Txbii(mu,b,i,i1)
      temp Tbbii(b,b1,i,i1)
      temp Tjbii(j,b,i,i1)
      temp Txbai(mu,b,a,i)
      temp Tbbai(b,b1,a,i)
#
      served Vxbii(mu,b,i,i1)
#
#     Permanent
#     ---------
      served Vbbii(b,b1,i,i1)
#
# End Arrays used in transformation for AO2 algorithm
# ---------------------------------------------------
#
# Declare one-particle density arrays
```

```
# ------------------------------------
#
      distributed Pij_aa(i,i1)
      distributed Pij_bb(j,j1)
      distributed Pab_aa(a,a1)
      distributed Pab_bb(b,b1)
      distributed Lai_aa(a,i)
      distributed Lai_bb(b,j)
      distributed Painew_aa(a,i)
      distributed Paiold_aa(a,i)
      distributed Painew_bb(b,j)
      distributed Paiold_bb(b,j)
      distributed Wab_aa(a,a1)
      distributed Wab_bb(b,b1)
      distributed Wij_aa(i,i1)
      distributed Wij_bb(j,j1)
      distributed Wai_aa(a,i)
      distributed Wai_bb(b,j)
      distributed P2_ao(mu,nu)
      distributed P2A_ao(mu,nu)
      distributed P2B_ao(mu,nu)
      distributed W2_ao(mu,nu)
      distributed Paa_ao(mu,nu)
      distributed Pbb_ao(mu,nu)
      distributed Whfa(mu,nu)
      distributed Whfb(mu,nu)
      distributed Dhfa(mu,nu)
      distributed Dhfb(mu,nu)
      distributed Dhf(mu,nu)
      distributed Lxi(mu,i)
      distributed Lxj(nu,j)
      distributed Yxi(mu,i)
      distributed Yxj(nu,j)
      local L0xxxi(mu,nu,lambda,i)
      local L0xxxj(mu,nu,lambda,j)
      temp V0xxxi(mu,nu,lambda,i)
      temp V0xxxj(mu,nu,lambda,j)
      local LDhfa(mu,nu)
      local LDhfb(mu,nu)
      local LP2A_ao(mu,nu)
      local LP2B_ao(mu,nu)
      local La(p,mu)
      local Lb(q,mu)
#
# Declare temporary arrays
# ------------------------
#
      temp Txxxx(mu,nu,lambda,sigma)
      temp TSxxxx(mu,nu,lambda,sigma)
      temp T1xxxx(mu,nu,lambda,sigma)
      temp Txxxi(mu,nu,lambda,i)
      temp T1xxxi(mu,nu,lambda,i)
      temp T2xxxi(mu,nu,lambda,i)
      temp T3xxxi(mu,nu,lambda,i)
      temp T4xxxi(mu,nu,lambda,i)
      temp Txxxj(mu,nu,lambda,j)
      temp T1xxxj(mu,nu,lambda,j)
```

```
temp T2xxxj(mu,nu,lambda,j)
temp T3xxxj(mu,nu,lambda,j)
temp T4xxxj(mu,nu,lambda,j)
temp Txixx(mu,i,nu,lambda)
temp T1xixx(mu,i,nu,lambda)
temp txxix(nu,mu,i,lambda)
temp txxjx(nu,mu,j,lambda)
temp Txjxx(mu,j,nu,lambda)
temp T1xjxx(mu,j,nu,lambda)
temp gaa(mu,i,nu,lambda)
temp gab(mu,i,nu,lambda)
temp gbb(mu,i,nu,lambda)
temp TAxxxi(mu,lambda,nu,i1)
temp TBxxxi(mu,lambda,sigma,i1)
temp TAxxxj(mu,lambda,nu,j1)
temp TBxxxj(mu,lambda,sigma,j1)
temp Tii(i,i1)
temp T1ii(i,i1)
temp Tjj(j,j1)
temp T1jj(j,j1)
temp Taa(a,a1)
temp Tbb(b,b1)
temp Tai(a,i)
temp T1ai(a,i)
temp Tbj(b,j)
temp T1bj(b,j)
temp Tiiaa(i,i1,a,a1)
temp T1iiaa(i,i1,a,a1)
temp Tjjbb(j,j1,b,b1)
temp T1jjbb(j,j1,b,b1)
temp Txi(mu,i)
temp Ixi(mu,i)
temp I1xi(mu,i)
temp Jxi(mu,i)
temp Kxi(mu,i)
temp Txj(mu,j)
temp Ixj(mu,j)
temp I1xj(mu,j)
temp Jxj(mu,j)
temp Kxj(mu,j)
temp Ixx(mu,nu)
temp I1xx(mu,nu)
temp Jxx(mu,nu)
temp J1xx(mu,nu)
temp Kxx(mu,nu)
temp K1xx(mu,nu)
temp Ixa(mu,a)
temp Jxa(mu,a)
temp Ixb(mu,b)
temp Jxb(mu,b)
temp Kxa(mu,a)
temp Kxb(mu,b)
temp Tpq(mu,nu)
temp Txx(mu,nu)
temp T1xx(mu,nu)
temp T2xx(mu,nu)
temp T3xx(mu,nu)
```

```
      temp T4xx(mu,nu)
      temp T5xx(mu,nu)
      temp T6xx(mu,nu)
      temp T7xx(mu,nu)
      temp T8xx(mu,nu)
      temp T9xx(mu,nu)
#
# Declare served arrays
# ---------------------
#
      temp AOINT(mu,nu,lambda,sigma)
      temp dx1(mu,nu,lambda,sigma)
      temp dy1(mu,nu,lambda,sigma)
      temp dz1(mu,nu,lambda,sigma)
      temp dx2(mu,nu,lambda,sigma)
      temp dy2(mu,nu,lambda,sigma)
      temp dz2(mu,nu,lambda,sigma)
      temp dx3(mu,nu,lambda,sigma)
      temp dy3(mu,nu,lambda,sigma)
      temp dz3(mu,nu,lambda,sigma)
      temp dx4(mu,nu,lambda,sigma)
      temp dy4(mu,nu,lambda,sigma)
      temp dz4(mu,nu,lambda,sigma)
#
# Declare Local arrays
# --------------------
#
      local Xa(lambda,i,sigma,mu)
      local Xb(lambda,j,sigma,mu)
      local xTa(sigma,mu,lambda,i)
      local xTb(sigma,mu,lambda,j)
      temp  D2(mu,lambda,nu,sigma)
      temp  D3(mu,lambda,nu,sigma)
#
      local Lxixi(mu,i1,lambda,i)
      local Lxixj(mu,i,lambda,j)
      local Lxjxj(mu,j1,lambda,j)
      local Lxxii(mu,nu,i1,i)
      local Lxxjj(mu,nu,j1,j)
      local Lixxi(i1,nu,lambda,i)
      local Lixxj(i,nu,lambda,j)
      local Ljxxj(j1,nu,lambda,j)
      local Lxipi(mu,i,p,i1)
      local Lpipi(p1,i,p,i1)
      local Lxaii(mu,a,i,i1)
      local Laaii(a1,a,i,i1)
      local Lixai(i,mu,a,i1)
      local Liaai(i,a1,a,i1)
#
      local Lxjqj(mu,j,q,j1)
      local Lqjqj(q1,j,q,j1)
      local Lxbjj(mu,b,j,j1)
      local Lbbjj(b1,b,j,j1)
      local Ljxbj(j,mu,b,j1)
      local Ljbbj(j,b1,b,j1)
      local Lxbii(mu,b,i,i1)
      local Lbbii(b1,b,i,i1)
```

```
      local Lxajj(mu,a,j,j1)
      local Laajj(a1,a,j,j1)
      local Lixbj(i,mu,b,j)
      local Liabj(i,a,b,j)
      local Liixb(i,i1,mu,b)
      local Liibb(i,i1,b1,b)
      local Lxiqj(mu,i,q,j)
      local Lpiqj(p,i,q,j)
#
      local Lxiai(mu,i,a1,i1)
      local Lxjbj(mu,j,b1,j1)
      local Lxibj(mu,i,b,j)
      local L1xixi(mu,i,nu,i1)
      local L1xjxj(mu,j,nu,j1)
      local L1xixj(mu,i,nu,j)
#
# Arrays and scalars  used in iterative computation of Dai
# -------------------------------------------------------
#
      distributed D0ai(a,i)
      distributed D1ai(a,i)
      distributed D2ai(a,i)
      distributed D3ai(a,i)
      distributed D4ai(a,i)
#
      distributed D0bj(b,j)
      distributed D1bj(b,j)
      distributed D2bj(b,j)
      distributed D3bj(b,j)
      distributed D4bj(b,j)
#
      distributed e1ai(a,i)
      distributed e2ai(a,i)
      distributed e3ai(a,i)
      distributed e4ai(a,i)
      distributed e5ai(a,i)
#
      distributed e1bj(b,j)
      distributed e2bj(b,j)
      distributed e3bj(b,j)
      distributed e4bj(b,j)
      distributed e5bj(b,j)
#
      scalar b11
      scalar b12
      scalar b13
      scalar b14
      scalar b15
      scalar b16
      scalar b17
      scalar b18
      scalar b19
      scalar b110
#
      scalar b22
      scalar b23
      scalar b24
```

```
      scalar b25
      scalar b26
      scalar b27
      scalar b28
      scalar b29
      scalar b210
#
      scalar b33
      scalar b34
      scalar b35
      scalar b36
      scalar b37
      scalar b38
      scalar b39
      scalar b310
      scalar b44
      scalar b45
      scalar b46
      scalar b47
      scalar b48
      scalar b49
      scalar b410
#
      scalar b55
      scalar b56
      scalar b57
      scalar b58
      scalar b59
      scalar b510
#
      scalar b66
      scalar b67
      scalar b68
      scalar b69
      scalar b610
#
      scalar b77
      scalar b78
      scalar b79
      scalar b710
#
      scalar b88
      scalar b89
      scalar b810
#
      scalar b99
      scalar b910
#
      scalar b1010
#
      scalar Tb11
      scalar Tb12
      scalar Tb13
      scalar Tb14
      scalar Tb15
      scalar Tb16
      scalar Tb17
```

```
      scalar Tb18
      scalar Tb19
      scalar Tb110
#
      scalar Tb22
      scalar Tb23
      scalar Tb24
      scalar Tb25
      scalar Tb26
      scalar Tb27
      scalar Tb28
      scalar Tb29
      scalar Tb210
#
      scalar Tb33
      scalar Tb34
      scalar Tb35
      scalar Tb36
      scalar Tb37
      scalar Tb38
      scalar Tb39
      scalar Tb310
      scalar Tb44
      scalar Tb45
      scalar Tb46
      scalar Tb47
      scalar Tb48
      scalar Tb49
      scalar Tb410
#
      scalar Tb55
      scalar Tb56
      scalar Tb57
      scalar Tb58
      scalar Tb59
      scalar Tb510
#
      scalar Tb66
      scalar Tb67
      scalar Tb68
      scalar Tb69
      scalar Tb610
#
      scalar Tb77
      scalar Tb78
      scalar Tb79
      scalar Tb710
#
      scalar Tb88
      scalar Tb89
      scalar Tb810
#
      scalar Tb99
      scalar Tb910
#
      scalar Tb1010
#
```

```
      scalar c1
      scalar c2
      scalar c3
      scalar c4
      scalar c5
      scalar c6
      scalar c7
      scalar c8
      scalar c9
      scalar c10
#
# Declare scalars
# ---------------
#
      scalar etemp
      scalar esum
      scalar esumaa
      scalar esumbb
      scalar esumab
      scalar ecorraa
      scalar ecorrbb
      scalar ecorrab
      scalar ecorrT
      scalar enew
      scalar eold
      scalar ecrit
      scalar ediff
      scalar mp2_energy
#
#
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
#
# ---------------------------------------------------------------------------
------
#
     PROC ENERGY
#
     esumaa    = 0.0
     ecorraa = 0.0
     PARDO a, a1, i, i1
#
           REQUEST            VSpipi(a,i,a1,i1) a
           Tpipi(a,i,a1,i1) = VSpipi(a,i,a1,i1)
           execute energy_denominator Tpipi
#
           etemp = Tpipi(a,i,a1,i1)*VSpipi(a,i,a1,i1)
           etemp = 0.25*etemp
           esumaa += etemp
     ENDPARDO a, a1, i, i1
#
     ecorrbb = 0.0
     esumbb    = 0.0
     PARDO b, b1, j, j1
#
           REQUEST VSqjqj(b,j,b1,j1) b
           Tqjqj(b,j,b1,j1)  = VSqjqj(b,j,b1,j1)
           execute energy_denominator Tqjqj
```

```
#
           etemp = Tqjqj(b,j,b1,j1)*VSqjqj(b,j,b1,j1)
           etemp = 0.25*etemp
           esumbb += etemp
#
     ENDPARDO b, b1, j, j1

     ecorrab = 0.0
     esumab    = 0.0
     PARDO a, b, i, j
#
           REQUEST Vpiqj(a,i,b,j) a
           Tpiqj(a,i,b,j) = Vpiqj(a,i,b,j)
           execute energy_denominator Tpiqj
#
           etemp = Tpiqj(a,i,b,j)*Vpiqj(a,i,b,j)
           esumab += etemp
#
     ENDPARDO a, b, i, j
     execute sip_barrier
     collective ecorraa += esumaa
     execute print_scalar ecorraa
#
     collective ecorrbb += esumbb
     execute print_scalar ecorrbb
#
     collective ecorrab += esumab
     execute print_scalar ecorrab
#
     mp2_energy  = ecorraa
     mp2_energy += ecorrbb
     mp2_energy += ecorrab
     execute print_scalar mp2_energy
#
      totenerg  = mp2_energy
      totenerg += scfeneg
      execute print_scalar totenerg
#
     ENDPROC ENERGY
#
#    ------------------------------------------------------------------------
#
#
#    ------------------------------------------------------------------------
#
     PROC TRAN_XXII
#
#    ------------------------------------------------------------------------
#
     execute sip_barrier
#
     PARDO mu, nu, lambda
#
           allocate L0xxxi(mu,nu,lambda,*)
           allocate L0xxxj(mu,nu,lambda,*)
#
           DO sigma
```

```
#
                  compute_integrals AOINT(mu,nu,lambda,sigma)
#
                  DO i
#
                     V0xxxi(mu,nu,lambda,i) =
AOINT(mu,nu,lambda,sigma)*ca(sigma,i)
                     L0xxxi(mu,nu,lambda,i) += V0xxxi(mu,nu,lambda,i)
#
                  ENDDO i
#
                  DO j
#
                     V0xxxj(mu,nu,lambda,j) =
AOINT(mu,nu,lambda,sigma)*cb(sigma,j)
                     L0xxxj(mu,nu,lambda,j) += V0xxxj(mu,nu,lambda,j)
#
                  ENDDO j
#
            ENDDO sigma
#
            DO i
#
               txixx(lambda,i,mu,nu) = L0xxxi(mu,nu,lambda,i)
               txxix(nu,mu,i,lambda) = L0xxxi(mu,nu,lambda,i)
#
               DO i1
#
                  Txixi(lambda,i,mu,i1)      = txixx(lambda,i,mu,nu)*ca(nu,i1)
                  prepare Vxixi(lambda,i,mu,i1) += Txixi(lambda,i,mu,i1)
#
               ENDDO i1
#
               DO j1
#
                  Txixj(lambda,i,mu,j1)      = txixx(lambda,i,mu,nu)*cb(nu,j1)
                  prepare Vxixj(lambda,i,mu,j1) += Txixj(lambda,i,mu,j1)
#
               ENDDO j1
#
               DO i1
#
                  Txxii(nu,mu,i,i1)  = txxix(nu,mu,i,lambda)*ca(lambda,i1)
                  prepare Vxxii(nu,mu,i,i1) += Txxii(nu,mu,i,i1)
#
               ENDDO i1
#
               DO i1
#
                  Tixxi(i1,nu,lambda,i)  = L0xxxi(mu,nu,lambda,i)*ca(mu,i1)
                  prepare Vixxi(i1,nu,lambda,i) += Tixxi(i1,nu,lambda,i)
#
               ENDDO i1
#
            ENDDO i
#
            DO j
```

```
#
               txjxx(lambda,j,mu,nu) = L0xxxj(mu,nu,lambda,j)
               txxjx(nu,mu,j,lambda) = L0xxxj(mu,nu,lambda,j)
#
               DO j1
#
                  Txjxj(lambda,j,mu,j1)      = txjxx(lambda,j,mu,nu)*cb(nu,j1)
                  prepare Vxjxj(lambda,j,mu,j1) += Txjxj(lambda,j,mu,j1)
#
               ENDDO j1
#
               DO j1
#
                  Txxjj(nu,mu,j,j1)  = txxjx(nu,mu,j,lambda)*cb(lambda,j1)
                  prepare Vxxjj(nu,mu,j,j1) += Txxjj(nu,mu,j,j1)
#
               ENDDO j1
#
               DO j1
#
                  Tjxxj(j1,nu,lambda,j)  = L0xxxj(mu,nu,lambda,j)*cb(mu,j1)
                  prepare Vjxxj(j1,nu,lambda,j) += Tjxxj(j1,nu,lambda,j)
#
               ENDDO j1
#
            ENDDO j
#
            DO j
#
               DO i1
#
                  Tixxj(i1,nu,lambda,j)  = L0xxxj(mu,nu,lambda,j)*ca(mu,i1)
                  prepare Vixxj(i1,nu,lambda,j) += Tixxj(i1,nu,lambda,j)
#
               ENDDO i1
#
            ENDDO j
#
            deallocate L0xxxi(mu,nu,lambda,*)
            deallocate L0xxxj(mu,nu,lambda,*)
#
      ENDPARDO mu, nu, lambda
#
      execute sip_barrier
      execute server_barrier
#
#     ------------------------------------------------------------------------
#
      ENDPROC TRAN_XXII
#
#     ------------------------------------------------------------------------
#
#     ------------------------------------------------------------------------
#
      PROC TRAN_PIPI
#
#     ------------------------------------------------------------------------
```

```
#
     execute sip_barrier
#
     PARDO mu, i, i1
#
           allocate Lxipi(mu,i,*,i1)
#
           DO nu
#
              REQUEST Vxixi(mu,i,nu,i1) i
#
              DO p
                 Txipi(mu,i,p,i1)  = Vxixi(mu,i,nu,i1)*ca(nu,p)
                 Lxipi(mu,i,p,i1) += Txipi(mu,i,p,i1)
              ENDDO p
#
           ENDDO nu
#
           DO p
#
              PREPARE Vxipi(mu,i,p,i1) = Lxipi(mu,i,p,i1)
#
           ENDDO p
#
           deallocate Lxipi(mu,i,*,i1)
#
     ENDPARDO mu, i, i1
#
     execute sip_barrier
     execute server_barrier
#    discard Vxixi
#
     PARDO p, i, i1
#
           allocate Lpipi(*,i,p,i1)
#
           DO mu
#
              REQUEST            Vxipi(mu,i,p,i1) i
              REQUEST            Vxipi(mu,i1,p,i) i
              Txipi(mu,i,p,i1) = Vxipi(mu,i1,p,i)
              Txipi(mu,i,p,i1)-= Vxipi(mu,i,p,i1)
              Txipi(mu,i,p,i1)*= -1.0
#
              DO p1
#
                #Tpipi(p1,i,p,i1)   = Vxipi(mu,i,p,i1)*ca(mu,p1)
                #T1pipi(p1,i,p,i1)  = Vxipi(mu,i1,p,i)*ca(mu,p1)
                #Tpipi(p1,i,p,i1)  -= T1pipi(p1,i,p,i1)
#
                 Tpipi(p1,i,p,i1)   = Txipi(mu,i,p,i1)*ca(mu,p1)
                 Lpipi(p1,i,p,i1)  += Tpipi(p1,i,p,i1)
#
              ENDDO p1
#
           ENDDO mu
#
```

```
            DO p1
#
               PREPARE VSpipi(p1,i,p,i1) = Lpipi(p1,i,p,i1)
#
            ENDDO p1
#
            deallocate Lpipi(*,i,p,i1)
#
      ENDPARDO p, i, i1
#
      execute sip_barrier
      execute server_barrier
#     DISCARD Vxipi
#
#     ------------------------------------------------------------------------
#
      ENDPROC TRAN_PIPI
#
#     ------------------------------------------------------------------------
#
#     ------------------------------------------------------------------------
#
      PROC TRAN_AAII
#
#     ------------------------------------------------------------------------
#
      execute sip_barrier
#
      PARDO mu, i, i1
#
            allocate Lxaii(mu,*,i,i1)
#
            DO nu
#
               REQUEST Vxxii(mu,nu,i,i1) i
#
               DO a
                  Txaii(mu,a,i,i1)  = Vxxii(mu,nu,i,i1)*ca(nu,a)
                  Lxaii(mu,a,i,i1) += Txaii(mu,a,i,i1)
               ENDDO a
#
            ENDDO nu
#
            DO a
#
               PREPARE Vxaii(mu,a,i,i1) = Lxaii(mu,a,i,i1)
#
            ENDDO a
#
            deallocate Lxaii(mu,*,i,i1)
#
      ENDPARDO mu, i, i1
#
      execute sip_barrier
      execute server_barrier
#
      PARDO a, i, i1
```

```
#
           allocate Laaii(*,a,i,i1)
#
           DO mu
#
              REQUEST Vxaii(mu,a,i,i1) i
#
              DO a1
#
                 Taaii(a1,a,i,i1)  = Vxaii(mu,a,i,i1)*ca(mu,a1)
                 Laaii(a1,a,i,i1) += Taaii(a1,a,i,i1)
#
              ENDDO a1
#
           ENDDO mu
#
           DO a1
#
              PREPARE Vaaii(a1,a,i,i1) = Laaii(a1,a,i,i1)
#
           ENDDO a1
#
           deallocate Laaii(*,a,i,i1)
#
     ENDPARDO a, i, i1
#
     execute sip_barrier
     execute server_barrier
#    DISCARD Vxaii
#
#    ------------------------------------------------------------------------
#
     ENDPROC TRAN_AAII
#
#    ------------------------------------------------------------------------
#
#    ------------------------------------------------------------------------
#
     PROC TRAN_IAAI
#
#    ------------------------------------------------------------------------
#
     execute sip_barrier
#
     PARDO mu, i, i1
#
           allocate Lixai(i,mu,*,i1)
#
           DO nu
#
              REQUEST Vixxi(i,mu,nu,i1) i
#
              DO a
#
                 Tixai(i,mu,a,i1)  = Vixxi(i,mu,nu,i1)*ca(nu,a)
                 Lixai(i,mu,a,i1) += Tixai(i,mu,a,i1)
#
```

```
            ENDDO a
#
         ENDDO nu
#
         DO a
#
            PREPARE Vixai(i,mu,a,i1) = Lixai(i,mu,a,i1)
#
         ENDDO a
#
         deallocate Lixai(i,mu,*,i1)
#
   ENDPARDO mu, i, i1
#
   execute sip_barrier
   execute server_barrier
#  DISCARD Vixxi
#
   PARDO a, i, i1
#
         allocate Liaai(i,*,a,i1)
#
         DO mu
#
            REQUEST Vixai(i,mu,a,i1) i
#
            DO a1
#
               Tiaai(i,a1,a,i1)  = Vixai(i,mu,a,i1)*ca(mu,a1)
               Liaai(i,a1,a,i1) += Tiaai(i,a1,a,i1)
#
            ENDDO a1
#
         ENDDO mu
#
         DO a1
#
            PREPARE Viaai(i,a1,a,i1) = Liaai(i,a1,a,i1)
#
         ENDDO a1
#
         deallocate Liaai(i,*,a,i1)
#
   ENDPARDO a, i, i1
#
   execute sip_barrier
   execute server_barrier
#  DISCARD Vixai
#
#  ------------------------------------------------------------------------
#
   ENDPROC TRAN_IAAI
#
#  ------------------------------------------------------------------------
#
#  ------------------------------------------------------------------------
#
```

