# DCQTP Programmers Guide

Large Scale Semi-Emprical Calculations
Last updated February, 2007

**Duane Williams**

# Table of Contents

# 1 CHG

## 1.1 chgRdKeys.F90

### 1.1.1 subroutine chgrdkeys

```
Read Charge Related Keywords
    - charge
    - Mulliken
    - cm1
    - cm2

reference J. Chem Phys. 1955, 23, 1833 (Mulliken)
reference J. Comp. Aided Mol. Design, 1995, 9, 87 (CM1)
reference J. Phys. Chem. A, 1998, 102, 1820-1831 (CM2)
```

For calls made by the subroutine, "chgrdkeys":

(See [rdnum], page 71.)

# 2 DC

## 2.1 atgrid.F90

### 2.1.1 subroutine atgrid

```
Grid Based Subsetting

author Arjan van der Vaart
date July-August 1997
date January 1998 (pbc)

Subsetting is atom-based for core and buffers.
```

For calls made by the subroutine, "atgrid":

(See [setncell], page 18.)
(See [mkagrid], page 13.)
(See [getacorners], page 8.)
(See [getaribs], page 8.)
(See [getaplanes], page 8.)
(See [getacore], page 9.)
(See [cmbcore], page 3.)
(See [pgetacorners], page 14.)
(See [pgetaribs], page 14.)
(See [pgetaplanes], page 14.)
(See [pgetainner], page 15.)
(See [busort], page 200.)
(See [getainner], page 9.)
(See [bsort], page 200.)

## 2.2 chkres.F90

### 2.2.1 subroutine chkres

```
Finds min and max inter-atomic distances for each residue
and writes them to the main output file.
```

## 2.3 clustsub.F90

### 2.3.1 subroutine clustsub

```
author Arjan van der Vaart

 Routine to do cluster subsetting.

 March 31 1998: added multiple core feature.
 the core will consist of ncore(1) residues selected from the
 group of residues icorel(1)..icorel(icorel1(2)-1), ncore(2) residues
 selected form the group of residues
 icorel(icorel1(2))..icorel(icorel1(3)-1) etc. Buffers will always
 be build from the entire set (1..nres) of residues.
```

For calls made by the subroutine, "clustsub":

(See [rsort], page 201.)
(See [pbcxyz], page 213.)
(See [bsort], page 200.)

## 2.4 cmbcore.F90

### 2.4.1 subroutine cmbcore

For calls made by the subroutine, "cmbcore":

(See [bsort1], page 200.)

## 2.5 cmbsub.F90

### 2.5.1 subroutine cmbsub

```
AuthorArjan van der Vaart
Date1997

Combined method of obtaining a subsystem for divide and conquer
using both cluster-based and grid-based subsetting.
```

For calls made by the subroutine, "cmbsub":

(See [clustsub], page 3.)
(See [resgrid], page 18.)
(See [atgrid], page 2.)
(See [mixgrid], page 11.)

## 2.6 dc_module.F90

### 2.6.1 module dc_module

```
Divide-and-Conquer Variables

 To view what this module contains:
```
(See [initialize_dcvars], page 4.)

### 2.6.2 subroutine initialize_dcvars [from module: dc_module]

```
Initializes Divide-and-Conquer Variables
```

## 2.7 dcRdKeys.F90

### 2.7.1 subroutine dcrdkeys

```
Read Divide-and-Conquer Keywords
   - cluster
   - manualsub
   - residue
   - atgrid
   - mixgrid
   - resgrid
   - combsub
   - pbc
   - rmin
   - no-overlap
   - cutbond
   - tempk
   - shift
   - chkres
```

For calls made by the subroutine, "dcrdkeys":

(See [rdnum], page 71.)

## 2.8 dcRdTail.F90

### 2.8.1 subroutine dcrdtail

```
Read Divide-and-Conquer Tail
   - combsub
   - cluster
   - manualsub
   - grid
   - group
   - neighbor
   - box
```

For calls made by the subroutine, "dcrdtail":

(See [rdcomb], page 61.)
(See [rdcluster], page 60.)
(See [rdsub], page 75.)
(See [rdgrid], page 63.)
(See [rdgroup], page 64.)
(See [rdneighb], page 69.)
(See [rdbox], page 59.)

## 2.9 doferm.F90

### 2.9.1 subroutine doferm

```
Uses an iterative bisection technique to determine the fermi
energy efermi.

Error flag is set if more than 100 iterations
are necessary.

more than one fermi energies when setch == .true.
```

For calls made by the subroutine, "doferm":

see system command "mpi_allreduce"

## 2.10 esqr.F90

### 2.10.1 subroutine esqr

```
Determines the coefficients evecsq for subsystem k:

evecsq(l0+l) = d11*evec1(1,l)**2 + d22*evec1(2,l)**2 + . . .

Here l0 is just a global pointer for subsystem k, and dii is
the normalization factor that accounts for subsystem overlap.

Once all of the entries of evecsq have been determined, then
the number of electrons can be computed for any set of fermi
occupation numbers:

no. of electrons = fermi(1)*evecsq(1) + fermi(2)*evecsq(2) + . . .
```

## 2.11 esqrfdm.F90

### 2.11.1 subroutine esqrfdm

```
Determines the coefficients evecsq for subsystem k:

evecsq(l0+l) = d11*evec1(1,l)**2 + d22*evec1(2,l)**2 + . . .

Here l0 is just a global pointer for subsystem k, and dii is
the normalization factor that accounts for subsystem overlap.

Once all of the entries of evecsq have been determined, then
the number of electrons can be computed for any set of fermi
occupation numbers:

no. of electrons = fermi(1)*evecsq(1) + fermi(2)*evecsq(2) + . . .
```

## 2.12 frmchk.F90

### 2.12.1 subroutine frmchk

```
Diagnostic tool for tracking fermi level.
```

## 2.13 fshift.F90

### 2.13.1 subroutine fshift

```
Routine to carry out dynamic or static level shift to prevent
auto-polarization of charge distribution.
```

## 2.14 gensub.F90

### 2.14.1 subroutine gensub

```
author Arjan van der Vaart
date 1997

Main driver for subset choices. This routine calls the different
methods of subsetting based on the keyword choice.
   eg. For standard calculations this routine will read the STANDARD
       keyword and set the number of subsystems to 1.
```

For calls made by the subroutine, "gensub":

(See [cmbsub], page 3.)
(See [resgrid], page 18.)
(See [mixgrid], page 11.)
(See [atgrid], page 2.)
(See [clustsub], page 3.)
(See [mpi_error_check], page 136.)

## 2.15 gridafnc.F90

```
author Arjan van der Vaart
date July-August 1997

This