```
      PROC TRAN_QJQJ
#
#     ------------------------------------------------------------------------
#
      execute sip_barrier
#
      PARDO mu, j, j1
#
            allocate Lxjqj(mu,j,*,j1)
#
            DO nu
#
               REQUEST Vxjxj(mu,j,nu,j1) j
#
               DO q
#
                  Txjqj(mu,j,q,j1)  = Vxjxj(mu,j,nu,j1)*cb(nu,q)
                  Lxjqj(mu,j,q,j1) += Txjqj(mu,j,q,j1)
#
               ENDDO q
#
            ENDDO nu
#
            DO q
#
               PREPARE Vxjqj(mu,j,q,j1) = Lxjqj(mu,j,q,j1)
#
            ENDDO q
#
            deallocate Lxjqj(mu,j,*,j1)
#
      ENDPARDO mu, j, j1
#
      execute sip_barrier
      execute server_barrier
#     DISCARD Vxjxj
#
      PARDO q, j, j1
#
            allocate Lqjqj(*,j,q,j1)
#
            DO mu
#
               REQUEST            Vxjqj(mu,j,q,j1) j
               REQUEST            Vxjqj(mu,j1,q,j) j
               Txjqj(mu,j,q,j1) = Vxjqj(mu,j1,q,j)
               Txjqj(mu,j,q,j1)-= Vxjqj(mu,j,q,j1)
               Txjqj(mu,j,q,j1)*= -1.0
#
               DO q1
#
                 #Tqjqj(q1,j,q,j1)   = Vxjqj(mu,j,q,j1)*cb(mu,q1)
                 #T1qjqj(q1,j,q,j1)  = Vxjqj(mu,j1,q,j)*cb(mu,q1)
                 #Tqjqj(q1,j,q,j1)  -= T1qjqj(q1,j,q,j1)
                  Tqjqj(q1,j,q,j1)   = Txjqj(mu,j,q,j1)*cb(mu,q1)
                  Lqjqj(q1,j,q,j1)  += Tqjqj(q1,j,q,j1)
#
```

```
            ENDDO q1
#
         ENDDO mu
#
         DO q1
#
            PREPARE VSqjqj(q1,j,q,j1) = Lqjqj(q1,j,q,j1)
#
         ENDDO q1
#
         deallocate Lqjqj(*,j,q,j1)
#
   ENDPARDO q, j, j1
#
   execute sip_barrier
   execute server_barrier
#  DISCARD Vxjqj
#
#  ------------------------------------------------------------------------
#
   ENDPROC TRAN_QJQJ
#
#  ------------------------------------------------------------------------
#
#  ------------------------------------------------------------------------
#
   PROC TRAN_BBJJ
#
#  ------------------------------------------------------------------------
#
   execute sip_barrier
#
   PARDO mu, j, j1
#
         allocate Lxbjj(mu,*,j,j1)
#
         DO nu
#
            REQUEST Vxxjj(mu,nu,j,j1) j
#
            DO b
#
               Txbjj(mu,b,j,j1)  = Vxxjj(mu,nu,j,j1)*cb(nu,b)
               Lxbjj(mu,b,j,j1) += Txbjj(mu,b,j,j1)
#
            ENDDO b
#
         ENDDO nu
#
         DO b
#
            PREPARE Vxbjj(mu,b,j,j1) = Lxbjj(mu,b,j,j1)
#
         ENDDO b
#
         deallocate Lxbjj(mu,*,j,j1)
#
```

```
      ENDPARDO mu, j, j1
#
      execute sip_barrier
      execute server_barrier
#
      PARDO b, j, j1
#
            allocate Lbbjj(*,b,j,j1)
#
            DO mu
#
               REQUEST Vxbjj(mu,b,j,j1) j
#
               DO b1
#
                  Tbbjj(b1,b,j,j1)  = Vxbjj(mu,b,j,j1)*cb(mu,b1)
                  Lbbjj(b1,b,j,j1) += Tbbjj(b1,b,j,j1)
#
               ENDDO b1
#
            ENDDO mu
#
            DO b1
#
               PREPARE Vbbjj(b1,b,j,j1) = Lbbjj(b1,b,j,j1)
#
            ENDDO b1
#
            deallocate Lbbjj(*,b,j,j1)
#
      ENDPARDO b, j, j1
#
      execute sip_barrier
      execute server_barrier
#     DISCARD Vxbjj
#
#     ------------------------------------------------------------------------
#
      ENDPROC TRAN_BBJJ
#
#     ------------------------------------------------------------------------
#
#     ------------------------------------------------------------------------
#
      PROC TRAN_JBBJ
#
#     ------------------------------------------------------------------------
#
      execute sip_barrier
#
      PARDO mu, j, j1
#
            allocate Ljxbj(j,nu,*,j1)
#
            DO nu
#
               REQUEST Vjxxj(j,mu,nu,j1) j
```

```
#
               DO b
#
                  Tjxbj(j,mu,b,j1)  = Vjxxj(j,mu,nu,j1)*cb(nu,b)
                  Ljxbj(j,mu,b,j1) += Tjxbj(j,mu,b,j1)
#
               ENDDO b
#
            ENDDO nu
#
            DO b
#
               PREPARE Vjxbj(j,mu,b,j1) = Ljxbj(j,mu,b,j1)
#
            ENDDO b
#
            deallocate Ljxbj(j,nu,*,j1)
#
      ENDPARDO mu, j, j1
#
      execute sip_barrier
      execute server_barrier
#     DISCARD Vjxxj
#
      PARDO b, j, j1
#
            allocate Ljbbj(j,*,b,j1)
#
            DO mu
#
               REQUEST Vjxbj(j,mu,b,j1) j
#
               DO b1
#
                  Tjbbj(j,b1,b,j1)  = Vjxbj(j,mu,b,j1)*cb(mu,b1)
                  Ljbbj(j,b1,b,j1) += Tjbbj(j,b1,b,j1)
#
               ENDDO b1
#
            ENDDO mu
#
            DO b1
#
               PREPARE Vjbbj(j,b1,b,j1) = Ljbbj(j,b1,b,j1)
#
            ENDDO b1
#
            deallocate Ljbbj(j,*,b,j1)
#
      ENDPARDO b, j, j1
#
      execute sip_barrier
      execute server_barrier
#     DISCARD Vjxbj
#
#     ------------------------------------------------------------------------
#
```

```
      ENDPROC TRAN_JBBJ
#
#     ------------------------------------------------------------------------
#
#     ------------------------------------------------------------------------
#
      PROC TRAN_BBII
#
#     ------------------------------------------------------------------------
#
      execute sip_barrier
#
      PARDO mu, i, i1
#
            allocate Lxbii(mu,*,i,i1)
#
            DO nu
#
               REQUEST Vxxii(mu,nu,i,i1) i
#
               DO b
#
                  Txbii(mu,b,i,i1)  = Vxxii(mu,nu,i,i1)*cb(nu,b)
                  Lxbii(mu,b,i,i1) += Txbii(mu,b,i,i1)
#
               ENDDO b
#
            ENDDO nu
#
            DO b
#
               PREPARE Vxbii(mu,b,i,i1) = Lxbii(mu,b,i,i1)
#
            ENDDO b
#
            deallocate Lxbii(mu,*,i,i1)
#
      ENDPARDO mu, i, i1
#
      execute sip_barrier
      execute server_barrier
#     DISCARD Vxxii
#
      PARDO b, i, i1
#
            allocate Lbbii(*,b,i,i1)
#
            DO mu
#
               REQUEST Vxbii(mu,b,i,i1) i
#
               DO b1
#
                  Tbbii(b1,b,i,i1)  = Vxbii(mu,b,i,i1)*cb(mu,b1)
                  Lbbii(b1,b,i,i1) += Tbbii(b1,b,i,i1)
#
               ENDDO b1
```

```
#
           ENDDO mu
#
           DO b1
#
              PREPARE Vbbii(b1,b,i,i1) = Lbbii(b1,b,i,i1)
#
           ENDDO b1
#
           deallocate Lbbii(*,b,i,i1)
#
     ENDPARDO  b, i, i1
#
     execute sip_barrier
     execute server_barrier
#    DISCARD Vxbii
#
#    ------------------------------------------------------------------------
#
     ENDPROC TRAN_BBII
#
#    ------------------------------------------------------------------------
#
#    ------------------------------------------------------------------------
#
     PROC TRAN_AAJJ
#
#    ------------------------------------------------------------------------
#
     execute sip_barrier
#
     PARDO mu, j, j1
#
           allocate Lxajj(mu,*,j,j1)
#
           DO nu
#
              REQUEST Vxxjj(mu,nu,j,j1) j
#
              DO a
#
                 Txajj(mu,a,j,j1)  = Vxxjj(mu,nu,j,j1)*ca(nu,a)
                 Lxajj(mu,a,j,j1) += Txajj(mu,a,j,j1)
#
              ENDDO a
#
           ENDDO nu
#
           DO a
#
              PREPARE Vxajj(mu,a,j,j1) = Lxajj(mu,a,j,j1)
#
           ENDDO a
#
           deallocate Lxajj(mu,*,j,j1)
#
     ENDPARDO mu, j, j1
```

```
#
     execute sip_barrier
     execute server_barrier
#    DISCARD Vxxjj
#
     PARDO a, j, j1
#
           allocate Laajj(*,a,j,j1)
#
           DO mu
#
              REQUEST Vxajj(mu,a,j,j1) j
#
              DO a1
#
                 Taajj(a1,a,j,j1)  = Vxajj(mu,a,j,j1)*ca(mu,a1)
                 Laajj(a1,a,j,j1) += Taajj(a1,a,j,j1)
#
              ENDDO a1
#
           ENDDO mu
#
           DO a1
#
              PREPARE Vaajj(a1,a,j,j1) = Laajj(a1,a,j,j1)
#
           ENDDO a1
#
           deallocate Laajj(*,a,j,j1)
#
     ENDPARDO  a, j, j1
#
     execute sip_barrier
     execute server_barrier
#    DISCARD Vxajj
#
#    ------------------------------------------------------------------------
#
     ENDPROC TRAN_AAJJ
#
#    ------------------------------------------------------------------------
#
#    ------------------------------------------------------------------------
#
     PROC TRAN_IABJ
#
#    ------------------------------------------------------------------------
#
     execute sip_barrier
#
     PARDO mu, i, j
#
           allocate Lixbj(i,mu,*,j)
#
           DO nu
#
              REQUEST Vixxj(i,mu,nu,j) i
```

```
#
               DO b
#
                  Tixbj(i,mu,b,j)  = Vixxj(i,mu,nu,j)*cb(nu,b)
                  Lixbj(i,mu,b,j) += Tixbj(i,mu,b,j)
#
               ENDDO b
#
            ENDDO nu
#
            DO b
#
               PREPARE Vixbj(i,mu,b,j) = Lixbj(i,mu,b,j)
#
            ENDDO b
#
            deallocate Lixbj(i,mu,*,j)
#
      ENDPARDO mu, i, j
#
      execute sip_barrier
      execute server_barrier
#     DISCARD Vixxj
#
      PARDO b, i, j
#
            allocate Liabj(i,*,b,j)
#
            DO mu
#
               REQUEST Vixbj(i,mu,b,j) i
#
               DO a
#
                  Tiabj(i,a,b,j)  = Vixbj(i,mu,b,j)*ca(mu,a)
                  Liabj(i,a,b,j) += Tiabj(i,a,b,j)
#
               ENDDO a
#
            ENDDO mu
#
            DO a
#
               PREPARE Viabj(i,a,b,j) = Liabj(i,a,b,j)
#
            ENDDO a
#
            deallocate Liabj(i,*,b,j)
#
      ENDPARDO b, i, j
#
      execute sip_barrier
      execute server_barrier
#     DISCARD Vixbj
#
#     ------------------------------------------------------------------------
#
```

```
      ENDPROC TRAN_IABJ
#
#     ------------------------------------------------------------------------
#
#     ------------------------------------------------------------------------
#
      PROC TRAN_IIBB
#
#     ------------------------------------------------------------------------
#
      execute sip_barrier
#
      PARDO mu, i, i1
#
            allocate Liixb(i,i1,mu,*)
#
            DO nu
#
               REQUEST Viixx(i,i1,mu,nu) i
#
               DO b
#
                  Tiixb(i,i1,mu,b)  = Viixx(i,i1,mu,nu)*cb(nu,b)
                  Liixb(i,i1,mu,b) += Tiixb(i,i1,mu,b)
#
               ENDDO b
#
            ENDDO nu
#
            DO b
#
               PREPARE Viixb(i,i1,mu,b) = Liixb(i,i1,mu,b)
#
            ENDDO b
#
            deallocate Liixb(i,i1,mu,*)
#
      ENDPARDO mu, i, i1
#
      execute sip_barrier
      execute server_barrier
#     DISCARD Viixx
#
      PARDO b, i, i1
#
            allocate Liibb(i,i1,*,b)
#
            DO mu
#
               REQUEST Viixb(i,i1,mu,b) i
#
               DO b1
#
                  Tiibb(i,i1,b1,b)  = Viixb(i,i1,mu,b)*cb(mu,b1)
                  Liibb(i,i1,b1,b) += Tiibb(i,i1,b1,b)
#
               ENDDO b1
```

```
#
            ENDDO mu
#
            DO b1
#
               PREPARE Viibb(i,i1,b1,b) = Liibb(i,i1,b1,b)
#
            ENDDO b1
#
            deallocate Liibb(i,i1,*,b)
#
      ENDPARDO b, i, i1
#
      execute sip_barrier
      execute server_barrier
#     DISCARD Viixb
#
#     ------------------------------------------------------------------------
#
      ENDPROC TRAN_IIBB
#
#     ------------------------------------------------------------------------
#
#     ------------------------------------------------------------------------
#
      PROC TRAN_PIQJ
#
#     ------------------------------------------------------------------------
#
      execute sip_barrier
#
      PARDO mu, i, j
#
            allocate Lxiqj(mu,i,*,j)
#
            DO nu
#
               REQUEST Vxixj(mu,i,nu,j) i
#
               DO q
#
                  Txiqj(mu,i,q,j)  = Vxixj(mu,i,nu,j)*cb(nu,q)
                  Lxiqj(mu,i,q,j) += Txiqj(mu,i,q,j)
#
               ENDDO q
#
            ENDDO nu
#
            DO q
#
               PREPARE Vxiqj(mu,i,q,j) = Lxiqj(mu,i,q,j)
#
            ENDDO q
#
            deallocate Lxiqj(mu,i,*,j)
#
      ENDPARDO mu, i, j
```

```
#
     execute sip_barrier
     execute server_barrier
#    DISCARD Vxixj
#
     PARDO q, i, j
#
           allocate Lpiqj(*,i,q,j)
#
           DO mu
#
              REQUEST Vxiqj(mu,i,q,j) i
#
              DO p
#
                 Tpiqj(p,i,q,j)  = Vxiqj(mu,i,q,j)*ca(mu,p)
                 Lpiqj(p,i,q,j) += Tpiqj(p,i,q,j)
#
              ENDDO p
#
           ENDDO mu
#
           DO p
#
              PREPARE Vpiqj(p,i,q,j) = Lpiqj(p,i,q,j)
#
           ENDDO p
#
           deallocate Lpiqj(*,i,q,j)
#
     ENDPARDO q, i, j
#
     execute sip_barrier
     execute server_barrier
#    DISCARD Vxiqj
#
#    ------------------------------------------------------------------------
#
     ENDPROC TRAN_PIQJ
#
#    ------------------------------------------------------------------------
#
#    ------------------------------------------------------------------------
#
     PROC TRAN_PPPP
#
#    ------------------------------------------------------------------------
#
          CALL TRAN_XXII
          CALL TRAN_PIPI
          CALL TRAN_AAII
          CALL TRAN_IAAI
#
          CALL TRAN_QJQJ
          CALL TRAN_BBJJ
          CALL TRAN_JBBJ
#
```

```
          CALL TRAN_PIQJ
          CALL TRAN_AAJJ
          CALL TRAN_IABJ
#
          CALL TRAN_BBII
#
#    ------------------------------------------------------------------------
#
     ENDPROC TRAN_PPPP
#
#    ------------------------------------------------------------------------
#
#    ------------------------------------------------------------------------
#
     PROC TRAN_UHF
#
         CALL TRAN_PPPP
         CALL ENERGY
#
     ENDPROC TRAN_UHF
#
#    ------------------------------------------------------------------------
#
# ------------------------------------------------------------------------
--
#
      PROC WHFDENS
#     ------------
#
      PARDO mu, nu, i
#
            Ixi(nu,i) = 0.0
#
            DO i1
#
               I1xi(nu,i) = ca(nu,i1)*fock_a(i,i1)
               Ixi(nu,i) -= I1xi(nu,i)
#
            ENDDO i1
#
            Tpq(mu,nu)       = ca(mu,i)*Ixi(nu,i)
            PUT Whfa(mu,nu) += Tpq(mu,nu)
#
      ENDPARDO mu, nu, i
#
      PARDO mu, nu, j
#
            Ixj(nu,j) = 0.0
#
            DO j1
#
               I1xj(nu,j) = cb(nu,j1)*fock_b(j,j1)
               Ixj(nu,j) -= I1xj(nu,j)
#
            ENDDO j1
#
            Tpq(mu,nu)       = cb(mu,j)*Ixj(nu,j)
```

```
            PUT Whfb(mu,nu) += Tpq(mu,nu)
#
      ENDPARDO mu, nu, j
      execute sip_barrier
#
      ENDPROC WHFDENS
#     --------------
#
# ------------------------------------------------------------------------
--
#
      PROC HFDENS
#     -----------
#
      PARDO mu, nu, i
#
            Txi(nu,i)         = ca(nu,i)
            Tpq(mu,nu)        = ca(mu,i)*Txi(nu,i)
            PUT Dhfa(mu,nu) += Tpq(mu,nu)
#
      ENDPARDO mu, nu, i
#
      PARDO mu, nu, j
#
            Txj(nu,j)         = cb(nu,j)
            Tpq(mu,nu)        = cb(mu,j)*Txj(nu,j)
            PUT Dhfb(mu,nu) += Tpq(mu,nu)
#
      ENDPARDO mu, nu, j
#
      ENDPROC HFDENS
#     --------------
#
      PROC D1TRANS
#     ------------
#
# Contract with the derivative integrals
# --------------------------------------
#
      PARDO mu, nu
#
            GET             P2A_ao(mu,nu)
            GET             P2B_ao(mu,nu)
            GET             DHFA(mu,nu)
            GET             DHFB(mu,nu)
#
            Tpq(mu,nu)    = DHFA(mu,nu)
            Tpq(mu,nu)   += DHFB(mu,nu)
            Tpq(mu,nu)   += P2A_ao(mu,nu)
            Tpq(mu,nu)   += P2B_ao(mu,nu)
#
            EXECUTE HCONT1 Tpq(mu,nu)
#
      ENDPARDO mu, nu
#
# ------------------------------------------------------------------------
--
```

```
#
      ENDPROC D1TRANS
#     ---------------
#
# -------------------------------------------------------------------------
--
#
      PROC S1TRANS
#     ------------
#
# Contract with the derivative integrals
# --------------------------------------
#
      PARDO mu, nu
#
            GET            W2_ao(mu,nu)
            GET            WHFa(mu,nu)
            GET            WHFb(mu,nu)
#
            Tpq(mu,nu)    = W2_ao(mu,nu)
            Tpq(mu,nu)   += WHFa(mu,nu)
            Tpq(mu,nu)   += WHFb(mu,nu)
#
            EXECUTE SCONT1 Tpq(mu,nu)
#
      ENDPARDO mu, nu
#
# -------------------------------------------------------------------------
--
#
      ENDPROC S1TRANS
#     ---------------
#
# -------------------------------------------------------------------------
#
# The following procedure computes the contribution to Lai which
# depends on the VVVO integrals directly.
#
      PROC LAIAO1
#     -----------
#
#     Zero-out half backtransform vvoo integral arrays.
#     -------------------------------------------------
#
      execute sip_barrier
#
      PARDO mu, nu, i, i1
            txixi(mu,i,nu,i1)         = 0.0
            PREPARE Vxixi(mu,i,nu,i1) = txixi(mu,i,nu,i1)
      ENDPARDO mu, nu, i, i1
#
      PARDO mu, nu, i, j
            txixj(mu,i,nu,j)         = 0.0
            PREPARE Vxixj(mu,i,nu,j) = txixj(mu,i,nu,j)
      ENDPARDO mu, nu, i, j
#
      PARDO mu, nu, j, j1
```

```
            txjxj(mu,j,nu,j1)         = 0.0
            PREPARE Vxjxj(mu,j,nu,j1) = txjxj(mu,j,nu,j1)
      ENDPARDO mu, nu, j, j1
#
      execute server_barrier
#
#     First half backtransform vvoo integrals
#     ---------------------------------------
#
#     AAAA spin case
#     --------------
#
      PARDO a1, i, i1
#
            allocate Lxiai(*,i,a1,i1)
#
            DO a
#
               REQUEST                    VSpipi(a,i,a1,i1) i
               Tpipi(a,i,a1,i1)         = VSpipi(a,i,a1,i1)
               execute energy_denominator Tpipi(a,i,a1,i1)
#
               DO mu
#
                  Txipi(mu,i,a1,i1)  = Tpipi(a,i,a1,i1)*ca(mu,a)
                  Lxiai(mu,i,a1,i1) += Txipi(mu,i,a1,i1)
#
               ENDDO mu
#
            ENDDO a
#
            DO mu
#
               PREPARE Vxipi(mu,i,a1,i1) = Lxiai(mu,i,a1,i1)
#
            ENDDO mu
#
            deallocate Lxiai(*,i,a1,i1)
#
      ENDPARDO a1, i, i1
#
      execute sip_barrier
      execute server_barrier
#
      PARDO mu, i, i1
#
            allocate L1xixi(mu,i,*,i1)
#
            DO a1
#
               REQUEST Vxipi(mu,i,a1,i1) i
#
               DO nu
#
                  Txixi(mu,i,nu,i1)   = Vxipi(mu,i,a1,i1)*La(a1,nu) #
ca(nu,a1)
                  L1xixi(mu,i,nu,i1) += Txixi(mu,i,nu,i1)
```

```
#
               ENDDO nu
#
            ENDDO a1
#
            DO nu
#
               PREPARE Vxixi(mu,i,nu,i1) = L1xixi(mu,i,nu,i1)
#
            ENDDO nu
#
            deallocate L1xixi(mu,i,*,i1)
#
      ENDPARDO mu, i, i1
#
#     BBBB spin case
#     --------------
#
      PARDO b1, j, j1
#
            allocate Lxjbj(*,j,b1,j1)
#
            DO b
#
               REQUEST                      VSqjqj(b,j,b1,j1) j
               Tqjqj(b,j,b1,j1)          = VSqjqj(b,j,b1,j1)
               execute energy_denominator Tqjqj(b,j,b1,j1)
#
               DO mu
#
                  Txjqj(mu,j,b1,j1)  = Tqjqj(b,j,b1,j1)*cb(mu,b)
                  Lxjbj(mu,j,b1,j1) += Txjqj(mu,j,b1,j1)
#
               ENDDO mu
#
            ENDDO b
#
            DO mu
#
               PREPARE Vxjqj(mu,j,b1,j1) = Lxjbj(mu,j,b1,j1)
#
            ENDDO mu
#
            deallocate Lxjbj(*,j,b1,j1)
#
      ENDPARDO b1, j, j1
#
      execute sip_barrier
      execute server_barrier
#
      PARDO mu, j, j1
#
            allocate L1xjxj(mu,j,*,j1)
#
            DO b1
#
               REQUEST Vxjqj(mu,j,b1,j1) j
```

```
#
               DO nu
#
                  Txjxj(mu,j,nu,j1)   = Vxjqj(mu,j,b1,j1)*Lb(b1,nu) #
cb(nu,b1)
                  L1xjxj(mu,j,nu,j1) += Txjxj(mu,j,nu,j1)
#
               ENDDO nu
#
            ENDDO b1
#
            DO nu
#
               PREPARE Vxjxj(mu,j,nu,j1) = L1xjxj(mu,j,nu,j1)
#
            ENDDO nu
#
            deallocate L1xjxj(mu,j,*,j1)
#
      ENDPARDO mu, j, j1
#
#     AABB spin case
#     --------------
#
      PARDO b, i, j
#
            allocate Lxibj(*,i,b,j)
#
            DO a
#
               REQUEST                    Vpiqj(a,i,b,j) i
               Tpiqj(a,i,b,j)           = Vpiqj(a,i,b,j)
               execute energy_denominator Tpiqj(a,i,b,j)
#
               DO mu
#
                  Txiqj(mu,i,b,j)  = Tpiqj(a,i,b,j)*ca(mu,a)
                  Lxibj(mu,i,b,j) += Txiqj(mu,i,b,j)
#
               ENDDO mu
#
            ENDDO a
#
            DO mu
#
               PREPARE Vxiqj(mu,i,b,j) = Lxibj(mu,i,b,j)
#
            ENDDO mu
#
            deallocate Lxibj(*,i,b,j)
#
      ENDPARDO b, i, j
#
      execute sip_barrier
      execute server_barrier
#
      PARDO mu, i, j
```

```
#
            allocate L1xixj(mu,i,*,j)
#
            DO b
#
               REQUEST Vxiqj(mu,i,b,j) i
#
               DO nu
#
                  Txixj(mu,i,nu,j)   = Vxiqj(mu,i,b,j)*Lb(b,nu) # cb(nu,b)
                  L1xixj(mu,i,nu,j) += Txixj(mu,i,nu,j)
#
               ENDDO nu
#
            ENDDO b
#
            DO nu
#
               PREPARE Vxixj(mu,i,nu,j) = L1xixj(mu,i,nu,j)
#
            ENDDO nu
#
            deallocate L1xixj(mu,i,*,j)
#
      ENDPARDO mu, i, j
#
      create Lxi
      create Lxj
      execute sip_barrier
      execute server_barrier
#
#     Perform intermediate contraction to get contribution to Lxi and Lxj
#     -------------------------------------------------------------------
#
      PARDO mu, lambda, sigma
#
            allocate xa(lambda,*,sigma,mu)
            allocate xb(lambda,*,sigma,mu)
            allocate xTa(sigma,mu,lambda,*)
            allocate xTb(sigma,mu,lambda,*)
#
#           Form Xa
#           -------
#
            DO i
#
               T1xixx(sigma,i,lambda,mu) = 0.0
#
               DO i1
#
                  REQUEST                     Vxixi(lambda,i,sigma,i1) i
                  REQUEST                     Vxixi(sigma,i,lambda,i1) i
#
                  Txixi(lambda,i,sigma,i1)  = Vxixi(lambda,i,sigma,i1)
                  T1xixi(lambda,i,sigma,i1) = Vxixi(sigma,i,lambda,i1)
                  Txixi(lambda,i,sigma,i1) -= T1xixi(lambda,i,sigma,i1)
#
```

```
                  Txixx(lambda,i,sigma,mu)  =
Txixi(lambda,i,sigma,i1)*La(i1,mu) # ca(mu,i1)
                  Txixx(lambda,i,sigma,mu) *= 0.5
                  xa(lambda,i,sigma,mu)    += Txixx(lambda,i,sigma,mu)
#
               ENDDO i1
#
               DO j
#
                  REQUEST                      Vxixj(sigma,i,lambda,j) i
                 #Txixx(lambda,i,sigma,mu)  =
Vxixj(sigma,i,lambda,j)*cb(mu,j)
                  Txixx(sigma,i,lambda,mu)  =
Vxixj(sigma,i,lambda,j)*Lb(j,mu)
                  T1xixx(sigma,i,lambda,mu)-= Txixx(sigma,i,lambda,mu)
#
               ENDDO j
#
               Txixx(lambda,i,sigma,mu) = T1xixx(sigma,i,lambda,mu)
               xa(lambda,i,sigma,mu)   += Txixx(lambda,i,sigma,mu)
               xTa(sigma,mu,lambda,i)   = xa(lambda,i,sigma,mu)
#
            ENDDO i
#
#           Form Xb
#           -------
#
            DO j
#
               T1xxxj(lambda,mu,sigma,j) = 0.0
#
               DO j1
#
                  REQUEST                     Vxjxj(lambda,j,sigma,j1) j
                  REQUEST                     Vxjxj(sigma,j,lambda,j1) j
#
                  Txjxj(lambda,j,sigma,j1)  = Vxjxj(lambda,j,sigma,j1)
                  T1xjxj(lambda,j,sigma,j1) = Vxjxj(sigma,j,lambda,j1)
                  Txjxj(lambda,j,sigma,j1) -= T1xjxj(lambda,j,sigma,j1)
#
                  Txjxx(lambda,j,sigma,mu)  =
Txjxj(lambda,j,sigma,j1)*Lb(j1,mu) # cb(mu,j1)
                  Txjxx(lambda,j,sigma,mu) *= 0.5
                  xb(lambda,j,sigma,mu)    += Txjxx(lambda,j,sigma,mu)
#
               ENDDO j1
#
               DO i
#
                  REQUEST                      Vxixj(lambda,i,sigma,j) i
                 #Txjxx(lambda,j,sigma,mu)  =
Vxixj(lambda,i,sigma,j)*ca(mu,i)
                  Txxxj(lambda,mu,sigma,j)  =
Vxixj(lambda,i,sigma,j)*La(i,mu)
                  T1xxxj(lambda,mu,sigma,j)-= Txxxj(lambda,mu,sigma,j)
#
               ENDDO i
```

```
#
               Txjxx(lambda,j,sigma,mu) = T1xxxj(lambda,mu,sigma,j)
               xb(lambda,j,sigma,mu)   += Txjxx(lambda,j,sigma,mu)
               xTb(sigma,mu,lambda,j)   = xb(lambda,j,sigma,mu)
#
            ENDDO j
#
            DO nu
#
               compute_integrals aoint(nu,sigma,mu,lambda)
#
#              Finish Lxi
#              ----------
#
               DO i
#
                 #Ixi(nu,i)      =
xa(lambda,i,sigma,mu)*aoint(mu,lambda,nu,sigma)
                  Ixi(nu,i)      =
aoint(nu,sigma,mu,lambda)*xTa(sigma,mu,lambda,i)
                  PUT Lxi(nu,i) += Ixi(nu,i)
#
               ENDDO i
#
#              Finish Lxj
#              ----------
#
               DO j
#
                 #Ixj(nu,j)      =
xb(lambda,j,sigma,mu)*aoint(mu,lambda,nu,sigma)
                  Ixj(nu,j)      =
aoint(nu,sigma,mu,lambda)*xTb(sigma,mu,lambda,j)
                  PUT Lxj(nu,j) += Ixj(nu,j)
#
               ENDDO j
#
#              Start third-term
#              ----------------
#
               GET               Paa_ao(sigma,nu)
               GET               Pbb_ao(sigma,nu)
               GET               Paa_ao(sigma,mu)
               GET               Pbb_ao(sigma,mu)
#
               I1xx(sigma,nu)  = Paa_ao(sigma,nu)
               I1xx(sigma,nu) += Pbb_ao(sigma,nu)
               Ixx(mu,lambda)  = aoint(nu,sigma,mu,lambda)*I1xx(sigma,nu)
               I1xx(nu,lambda) = aoint(nu,sigma,mu,lambda)*Paa_ao(sigma,mu)
#
               DO i
#
                  Ixi(lambda,i)      = Ixx(mu,lambda)*ca(mu,i)
                  I1xi(lambda,i)     = I1xx(nu,lambda)*ca(nu,i)
                  Ixi(lambda,i)     -= I1xi(lambda,i)
#
                  PUT Yxi(lambda,i) += Ixi(lambda,i)
```

```
#
               ENDDO i
#
               I1xx(sigma,nu)  = Pbb_ao(sigma,nu)
               I1xx(sigma,nu) += Paa_ao(sigma,nu)
               Ixx(mu,lambda)  = aoint(nu,sigma,mu,lambda)*I1xx(sigma,nu)
               I1xx(nu,lambda) = aoint(nu,sigma,mu,lambda)*Pbb_ao(sigma,mu)
#
               DO j
#
                  Ixj(lambda,j)      = Ixx(mu,lambda)*cb(mu,j)
                  I1xj(lambda,j)     = I1xx(nu,lambda)*cb(nu,j)
                  Ixj(lambda,j)     -= I1xj(lambda,j)
#
                  PUT Yxj(lambda,j) += Ixj(lambda,j)
#
               ENDDO j
#
            ENDDO nu
#
            deallocate xa(lambda,*,sigma,mu)
            deallocate xb(lambda,*,sigma,mu)
            deallocate xTa(sigma,mu,lambda,*)
            deallocate xTb(sigma,mu,lambda,*)
#
      ENDPARDO mu, lambda, sigma
      execute sip_barrier
#
      PARDO lambda, a, i
#
            GET                Yxi(lambda,i)
            Tai(a,i)         = ca(lambda,a)*Yxi(lambda,i)
            Tai(a,i)        *= -1.0
            PUT Lai_aa(a,i) += tai(a,i)
#
      ENDPARDO lambda, a, i
#
      PARDO lambda, b, j
#
            GET                Yxj(lambda,j)
            Tbj(b,j)         = cb(lambda,b)*Yxj(lambda,j)
            Tbj(b,j)        *= -1.0
            PUT Lai_bb(b,j) += tbj(b,j)
#
      ENDPARDO lambda, b, j
#
#     Perform final transformation to get contribution to Lai
#     -------------------------------------------------------
#
      PARDO a, i, nu
#
           GET               Lxi(nu,i)
           tai(a,i)        = Lxi(nu,i)*ca(nu,a)
           PUT Lai_aa(a,i) += tai(a,i)
#
      ENDPARDO a, i, nu
#
```

```
      PARDO b, j, nu
#
           GET                Lxj(nu,j)
           tbj(b,j)         = Lxj(nu,j)*cb(nu,b)
           PUT Lai_bb(b,j) += tbj(b,j)
#
      ENDPARDO b, j, nu
#
#     Done direct contribution of Lai and Lbj
#     ---------------------------------------
#
      ENDPROC LAIAO1
#     --------------
#
# ------------------------------------------------------------------------
#
# ------------------------------------------------------------------------
#
# Procedure DPQRSSEP computes the seperable part of the two-particle
# 'density' matrix.
#
      PROC DPQRSSEP
#     -------------
#
#          Get 1-particle pieces
#          ---------------------
#
           GET DHFa(mu,lambda)
           GET DHFa(mu,sigma)
           GET DHFa(mu,nu)
           GET DHFa(nu,sigma)
           GET DHFa(nu,lambda)
           GET DHFa(sigma,lambda)
#
           GET DHFb(mu,lambda)
           GET DHFb(mu,sigma)
           GET DHFb(mu,nu)
           GET DHFb(nu,sigma)
           GET DHFb(nu,lambda)
           GET DHFb(sigma,lambda)
#
           GET P2A_ao(mu,lambda)
           GET P2A_ao(mu,sigma)
           GET P2A_ao(mu,nu)
           GET P2A_ao(nu,lambda)
           GET P2A_ao(nu,sigma)
           GET P2A_ao(sigma,lambda)
#
           GET P2B_ao(mu,lambda)
           GET P2B_ao(mu,nu)
           GET P2B_ao(mu,sigma)
           GET P2B_ao(nu,lambda)
           GET P2B_ao(nu,sigma)
           GET P2B_ao(sigma,lambda)
#
#          HF only
#          -------
```

```
            Txx(nu,sigma)               = DHFa(nu,sigma)
            Txx(nu,sigma)              += DHFb(nu,sigma)
            Txx(nu,sigma)              += P2A_ao(nu,sigma)
            Txx(nu,sigma)              += P2B_ao(nu,sigma)
            T1xx(mu,lambda)             = DHFa(mu,lambda)
            T1xx(mu,lambda)            += DHFb(mu,lambda)

            Txxxx(mu,lambda,nu,sigma)  = T1xx(mu,lambda)^Txx(nu,sigma)
#
            Txx(nu,lambda)              = DHFa(nu,lambda)
            Txx(nu,lambda)             += P2A_ao(nu,lambda)
            T1xxxx(mu,lambda,nu,sigma) = DHFa(mu,sigma)^Txx(nu,lambda)
            T1xxxx(mu,lambda,nu,sigma)*= 0.5
            Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
            Txx(nu,lambda)              = DHFb(nu,lambda)
            Txx(nu,lambda)             += P2B_ao(nu,lambda)
            T1xxxx(mu,lambda,nu,sigma) = DHFb(mu,sigma)^Txx(nu,lambda)
            T1xxxx(mu,lambda,nu,sigma)*= 0.5
            Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
            Txx(sigma,lambda)           = DHFa(sigma,lambda)
            Txx(sigma,lambda)          += P2A_ao(sigma,lambda)
            T1xxxx(mu,lambda,nu,sigma) = DHFa(mu,nu)^Txx(sigma,lambda)
            T1xxxx(mu,lambda,nu,sigma)*= 0.5
            Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
            Txx(sigma,lambda)           = DHFb(sigma,lambda)
            Txx(sigma,lambda)          += P2B_ao(sigma,lambda)
            T1xxxx(mu,lambda,nu,sigma) = DHFb(mu,nu)^Txx(sigma,lambda)
            T1xxxx(mu,lambda,nu,sigma)*= 0.5
            Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
#           Correlation Only
#           ----------------
#
            Txx(nu,sigma)               = DHFA(nu,sigma)
            Txx(nu,sigma)              += DHFB(nu,sigma)
            T1xxxx(mu,lambda,nu,sigma) = P2A_ao(mu,lambda)^Txx(nu,sigma)
            Txxxx(mu,lambda,nu,sigma) += T1xxxx(mu,lambda,nu,sigma)
#
            T1xxxx(mu,lambda,nu,sigma) = P2B_ao(mu,lambda)^Txx(nu,sigma)
            Txxxx(mu,lambda,nu,sigma) += T1xxxx(mu,lambda,nu,sigma)
#
            T1xxxx(mu,lambda,nu,sigma) = P2A_ao(mu,sigma)^DHFA(nu,lambda)
            T1xxxx(mu,lambda,nu,sigma)*= 0.5
            Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
            T1xxxx(mu,lambda,nu,sigma) = P2A_ao(mu,nu)^DHFA(sigma,lambda)
            T1xxxx(mu,lambda,nu,sigma)*= 0.5
            Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
            T1xxxx(mu,lambda,nu,sigma) = P2B_ao(mu,sigma)^DHFB(nu,lambda)
            T1xxxx(mu,lambda,nu,sigma)*= 0.5
            Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
            T1xxxx(mu,lambda,nu,sigma) = P2B_ao(mu,nu)^DHFB(sigma,lambda)
```

```
            T1xxxx(mu,lambda,nu,sigma)*= 0.5
            Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
            Txxxx(mu,lambda,nu,sigma) *= 0.5
#
      ENDPROC DPQRSSEP
#     ----------------
#
# ---------------------------------------------------------------------------
#
# ---------------------------------------------------------------------------
--
#
      PROC D2TRANS
#     ------------
#
      allocate LDhfa(*,*)
      allocate LDhfb(*,*)
      allocate LP2A_ao(*,*)
      allocate LP2B_ao(*,*)
      execute sip_barrier
      DO mu
      DO nu
         GET              DHFa(mu,nu)
         GET              P2A_ao(mu,nu)
         GET              DHFb(mu,nu)
         GET              P2B_ao(mu,nu)
         LDhfa(mu,nu)   = DHFa(mu,nu)
         LDhfb(mu,nu)   = DHFb(mu,nu)
         LP2A_ao(mu,nu) = P2A_ao(mu,nu)
         LP2B_ao(mu,nu) = P2B_ao(mu,nu)
      ENDDO nu
      ENDDO mu
      execute sip_barrier
#
      PARDO mu, nu
#
           DO lambda
           DO sigma
#
              IF mu < lambda
              IF nu < sigma
#
                 D2(mu,lambda,nu,sigma) = 0.0
                 D3(mu,lambda,sigma,nu) = 0.0
#
                 DO i1
                    TAxxxi(mu,lambda,nu,i1)     = 0.0
                    TBxxxi(mu,lambda,sigma,i1)  = 0.0
                    DO i
                       REQUEST Vxixi(mu,i,nu,i1)        i
                       REQUEST Vxixi(lambda,i,nu,i1)    i
                       REQUEST Vxixi(mu,i,sigma,i1)     i
                       REQUEST Vxixi(lambda,i,sigma,i1) i
#
                       T1xxxi(mu,lambda,nu,i1)    =
Vxixi(mu,i,nu,i1)*La(i,lambda) # ca(lambda,i)
```

```
                        T2xxxi(mu,lambda,nu,i1)    =
Vxixi(lambda,i,nu,i1)*ca(mu,i)
#
                        TAxxxi(mu,lambda,nu,i1)   += T1xxxi(mu,lambda,nu,i1)
                        TAxxxi(mu,lambda,nu,i1)   += T2xxxi(mu,lambda,nu,i1) #
-
#
                        T3xxxi(mu,lambda,sigma,i1) =
Vxixi(mu,i,sigma,i1)*La(i,lambda) # ca(lambda,i)
                        T4xxxi(mu,lambda,sigma,i1) =
Vxixi(lambda,i,sigma,i1)*ca(mu,i)
#
                        TBxxxi(mu,lambda,sigma,i1)+=
T4xxxi(mu,lambda,sigma,i1)
                        TBxxxi(mu,lambda,sigma,i1)+=
T3xxxi(mu,lambda,sigma,i1) # -
#
                     ENDDO i
#
                    #Txxxx(mu,lambda,nu,sigma)  =
TBxxxi(mu,lambda,sigma,i1)*ca(nu,i1)
                    #D2(mu,lambda,nu,sigma)    += Txxxx(mu,lambda,nu,sigma)
                     Txxxx(mu,lambda,sigma,nu)  =
TBxxxi(mu,lambda,sigma,i1)*La(i1,nu)
                     D3(mu,lambda,sigma,nu)    += Txxxx(mu,lambda,sigma,nu)
#
                     T1xxxx(mu,lambda,nu,sigma) =
TAxxxi(mu,lambda,nu,i1)*La(i1,sigma) # ca(sigma,i1)
                     D2(mu,lambda,nu,sigma)    += T1xxxx(mu,lambda,nu,sigma)
                  ENDDO i1
#
                  DO j1
                     TAxxxj(mu,lambda,nu,j1)     = 0.0
                     TBxxxj(mu,lambda,sigma,j1)  = 0.0
                     DO j
                        REQUEST Vxjxj(mu,j,nu,j1)         j
                        REQUEST Vxjxj(lambda,j,nu,j1)     j
                        REQUEST Vxjxj(mu,j,sigma,j1)      j
                        REQUEST Vxjxj(lambda,j,sigma,j1)  j
#
                        T1xxxj(mu,lambda,nu,j1)    =
Vxjxj(mu,j,nu,j1)*Lb(j,lambda) # cb(lambda,j)
                        T2xxxj(mu,lambda,nu,j1)    =
Vxjxj(lambda,j,nu,j1)*cb(mu,j)
#
                        TAxxxj(mu,lambda,nu,j1)   += T1xxxj(mu,lambda,nu,j1)
                        TAxxxj(mu,lambda,nu,j1)   += T2xxxj(mu,lambda,nu,j1) #
-
#
                        T3xxxj(mu,lambda,sigma,j1) =
Vxjxj(mu,j,sigma,j1)*Lb(j,lambda) # cb(lambda,j)
                        T4xxxj(mu,lambda,sigma,j1) =
Vxjxj(lambda,j,sigma,j1)*cb(mu,j)
#
                        TBxxxj(mu,lambda,sigma,j1)+=
T4xxxj(mu,lambda,sigma,j1)
```

```
                        TBxxxj(mu,lambda,sigma,j1)+=
T3xxxj(mu,lambda,sigma,j1) # -
#
                     ENDDO j
#
                    #Txxxx(mu,lambda,nu,sigma)  =
TBxxxj(mu,lambda,sigma,j1)*cb(nu,j1)
                    #D2(mu,lambda,nu,sigma)    += Txxxx(mu,lambda,nu,sigma)
                     Txxxx(mu,lambda,sigma,nu)  =
TBxxxj(mu,lambda,sigma,j1)*Lb(j1,nu)
                     D3(mu,lambda,sigma,nu)    += Txxxx(mu,lambda,sigma,nu)
#
                     T1xxxx(mu,lambda,nu,sigma) =
TAxxxj(mu,lambda,nu,j1)*Lb(j1,sigma) # cb(sigma,j1)
                     D2(mu,lambda,nu,sigma)    += T1xxxx(mu,lambda,nu,sigma)
                  ENDDO j1
#
                  DO j
                     TAxxxj(mu,lambda,nu,j)     = 0.0
                     TBxxxj(mu,lambda,sigma,j)  = 0.0
                     DO i
                        REQUEST Vxixj(mu,i,nu,j)        j
                        REQUEST Vxixj(lambda,i,nu,j)    j
                        REQUEST Vxixj(mu,i,sigma,j)     j
                        REQUEST Vxixj(lambda,i,sigma,j) j
#
                        T1xxxj(mu,lambda,nu,j)    =
Vxixj(mu,i,nu,j)*La(i,lambda) # ca(lambda,i)
                        T2xxxj(mu,lambda,nu,j)    =
Vxixj(lambda,i,nu,j)*ca(mu,i)
#
                        TAxxxj(mu,lambda,nu,j)   += T1xxxj(mu,lambda,nu,j)
                        TAxxxj(mu,lambda,nu,j)   += T2xxxj(mu,lambda,nu,j) # -
#
                        T3xxxj(mu,lambda,sigma,j) =
Vxixj(mu,i,sigma,j)*La(i,lambda) # ca(lambda,i)
                        T4xxxj(mu,lambda,sigma,j) =
Vxixj(lambda,i,sigma,j)*ca(mu,i)
#
                        TBxxxj(mu,lambda,sigma,j)+= T4xxxj(mu,lambda,sigma,j)
                        TBxxxj(mu,lambda,sigma,j)+= T3xxxj(mu,lambda,sigma,j)
# -
#
                     ENDDO i
#
                    #Txxxx(mu,lambda,nu,sigma)  =
TBxxxj(mu,lambda,sigma,j)*cb(nu,j)
                    #D2(mu,lambda,nu,sigma)    += Txxxx(mu,lambda,nu,sigma)
                    #D2(mu,lambda,nu,sigma)    += Txxxx(mu,lambda,nu,sigma)
                     Txxxx(mu,lambda,sigma,nu)  =
TBxxxj(mu,lambda,sigma,j)*Lb(j,nu)
                     Txxxx(mu,lambda,sigma,nu) *= 2.0
                     D3(mu,lambda,sigma,nu)    += Txxxx(mu,lambda,sigma,nu)
#
                     T1xxxx(mu,lambda,nu,sigma) =
TAxxxj(mu,lambda,nu,j)*Lb(j,sigma) # cb(sigma,j)
                     T1xxxx(mu,lambda,nu,sigma)*= 2.0
```

```
                   D2(mu,lambda,nu,sigma)    += T1xxxx(mu,lambda,nu,sigma)
                  #D2(mu,lambda,nu,sigma)    += Txxxx(mu,lambda,nu,sigma)
                ENDDO j
#
#               Get the separable part
#               ----------------------
#
#               Get 1-particle pieces
#               ---------------------
#
#               HF only
#               -------
                Txx(nu,sigma)              = LDHFa(nu,sigma)
                Txx(nu,sigma)             += LDHFb(nu,sigma)
                Txx(nu,sigma)             += LP2A_ao(nu,sigma)
                Txx(nu,sigma)             += LP2B_ao(nu,sigma)
                T1xx(mu,lambda)            = LDHFa(mu,lambda)
                T1xx(mu,lambda)           += LDHFb(mu,lambda)

                Txxxx(mu,lambda,nu,sigma)  = T1xx(mu,lambda)^Txx(nu,sigma)
#
                T2xx(nu,lambda)             = LDHFa(nu,lambda)
                T2xx(nu,lambda)            += LP2A_ao(nu,lambda)
                T1xxxx(mu,lambda,nu,sigma) = LDHFa(mu,sigma)^T2xx(nu,lambda)
                T1xxxx(mu,lambda,nu,sigma)*= 0.5
                Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                T3xx(nu,lambda)             = LDHFb(nu,lambda)
                T3xx(nu,lambda)            += LP2B_ao(nu,lambda)
                T1xxxx(mu,lambda,nu,sigma) = LDHFb(mu,sigma)^T3xx(nu,lambda)
                T1xxxx(mu,lambda,nu,sigma)*= 0.5
                Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                T4xx(sigma,lambda)          = LDHFa(sigma,lambda)
                T4xx(sigma,lambda)         += LP2A_ao(sigma,lambda)
                T1xxxx(mu,lambda,nu,sigma) = LDHFa(mu,nu)^T4xx(sigma,lambda)
                T1xxxx(mu,lambda,nu,sigma)*= 0.5
                Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                T5xx(sigma,lambda)          = LDHFb(sigma,lambda)
                T5xx(sigma,lambda)         += LP2B_ao(sigma,lambda)
                T1xxxx(mu,lambda,nu,sigma) = LDHFb(mu,nu)^T5xx(sigma,lambda)
                T1xxxx(mu,lambda,nu,sigma)*= 0.5
                Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
#               Correlation Only
#               ----------------
#
                T6xx(nu,sigma)              = LDHFA(nu,sigma)
                T6xx(nu,sigma)             += LDHFB(nu,sigma)
                T1xxxx(mu,lambda,nu,sigma) =
LP2A_ao(mu,lambda)^T6xx(nu,sigma)
                Txxxx(mu,lambda,nu,sigma) += T1xxxx(mu,lambda,nu,sigma)
#
                T1xxxx(mu,lambda,nu,sigma) =
LP2B_ao(mu,lambda)^T6xx(nu,sigma)
                Txxxx(mu,lambda,nu,sigma) += T1xxxx(mu,lambda,nu,sigma)
```

```
#
                  T1xxxx(mu,lambda,nu,sigma) =
LP2A_ao(mu,sigma)^LDHFA(nu,lambda)
                  T1xxxx(mu,lambda,nu,sigma)*= 0.5
                  Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                  T1xxxx(mu,lambda,nu,sigma) =
LP2A_ao(mu,nu)^LDHFA(sigma,lambda)
                  T1xxxx(mu,lambda,nu,sigma)*= 0.5
                  Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                  T1xxxx(mu,lambda,nu,sigma) =
LP2B_ao(mu,sigma)^LDHFB(nu,lambda)
                  T1xxxx(mu,lambda,nu,sigma)*= 0.5
                  Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                  T1xxxx(mu,lambda,nu,sigma) =
LP2B_ao(mu,nu)^LDHFB(sigma,lambda)
                  T1xxxx(mu,lambda,nu,sigma)*= 0.5
                  Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                  Txxxx(mu,lambda,nu,sigma) *= 0.5
                  TSxxxx(mu,lambda,nu,sigma) = Txxxx(mu,lambda,nu,sigma)
                  TSxxxx(mu,lambda,nu,sigma) *= 4.0
#
#                 Add the the nonseparable part
#                 -----------------------------
                  Txxxx(mu,lambda,nu,sigma)   = D3(mu,lambda,sigma,nu)
                  TSxxxx(mu,lambda,nu,sigma) += D2(mu,lambda,nu,sigma)
                  TSxxxx(mu,lambda,nu,sigma) += txxxx(mu,lambda,nu,sigma)
#
#                 Set up integrals
#                 ----------------
                  execute der_int_setup dx1(mu,lambda,nu,sigma)
                  execute der_int_setup dy1(mu,lambda,nu,sigma)
                  execute der_int_setup dz1(mu,lambda,nu,sigma)
                  execute der_int_setup dx2(mu,lambda,nu,sigma)
                  execute der_int_setup dy2(mu,lambda,nu,sigma)
                  execute der_int_setup dz2(mu,lambda,nu,sigma)
                  execute der_int_setup dx3(mu,lambda,nu,sigma)
                  execute der_int_setup dy3(mu,lambda,nu,sigma)
                  execute der_int_setup dz3(mu,lambda,nu,sigma)
                  execute der_int_setup dx4(mu,lambda,nu,sigma)
                  execute der_int_setup dy4(mu,lambda,nu,sigma)
                  execute der_int_setup dz4(mu,lambda,nu,sigma)
#
#                 Compute integral block
#                 ----------------------
                  execute compute_derivative_integrals
#
#                 Contract density with integral derivatives
#                 ------------------------------------------
                  execute DCONT2 TSxxxx(mu,lambda,nu,sigma)
#
               ENDIF # nu < sigma
               ENDIF # mu < lambda
#
```

```
           ENDDO sigma
           ENDDO lambda
#
      ENDPARDO mu, nu
#
      PARDO mu, nu
#
           DO lambda
           DO sigma
#
              IF mu == lambda
              IF nu < sigma
#
                 D2(mu,lambda,nu,sigma) = 0.0
                 D3(mu,lambda,sigma,nu) = 0.0
#
                 DO i1
                    TAxxxi(mu,lambda,nu,i1)     = 0.0
                    TBxxxi(mu,lambda,sigma,i1)  = 0.0
                    DO i
                       REQUEST Vxixi(mu,i,nu,i1)    i
                       REQUEST Vxixi(mu,i,sigma,i1) i
#
                       T1xxxi(mu,lambda,nu,i1)    =
Vxixi(mu,i,nu,i1)*La(i,lambda) # ca(lambda,i)
                       TAxxxi(mu,lambda,nu,i1)   += T1xxxi(mu,lambda,nu,i1)
                       T3xxxi(mu,lambda,sigma,i1) =
Vxixi(mu,i,sigma,i1)*La(i,lambda) # ca(lambda,i)
                       TBxxxi(mu,lambda,sigma,i1)+=
T3xxxi(mu,lambda,sigma,i1) # -
#
                    ENDDO i
#
                   #Txxxx(mu,lambda,nu,sigma) =
TBxxxi(mu,lambda,sigma,i1)*ca(nu,i1)
                   #D2(mu,lambda,nu,sigma)   += Txxxx(mu,lambda,nu,sigma)
                    Txxxx(mu,lambda,sigma,nu) =
TBxxxi(mu,lambda,sigma,i1)*La(i1,nu)
                    D3(mu,lambda,sigma,nu)   += Txxxx(mu,lambda,sigma,nu)
#
                    T1xxxx(mu,lambda,nu,sigma) =
TAxxxi(mu,lambda,nu,i1)*La(i1,sigma) # ca(sigma,i1)
                    D2(mu,lambda,nu,sigma)   += T1xxxx(mu,lambda,nu,sigma)
                 ENDDO i1
#
                 DO j1
                    TAxxxj(mu,lambda,nu,j1)     = 0.0
                    TBxxxj(mu,lambda,sigma,j1)  = 0.0
                    DO j
                       REQUEST Vxjxj(mu,j,nu,j1)    j
                       REQUEST Vxjxj(mu,j,sigma,j1) j
#
                       T1xxxj(mu,lambda,nu,j1)    =
Vxjxj(mu,j,nu,j1)*Lb(j,lambda) # cb(lambda,j)
                       TAxxxj(mu,lambda,nu,j1)   += T1xxxj(mu,lambda,nu,j1)
                       T3xxxj(mu,lambda,sigma,j1) =
Vxjxj(mu,j,sigma,j1)*Lb(j,lambda) # cb(lambda,j)
```

```
                        TBxxxj(mu,lambda,sigma,j1)+=
T3xxxj(mu,lambda,sigma,j1) # -
#
                     ENDDO j
#
                    #Txxxx(mu,lambda,nu,sigma) =
TBxxxj(mu,lambda,sigma,j1)*cb(nu,j1)
                    #D2(mu,lambda,nu,sigma)   += Txxxx(mu,lambda,nu,sigma)
                     Txxxx(mu,lambda,sigma,nu) =
TBxxxj(mu,lambda,sigma,j1)*Lb(j1,nu)
                     D3(mu,lambda,sigma,nu)   += Txxxx(mu,lambda,sigma,nu)
#
                     T1xxxx(mu,lambda,nu,sigma) =
TAxxxj(mu,lambda,nu,j1)*Lb(j1,sigma) # cb(sigma,j1)
                     D2(mu,lambda,nu,sigma)   += T1xxxx(mu,lambda,nu,sigma)
                  ENDDO j1
#
                  DO j
                     TAxxxj(mu,lambda,nu,j)    = 0.0
                     TBxxxj(mu,lambda,sigma,j) = 0.0
                     DO i
                        REQUEST Vxixj(mu,i,nu,j)    j
                        REQUEST Vxixj(mu,i,sigma,j) j
#
                        T1xxxj(mu,lambda,nu,j)    =
Vxixj(mu,i,nu,j)*La(i,lambda) # ca(lambda,i)
                        TAxxxj(mu,lambda,nu,j)   += T1xxxj(mu,lambda,nu,j)
                        T3xxxj(mu,lambda,sigma,j) =
Vxixj(mu,i,sigma,j)*La(i,lambda) # ca(lambda,i)
                        TBxxxj(mu,lambda,sigma,j)+= T3xxxj(mu,lambda,sigma,j)
# -
#
                     ENDDO i
#
                    #Txxxx(mu,lambda,nu,sigma) =
TBxxxj(mu,lambda,sigma,j)*cb(nu,j)
                    #D2(mu,lambda,nu,sigma)   += Txxxx(mu,lambda,nu,sigma)
                    #D2(mu,lambda,nu,sigma)   += Txxxx(mu,lambda,nu,sigma)
                     Txxxx(mu,lambda,sigma,nu) =
TBxxxj(mu,lambda,sigma,j)*Lb(j,nu)
                     Txxxx(mu,lambda,sigma,nu)*= 2.0
                     D3(mu,lambda,sigma,nu)   += Txxxx(mu,lambda,sigma,nu)
#
                     T1xxxx(mu,lambda,nu,sigma) =
TAxxxj(mu,lambda,nu,j)*Lb(j,sigma) # cb(sigma,j)
                     T1xxxx(mu,lambda,nu,sigma)*= 2.0
                     D2(mu,lambda,nu,sigma)   += T1xxxx(mu,lambda,nu,sigma)
                    #D2(mu,lambda,nu,sigma)   += Txxxx(mu,lambda,nu,sigma)
                  ENDDO j
#
#                 Get the separable part
#                 ----------------------
#
#                 Get 1-particle pieces
#                 ---------------------
#
#                 HF only
```

```
#                 -------
                  Txx(nu,sigma)               = LDHFa(nu,sigma)
                  Txx(nu,sigma)              += LDHFb(nu,sigma)
                  Txx(nu,sigma)              += LP2A_ao(nu,sigma)
                  Txx(nu,sigma)              += LP2B_ao(nu,sigma)
                  T1xx(mu,lambda)             = LDHFa(mu,lambda)
                  T1xx(mu,lambda)            += LDHFb(mu,lambda)

                  Txxxx(mu,lambda,nu,sigma)   = T1xx(mu,lambda)^Txx(nu,sigma)
#
                  T2xx(nu,lambda)              = LDHFa(nu,lambda)
                  T2xx(nu,lambda)             += LP2A_ao(nu,lambda)
                  T1xxxx(mu,lambda,nu,sigma) = LDHFa(mu,sigma)^T2xx(nu,lambda)
                  T1xxxx(mu,lambda,nu,sigma)*= 0.5
                  Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                  T3xx(nu,lambda)              = LDHFb(nu,lambda)
                  T3xx(nu,lambda)             += LP2B_ao(nu,lambda)
                  T1xxxx(mu,lambda,nu,sigma) = LDHFb(mu,sigma)^T3xx(nu,lambda)
                  T1xxxx(mu,lambda,nu,sigma)*= 0.5
                  Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                  T4xx(sigma,lambda)           = LDHFa(sigma,lambda)
                  T4xx(sigma,lambda)          += LP2A_ao(sigma,lambda)
                  T1xxxx(mu,lambda,nu,sigma) = LDHFa(mu,nu)^T4xx(sigma,lambda)
                  T1xxxx(mu,lambda,nu,sigma)*= 0.5
                  Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                  T5xx(sigma,lambda)           = LDHFb(sigma,lambda)
                  T5xx(sigma,lambda)          += LP2B_ao(sigma,lambda)
                  T1xxxx(mu,lambda,nu,sigma) = LDHFb(mu,nu)^T5xx(sigma,lambda)
                  T1xxxx(mu,lambda,nu,sigma)*= 0.5
                  Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
#                 Correlation Only
#                 ----------------
#
                  T6xx(nu,sigma)               = LDHFA(nu,sigma)
                  T6xx(nu,sigma)              += LDHFB(nu,sigma)
                  T1xxxx(mu,lambda,nu,sigma) =
LP2A_ao(mu,lambda)^T6xx(nu,sigma)
                  Txxxx(mu,lambda,nu,sigma) += T1xxxx(mu,lambda,nu,sigma)
#
                  T1xxxx(mu,lambda,nu,sigma) =
LP2B_ao(mu,lambda)^T6xx(nu,sigma)
                  Txxxx(mu,lambda,nu,sigma) += T1xxxx(mu,lambda,nu,sigma)
#
                  T1xxxx(mu,lambda,nu,sigma) =
LP2A_ao(mu,sigma)^LDHFA(nu,lambda)
                  T1xxxx(mu,lambda,nu,sigma)*= 0.5
                  Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                  T1xxxx(mu,lambda,nu,sigma) =
LP2A_ao(mu,nu)^LDHFA(sigma,lambda)
                  T1xxxx(mu,lambda,nu,sigma)*= 0.5
                  Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
```

```
                  T1xxxx(mu,lambda,nu,sigma) =
LP2B_ao(mu,sigma)^LDHFB(nu,lambda)
                  T1xxxx(mu,lambda,nu,sigma)*= 0.5
                  Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                  T1xxxx(mu,lambda,nu,sigma) =
LP2B_ao(mu,nu)^LDHFB(sigma,lambda)
                  T1xxxx(mu,lambda,nu,sigma)*= 0.5
                  Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                  Txxxx(mu,lambda,nu,sigma) *= 0.5
                  TSxxxx(mu,lambda,nu,sigma) = Txxxx(mu,lambda,nu,sigma)
                  TSxxxx(mu,lambda,nu,sigma) *= 2.0
#
#                 Add the the nonseparable part
#                 -----------------------------
                  txxxx(mu,lambda,nu,sigma)   = D3(mu,lambda,sigma,nu)
                  TSxxxx(mu,lambda,nu,sigma) += D2(mu,lambda,nu,sigma)
                  TSxxxx(mu,lambda,nu,sigma) += txxxx(mu,lambda,nu,sigma)
#
#                 Set up integrals
#                 ----------------
                  execute der_int_setup dx1(mu,lambda,nu,sigma)
                  execute der_int_setup dy1(mu,lambda,nu,sigma)
                  execute der_int_setup dz1(mu,lambda,nu,sigma)
                  execute der_int_setup dx2(mu,lambda,nu,sigma)
                  execute der_int_setup dy2(mu,lambda,nu,sigma)
                  execute der_int_setup dz2(mu,lambda,nu,sigma)
                  execute der_int_setup dx3(mu,lambda,nu,sigma)
                  execute der_int_setup dy3(mu,lambda,nu,sigma)
                  execute der_int_setup dz3(mu,lambda,nu,sigma)
                  execute der_int_setup dx4(mu,lambda,nu,sigma)
                  execute der_int_setup dy4(mu,lambda,nu,sigma)
                  execute der_int_setup dz4(mu,lambda,nu,sigma)
#
#                 Compute integral block
#                 ----------------------
                  execute compute_derivative_integrals
#
#                 Contract density with integral derivatives
#                 ------------------------------------------
                  execute DCONT2 TSxxxx(mu,lambda,nu,sigma)
#
               ENDIF # nu < sigma
               ENDIF # mu == lambda
#
            ENDDO sigma
            ENDDO lambda
#
      ENDPARDO mu, nu
#
      PARDO mu, nu
#
            DO lambda
            DO sigma
#
               IF mu < lambda
```

```
               IF nu == sigma
#
                  D2(mu,lambda,nu,sigma) = 0.0
                  D3(mu,lambda,sigma,nu) = 0.0
#
                  DO i1
                     TAxxxi(mu,lambda,nu,i1)  = 0.0
                     TBxxxi(mu,lambda,nu,i1)  = 0.0
                     DO i
                        REQUEST Vxixi(mu,i,nu,i1)     i
                        REQUEST Vxixi(lambda,i,nu,i1) i
#
                        T1xxxi(mu,lambda,nu,i1)  =
Vxixi(mu,i,nu,i1)*La(i,lambda) # ca(lambda,i)
                        TAxxxi(mu,lambda,nu,i1) += T1xxxi(mu,lambda,nu,i1)
                        T2xxxi(mu,lambda,nu,i1)  =
Vxixi(lambda,i,nu,i1)*ca(mu,i)
                        TBxxxi(mu,lambda,nu,i1) += T2xxxi(mu,lambda,nu,i1) # -
#
                     ENDDO i
                     Txxxx(mu,lambda,nu,sigma) =
TBxxxi(mu,lambda,nu,i1)*La(i1,sigma) # ca(sigma,i1)
                     D2(mu,lambda,nu,sigma)   += Txxxx(mu,lambda,nu,sigma)
                     Txxxx(mu,lambda,nu,sigma) =
TAxxxi(mu,lambda,nu,i1)*La(i1,sigma) # ca(sigma,i1)
                     D2(mu,lambda,nu,sigma)   += Txxxx(mu,lambda,nu,sigma)
                  ENDDO i1
#
                  DO j1
                     TAxxxj(mu,lambda,nu,j1)  = 0.0
                     TBxxxj(mu,lambda,nu,j1)  = 0.0
                     DO j
                        REQUEST Vxjxj(mu,j,nu,j1)     j
                        REQUEST Vxjxj(lambda,j,nu,j1) j
#
                        T1xxxj(mu,lambda,nu,j1)  =
Vxjxj(mu,j,nu,j1)*Lb(j,lambda) # cb(lambda,j)
                        TAxxxj(mu,lambda,nu,j1) += T1xxxj(mu,lambda,nu,j1)
                        T2xxxj(mu,lambda,nu,j1)  =
Vxjxj(lambda,j,nu,j1)*cb(mu,j)
                        TBxxxj(mu,lambda,nu,j1) += T2xxxj(mu,lambda,nu,j1) # -
#
                     ENDDO j
                     Txxxx(mu,lambda,nu,sigma) =
TBxxxj(mu,lambda,nu,j1)*Lb(j1,sigma) # cb(sigma,j1)
                     D2(mu,lambda,nu,sigma)   += Txxxx(mu,lambda,nu,sigma)
                     Txxxx(mu,lambda,nu,sigma) =
TAxxxj(mu,lambda,nu,j1)*Lb(j1,sigma) # cb(sigma,j1)
                     D2(mu,lambda,nu,sigma)   += Txxxx(mu,lambda,nu,sigma)
                  ENDDO j1
#
                  DO j
                     TAxxxj(mu,lambda,nu,j)  = 0.0
                     TBxxxj(mu,lambda,nu,j)  = 0.0
                     DO i
                        REQUEST Vxixj(mu,i,nu,j)     j
                        REQUEST Vxixj(lambda,i,nu,j) j
```

```
#
                        T1xxxj(mu,lambda,nu,j)  =
Vxixj(mu,i,nu,j)*La(i,lambda) # ca(lambda,i)
                        TAxxxj(mu,lambda,nu,j) += T1xxxj(mu,lambda,nu,j)
                        T2xxxj(mu,lambda,nu,j)  =
Vxixj(lambda,i,nu,j)*ca(mu,i)
                        TBxxxj(mu,lambda,nu,j) += T2xxxj(mu,lambda,nu,j) # -
#
                     ENDDO i
#
                     Txxxx(mu,lambda,nu,sigma) =
TBxxxj(mu,lambda,nu,j)*Lb(j,sigma) # cb(sigma,j)
                     D2(mu,lambda,nu,sigma)   += Txxxx(mu,lambda,nu,sigma)
                     D2(mu,lambda,nu,sigma)   += Txxxx(mu,lambda,nu,sigma)
#
                     Txxxx(mu,lambda,nu,sigma) =
TAxxxj(mu,lambda,nu,j)*Lb(j,sigma) # cb(sigma,j)
                     D2(mu,lambda,nu,sigma)   += Txxxx(mu,lambda,nu,sigma)
                     D2(mu,lambda,nu,sigma)   += Txxxx(mu,lambda,nu,sigma)
                  ENDDO j
#
#                 Get the separable part
#                 ----------------------
#
#                 Get 1-particle pieces
#                 ---------------------
#
#                 HF only
#                 -------
                  Txx(nu,sigma)              = LDHFa(nu,sigma)
                  Txx(nu,sigma)             += LDHFb(nu,sigma)
                  Txx(nu,sigma)             += LP2A_ao(nu,sigma)
                  Txx(nu,sigma)             += LP2B_ao(nu,sigma)
                  T1xx(mu,lambda)            = LDHFa(mu,lambda)
                  T1xx(mu,lambda)           += LDHFb(mu,lambda)

                  Txxxx(mu,lambda,nu,sigma)  = T1xx(mu,lambda)^Txx(nu,sigma)
#
                  T2xx(nu,lambda)             = LDHFa(nu,lambda)
                  T2xx(nu,lambda)            += LP2A_ao(nu,lambda)
                  T1xxxx(mu,lambda,nu,sigma) = LDHFa(mu,sigma)^T2xx(nu,lambda)
                  T1xxxx(mu,lambda,nu,sigma)*= 0.5
                  Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                  T3xx(nu,lambda)             = LDHFb(nu,lambda)
                  T3xx(nu,lambda)            += LP2B_ao(nu,lambda)
                  T1xxxx(mu,lambda,nu,sigma) = LDHFb(mu,sigma)^T3xx(nu,lambda)
                  T1xxxx(mu,lambda,nu,sigma)*= 0.5
                  Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                  T4xx(sigma,lambda)          = LDHFa(sigma,lambda)
                  T4xx(sigma,lambda)         += LP2A_ao(sigma,lambda)
                  T1xxxx(mu,lambda,nu,sigma) = LDHFa(mu,nu)^T4xx(sigma,lambda)
                  T1xxxx(mu,lambda,nu,sigma)*= 0.5
                  Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                  T5xx(sigma,lambda)          = LDHFb(sigma,lambda)
```

```
                T5xx(sigma,lambda)          += LP2B_ao(sigma,lambda)
                T1xxxx(mu,lambda,nu,sigma) = LDHFb(mu,nu)^T5xx(sigma,lambda)
                T1xxxx(mu,lambda,nu,sigma)*= 0.5
                Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
#               Correlation Only
#               ----------------
#
                T6xx(nu,sigma)               = LDHFA(nu,sigma)
                T6xx(nu,sigma)              += LDHFB(nu,sigma)
                T1xxxx(mu,lambda,nu,sigma) =
LP2A_ao(mu,lambda)^T6xx(nu,sigma)
                Txxxx(mu,lambda,nu,sigma) += T1xxxx(mu,lambda,nu,sigma)
#
                T1xxxx(mu,lambda,nu,sigma) =
LP2B_ao(mu,lambda)^T6xx(nu,sigma)
                Txxxx(mu,lambda,nu,sigma) += T1xxxx(mu,lambda,nu,sigma)
#
                T1xxxx(mu,lambda,nu,sigma) =
LP2A_ao(mu,sigma)^LDHFA(nu,lambda)
                T1xxxx(mu,lambda,nu,sigma)*= 0.5
                Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                T1xxxx(mu,lambda,nu,sigma) =
LP2A_ao(mu,nu)^LDHFA(sigma,lambda)
                T1xxxx(mu,lambda,nu,sigma)*= 0.5
                Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                T1xxxx(mu,lambda,nu,sigma) =
LP2B_ao(mu,sigma)^LDHFB(nu,lambda)
                T1xxxx(mu,lambda,nu,sigma)*= 0.5
                Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                T1xxxx(mu,lambda,nu,sigma) =
LP2B_ao(mu,nu)^LDHFB(sigma,lambda)
                T1xxxx(mu,lambda,nu,sigma)*= 0.5
                Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                Txxxx(mu,lambda,nu,sigma) *= 0.5
                TSxxxx(mu,lambda,nu,sigma) = Txxxx(mu,lambda,nu,sigma)
                TSxxxx(mu,lambda,nu,sigma) *= 2.0
#
#               Add the the nonseparable part
#               -----------------------------
                TSxxxx(mu,lambda,nu,sigma) += D2(mu,lambda,nu,sigma)
#
#               Set up integrals
#               ----------------
                execute der_int_setup dx1(mu,lambda,nu,sigma)
                execute der_int_setup dy1(mu,lambda,nu,sigma)
                execute der_int_setup dz1(mu,lambda,nu,sigma)
                execute der_int_setup dx2(mu,lambda,nu,sigma)
                execute der_int_setup dy2(mu,lambda,nu,sigma)
                execute der_int_setup dz2(mu,lambda,nu,sigma)
                execute der_int_setup dx3(mu,lambda,nu,sigma)
                execute der_int_setup dy3(mu,lambda,nu,sigma)
                execute der_int_setup dz3(mu,lambda,nu,sigma)
```

```
                execute der_int_setup dx4(mu,lambda,nu,sigma)
                execute der_int_setup dy4(mu,lambda,nu,sigma)
                execute der_int_setup dz4(mu,lambda,nu,sigma)
#
#               Compute integral block
#               ----------------------
                execute compute_derivative_integrals
#
#               Contract density with integral derivatives
#               ------------------------------------------
                execute DCONT2 TSxxxx(mu,lambda,nu,sigma)
#
             ENDIF # nu == sigma
             ENDIF # mu < lambda
#
          ENDDO sigma
          ENDDO lambda
#
     ENDPARDO mu, nu
#
     PARDO mu, nu
#
          DO lambda
          DO sigma
#
             IF mu == lambda
             IF nu == sigma
#
                D2(mu,lambda,nu,sigma) = 0.0
#
                DO i1
                   TAxxxi(mu,lambda,nu,i1)  = 0.0
                   DO i
                      REQUEST Vxixi(mu,i,nu,i1) i
                      T1xxxi(mu,lambda,nu,i1)  =
Vxixi(mu,i,nu,i1)*La(i,lambda) # ca(lambda,i)
                      TAxxxi(mu,lambda,nu,i1) += T1xxxi(mu,lambda,nu,i1)
                   ENDDO i
                   Txxxx(mu,lambda,nu,sigma) =
TAxxxi(mu,lambda,nu,i1)*La(i1,sigma) # ca(sigma,i1)
                   D2(mu,lambda,nu,sigma)   += Txxxx(mu,lambda,nu,sigma)
                ENDDO i1
#
                DO j1
                   TAxxxj(mu,lambda,nu,j1)  = 0.0
                   DO j
                      REQUEST Vxjxj(mu,j,nu,j1) j
                      T1xxxj(mu,lambda,nu,j1)  =
Vxjxj(mu,j,nu,j1)*Lb(j,lambda) # cb(lambda,j)
                      TAxxxj(mu,lambda,nu,j1) += T1xxxj(mu,lambda,nu,j1)
                   ENDDO j
                   Txxxx(mu,lambda,nu,sigma) =
TAxxxj(mu,lambda,nu,j1)*Lb(j1,sigma) # cb(sigma,j1)
                   D2(mu,lambda,nu,sigma)   += Txxxx(mu,lambda,nu,sigma)
                ENDDO j1
#
                DO j
```

```
                    TAxxxj(mu,lambda,nu,j)  = 0.0
                    DO i
                       REQUEST Vxixj(mu,i,nu,j) i
                       T1xxxj(mu,lambda,nu,j)  =
Vxixj(mu,i,nu,j)*La(i,lambda) # ca(lambda,i)
                       TAxxxj(mu,lambda,nu,j) += T1xxxj(mu,lambda,nu,j)
                    ENDDO i
                    Txxxx(mu,lambda,nu,sigma) =
TAxxxj(mu,lambda,nu,j)*Lb(j,sigma) # cb(sigma,j)
                    D2(mu,lambda,nu,sigma)   += Txxxx(mu,lambda,nu,sigma)
                    D2(mu,lambda,nu,sigma)   += Txxxx(mu,lambda,nu,sigma)
                 ENDDO j
#
#                Get the separable part
#                ----------------------
#
#                Get 1-particle pieces
#                ---------------------
#
#                HF only
#                -------
                 Txx(nu,sigma)              = LDHFa(nu,sigma)
                 Txx(nu,sigma)             += LDHFb(nu,sigma)
                 Txx(nu,sigma)             += LP2A_ao(nu,sigma)
                 Txx(nu,sigma)             += LP2B_ao(nu,sigma)
                 T1xx(mu,lambda)            = LDHFa(mu,lambda)
                 T1xx(mu,lambda)           += LDHFb(mu,lambda)

                 Txxxx(mu,lambda,nu,sigma)  = T1xx(mu,lambda)^Txx(nu,sigma)
#
                 T2xx(nu,lambda)             = LDHFa(nu,lambda)
                 T2xx(nu,lambda)            += LP2A_ao(nu,lambda)
                 T1xxxx(mu,lambda,nu,sigma) = LDHFa(mu,sigma)^T2xx(nu,lambda)
                 T1xxxx(mu,lambda,nu,sigma)*= 0.5
                 Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                 T3xx(nu,lambda)             = LDHFb(nu,lambda)
                 T3xx(nu,lambda)            += LP2B_ao(nu,lambda)
                 T1xxxx(mu,lambda,nu,sigma) = LDHFb(mu,sigma)^T3xx(nu,lambda)
                 T1xxxx(mu,lambda,nu,sigma)*= 0.5
                 Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                 T4xx(sigma,lambda)          = LDHFa(sigma,lambda)
                 T4xx(sigma,lambda)         += LP2A_ao(sigma,lambda)
                 T1xxxx(mu,lambda,nu,sigma) = LDHFa(mu,nu)^T4xx(sigma,lambda)
                 T1xxxx(mu,lambda,nu,sigma)*= 0.5
                 Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                 T5xx(sigma,lambda)          = LDHFb(sigma,lambda)
                 T5xx(sigma,lambda)         += LP2B_ao(sigma,lambda)
                 T1xxxx(mu,lambda,nu,sigma) = LDHFb(mu,nu)^T5xx(sigma,lambda)
                 T1xxxx(mu,lambda,nu,sigma)*= 0.5
                 Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
#                Correlation Only
#                ----------------
#
```

```
                T6xx(nu,sigma)              = LDHFA(nu,sigma)
                T6xx(nu,sigma)             += LDHFB(nu,sigma)
                T1xxxx(mu,lambda,nu,sigma) =
LP2A_ao(mu,lambda)^T6xx(nu,sigma)
                Txxxx(mu,lambda,nu,sigma) += T1xxxx(mu,lambda,nu,sigma)
#
                T1xxxx(mu,lambda,nu,sigma) =
LP2B_ao(mu,lambda)^T6xx(nu,sigma)
                Txxxx(mu,lambda,nu,sigma) += T1xxxx(mu,lambda,nu,sigma)
#
                T1xxxx(mu,lambda,nu,sigma) =
LP2A_ao(mu,sigma)^LDHFA(nu,lambda)
                T1xxxx(mu,lambda,nu,sigma)*= 0.5
                Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                T1xxxx(mu,lambda,nu,sigma) =
LP2A_ao(mu,nu)^LDHFA(sigma,lambda)
                T1xxxx(mu,lambda,nu,sigma)*= 0.5
                Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                T1xxxx(mu,lambda,nu,sigma) =
LP2B_ao(mu,sigma)^LDHFB(nu,lambda)
                T1xxxx(mu,lambda,nu,sigma)*= 0.5
                Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                T1xxxx(mu,lambda,nu,sigma) =
LP2B_ao(mu,nu)^LDHFB(sigma,lambda)
                T1xxxx(mu,lambda,nu,sigma)*= 0.5
                Txxxx(mu,lambda,nu,sigma) -= T1xxxx(mu,lambda,nu,sigma)
#
                Txxxx(mu,lambda,nu,sigma) *= 0.5
                TSxxxx(mu,lambda,nu,sigma) = Txxxx(mu,lambda,nu,sigma)
#
#               Add the the nonseparable part
#               -----------------------------
                TSxxxx(mu,lambda,nu,sigma) += D2(mu,lambda,nu,sigma)
#
#               Set up integrals
#               ----------------
                execute der_int_setup dx1(mu,lambda,nu,sigma)
                execute der_int_setup dy1(mu,lambda,nu,sigma)
                execute der_int_setup dz1(mu,lambda,nu,sigma)
                execute der_int_setup dx2(mu,lambda,nu,sigma)
                execute der_int_setup dy2(mu,lambda,nu,sigma)
                execute der_int_setup dz2(mu,lambda,nu,sigma)
                execute der_int_setup dx3(mu,lambda,nu,sigma)
                execute der_int_setup dy3(mu,lambda,nu,sigma)
                execute der_int_setup dz3(mu,lambda,nu,sigma)
                execute der_int_setup dx4(mu,lambda,nu,sigma)
                execute der_int_setup dy4(mu,lambda,nu,sigma)
                execute der_int_setup dz4(mu,lambda,nu,sigma)
#
#               Compute integral block
#               ----------------------
                execute compute_derivative_integrals
#
#               Contract density with integral derivatives
```

```
#                -----------------------------------------
                 execute DCONT2 TSxxxx(mu,lambda,nu,sigma)
#
              ENDIF # nu == sigma
              ENDIF # mu == lambda
#
            ENDDO sigma
            ENDDO lambda
#
      ENDPARDO mu, nu
#
      ENDPROC D2TRANS
#     ---------------
#
# ------------------------------------------------------------------------
--
#
# ------------------------------------------------------------------------
-------------
#
         PROC UPDATE_PAI
#        ---------------
#
         PARDO a, i
#
               GET                          Painew_aa(a,i)
               GET                          Paiold_aa(a,i)
               Tai(a,i)                  = Painew_aa(a,i)
               execute energy_denominator Tai(a,i)
               Tai(a,i)                 -= Paiold_aa(a,i)
#
               if kiter == 1
                  PUT e1ai(a,i) = tai(a,i)
               endif
#
               if kiter == 2
                  PUT e2ai(a,i) = tai(a,i)
               endif
#
               if kiter == 3
                  PUT e3ai(a,i) = tai(a,i)
               endif
#
               if kiter == 4
                  PUT e4ai(a,i) = tai(a,i)
               endif
#
               if kiter >= 5
                  PUT e5ai(a,i) = tai(a,i)
               endif
#
         ENDPARDO a, i
#
         PARDO b, j
#
               GET                          Painew_bb(b,j)
               GET                          Paiold_bb(b,j)
```

```
               Tbj(b,j)                  = Painew_bb(b,j)
               execute energy_denominator Tbj(b,j)
               Tbj(b,j)                 -= Paiold_bb(b,j)
#
               if kiter == 1
                  PUT e1bj(b,j) = tbj(b,j)
               endif
#
               if kiter == 2
                  PUT e2bj(b,j) = tbj(b,j)
               endif
#
               if kiter == 3
                  PUT e3bj(b,j) = tbj(b,j)
               endif
#
               if kiter == 4
                  PUT e4bj(b,j) = tbj(b,j)
               endif
#
               if kiter >= 5
                  PUT e5bj(b,j) = tbj(b,j)
               endif
#
         ENDPARDO b, j
#
         ENDPROC UPDATE_PAI
#        ------------------
#
# ------------------------------------------------------------------------------
-------------
#
# ------------------------------------------------------------------------------
-------------
#
      PROC MOVE_PAI
#     -------------
#
      PARDO a, i
#
            GET        Paiold_aa(a,i)
            tai(a,i) = Paiold_aa(a,i)
#
            if kiter == 1
               PUT d1ai(a,i) = tai(a,i)
            endif
#
            if kiter == 2
               PUT d2ai(a,i) = Tai(a,i)
            endif
#
            if kiter == 3
               PUT d3ai(a,i) = Tai(a,i)
            endif
#
            if kiter == 4
               PUT d4ai(a,i) = Tai(a,i)
```

```
            endif
#
            if kiter >= 5
               PUT d4ai(a,i) = Tai(a,i)
            endif
#
      ENDPARDO a, i
#
      PARDO b, j
#
            GET        Paiold_bb(b,j)
            tbj(b,j) = Paiold_bb(b,j)
#
            if kiter == 1
               PUT d1bj(b,j) = tbj(b,j)
            endif
#
            if kiter == 2
               PUT d2bj(b,j) = tbj(b,j)
            endif
#
            if kiter == 3
               PUT d3bj(b,j) = tbj(b,j)
            endif
#
            if kiter == 4
               PUT d4bj(b,j) = tbj(b,j)
            endif
#
            if kiter >= 5
               PUT d4bj(b,j) = tbj(b,j)
            endif
#
      ENDPARDO b, j
#
      ENDPROC MOVE_PAI
#     ----------------
#
# ------------------------------------------------------------------------
-------------
#
# ------------------------------------------------------------------------
------
#
     PROC ZERO_DSCALAR
#    -----------------
#
#    The scalars used (overlap of error arrays and coeffients) in the DIIS
#    expansion are zero'd out.
#
#      Zero out scalars.
#      -----------------
#
       b11 = 0.0
       b12 = 0.0
       b13 = 0.0
       b14 = 0.0
```

```
        b15 = 0.0
        b16 = 0.0
        b17 = 0.0
        b18 = 0.0
        b19 = 0.0
        b110 = 0.0
#
        b22 = 0.0
        b23 = 0.0
        b24 = 0.0
        b25 = 0.0
        b26 = 0.0
        b27 = 0.0
        b28 = 0.0
        b29 = 0.0
        b210 = 0.0
#
        b33 = 0.0
        b34 = 0.0
        b35 = 0.0
        b36 = 0.0
        b37 = 0.0
        b38 = 0.0
        b39 = 0.0
        b310 = 0.0
#
        b44 = 0.0
        b45 = 0.0
        b46 = 0.0
        b47 = 0.0
        b48 = 0.0
        b49 = 0.0
        b410 = 0.0
#
        b55 = 0.0
        b56 = 0.0
        b57 = 0.0
        b58 = 0.0
        b59 = 0.0
        b510 = 0.0
#
        b66 = 0.0
        b67 = 0.0
        b68 = 0.0
        b69 = 0.0
        b610 = 0.0
#
        b77 = 0.0
        b78 = 0.0
        b79 = 0.0
        b710 = 0.0
#
        b88 = 0.0
        b89 = 0.0
        b810 = 0.0
#
        b99 = 0.0
```

```
        b910 = 0.0
#
        b1010 = 0.0
#
        Tb11 = 0.0
        Tb12 = 0.0
        Tb13 = 0.0
        Tb14 = 0.0
        Tb15 = 0.0
        Tb16 = 0.0
        Tb17 = 0.0
        Tb18 = 0.0
        Tb19 = 0.0
        Tb110 = 0.0
#
        Tb22 = 0.0
        Tb23 = 0.0
        Tb24 = 0.0
        Tb25 = 0.0
        Tb26 = 0.0
        Tb27 = 0.0
        Tb28 = 0.0
        Tb29 = 0.0
        Tb210 = 0.0
#
        Tb33 = 0.0
        Tb34 = 0.0
        Tb35 = 0.0
        Tb36 = 0.0
        Tb37 = 0.0
        Tb38 = 0.0
        Tb39 = 0.0
        Tb310 = 0.0
#
        Tb44 = 0.0
        Tb45 = 0.0
        Tb46 = 0.0
        Tb47 = 0.0
        Tb48 = 0.0
        Tb49 = 0.0
        Tb410 = 0.0
#
        Tb55 = 0.0
        Tb56 = 0.0
        Tb57 = 0.0
        Tb58 = 0.0
        Tb59 = 0.0
        Tb510 = 0.0
#
        Tb66 = 0.0
        Tb67 = 0.0
        Tb68 = 0.0
        Tb69 = 0.0
        Tb610 = 0.0
#
        Tb77 = 0.0
        Tb78 = 0.0
```

```
       Tb79 = 0.0
       Tb710 = 0.0
#
       Tb88 = 0.0
       Tb89 = 0.0
       Tb810 = 0.0
#
       Tb99 = 0.0
       Tb910 = 0.0
#
       Tb1010 = 0.0
#
       c1 = 0.0
       c2 = 0.0
       c3 = 0.0
       c4 = 0.0
       c5 = 0.0
       c6 = 0.0
       c7 = 0.0
       c8 = 0.0
       c9 = 0.0
       c10 = 0.0
#
       execute sip_barrier
#
     ENDPROC ZERO_DSCALAR
#    --------------------
#
# ------------------------------------------------------------------------------
------
#
# ------------------------------------------------------------------------------
------
#
     PROC SETUP_DIIS
#    ---------------
#
#    Put the matrix elements of B into the sip 'B' matrix.
#
       execute diis_setup Tb11
       execute diis_setup Tb12
       execute diis_setup Tb13
       execute diis_setup Tb14
       execute diis_setup Tb15
       execute diis_setup Tb16
       execute diis_setup Tb17
       execute diis_setup Tb18
       execute diis_setup Tb19
       execute diis_setup Tb110

       execute diis_setup Tb22
       execute diis_setup Tb23
       execute diis_setup Tb24
       execute diis_setup Tb25
       execute diis_setup Tb26
       execute diis_setup Tb27
       execute diis_setup Tb28
```

```
        execute diis_setup Tb29
        execute diis_setup Tb210

        execute diis_setup Tb33
        execute diis_setup Tb34
        execute diis_setup Tb35
        execute diis_setup Tb36
        execute diis_setup Tb37
        execute diis_setup Tb38
        execute diis_setup Tb39
        execute diis_setup Tb310

        execute diis_setup Tb44
        execute diis_setup Tb45
        execute diis_setup Tb46
        execute diis_setup Tb47
        execute diis_setup Tb48
        execute diis_setup Tb49
        execute diis_setup Tb410

        execute diis_setup Tb55
        execute diis_setup Tb56
        execute diis_setup Tb57
        execute diis_setup Tb58
        execute diis_setup Tb59
        execute diis_setup Tb510

        execute diis_setup Tb66
        execute diis_setup Tb67
        execute diis_setup Tb68
        execute diis_setup Tb69
        execute diis_setup Tb610

        execute diis_setup Tb77
        execute diis_setup Tb78
        execute diis_setup Tb79
        execute diis_setup Tb710

        execute diis_setup Tb88
        execute diis_setup Tb89
        execute diis_setup Tb810

        execute diis_setup Tb99
        execute diis_setup Tb910

        execute diis_setup Tb1010
#
        execute sip_barrier
#
     ENDPROC SETUP_DIIS
#    ------------------
#
# ------------------------------------------------------------------------
------
#
# ------------------------------------------------------------------------
------
```

```
#
      PROC DIIS1
#     ----------
#
#     Zero out scalars.
#     -----------------
#
      CALL ZERO_DSCALAR
#
      execute sip_barrier
#
#     Determine the 'B-matrix'.
#     -------------------------
#
#     Compute contributions due to Dai amplitudes.
#     --------------------------------------------
#
      PARDO a, i
#
#           REQUEST/GET amplitude data from all 2 previous iterations.
#           ----------------------------------------------------------
#
            GET e1ai(a,i)     # kiter-1 amplitudes
            GET e2ai(a,i)     # kiter-0 amplitudes
#
#           Compute contributions to the 'B-matrix'.
#           ----------------------------------------
#
#              b1x x=1,2
#              ---------
#
               etemp = e1ai(a,i)*e1ai(a,i)
               b11  += etemp
#
               etemp = e1ai(a,i)*e2ai(a,i)
               b12  += etemp
#
#              b1x x=2,2
#              ---------
#
               etemp = e2ai(a,i)*e2ai(a,i)
               b22  += etemp
#
      ENDPARDO a, i
#
#     Compute contributions due to Dbj amplitudes.
#     --------------------------------------------
#
      PARDO b, j
#
#           REQUEST/GET amplitude data from all 2 previous iterations.
#           ----------------------------------------------------------
#
            GET e1bj(b,j)     # kiter-1 amplitudes
            GET e2bj(b,j)     # kiter-0 amplitudes
#
#           Compute contributions to the 'B-matrix'.
```

```
#          ----------------------------------------
#
#             b1x x=1,2
#             ---------
#
              etemp = e1bj(b,j)*e1bj(b,j)
              b11  += etemp
#
              etemp = e1bj(b,j)*e2bj(b,j)
              b12  += etemp
#
#             b1x x=2,2
#             ---------
#
              etemp = e2bj(b,j)*e2bj(b,j)
              b22  += etemp
#
     ENDPARDO b, j
#
     execute sip_barrier
#
#    Collectively sum B-matrix elements.
#    -----------------------------------
#
     collective Tb11 += b11
     collective Tb12 += b12
     collective Tb22 += b22
     execute server_barrier
#
#    Now the unique elements of the 'B-matrix' have been computed, the array
is filled out
#    in the setup_diis.
#    ------------------------------------------------------------------------
--------------
#
#    Put the elements of the 'B-matrix', which have been computed as scalars
into the
#    R-matrix.
#    ------------------------------------------------------------------------
--------
#
     CALL SETUP_DIIS
#
     execute compute_diis # --> New instruction
#
     c1 = Tb11
     c2 = Tb22
     execute print_scalar c1
     execute print_scalar c2
     execute server_barrier
     execute sip_barrier
#
#    Done computing the c-vector.
#    ----------------------------
#
#    Form Pai_old.
#    -------------
```

```
#
     PARDO a, i
#
#          REQUEST/GET amplitude data from all 2 previous iterations.
#          ----------------------------------------------------------
#
           GET D0ai(a,i)     # kiter-2 amplitudes
           GET D1ai(a,i)     # kiter-1 amplitudes
#
           GET e1ai(a,i)     # kiter-1 amplitudes
           GET e2ai(a,i)     # kiter-0 amplitudes
#
#          Compute contributions to updated amplitudes --> tai_old.
#          ---------------------------------------------------------
#
           t1ai(a,i)     = d0ai(a,i)
           t1ai(a,i)    += e1ai(a,i)
           t1ai(a,i)    *= c1
           tai(a,i)      = t1ai(a,i)
#
           t1ai(a,i)     = d1ai(a,i)
           t1ai(a,i)    += e2ai(a,i)
           t1ai(a,i)    *= c2
           tai(a,i)     += t1ai(a,i)
#
           PUT Paiold_aa(a,i) = tai(a,i)
#
     ENDPARDO a, i
#
#    Form Dbj_old.
#    -------------
#
     PARDO b, j
#
#          REQUEST/GET amplitude data from all 2 previous iterations.
#          ----------------------------------------------------------
#
           GET D0bj(b,j)     # kiter-2 amplitudes
           GET D1bj(b,j)     # kiter-1 amplitudes
#
           GET e1bj(b,j)     # kiter-1 amplitudes
           GET e2bj(b,j)     # kiter-0 amplitudes
#
#          Compute contributions to updated amplitudes --> tbj_old.
#          ---------------------------------------------------------
#
           t1bj(b,j)     = d0bj(b,j)
           t1bj(b,j)    += e1bj(b,j)
           t1bj(b,j)    *= c1
           tbj(b,j)      = t1bj(b,j)
#
           t1bj(b,j)     = d1bj(b,j)
           t1bj(b,j)    += e2bj(b,j)
           t1bj(b,j)    *= c2
           tbj(b,j)     += t1bj(b,j)
#
           PUT Paiold_bb(b,j) = tbj(b,j)
```

```
#
     ENDPARDO b, j
#
     execute sip_barrier
#
     ENDPROC DIIS1
#    -------------
#
# ------------------------------------------------------------------------
------
#
     PROC DIIS2
#    ----------
#
#    Zero out scalars.
#    -----------------
#
     CALL ZERO_DSCALAR
#
     execute server_barrier
     execute sip_barrier
#
#    Determine the 'B-matrix'.
#    -------------------------
#
#    Compute contributions due to dai amplitudes.
#    --------------------------------------------
#
     PARDO a, i
#
#          REQUEST/GET amplitude data from all 4 previous iterations.
#          ----------------------------------------------------------
#
           GET e1ai(a,i)    # kiter-2 amplitudes
           GET e2ai(a,i)    # kiter-1 amplitudes
           GET e3ai(a,i)    # kiter-0 amplitudes
#
#          Compute contributions to the 'B-matrix'.
#          ----------------------------------------
#
#             b1x x=1,3
#             ---------
#
              etemp = e1ai(a,i)*e1ai(a,i)
              b11  += etemp
#
              etemp = e1ai(a,i)*e2ai(a,i)
              b12  += etemp
#
              etemp = e1ai(a,i)*e3ai(a,i)
              b13  += etemp
#
#             b1x x=2,3
#             ---------
#
              etemp = e2ai(a,i)*e2ai(a,i)
              b22  += etemp
```

```
#
              etemp = e2ai(a,i)*e3ai(a,i)
              b23  += etemp
#
#             b1x x=3,3
#             ---------
#
              etemp = e3ai(a,i)*e3ai(a,i)
              b33  += etemp
#
     ENDPARDO a, i
#
#    Compute contributions due to dbj amplitudes.
#    --------------------------------------------
#
     PARDO b, j
#
#          REQUEST/GET amplitude data from all 3 previous iterations.
#          ----------------------------------------------------------
#
           GET e1bj(b,j)    # kiter-2 amplitudes
           GET e2bj(b,j)    # kiter-1 amplitudes
           GET e3bj(b,j)    # kiter-0 amplitudes
#
#          Compute contributions to the 'B-matrix'.
#          ----------------------------------------
#
#             b1x x=1,3
#             ---------
#
              etemp = e1bj(b,j)*e1bj(b,j)
              b11  += etemp
#
              etemp = e1bj(b,j)*e2bj(b,j)
              b12  += etemp
#
              etemp = e1bj(b,j)*e3bj(b,j)
              b13  += etemp
#
#             b1x x=2,3
#             ---------
#
              etemp = e2bj(b,j)*e2bj(b,j)
              b22  += etemp
#
              etemp = e2bj(b,j)*e3bj(b,j)
              b23  += etemp
#
#             b1x x=3,3
#             ---------
#
              etemp = e3bj(b,j)*e3bj(b,j)
              b33  += etemp
#
     ENDPARDO b, j
#
     execute sip_barrier
```

```
#
#    Collectively sum B-matrix elements.
#    -----------------------------------
#
     collective Tb11 += b11
     collective Tb12 += b12
     collective Tb13 += b13
     collective Tb22 += b22
     collective Tb23 += b23
     collective Tb33 += b33
     execute server_barrier
#
#    Now the unique elements of the 'B-matrix' have been computed and the
array filled out.
#    ------------------------------------------------------------------------
--------------
#
#    Put the elements of the 'B-matrix', which have been computed as scalars
into the
#    R-matrix.
#    ------------------------------------------------------------------------
--------

     CALL SETUP_DIIS
#
     execute compute_diis # --> New instruction
#
     c1 = Tb11
     c2 = Tb22
     c3 = Tb33
     execute print_scalar c1
     execute print_scalar c2
     execute print_scalar c3
     execute server_barrier
     execute sip_barrier
#
#    Done computing the c-vector.
#    ----------------------------
#
#    Form the updated amplitudes using the c-vector.
#    -----------------------#
#    Form Dai_old.
#    -------------
#
     PARDO a, i
#
#          REQUEST/GET amplitude data from all 3 previous iterations.
#          ----------------------------------------------------------
#
           GET D0ai(a,i)     # kiter-3 amplitudes
           GET D1ai(a,i)     # kiter-2 amplitudes
           GET D2ai(a,i)     # kiter-1 amplitudes
#
           GET e1ai(a,i)     # kiter-2 amplitudes
           GET e2ai(a,i)     # kiter-1 amplitudes
           GET e3ai(a,i)     # kiter-0 amplitudes
#
```

```
#           Compute contributions to updated amplitudes --> tai_old.
#           --------------------------------------------------------
#
            t1ai(a,i)     = d0ai(a,i)
            t1ai(a,i)    += e1ai(a,i)
            t1ai(a,i)    *= c1
            tai(a,i)      = t1ai(a,i)
#
            t1ai(a,i)     = d1ai(a,i)
            t1ai(a,i)    += e2ai(a,i)
            t1ai(a,i)    *= c2
            tai(a,i)     += t1ai(a,i)
#
            t1ai(a,i)     = d2ai(a,i)
            t1ai(a,i)    += e3ai(a,i)
            t1ai(a,i)    *= c3
            tai(a,i)     += t1ai(a,i)
#
            PUT Paiold_aa(a,i) = tai(a,i)
#
      ENDPARDO a, i
#
#     Form tbj_old.
#     -------------
#
      PARDO b, j
#
#           REQUEST/GET amplitude data from all 3 previous iterations.
#           ----------------------------------------------------------
#
            GET D0bj(b,j)     # kiter-3 amplitudes
            GET D1bj(b,j)     # kiter-2 amplitudes
            GET D2bj(b,j)     # kiter-1 amplitudes
#
            GET e1bj(b,j)     # kiter-2 amplitudes
            GET e2bj(b,j)     # kiter-1 amplitudes
            GET e3bj(b,j)     # kiter-0 amplitudes
#
#           Compute contributions to updated amplitudes --> tbj_old.
#           --------------------------------------------------------
#
            t1bj(b,j)     = d0bj(b,j)
            t1bj(b,j)    += e1bj(b,j)
            t1bj(b,j)    *= c1
            tbj(b,j)      = t1bj(b,j)
#
            t1bj(b,j)     = d1bj(b,j)
            t1bj(b,j)    += e2bj(b,j)
            t1bj(b,j)    *= c2
            tbj(b,j)     += t1bj(b,j)
#
            t1bj(b,j)     = d2bj(b,j)
            t1bj(b,j)    += e3bj(b,j)
            t1bj(b,j)    *= c3
            tbj(b,j)     += t1bj(b,j)
#
            PUT Paiold_bb(b,j) = tbj(b,j)
```

```
#
     ENDPARDO b, j
#
     execute sip_barrier
#
     ENDPROC DIIS2
#    -------------
#
# ------------------------------------------------------------------------
------
#
# ------------------------------------------------------------------------
------
#
     PROC DIIS3
#    ----------
#
#    Zero out scalars.
#    -----------------
#
     CALL ZERO_DSCALAR
#
     execute sip_barrier
#
#    Determine the 'B-matrix'.
#    -------------------------
#
#    Compute contributions due to dai amplitudes.
#    --------------------------------------------
#
     PARDO a, i
#
#          REQUEST/GET amplitude data from all 4 previous iterations.
#          ----------------------------------------------------------
#
           GET e1ai(a,i)    # kiter-3 amplitudes
           GET e2ai(a,i)    # kiter-2 amplitudes
           GET e3ai(a,i)    # kiter-1 amplitudes
           GET e4ai(a,i)    # kiter-0 amplitudes
#
#          Compute contributions to the 'B-matrix'.
#          ----------------------------------------
#
#             b1x x=1,4
#             ---------
#
              etemp = e1ai(a,i)*e1ai(a,i)
              b11  += etemp
#
              etemp = e1ai(a,i)*e2ai(a,i)
              b12  += etemp
#
              etemp = e1ai(a,i)*e3ai(a,i)
              b13  += etemp
#
              etemp = e1ai(a,i)*e4ai(a,i)
              b14  += etemp
```

```
#
#             b1x x=2,4
#             ---------
#
              etemp = e2ai(a,i)*e2ai(a,i)
              b22  += etemp
#
              etemp = e2ai(a,i)*e3ai(a,i)
              b23  += etemp
#
              etemp = e2ai(a,i)*e4ai(a,i)
              b24  += etemp
#
#             b1x x=3,4
#             ---------
#
              etemp = e3ai(a,i)*e3ai(a,i)
              b33  += etemp
#
              etemp = e3ai(a,i)*e4ai(a,i)
              b34  += etemp
#
#             b1x x=4,4
#             ---------
#
              etemp = e4ai(a,i)*e4ai(a,i)
              b44  += etemp
#
     ENDPARDO a, i
#
#    Compute contributions due to dbj amplitudes.
#    --------------------------------------------
#
     PARDO b, j
#
#          REQUEST/GET amplitude data from all 3 previous iterations.
#          ----------------------------------------------------------
#
           GET e1bj(b,j)    # kiter-3 amplitudes
           GET e2bj(b,j)    # kiter-2 amplitudes
           GET e3bj(b,j)    # kiter-1 amplitudes
           GET e4bj(b,j)    # kiter-0 amplitudes
#
#          Compute contributions to the 'B-matrix'.
#          ----------------------------------------
#
#             b1x x=1,4
#             ---------
#
              etemp = e1bj(b,j)*e1bj(b,j)
              b11  += etemp
#
              etemp = e1bj(b,j)*e2bj(b,j)
              b12  += etemp
#
              etemp = e1bj(b,j)*e3bj(b,j)
              b13  += etemp
```

```
#
              etemp = e1bj(b,j)*e4bj(b,j)
              b14  += etemp
#
#             b1x x=2,4
#             ---------
#
              etemp = e2bj(b,j)*e2bj(b,j)
              b22  += etemp
#
              etemp = e2bj(b,j)*e3bj(b,j)
              b23  += etemp
#
              etemp = e2bj(b,j)*e4bj(b,j)
              b24  += etemp
#
#             b1x x=3,4
#             ---------
#
              etemp = e3bj(b,j)*e3bj(b,j)
              b33  += etemp
#
              etemp = e3bj(b,j)*e4bj(b,j)
              b34  += etemp
#
#             b1x x=4,4
#             ---------
#
              etemp = e4bj(b,j)*e4bj(b,j)
              b44  += etemp
#
     ENDPARDO b, j
#
     execute sip_barrier
#
#    Collectively sum B-matrix elements.
#    -----------------------------------
#
     collective Tb11 += b11
     collective Tb12 += b12
     collective Tb13 += b13
     collective Tb14 += b14
     collective Tb22 += b22
     collective Tb23 += b23
     collective Tb24 += b24
     collective Tb33 += b33
     collective Tb34 += b34
     collective Tb44 += b44
     execute server_barrier
#
#    Now the unique elements of the 'B-matrix' have been computed and the
array filled out.
#    ------------------------------------------------------------------------
--------------
#
#    Put the elements of the 'B-matrix', which have been computed as scalars
into the
```

```
#    R-matrix.
#    ------------------------------------------------------------------------
--------

     CALL SETUP_DIIS
#
     execute compute_diis # --> New instruction
#
     c1 = Tb11
     c2 = Tb22
     c3 = Tb33
     c4 = Tb44
     execute print_scalar c1
     execute print_scalar c2
     execute print_scalar c3
     execute print_scalar c4
     execute server_barrier
     execute sip_barrier
#
#    Done computing the c-vector.
#    ----------------------------
#
#    Form the updated amplitudes using the c-vector.
#    -----------------------#
#
#    Form Dai_old.
#    -------------
#
     PARDO a, i
#
#          REQUEST/GET amplitude data from all 3 previous iterations.
#          ----------------------------------------------------------
#
           GET D0ai(a,i)     # kiter-4 amplitudes
           GET D1ai(a,i)     # kiter-3 amplitudes
           GET D2ai(a,i)     # kiter-2 amplitudes
           GET D3ai(a,i)     # kiter-1 amplitudes
#
           GET e1ai(a,i)     # kiter-3 amplitudes
           GET e2ai(a,i)     # kiter-2 amplitudes
           GET e3ai(a,i)     # kiter-1 amplitudes
           GET e4ai(a,i)     # kiter-0 amplitudes
#
#          Compute contributions to updated amplitudes --> tai_old.
#          ---------------------------------------------------------
#
           t1ai(a,i)     = d0ai(a,i)
           t1ai(a,i)    += e1ai(a,i)
           t1ai(a,i)    *= c1
           tai(a,i)      = t1ai(a,i)
#
           t1ai(a,i)     = d1ai(a,i)
           t1ai(a,i)    += e2ai(a,i)
           t1ai(a,i)    *= c2
           tai(a,i)     += t1ai(a,i)
#
           t1ai(a,i)     = d2ai(a,i)
```

```
            t1ai(a,i)     += e3ai(a,i)
            t1ai(a,i)     *= c3
            tai(a,i)      += t1ai(a,i)
#
            t1ai(a,i)      = d3ai(a,i)
            t1ai(a,i)     += e4ai(a,i)
            t1ai(a,i)     *= c4
            tai(a,i)      += t1ai(a,i)
#
            PUT Paiold_aa(a,i) = tai(a,i)
#
      ENDPARDO a, i
#
#     Form tbj_old.
#     -------------
#
      PARDO b, j
#
#           REQUEST/GET amplitude data from all 3 previous iterations.
#           ----------------------------------------------------------
#
            GET D0bj(b,j)      # kiter-4 amplitudes
            GET D1bj(b,j)      # kiter-3 amplitudes
            GET D2bj(b,j)      # kiter-2 amplitudes
            GET D3bj(b,j)      # kiter-1 amplitudes
#
            GET e1bj(b,j)      # kiter-3 amplitudes
            GET e2bj(b,j)      # kiter-2 amplitudes
            GET e3bj(b,j)      # kiter-1 amplitudes
            GET e4bj(b,j)      # kiter-0 amplitudes
#
#           Compute contributions to updated amplitudes --> tbj_old.
#           --------------------------------------------------------
#
            t1bj(b,j)      = d0bj(b,j)
            t1bj(b,j)     += e1bj(b,j)
            t1bj(b,j)     *= c1
            tbj(b,j)       = t1bj(b,j)
#
            t1bj(b,j)      = d1bj(b,j)
            t1bj(b,j)     += e2bj(b,j)
            t1bj(b,j)     *= c2
            tbj(b,j)      += t1bj(b,j)
#
            t1bj(b,j)      = d2bj(b,j)
            t1bj(b,j)     += e3bj(b,j)
            t1bj(b,j)     *= c3
            tbj(b,j)      += t1bj(b,j)
#
            t1bj(b,j)      = d3bj(b,j)
            t1bj(b,j)     += e4bj(b,j)
            t1bj(b,j)     *= c4
            tbj(b,j)      += t1bj(b,j)
#
            PUT Paiold_bb(b,j) = tbj(b,j)
#
      ENDPARDO b, j
```

```
#
     execute sip_barrier
#
     ENDPROC DIIS3
#    -------------
#
# ------------------------------------------------------------------------
------
#
# ------------------------------------------------------------------------
------
#
     PROC DIIS4
#    ----------
#
#    Zero out scalars.
#    -----------------
#
     CALL ZERO_DSCALAR
#
     execute sip_barrier
#
#    Determine the 'B-matrix'.
#    -------------------------
#
#    Compute contributions due to dai amplitudes.
#    --------------------------------------------
#
     PARDO a, i
#
#          REQUEST/GET amplitude data from all 4 previous iterations.
#          ----------------------------------------------------------
#
           GET e1ai(a,i)    # kiter-4 amplitudes
           GET e2ai(a,i)    # kiter-3 amplitudes
           GET e3ai(a,i)    # kiter-2 amplitudes
           GET e4ai(a,i)    # kiter-1 amplitudes
           GET e5ai(a,i)    # kiter-0 amplitudes
#
#          Compute contributions to the 'B-matrix'.
#          ----------------------------------------
#
#             b1x x=1,5
#             ---------
#
              etemp = e1ai(a,i)*e1ai(a,i)
              b11  += etemp
#
              etemp = e1ai(a,i)*e2ai(a,i)
              b12  += etemp
#
              etemp = e1ai(a,i)*e3ai(a,i)
              b13  += etemp
#
              etemp = e1ai(a,i)*e4ai(a,i)
              b14  += etemp
#
```

```
               etemp = e1ai(a,i)*e5ai(a,i)
               b15  += etemp
#
#              b1x x=2,5
#              ---------
#
               etemp = e2ai(a,i)*e2ai(a,i)
               b22  += etemp
#
               etemp = e2ai(a,i)*e3ai(a,i)
               b23  += etemp
#
               etemp = e2ai(a,i)*e4ai(a,i)
               b24  += etemp
#
               etemp = e2ai(a,i)*e5ai(a,i)
               b25  += etemp
#
#              b1x x=3,5
#              ---------
#
               etemp = e3ai(a,i)*e3ai(a,i)
               b33  += etemp
#
               etemp = e3ai(a,i)*e4ai(a,i)
               b34  += etemp
#
               etemp = e3ai(a,i)*e5ai(a,i)
               b35  += etemp
#
#              b1x x=4,5
#              ---------
#
               etemp = e4ai(a,i)*e4ai(a,i)
               b44  += etemp
#
               etemp = e4ai(a,i)*e5ai(a,i)
               b45  += etemp
#
#              b1x x=5,5
#              ---------
#
               etemp = e5ai(a,i)*e5ai(a,i)
               b55  += etemp
#
     ENDPARDO a, i
#
#    Compute contributions due to dbj amplitudes.
#    --------------------------------------------
#
     PARDO b, j
#
#          REQUEST/GET amplitude data from all 3 previous iterations.
#          ----------------------------------------------------------
#
           GET e1bj(b,j)    # kiter-4 amplitudes
           GET e2bj(b,j)    # kiter-3 amplitudes
```

```
            GET e3bj(b,j)    # kiter-2 amplitudes
            GET e4bj(b,j)    # kiter-1 amplitudes
            GET e5bj(b,j)    # kiter-0 amplitudes
#
#           Compute contributions to the 'B-matrix'.
#           ----------------------------------------
#
#              b1x x=1,5
#              ---------
#
               etemp = e1bj(b,j)*e1bj(b,j)
               b11  += etemp
#
               etemp = e1bj(b,j)*e2bj(b,j)
               b12  += etemp
#
               etemp = e1bj(b,j)*e3bj(b,j)
               b13  += etemp
#
               etemp = e1bj(b,j)*e4bj(b,j)
               b14  += etemp
#
               etemp = e1bj(b,j)*e5bj(b,j)
               b15  += etemp
#
#              b1x x=2,5
#              ---------
#
               etemp = e2bj(b,j)*e2bj(b,j)
               b22  += etemp
#
               etemp = e2bj(b,j)*e3bj(b,j)
               b23  += etemp
#
               etemp = e2bj(b,j)*e4bj(b,j)
               b24  += etemp
#
               etemp = e2bj(b,j)*e5bj(b,j)
               b25  += etemp
#
#              b1x x=3,5
#              ---------
#
               etemp = e3bj(b,j)*e3bj(b,j)
               b33  += etemp
#
               etemp = e3bj(b,j)*e4bj(b,j)
               b34  += etemp
#
               etemp = e3bj(b,j)*e5bj(b,j)
               b35  += etemp
#
#              b1x x=4,5
#              ---------
#
               etemp = e4bj(b,j)*e4bj(b,j)
               b44  += etemp
```

```
#
              etemp = e4bj(b,j)*e5bj(b,j)
              b45  += etemp
#
#             b1x x=5,5
#             ---------
#
              etemp = e5bj(b,j)*e5bj(b,j)
              b55  += etemp
#
     ENDPARDO b, j
#
     execute sip_barrier
#
#    Collectively sum B-matrix elements.
#    -----------------------------------
#
     collective Tb11 += b11
     collective Tb12 += b12
     collective Tb13 += b13
     collective Tb14 += b14
     collective Tb15 += b15
     collective Tb22 += b22
     collective Tb23 += b23
     collective Tb24 += b24
     collective Tb25 += b25
     collective Tb33 += b33
     collective Tb34 += b34
     collective Tb35 += b35
     collective Tb44 += b44
     collective Tb45 += b45
     collective Tb55 += b55
     execute server_barrier
#
#    Now the unique elements of the 'B-matrix' have been computed and the
array filled out.
#    ------------------------------------------------------------------------
--------------
#
#    Put the elements of the 'B-matrix', which have been computed as scalars
into the
#    R-matrix.
#    ------------------------------------------------------------------------
--------

     CALL SETUP_DIIS
#
     execute compute_diis # --> New instruction
#
     c1 = Tb11
     c2 = Tb22
     c3 = Tb33
     c4 = Tb44
     c5 = Tb55
     execute print_scalar c1
     execute print_scalar c2
     execute print_scalar c3
```

```
      execute print_scalar c4
      execute print_scalar c5
      execute sip_barrier
#
#     Done computing the c-vector.
#     ----------------------------
#
#     Form the updated amplitudes using the c-vector.
#     -----------------------#
#     Form Dai_old.
#     -------------
#
      PARDO a, i
#
#           REQUEST/GET amplitude data from all 3 previous iterations.
#           ----------------------------------------------------------
#
            GET D0ai(a,i)      # kiter-5 amplitudes
            GET D1ai(a,i)      # kiter-4 amplitudes
            GET D2ai(a,i)      # kiter-3 amplitudes
            GET D3ai(a,i)      # kiter-2 amplitudes
            GET D4ai(a,i)      # kiter-1 amplitudes
#
            GET e1ai(a,i)      # kiter-4 amplitudes
            GET e2ai(a,i)      # kiter-3 amplitudes
            GET e3ai(a,i)      # kiter-2 amplitudes
            GET e4ai(a,i)      # kiter-1 amplitudes
            GET e5ai(a,i)      # kiter-0 amplitudes
#
#           Compute contributions to updated amplitudes --> tai_old.
#           ---------------------------------------------------------
#
            t1ai(a,i)      = d0ai(a,i)
            t1ai(a,i)     += e1ai(a,i)
            t1ai(a,i)     *= c1
            tai(a,i)       = t1ai(a,i)
#
            t1ai(a,i)      = d1ai(a,i)
            t1ai(a,i)     += e2ai(a,i)
            t1ai(a,i)     *= c2
            tai(a,i)      += t1ai(a,i)
#
            t1ai(a,i)      = d2ai(a,i)
            t1ai(a,i)     += e3ai(a,i)
            t1ai(a,i)     *= c3
            tai(a,i)      += t1ai(a,i)
#
            t1ai(a,i)      = d3ai(a,i)
            t1ai(a,i)     += e4ai(a,i)
            t1ai(a,i)     *= c4
            tai(a,i)      += t1ai(a,i)
#
            t1ai(a,i)      = d4ai(a,i)
            t1ai(a,i)     += e5ai(a,i)
            t1ai(a,i)     *= c5
            tai(a,i)      += t1ai(a,i)
#
```

```
          PUT Paiold_aa(a,i) = tai(a,i)
#
     ENDPARDO a, i
#
#    Form tbj_old.
#    -------------
#
     PARDO b, j
#
#         REQUEST/GET amplitude data from all 3 previous iterations.
#         ----------------------------------------------------------
#
          GET D0bj(b,j)     # kiter-5 amplitudes
          GET D1bj(b,j)     # kiter-4 amplitudes
          GET D2bj(b,j)     # kiter-3 amplitudes
          GET D3bj(b,j)     # kiter-2 amplitudes
          GET D4bj(b,j)     # kiter-1 amplitudes
#
          GET e1bj(b,j)     # kiter-4 amplitudes
          GET e2bj(b,j)     # kiter-3 amplitudes
          GET e3bj(b,j)     # kiter-2 amplitudes
          GET e4bj(b,j)     # kiter-1 amplitudes
          GET e5bj(b,j)     # kiter-0 amplitudes
#
#         Compute contributions to updated amplitudes --> tbj_old.
#         ---------------------------------------------------------
#
          t1bj(b,j)     = d0bj(b,j)
          t1bj(b,j)    += e1bj(b,j)
          t1bj(b,j)    *= c1
          tbj(b,j)      = t1bj(b,j)
#
          t1bj(b,j)     = d1bj(b,j)
          t1bj(b,j)    += e2bj(b,j)
          t1bj(b,j)    *= c2
          tbj(b,j)     += t1bj(b,j)
#
          t1bj(b,j)     = d2bj(b,j)
          t1bj(b,j)    += e3bj(b,j)
          t1bj(b,j)    *= c3
          tbj(b,j)     += t1bj(b,j)
#
          t1bj(b,j)     = d3bj(b,j)
          t1bj(b,j)    += e4bj(b,j)
          t1bj(b,j)    *= c4
          tbj(b,j)     += t1bj(b,j)
#
          t1bj(b,j)     = d4bj(b,j)
          t1bj(b,j)    += e5bj(b,j)
          t1bj(b,j)    *= c5
          tbj(b,j)     += t1bj(b,j)
#
          PUT Paiold_bb(b,j) = tbj(b,j)
#
     ENDPARDO b, j
#
     execute sip_barrier
```

```
#
     ENDPROC DIIS4
#    -------------
#
# ------------------------------------------------------------------------
------
#
# ------------------------------------------------------------------------
------
#
     PROC MOVE4
#    ----------
#
#    0 --> 1
#    ------------------------------------------------
#
     PARDO a, i
           GET d1ai(a,i)
           PUT d0ai(a,i) = d1ai(a,i)
     ENDPARDO a, i
#
     PARDO b, j
           GET d1bj(b,j)
           PUT d0bj(b,j) = d1bj(b,j)
     ENDPARDO b, j
#
     execute sip_barrier
#
#    2 --> 1
#    ------------------------------------------------
#
     PARDO a, i
           GET e2ai(a,i)
           PUT e1ai(a,i) = e2ai(a,i)
     ENDPARDO a, i
#
     PARDO b, j
           GET e2bj(b,j)
           PUT e1bj(b,j) = e2bj(b,j)
     ENDPARDO b, j
#
     PARDO a, i
           GET d2ai(a,i)
           PUT d1ai(a,i) = d2ai(a,i)
     ENDPARDO a, i
#
     PARDO b, j
           GET d2bj(b,j)
           PUT d1bj(b,j) = d2bj(b,j)
     ENDPARDO b, j
#
     execute sip_barrier
#
#    3 --> 2
#    ------------------------------------------------
#
     PARDO a, i
```

```
            GET e3ai(a,i)
            PUT e2ai(a,i) = e3ai(a,i)
      ENDPARDO a, i
#
      PARDO b, j
            GET e3bj(b,j)
            PUT e2bj(b,j) = e3bj(b,j)
      ENDPARDO b, j
#
      PARDO a, i
            GET d3ai(a,i)
            PUT d2ai(a,i) = d3ai(a,i)
      ENDPARDO a, i
#
      PARDO b, j
            GET d3bj(b,j)
            PUT d2bj(b,j) = d3bj(b,j)
      ENDPARDO b, j
#
      execute sip_barrier
#
#     4 --> 3
#     ------------------------------------------------
#
      PARDO a, i
            GET e4ai(a,i)
            PUT e3ai(a,i) = e4ai(a,i)
      ENDPARDO a, i
#
      PARDO b, j
            GET e4bj(b,j)
            PUT e3bj(b,j) = e4bj(b,j)
      ENDPARDO b, j
#
      PARDO a, i
            GET d4ai(a,i)
            PUT d3ai(a,i) = d4ai(a,i)
      ENDPARDO a, i
#
      PARDO b, j
            GET d4bj(b,j)
            PUT d3bj(b,j) = d4bj(b,j)
      ENDPARDO b, j
#
      execute sip_barrier
#
#     5 --> 4
#     ------------------------------------------------
#
      PARDO a, i
            GET e5ai(a,i)
            PUT e4ai(a,i) = e5ai(a,i)
      ENDPARDO a, i
#
      PARDO b, j
            GET e5bj(b,j)
            PUT e4bj(b,j) = e5bj(b,j)
```

```
      ENDPARDO b, j
#
      PARDO a, i
            GET Paiold_aa(a,i)
            PUT d4ai(a,i) = Paiold_aa(a,i)
      ENDPARDO a, i
#
      PARDO b, j
            GET Paiold_bb(b,j)
            PUT d4bj(b,j) = Paiold_bb(b,j)
      ENDPARDO b, j
#
      execute sip_barrier
#
      ENDPROC MOVE4
#     -------------
#
#################################################################################
#
#
#                           START OF MAIN PROGRAM
#
#################################################################################
#
#
# Transform integrals
# -------------------
#
       execute sip_barrier
       CALL TRAN_UHF
       allocate La(*,*)
       allocate Lb(*,*)
       DO mu
       DO p
          La(p,mu) = ca(mu,p)
       ENDDO p
       ENDDO mu
       DO mu
       DO q
          Lb(q,mu) = cb(mu,q)
       ENDDO q
       ENDDO mu
#
# Create one-particle arrays
# --------------------------
#
       CREATE Pij_aa
       CREATE Pij_bb
       CREATE Pab_aa
       CREATE Pab_bb
       CREATE Lai_aa
       CREATE Lai_bb
       CREATE Painew_aa
       CREATE Paiold_aa
       CREATE Painew_bb
       CREATE Paiold_bb
       CREATE Wab_aa
```

```
      CREATE Wab_bb
      CREATE Wij_aa
      CREATE Wij_bb
      CREATE Wai_aa
      CREATE Wai_bb
      CREATE P2_ao
      CREATE P2A_ao
      CREATE P2B_ao
      CREATE W2_ao
      CREATE Paa_ao
      CREATE Pbb_ao
      CREATE Yxi
      CREATE Yxj
      CREATE WHFa
      CREATE WHFb
      CREATE DHFa
      CREATE DHFb
      CREATE DHF
      execute sip_barrier
#
#     Compute the HF contribution to the weighted density matrix
#     ----------------------------------------------------------
#
      call WHFDENS
      execute sip_barrier
#
# First compute the occupied-occupied block of the density correction
# -------------------------------------------------------------------
#
#     AAAA/AAAA piece
#     ---------------
      PARDO a, i, a1, i2
#
            REQUEST                     VSpipi(a,i,a1,i2) i
            Tpipi(a,i,a1,i2)          = VSpipi(a,i,a1,i2)
            execute energy_denominator Tpipi(a,i,a1,i2)
#
            DO i1
#
               REQUEST                    VSpipi(a,i1,a1,i2) i1
               T1pipi(a,i1,a1,i2)        = VSpipi(a,i1,a1,i2)
               execute energy_denominator T1pipi(a,i1,a1,i2)
#
               Tii(i,i1)                  = Tpipi(a,i,a1,i2)*T1pipi(a,i1,a1,i2)
               Tii(i,i1)                 *= -0.5
               PUT Pij_aa(i,i1)          += Tii(i,i1)
#
            ENDDO i1
#
      ENDPARDO a, i, a1, i2
#
#     AABB/AABB piece
#     ---------------
      PARDO a, i, b, j
#
            REQUEST                     Vpiqj(a,i,b,j) i
            Tpiqj(a,i,b,j)            = Vpiqj(a,i,b,j)
```

```
            execute energy_denominator Tpiqj(a,i,b,j)
#
            DO i1
#
               REQUEST                    Vpiqj(a,i1,b,j) j
               T1piqj(a,i1,b,j)         = Vpiqj(a,i1,b,j)
               execute energy_denominator T1piqj(a,i1,b,j)
#
               Tii(i,i1)                 = Tpiqj(a,i,b,j)*T1piqj(a,i1,b,j)
               Tii(i,i1)                *= -1.0
               PUT Pij_aa(i,i1)         += Tii(i,i1)
#
            ENDDO i1
#
      ENDPARDO a, i, b, j
#
#     BBBB/BBBB piece
#     ---------------
      PARDO b, j, b1, j2
#
            REQUEST                    VSqjqj(b,j,b1,j2) j
            Tqjqj(b,j,b1,j2)         = VSqjqj(b,j,b1,j2)
            execute energy_denominator Tqjqj(b,j,b1,j2)
#
            DO j1
#
               REQUEST                    VSqjqj(b,j1,b1,j2) j1
               T1qjqj(b,j1,b1,j2)       = VSqjqj(b,j1,b1,j2)
               execute energy_denominator T1qjqj(b,j1,b1,j2)
#
               Tjj(j,j1)                 = Tqjqj(b,j,b1,j2)*T1qjqj(b,j1,b1,j2)
               Tjj(j,j1)                *= -0.5
               PUT Pij_bb(j,j1)         += Tjj(j,j1)
#
            ENDDO j1
#
      ENDPARDO b, j, b1, j2
#
#     BBAA/BBAA piece
#     ---------------
      PARDO b, j, a, i
#
            REQUEST                    Vpiqj(a,i,b,j) i
            Tpiqj(a,i,b,j)           = Vpiqj(a,i,b,j)
            execute energy_denominator Tpiqj(a,i,b,j)
#
            DO j1
#
               REQUEST                    Vpiqj(a,i,b,j1) i
               T1piqj(a,i,b,j1)         = Vpiqj(a,i,b,j1)
               execute energy_denominator T1piqj(a,i,b,j1)
#
               Tjj(j,j1)                 = Tpiqj(a,i,b,j)*T1piqj(a,i,b,j1)
               Tjj(j,j1)                *= -1.0
               PUT Pij_bb(j,j1)         += Tjj(j,j1)
#
            ENDDO j1
```

```
#
      ENDPARDO b, j, a, i
#
# Done compute the occupied-occupied block of the density correction
# ---------------------------------------------------------------------
#
# Compute the virtual-virtual block of the density correction
# ---------------------------------------------------------------------
#
#     AAAA/AAAA piece
#     ---------------
#
      PARDO a, a2, i, i1
#
            REQUEST                     VSpipi(a,i,a2,i1) i
            Tpipi(a,i,a2,i1)          = VSpipi(a,i,a2,i1)
            execute energy_denominator Tpipi(a,i,a2,i1)
#
            DO a1
#
               REQUEST                     VSpipi(a1,i,a2,i1) i
               T1pipi(a1,i,a2,i1)        = VSpipi(a1,i,a2,i1)
               execute energy_denominator T1pipi(a1,i,a2,i1)
#
               Taa(a,a1) = Tpipi(a,i,a2,i1)*T1pipi(a1,i,a2,i1)
               Taa(a,a1) *= 0.5
               PUT Pab_aa(a,a1) += Taa(a,a1)
#
            ENDDO a1
#
      ENDPARDO a, a2, i, i1
#
#     AABB/AABB piece
#     ---------------
#
      PARDO a, b, i, j
#
            REQUEST                     Vpiqj(a,i,b,j) i
            Tpiqj(a,i,b,j)            = Vpiqj(a,i,b,j)
            execute energy_denominator Tpiqj(a,i,b,j)
#
            DO a1
#
               REQUEST                     Vpiqj(a1,i,b,j) i
               T1piqj(a1,i,b,j)          = Vpiqj(a1,i,b,j)
               execute energy_denominator T1piqj(a1,i,b,j)
#
               Taa(a,a1) = Tpiqj(a,i,b,j)*T1piqj(a1,i,b,j)
               PUT Pab_aa(a,a1) += Taa(a,a1)
#
            ENDDO a1
#
      ENDPARDO a, b, i, j
#
#     BBBB/BBBB piece
#     ---------------
#
```

```
      PARDO b, b2, j, j1
#
            REQUEST                     VSqjqj(b,j,b2,j1) j
            Tqjqj(b,j,b2,j1)          = VSqjqj(b,j,b2,j1)
            execute energy_denominator Tqjqj(b,j,b2,j1)
#
            DO b1
#
               REQUEST                     VSqjqj(b1,j,b2,j1) j
               T1qjqj(b1,j,b2,j1)        = VSqjqj(b1,j,b2,j1)
               execute energy_denominator T1qjqj(b1,j,b2,j1)
#
               Tbb(b,b1) = Tqjqj(b,j,b2,j1)*T1qjqj(b1,j,b2,j1)
               Tbb(b,b1) *= 0.5
               PUT Pab_bb(b,b1) += Tbb(b,b1)
#
            ENDDO b1
#
      ENDPARDO b, b2, j, j1
#
#     BBAA/BBAA piece
#     ---------------
#
      PARDO b, a, j, i
#
            REQUEST                     Vpiqj(a,i,b,j) j
            Tpiqj(a,i,b,j)            = Vpiqj(a,i,b,j)
            execute energy_denominator Tpiqj(a,i,b,j)
#
            DO b1
#
               REQUEST                     Vpiqj(a,i,b1,j) j
               T1piqj(a,i,b1,j)          = Vpiqj(a,i,b1,j)
               execute energy_denominator T1piqj(a,i,b1,j)
#
               Tbb(b,b1) = Tpiqj(a,i,b,j)*T1piqj(a,i,b1,j)
               PUT Pab_bb(b,b1) += Tbb(b,b1)
#
            ENDDO b1
#
      ENDPARDO b, a, j, i
#
# End compute the virtual-virtual block of the density correction
# ---------------------------------------------------------------------
      execute sip_barrier
#
# Backtransform Pab to be used in the 'direct' contribution to Lai
# ---------------------------------------------------------------
#
#     Transform Pab_aa
#     ----------------
      PARDO a, a1
#
            GET Pab_aa(a,a1)
#
            DO mu
#
```

```
               Ixa(mu,a1) = Pab_aa(a,a1)*ca(mu,a)
#
               DO nu
#
                  Ixx(mu,nu)          = Ixa(mu,a1)*ca(nu,a1)
                  PUT Paa_ao(mu,nu) += Ixx(mu,nu)
#
               ENDDO nu
#
            ENDDO mu
#
      ENDPARDO a, a1
#
#     Transform Pab_bb
#     ----------------
      PARDO b, b1
#
            GET Pab_bb(b,b1)
#
            DO mu
#
               Ixb(mu,b1) = Pab_bb(b,b1)*cb(mu,b)
#
               DO nu
#
                  Ixx(mu,nu)          = Ixb(mu,b1)*cb(nu,b1)
                  PUT Pbb_ao(mu,nu) += Ixx(mu,nu)
#
               ENDDO nu
#
            ENDDO mu
#
      ENDPARDO b, b1
      execute sip_barrier
#
# Compute the right-hand side of Eq. 10 --> Lai_aa
# ------------------------------------------------
#
#     Compte the 'direct' contributions
#     ---------------------------------
      CALL LAIAO1
#
#     Second-term
#     -----------
#
      PARDO a, a1, i1, i2
#
            REQUEST                    VSpipi(a,i1,a1,i2) i1
            Tpipi(a,i1,a1,i2)        = VSpipi(a,i1,a1,i2)
            execute energy_denominator Tpipi(a,i1,a1,i2)
#
            DO i
#
               REQUEST            VSpipi(a1,i2,i,i1) i
               Tai(a,i)         = Tpipi(a,i1,a1,i2)*VSpipi(a1,i2,i,i1)
               Tai(a,i)        *= 0.5
               PUT Lai_aa(a,i) += Tai(a,i)
```

```
#
            ENDDO i
#
      ENDPARDO a, a1, i1, i2
#
      PARDO a, b, i1, j
#
            REQUEST                     Vpiqj(a,i1,b,j) j
            Tpiqj(a,i1,b,j)           = Vpiqj(a,i1,b,j)
            execute energy_denominator Tpiqj(a,i1,b,j)
#
            DO i
#
               REQUEST            Vpiqj(i,i1,b,j) i
               Tai(a,i)          = Tpiqj(a,i1,b,j)*Vpiqj(i,i1,b,j)
               PUT Lai_aa(a,i) += Tai(a,i)
#
            ENDDO i
#
      ENDPARDO a, b, i1, j
#
#     Fourth-term
#     -----------
#
      PARDO i, i1, i2, a
#
            REQUEST            VSpipi(a,i,i1,i2) i
            GET                Pij_aa(i1,i2)
#
            Tai(a,i)          = VSpipi(a,i,i1,i2)*Pij_aa(i1,i2)
            Tai(a,i)         *= -1.0
            PUT Lai_aa(a,i) += Tai(a,i)
#
      ENDPARDO i, i1, i2, a
#
      PARDO i, j, j1, a
#
            REQUEST            Vpiqj(a,i,j,j1) i
            GET                Pij_bb(j,j1)
#
            Tai(a,i)          = Vpiqj(a,i,j,j1)*Pij_bb(j,j1)
            Tai(a,i)         *= -1.0
            PUT Lai_aa(a,i) += Tai(a,i)
#
      ENDPARDO i, j, j1, a
#
# Compute the right-hand side of Eq. 10 --> Lai_bb
# ------------------------------------------------
#
#     Second-term
#     -----------
#
      PARDO b, b1, j1, j2
#
            REQUEST                     VSqjqj(b,j1,b1,j2) j1
            Tqjqj(b,j1,b1,j2)         = VSqjqj(b,j1,b1,j2)
            execute energy_denominator Tqjqj(b,j1,b1,j2)
```

```
#
            DO j
#
               REQUEST            VSqjqj(b1,j2,j,j1) j
               Tbj(b,j)         = Tqjqj(b,j1,b1,j2)*VSqjqj(b1,j2,j,j1)
               Tbj(b,j)        *= 0.5
               PUT Lai_bb(b,j) += Tbj(b,j)
#
            ENDDO j
#
      ENDPARDO b, b1, j1, j2
#
      PARDO a, b, i, j1
#
            REQUEST                    Vpiqj(a,i,b,j1) i
            Tpiqj(a,i,b,j1)          = Vpiqj(a,i,b,j1)
            execute energy_denominator Tpiqj(a,i,b,j1)
#
            DO j
#
               REQUEST            Vpiqj(a,i,j,j1) i
               Tbj(b,j)         = Tpiqj(a,i,b,j1)*Vpiqj(a,i,j,j1)
               PUT Lai_bb(b,j) += Tbj(b,j)
#
            ENDDO j
#
      ENDPARDO a, b, i, j1
#
#     Fourth-term
#     -----------
#
      PARDO j, j1, j2, b
#
            REQUEST            VSqjqj(b,j,j1,j2) j
            GET                Pij_bb(j1,j2)
#
            Tbj(b,j)         = VSqjqj(b,j,j1,j2)*Pij_bb(j1,j2)
            Tbj(b,j)        *= -1.0
            PUT Lai_bb(b,j) += Tbj(b,j)
#
      ENDPARDO j, j1, j2, b
#
      PARDO j, i, i1, b
#
            REQUEST            Vpiqj(i,i1,b,j) i
            GET                Pij_aa(i,i1)
#
            Tbj(b,j)         = Vpiqj(i,i1,b,j)*Pij_aa(i,i1)
            Tbj(b,j)        *= -1.0
            PUT Lai_bb(b,j) += Tbj(b,j)
#
      ENDPARDO j, i, i1, b
#
# Done compute the right-hand side of Eq. 10 --> Lai_bb
# -----------------------------------------------------
#
      execute sip_barrier
```

```
#
# Compute the occupied-virtual block of correlated density
# --------------------------------------------------------
#
# Create error arrays used in the DIIS procedure.
# -----------------------------------------------
#
      CREATE D0ai
      CREATE D1ai
      CREATE D2ai
      CREATE D3ai
      CREATE D4ai
#
      CREATE D0bj
      CREATE D1bj
      CREATE D2bj
      CREATE D3bj
      CREATE D4bj
#
      CREATE e1ai
      CREATE e2ai
      CREATE e3ai
      CREATE e4ai
      CREATE e5ai
#
      CREATE e1bj
      CREATE e2bj
      CREATE e3bj
      CREATE e4bj
      CREATE e5bj
#
# Get initial guess and form a few intermediates
# ----------------------------------------------
#
      eold = 0.0
      esum = 0.0
      ecrit = 0.0000000001
#
      PARDO a, a1, i, i1
#
            REQUEST                     VSpipi(a,i,a1,i1) i
            REQUEST                     Vaaii(a,a1,i1,i)  i
            REQUEST                     Viaai(i,a,a1,i1)  i
#
            Tpipi(a,i,a1,i1)          = VSpipi(a,i,a1,i1)
            T1pipi(a,i,a1,i1)         = Vaaii(a,a1,i1,i)
            T2pipi(a,i,a1,i1)         = Viaai(i,a,a1,i1)
#
            Tpipi(a,i,a1,i1)         -= T1pipi(a,i,a1,i1)
            Tpipi(a,i,a1,i1)         += T2pipi(a,i,a1,i1)
            PREPARE ASpipi(a,i,a1,i1) = Tpipi(a,i,a1,i1)
#
      ENDPARDO a, a1, i, i1
#
      PARDO a, b, i, j
#
            REQUEST                     Vpiqj(a,i,b,j) j
```

```
            REQUEST                      Viabj(i,a,b,j) j
#
            Tpiqj(a,i,b,j)             = Vpiqj(a,i,b,j)
            T2piqj(a,i,b,j)            = Viabj(i,a,b,j)
#
            Tpiqj(a,i,b,j)            += T2piqj(a,i,b,j)
#
            PREPARE Apiqj(a,i,b,j) = Tpiqj(a,i,b,j)
#
      ENDPARDO a, b, i, j
#
      PARDO b, b1, j, j1
#
            REQUEST                      VSqjqj(b,j,b1,j1) j
            REQUEST                      Vbbjj(b,b1,j1,j) j
            REQUEST                      Vjbbj(j,b,b1,j1) j
#
            Tqjqj(b,j,b1,j1)           = VSqjqj(b,j,b1,j1)
            T1qjqj(b,j,b1,j1)          = Vbbjj(b,b1,j1,j)
            T2qjqj(b,j,b1,j1)          = Vjbbj(j,b,b1,j1)
#
            Tqjqj(b,j,b1,j1)          -= T1qjqj(b,j,b1,j1)
            Tqjqj(b,j,b1,j1)          += T2qjqj(b,j,b1,j1)
            PREPARE ASqjqj(b,j,b1,j1) = Tqjqj(b,j,b1,j1)
#
      ENDPARDO b, b1, j, j1
#
      PARDO a, i
#
            GET                          Lai_aa(a,i)
            Tai(a,i)                   = Lai_aa(a,i)
            Tai(a,i)                  *= -1.0
            execute energy_denominator Tai(a,i)
            etemp                      = Tai(a,i)*Tai(a,i)
            esum                      += etemp
            PUT Paiold_aa(a,i)        = Tai(a,i)
#
      ENDPARDO a, i
#
      PARDO b, j
#
            GET                          Lai_bb(b,j)
            Tbj(b,j)                   = Lai_bb(b,j)
            Tbj(b,j)                  *= -1.0
            execute energy_denominator Tbj(b,j)
            etemp                      = Tbj(b,j)*Tbj(b,j)
            esum                      += etemp
            PUT Paiold_bb(b,j)         = Tbj(b,j)
#
      ENDPARDO b, j
#
      execute sip_barrier
      execute server_barrier
      collective eold += esum
#
# Done initial guess
# ------------------
```

```
#
#     Start iterations
#     ----------------
#
      DO kiter
#
         PARDO a, i
#
               GET                          Lai_aa(a,i)
               Tai(a,i)                   = Lai_aa(a,i)
               Tai(a,i)                  *= -1.0
               PUT Painew_aa(a,i)        += Tai(a,i)
#
         ENDPARDO a, i
#
         PARDO b, j
#
               GET                          Lai_bb(b,j)
               Tbj(b,j)                   = Lai_bb(b,j)
               Tbj(b,j)                  *= -1.0
               PUT Painew_bb(b,j)        += Tbj(b,j)
#
         ENDPARDO b, j
#
         PARDO a, a1, i, i1
#
              #REQUEST                     VSpipi(a,i,a1,i1) i
              #REQUEST                     Vaaii(a,a1,i1,i)  i
              #REQUEST                     Viaai(i,a,a1,i1)  i
               REQUEST                     ASpipi(a,i,a1,i1) i
               GET                         Paiold_aa(a1,i1)
#
              #Tpipi(a,i,a1,i1)           = VSpipi(a,i,a1,i1)
              #T1pipi(a,i,a1,i1)          = Vaaii(a,a1,i1,i)
              #T2pipi(a,i,a1,i1)          = Viaai(i,a,a1,i1)
#
              #Tpipi(a,i,a1,i1)          -= T1pipi(a,i,a1,i1)
              #Tpipi(a,i,a1,i1)          += T2pipi(a,i,a1,i1)
#
               Tai(a,i)                   = ASpipi(a,i,a1,i1)*Paiold_aa(a1,i1)
               PUT Painew_aa(a,i)        += Tai(a,i)
#
         ENDPARDO a, a1, i, i1
#
         PARDO a, b, i, j
#
              #REQUEST                     Vpiqj(a,i,b,j) j
              #REQUEST                     Viabj(i,a,b,j) j
               REQUEST                     Apiqj(a,i,b,j) j
               GET                         Paiold_bb(b,j)
               GET                         Paiold_aa(a,i)
#
              #Tpiqj(a,i,b,j)             = Vpiqj(a,i,b,j)
              #T2piqj(a,i,b,j)            = Viabj(i,a,b,j)
              #Tpiqj(a,i,b,j)            += T2piqj(a,i,b,j)
#
               Tai(a,i)                   = Apiqj(a,i,b,j)*Paiold_bb(b,j)
```

```
             PUT Painew_aa(a,i)      += Tai(a,i)
#
            #Tpiqj(a,i,b,j)           = Vpiqj(a,i,b,j)
            #T2piqj(a,i,b,j)          = Viabj(i,a,b,j)
            #Tpiqj(a,i,b,j)          += T2piqj(a,i,b,j)
#
             Tbj(b,j)                 = Apiqj(a,i,b,j)*Paiold_aa(a,i)
             PUT Painew_bb(b,j)      += Tbj(b,j)
#
        ENDPARDO a, b, i, j
#
        PARDO b, b1, j, j1
#
            #REQUEST                    VSqjqj(b,j,b1,j1) j
            #REQUEST                    Vbbjj(b,b1,j1,j) j
            #REQUEST                    Vjbbj(j,b,b1,j1) j
             REQUEST                    ASqjqj(b,j,b1,j1) j
             GET                        Paiold_bb(b1,j1)
#
            #Tqjqj(b,j,b1,j1)          = VSqjqj(b,j,b1,j1)
            #T1qjqj(b,j,b1,j1)         = Vbbjj(b,b1,j1,j)
            #T2qjqj(b,j,b1,j1)         = Vjbbj(j,b,b1,j1)
#
            #Tqjqj(b,j,b1,j1)         -= T1qjqj(b,j,b1,j1)
            #Tqjqj(b,j,b1,j1)         += T2qjqj(b,j,b1,j1)
#
             Tbj(b,j)                 = ASqjqj(b,j,b1,j1)*Paiold_bb(b1,j1)
             PUT Painew_bb(b,j)      += Tbj(b,j)
#
        ENDPARDO b, b1, j, j1
#
        execute sip_barrier
#
#       Update error vector for diis
#       ----------------------------
#
        CALL UPDATE_PAI
#
        execute sip_barrier
#
        esum = 0.0
        enew = 0.0
        PARDO a, i
#
             GET                        Painew_aa(a,i)
             Tai(a,i)                  = Painew_aa(a,i)
             execute energy_denominator Tai(a,i)
             PUT                        Paiold_aa(a,i) = Tai(a,i)
             etemp                     = Painew_aa(a,i)*Painew_aa(a,i)
             esum                      += etemp
             Tai(a,i)                  = 0.0
             PUT Painew_aa(a,i)        = Tai(a,i)
#
        ENDPARDO a, i
#
        PARDO b, j
#
```

```
            GET                          Painew_bb(b,j)
            Tbj(b,j)                   = Painew_bb(b,j)
            execute energy_denominator Tbj(b,j)
            PUT                          Paiold_bb(b,j) = Tbj(b,j)
            etemp                      = Painew_bb(b,j)*Painew_bb(b,j)
            esum                      += etemp
            Tbj(b,j)                   = 0.0
            PUT Painew_bb(b,j)         = Tbj(b,j)
#
         ENDPARDO b, j
#
         execute sip_barrier
         collective enew += esum
#
#        Check on convergence
#        --------------------
#
         IF enew < eold
            ediff = eold - enew
            IF ediff < ecrit
               exit # kiter
            ENDIF
         ENDIF
#
         IF enew > eold
            ediff = enew - eold
            IF ediff < ecrit
               exit # kiter
            ENDIF
         ENDIF
#
#        Reset eold --> enew
#        -------------------
#
         eold = enew
#
         if kiter == 2
#
#           Get uptated amplitudes based on DIIS procedure.
#           -----------------------------------------------
#
            CALL DIIS1
#
         endif # kiter == 2
#
         if kiter == 3
#
#           Get uptated amplitudes based on DIIS procedure.
#           -----------------------------------------------
#
            CALL DIIS2
#
         endif # kiter == 3
#
         if kiter == 4
#
#           Get uptated amplitudes based on DIIS procedure.
```

```
#           ------------------------------------------------
#
            CALL DIIS3
#
         endif # kiter == 4
#
         if kiter >= 5
#
#           Get uptated amplitudes based on DIIS procedure.
#           ------------------------------------------------
#
            CALL DIIS4
            CALL MOVE4
#
         endif # kiter == 5
#
         CALL MOVE_PAI
#
      ENDDO kiter
#
      execute sip_barrier
#
# Delete error arrays used in the DIIS procedure.
# -----------------------------------------------
#
      DELETE D0ai
      DELETE D1ai
      DELETE D2ai
      DELETE D3ai
      DELETE D4ai
#
      DELETE D0bj
      DELETE D1bj
      DELETE D2bj
      DELETE D3bj
      DELETE D4bj
#
      DELETE e1ai
      DELETE e2ai
      DELETE e3ai
      DELETE e4ai
      DELETE e5ai
#
      DELETE e1bj
      DELETE e2bj
      DELETE e3bj
      DELETE e4bj
      DELETE e5bj
#
# Done compute the occupied-virtual block of correlated density
# -------------------------------------------------------------
#
# Compute the second-order corrections to the energy weighted
# density matrix.
# -----------------------------------------------------------
#
#      Compute Wab_aa
```

```
#       --------------
#
        PARDO a, a1, a2
#
              GET Pab_aa(a2,a1)
              Taa(a,a1)         = Pab_aa(a2,a1)*Fock_a(a2,a)
              Taa(a,a1)        *= -1.0
              PUT Wab_aa(a,a1) += Taa(a,a1)
#
        ENDPARDO a, a1, a2
#
        PARDO a1, a2, i, i1
#
              REQUEST                    VSpipi(a1,i,a2,i1) i
              Tpipi(a1,i,a2,i1)        = VSpipi(a1,i,a2,i1)
              execute energy_denominator Tpipi(a1,i,a2,i1)
#
              DO a
#
                 REQUEST             VSpipi(a,i1,a2,i) i
#
                 Taa(a,a1)         = VSpipi(a,i1,a2,i)*Tpipi(a1,i,a2,i1)
                 Taa(a,a1)        *= 0.5
                 PUT Wab_aa(a,a1) += Taa(a,a1)
#
              ENDDO a
#
        ENDPARDO a1, a2, i, i1
#
        PARDO a1, b, i, j
#
              REQUEST                    Vpiqj(a1,i,b,j) i
              Tpiqj(a1,i,b,j)          = Vpiqj(a1,i,b,j)
              execute energy_denominator Tpiqj(a1,i,b,j)
#
              DO a
#
                 REQUEST             Vpiqj(a,i,b,j) i
#
                 Taa(a,a1)         = Vpiqj(a,i,b,j)*Tpiqj(a1,i,b,j)
                 Taa(a,a1)        *= -1.0
                 PUT Wab_aa(a,a1) += Taa(a,a1)
#
              ENDDO a
#
        ENDPARDO a1, b, i, j
#
#       Done compute Wab_aa
#       -------------------
#
#       Compute Wab_bb
#       --------------
#
        PARDO b, b1, b2
#
              GET Pab_bb(b2,b1)
              Tbb(b,b1)         = Pab_bb(b2,b1)*Fock_b(b2,b)
```

```
            Tbb(b,b1)         *= -1.0
            PUT Wab_bb(b,b1) += Tbb(b,b1)
#
      ENDPARDO b, b1, b2
#
      PARDO b1, b2, j, j1
#
            REQUEST                     VSqjqj(b1,j,b2,j1) j1
            Tqjqj(b1,j,b2,j1)         = VSqjqj(b1,j,b2,j1)
            execute energy_denominator Tqjqj(b1,j,b2,j1)
#
            DO b
#
               REQUEST             VSqjqj(b,j1,b2,j) j
#
               Tbb(b,b1)          = VSqjqj(b,j1,b2,j)*Tqjqj(b1,j,b2,j1)
               Tbb(b,b1)         *= 0.5
               PUT Wab_bb(b,b1) += Tbb(b,b1)
#
            ENDDO b
#
      ENDPARDO b1, b2, j, j1
#
      PARDO a, b1, i, j
#
            REQUEST                     Vpiqj(a,i,b1,j) i
            Tpiqj(a,i,b1,j)           = Vpiqj(a,i,b1,j)
            execute energy_denominator Tpiqj(a,i,b1,j)
#
            DO b
#
               REQUEST             Vpiqj(a,i,b,j) i
#
               Tbb(b,b1)          = Vpiqj(a,i,b,j)*Tpiqj(a,i,b1,j)
               Tbb(b,b1)         *= -1.0
               PUT Wab_bb(b,b1) += Tbb(b,b1)
#
            ENDDO b
#
      ENDPARDO a, b1, i, j
#
#       PARDO a, a1
#
#             GET Wab_aa(a,a1)
#             execute dump_block Wab_aa(a,a1)
#
#       ENDPARDO a, a1
#
#       PARDO b, b1
#
#             GET Wab_bb(b,b1)
#             execute dump_block Wab_bb(b,b1)
#
#       ENDPARDO b, b1
#
#      Done compute Wab_bb
#      -------------------
```

```
#
#       Compute Wij_aa
#       --------------
#
#       Second-term in Eq. 12
#       ---------------------
        PARDO i, i1, i2
#
              GET Pij_aa(i2,i1)
#
              T1ii(i,i1)         = Pij_aa(i2,i1)*Fock_a(i2,i)
              T1ii(i,i1)        *= -1.0
              PUT Wij_aa(i,i1) += T1ii(i,i1)
#
        ENDPARDO i, i1, i2
#
#       Fourth-term in Eq. 12
#       ---------------------
        PARDO a, a1, i1, i2
#
              REQUEST                     VSpipi(a,i1,a1,i2) i1
              Tpipi(a,i1,a1,i2)         = VSpipi(a,i1,a1,i2)
              execute energy_denominator Tpipi(a,i1,a1,i2)
#
              DO i
#
                 REQUEST             VSpipi(a,i2,a1,i) i
#
                 Tii(i,i1)          = VSpipi(a,i2,a1,i)*Tpipi(a,i1,a1,i2)
                 Tii(i,i1)         *= 0.5
                 PUT Wij_aa(i,i1) += Tii(i,i1)
#
              ENDDO i
#
        ENDPARDO a, a1, i1, i2
#
        PARDO a, b, i1, j
#
              REQUEST                     Vpiqj(a,i1,b,j) i1
              Tpiqj(a,i1,b,j)           = Vpiqj(a,i1,b,j)
              execute energy_denominator Tpiqj(a,i1,b,j)
#
              DO i
#
                 REQUEST             Vpiqj(a,i,b,j) i
#
                 Tii(i,i1)          = Vpiqj(a,i,b,j)*Tpiqj(a,i1,b,j)
                 Tii(i,i1)         *= -1.0
                 PUT Wij_aa(i,i1) += Tii(i,i1)
#
              ENDDO i
#
        ENDPARDO a, b, i1, j
#
#       Third-term in Eq. 12
#       --------------------
#
```

```
#       occupied-occupied contribution
#       ------------------------------
#
        PARDO i, i1, i2, i3
#
              REQUEST             VSpipi(i,i1,i2,i3) i
              GET                 Pij_aa(i2,i3)
#
              Tii(i,i1)          = VSpipi(i,i1,i2,i3)*Pij_aa(i2,i3)
              Tii(i,i1)         *= -1.0
              PUT Wij_aa(i,i1) += Tii(i,i1)
#
        ENDPARDO i, i1, i2, i3
#
        PARDO i, i1, j, j1
#
              REQUEST             Vpiqj(i,i1,j,j1) i
              GET                 Pij_bb(j,j1)
#
              Tii(i,i1)          = Vpiqj(i,i1,j,j1)*Pij_bb(j,j1)
              Tii(i,i1)         *= -1.0
              PUT Wij_aa(i,i1) += Tii(i,i1)
#
        ENDPARDO i, i1, j, j1
#
#       virtual-virtual contribution
#       ----------------------------
        PARDO i, i1, a, a1
#
              REQUEST Vaaii(a,a1,i,i1) i
              REQUEST Viaai(i,a1,a,i1) i
              GET Pab_aa(a,a1)
#
              Tiiaa(i,i1,a,a1)  = Vaaii(a,a1,i,i1)
              T1iiaa(i,i1,a,a1) = Viaai(i,a1,a,i1)
              Tiiaa(i,i1,a,a1) -= T1iiaa(i,i1,a,a1)
#
              Tii(i,i1)          = Tiiaa(i,i1,a,a1)*Pab_aa(a,a1)
              Tii(i,i1)         *= -1.0
              PUT Wij_aa(i,i1) += Tii(i,i1)
#
        ENDPARDO i, i1, a, a1
#
        PARDO i, i1, b, b1
#
              REQUEST Vbbii(b,b1,i,i1) i
              GET Pab_bb(b,b1)
#
              Tii(i,i1)          = Vbbii(b,b1,i,i1)*Pab_bb(b,b1)
              Tii(i,i1)         *= -1.0
              PUT Wij_aa(i,i1) += Tii(i,i1)
#
        ENDPARDO i, i1, b, b1
#
#       virtual-occupied contribution --> Needs checked VFL
#       ---------------------------
        PARDO i, i1, i2, a
```

```
#
            REQUEST              VSpipi(i,i1,a,i2) i
            GET                  Paiold_aa(a,i2)
#
            Tii(i,i1)          = VSpipi(i,i1,a,i2)*Paiold_aa(a,i2)
            Tii(i,i1)         *= -1.0
            PUT Wij_aa(i,i1)  += Tii(i,i1)
            T1ii(i1,i)         = Tii(i,i1)
            PUT Wij_aa(i1,i)  += T1ii(i1,i)
#
      ENDPARDO i, i1, i2, a
#
      PARDO i, i1, j, b
#
            REQUEST              Vpiqj(i,i1,b,j) i
            GET                  Paiold_bb(b,j)
#
            Tii(i,i1)          = Vpiqj(i,i1,b,j)*Paiold_bb(b,j)
            Tii(i,i1)         *= -1.0
            PUT Wij_aa(i,i1)  += Tii(i,i1)
            T1ii(i1,i)         = Tii(i,i1)
            PUT Wij_aa(i1,i)  += T1ii(i1,i)
#
      ENDPARDO i, i1, j, b
#
#     Compute Wij_bb
#     --------------
#
#     Second-term in Eq. 12
#     ---------------------
      PARDO j, j1, j2
#
            GET Pij_bb(j2,j1)
#
            T1jj(j2,j1)       = Pij_bb(j2,j1)
            Tjj(j,j1)         = T1jj(j2,j1)*Fock_b(j2,j)
            Tjj(j,j1)         *= -1.0
            PUT Wij_bb(j,j1)  += Tjj(j,j1)
#
      ENDPARDO j, j1, j2
#
#     Fourth-term in Eq. 12
#     ---------------------
      PARDO b, b1, j1, j2
#
            REQUEST                     VSqjqj(b,j1,b1,j2) j1
            Tqjqj(b,j1,b1,j2)         = VSqjqj(b,j1,b1,j2)
            execute energy_denominator Tqjqj(b,j1,b1,j2)
#
            DO j
#
               REQUEST             VSqjqj(b,j2,b1,j) j
#
               Tjj(j,j1)          = VSqjqj(b,j2,b1,j)*Tqjqj(b,j1,b1,j2)
               Tjj(j,j1)         *= 0.5
               PUT Wij_bb(j,j1)  += Tjj(j,j1)
#
```

```
            ENDDO j
#
      ENDPARDO b, b1, j1, j2
#
      PARDO a, b, i, j1
#
            REQUEST                    Vpiqj(a,i,b,j1) i
            Tpiqj(a,i,b,j1)          = Vpiqj(a,i,b,j1)
            execute energy_denominator Tpiqj(a,i,b,j1)
#
            DO j
#
               REQUEST             Vpiqj(a,i,b,j) i
#
               Tjj(j,j1)          = Vpiqj(a,i,b,j)*Tpiqj(a,i,b,j1)
               Tjj(j,j1)         *= -1.0
               PUT Wij_bb(j,j1) += Tjj(j,j1)
#
            ENDDO j
#
      ENDPARDO a, b, i, j1
#
#     Third-term in Eq. 12
#     --------------------
#
#     occupied-occupied contribution
#     ------------------------------
      PARDO j, j1, j2, j3
#
            REQUEST             VSqjqj(j,j1,j2,j3) j
            GET                 Pij_bb(j2,j3)
#
            Tjj(j,j1)          = VSqjqj(j,j1,j2,j3)*Pij_bb(j2,j3)
            Tjj(j,j1)         *= -1.0
            PUT Wij_bb(j,j1) += Tjj(j,j1)
#
      ENDPARDO j, j1, j2, j3
#
      PARDO i, i1, j, j1
#
            REQUEST             Vpiqj(i,i1,j,j1) j
            GET                 Pij_aa(i,i1)
#
            Tjj(j,j1)          = Vpiqj(i,i1,j,j1)*Pij_aa(i,i1)
            Tjj(j,j1)         *= -1.0
            PUT Wij_bb(j,j1) += Tjj(j,j1)
#
      ENDPARDO i, i1, j, j1
#
#     virtual-virtual contribution
#     ----------------------------
      PARDO j, j1, b, b1
#
            REQUEST Vbbjj(b,b1,j,j1) j
            REQUEST Vjbbj(j,b1,b,j1) j
            GET Pab_bb(b,b1)
#
```

```
            Tjjbb(j,j1,b,b1)  = Vbbjj(b,b1,j,j1)
            T1jjbb(j,j1,b,b1) = Vjbbj(j,b1,b,j1)
            Tjjbb(j,j1,b,b1) -= T1jjbb(j,j1,b,b1)
#
            Tjj(j,j1)         = Tjjbb(j,j1,b,b1)*Pab_bb(b,b1)
            Tjj(j,j1)        *= -1.0
            PUT Wij_bb(j,j1) += Tjj(j,j1)
#
      ENDPARDO j, j1, b, b1
#
      PARDO j, j1, a, a1
#
            REQUEST Vaajj(a,a1,j,j1) j
            GET Pab_aa(a,a1)
#
            Tjj(j,j1)         = Vaajj(a,a1,j,j1)*Pab_aa(a,a1)
            Tjj(j,j1)        *= -1.0
            PUT Wij_bb(j,j1) += Tjj(j,j1)
#
      ENDPARDO j, j1, a, a1
#
#     virtual-occupied contribution --> Needs checked VFL
#     ---------------------------
      PARDO j, j1, j2, b
#
            REQUEST             VSqjqj(j,j1,b,j2) j
            GET                 Paiold_bb(b,j2)
#
            Tjj(j,j1)         = VSqjqj(j,j1,b,j2)*Paiold_bb(b,j2)
            Tjj(j,j1)        *= -1.0
            PUT Wij_bb(j,j1) += Tjj(j,j1)
            T1jj(j1,j)        = Tjj(j,j1)
            PUT Wij_bb(j1,j) += T1jj(j1,j)
#
      ENDPARDO j, j1, j2, b
#
      PARDO j, j1, i, a
#
            REQUEST             Vpiqj(a,i,j,j1) i
            GET                 Paiold_aa(a,i)
#
            Tjj(j,j1)         = Vpiqj(a,i,j,j1)*Paiold_aa(a,i)
            Tjj(j,j1)        *= -1.0
            PUT Wij_bb(j,j1) += Tjj(j,j1)
            T1jj(j1,j)        = Tjj(j,j1)
            PUT Wij_bb(j1,j) += T1jj(j1,j)
#
      ENDPARDO j, j1, i, a
#
#
#       PARDO i, i1
#
#             GET Wij_aa(i,i1)
#             execute dump_block Wij_aa(i,i1)
#
#       ENDPARDO i, i1
#
```

```
#         PARDO j, j1
#
#               GET Wij_bb(j,j1)
#               execute dump_block Wij_bb(j,j1)
#
#         ENDPARDO j, j1
#
#       Compute Wai_aa
#       --------------
        PARDO a, i, i1
#
              GET Paiold_aa(a,i1)
              Tai(a,i)         = Paiold_aa(a,i1)*Fock_a(i1,i)
              Tai(a,i)        *= -1.0
              PUT Wai_aa(a,i) += Tai(a,i)
#
        ENDPARDO a, i, i1
#
        PARDO a, a1, i1, i2
#
              REQUEST                    VSpipi(a1,i1,a,i2) i1
              Tpipi(a1,i1,a,i2)        = VSpipi(a1,i1,a,i2)
              execute energy_denominator Tpipi(a1,i1,a,i2)
#
              DO i
#
                 REQUEST           VSpipi(a1,i2,i,i1) i
#
                 Tai(a,i)         = Tpipi(a1,i1,a,i2)*VSpipi(a1,i2,i,i1)
                 Tai(a,i)        *= 0.5
                 PUT Wai_aa(a,i) += Tai(a,i)
#
              ENDDO i
#
        ENDPARDO a, a1, i1, i2
#
        PARDO a, b, j, i2
#
              REQUEST                    Vpiqj(a,i2,b,j) j
              Tpiqj(a,i2,b,j)          = Vpiqj(a,i2,b,j)
              execute energy_denominator Tpiqj(a,i2,b,j)
#
              DO i
#
                 REQUEST           Vpiqj(i,i2,b,j) i
#
                 Tai(a,i)         = Tpiqj(a,i2,b,j)*Vpiqj(i,i2,b,j)
                 Tai(a,i)        *= -1.0
                 PUT Wai_aa(a,i) += Tai(a,i)
#
              ENDDO i
#
        ENDPARDO a, b, j, i2
#
#       Compute Wai_bb
#       --------------
        PARDO b, j, j1
```

```
#
            GET Paiold_bb(b,j1)
            Tbj(b,j)         = Paiold_bb(b,j1)*Fock_b(j1,j)
            Tbj(b,j)        *= -1.0
            PUT Wai_bb(b,j) += Tbj(b,j)
#
      ENDPARDO b, j, j1
#
      PARDO b, b1, j1, j2
#
            REQUEST                    VSqjqj(b1,j1,b,j2) j1
            Tqjqj(b1,j1,b,j2)        = VSqjqj(b1,j1,b,j2)
            execute energy_denominator Tqjqj(b1,j1,b,j2)
#
            DO j
#
               REQUEST            VSqjqj(b1,j2,j,j1) j
#
               Tbj(b,j)         = Tqjqj(b1,j1,b,j2)*VSqjqj(b1,j2,j,j1)
               Tbj(b,j)        *= 0.5
               PUT Wai_bb(b,j) += Tbj(b,j)
#
            ENDDO j
#
      ENDPARDO b, b1, j1, j2
#
      PARDO a, b, i, j2
#
            REQUEST                    Vpiqj(a,i,b,j2) i
            Tpiqj(a,i,b,j2)          = Vpiqj(a,i,b,j2)
            execute energy_denominator Tpiqj(a,i,b,j2)
#
            DO j
#
               REQUEST            Vpiqj(a,i,j,j2) i
#
               Tbj(b,j)         = Tpiqj(a,i,b,j2)*Vpiqj(a,i,j,j2)
               Tbj(b,j)        *= -1.0
               PUT Wai_bb(b,j) += Tbj(b,j)
#
            ENDDO j
#
      ENDPARDO a, b, i, j2
#
#       PARDO a, i
#
#             GET Wai_aa(a,i)
#             execute dump_block Wai_aa(a,i)
#
#       ENDPARDO a, i
#
#       PARDO b, j
#
#             GET Wai_bb(b,j)
#             execute dump_block Wai_bb(b,j)
#
#       ENDPARDO b, j
```

```
#
# Done compute the second-order corrections to the energy
# weighted density matrix.
# -------------------------------------------------------------
      execute sip_barrier
#     discard VSpipi
#     discard VSqjqj
#     discard Vpiqj
#     discard Vaaii
#     discard Viaai
#     discard Vbbjj
#     discard Vjbbj
#     discard Viabj
#     discard Vaajj
#     discard Vbbii
#
# Backtransform Ppq --> P2_ao Wpq --> W2_ao
# ----------------------------------------
#
#     Transform Pij_aa
#     ----------------
      PARDO i, i1
#
            GET Pij_aa(i,i1)
            GET Wij_aa(i,i1)
#
            DO mu
#
               Ixi(mu,i1) = Pij_aa(i,i1)*ca(mu,i)
               Jxi(mu,i1) = Wij_aa(i,i1)*ca(mu,i)
#
               DO nu
#
                  Ixx(mu,nu)          = Ixi(mu,i1)*ca(nu,i1)
                  Jxx(mu,nu)          = Jxi(mu,i1)*ca(nu,i1)
                  PUT P2A_ao(mu,nu) += Ixx(mu,nu)
                  PUT W2_ao(mu,nu)  += Jxx(mu,nu)
#
               ENDDO nu
#
            ENDDO mu
#
      ENDPARDO i, i1
#
#     Transform Pij_bb
#     ----------------
      PARDO j, j1
#
            GET Pij_bb(j,j1)
            GET Wij_bb(j,j1)
#
            DO mu
#
               Ixj(mu,j1) = Pij_bb(j,j1)*cb(mu,j)
               Jxj(mu,j1) = Wij_bb(j,j1)*cb(mu,j)
#
               DO nu
```

```
#
                  Ixx(mu,nu)          = Ixj(mu,j1)*cb(nu,j1)
                  Jxx(mu,nu)          = Jxj(mu,j1)*cb(nu,j1)
                  PUT P2B_ao(mu,nu) += Ixx(mu,nu)
                  PUT W2_ao(mu,nu)  += Jxx(mu,nu)
#
               ENDDO nu
#
            ENDDO mu
#
      ENDPARDO j, j1
#
#     Transform Pab_aa
#     ----------------
      PARDO a, a1
#
            GET Pab_aa(a,a1)
            GET Wab_aa(a,a1)
#
            DO mu
#
               Ixa(mu,a1) = Pab_aa(a,a1)*ca(mu,a)
               Jxa(mu,a1) = Wab_aa(a,a1)*ca(mu,a)
#
               DO nu
#
                  Ixx(mu,nu)          = Ixa(mu,a1)*ca(nu,a1)
                  Jxx(mu,nu)          = Jxa(mu,a1)*ca(nu,a1)
                  PUT P2A_ao(mu,nu) += Ixx(mu,nu)
                  PUT W2_ao(mu,nu)  += Jxx(mu,nu)
#
               ENDDO nu
#
            ENDDO mu
#
      ENDPARDO a, a1
#
#     Transform Pab_bb
#     ----------------
      PARDO b, b1
#
            GET Pab_bb(b,b1)
            GET Wab_bb(b,b1)
#
            DO mu
#
               Ixb(mu,b1) = Pab_bb(b,b1)*cb(mu,b)
               Jxb(mu,b1) = Wab_bb(b,b1)*cb(mu,b)
#
               DO nu
#
                  Ixx(mu,nu)          = Ixb(mu,b1)*cb(nu,b1)
                  Jxx(mu,nu)          = Jxb(mu,b1)*cb(nu,b1)
                  PUT P2B_ao(mu,nu) += Ixx(mu,nu)
                  PUT W2_ao(mu,nu)  += Jxx(mu,nu)
#
               ENDDO nu
```

```
#
            ENDDO mu
#
      ENDPARDO b, b1
#
#     Transform Pai_aa
#     ----------------
      PARDO a, i
#
            GET Paiold_aa(a,i)
            GET Wai_aa(a,i)
#
            DO mu
#
               Ixi(mu,i) = Paiold_aa(a,i)*ca(mu,a)
               Jxi(mu,i) = Wai_aa(a,i)*ca(mu,a)
#
               DO nu
#
                  Ixx(mu,nu)  = Ixi(mu,i)*ca(nu,i)
                  Jxx(mu,nu)  = Jxi(mu,i)*ca(nu,i)
                  I1xx(nu,mu) = Ixx(mu,nu)
                  J1xx(nu,mu) = Jxx(mu,nu)
#
                  PUT P2A_ao(mu,nu) += Ixx(mu,nu)
                  PUT W2_ao(mu,nu)  += Jxx(mu,nu)
                  PUT P2A_ao(nu,mu) += I1xx(nu,mu)
                  PUT W2_ao(nu,mu)  += J1xx(nu,mu)
#
               ENDDO nu
#
            ENDDO mu
#
      ENDPARDO a, i
#
#     Transform Pai_bb
#     ----------------
      PARDO b, j
#
            GET Paiold_bb(b,j)
            GET Wai_bb(b,j)
            GET Lai_bb(b,j)
#
            DO mu
#
               Ixj(mu,j) = Paiold_bb(b,j)*cb(mu,b)
               Jxj(mu,j) = Wai_bb(b,j)*cb(mu,b)
#
               DO nu
#
                  Ixx(mu,nu)  = Ixj(mu,j)*cb(nu,j)
                  Jxx(mu,nu)  = Jxj(mu,j)*cb(nu,j)
                  I1xx(nu,mu) = Ixx(mu,nu)
                  J1xx(nu,mu) = Jxx(mu,nu)
#
                  PUT P2B_ao(mu,nu) += Ixx(mu,nu)
                  PUT W2_ao(mu,nu)  += Jxx(mu,nu)
```

```
                  PUT P2B_ao(nu,mu) += I1xx(nu,mu)
                  PUT W2_ao(nu,mu)  += J1xx(nu,mu)
#
               ENDDO nu
#
            ENDDO mu
#
      ENDPARDO b, j
#
# Done backtransform Ppq --> P2_ao Wpq --> W2_ao
# ---------------------------------------------
      execute sip_barrier
      delete Paiold_bb
      delete Wai_bb
      delete Lai_bb
      delete Paiold_aa
      delete Wai_aa
      delete Lai_aa
      execute sip_barrier
#
# Form the HF density
# -------------------
#
      CALL HFDENS
      execute sip_barrier
#
# Contract the density with the AO basis core Hamiltonian
# -------------------------------------------------------
#
      CALL D1TRANS
      CALL S1TRANS
#
# Contract the 'two-particle' contributions
# -----------------------------------------
#
      CALL D2TRANS
#
# Calculate the mp2 energy and write to JOBARC
# --------------------------------------------
#
     #CALL ENERGY
     #execute sip_barrier
#
                           ENDSIAL MBPT2_GRAD_AO2
#
###############################################################################
##
#
```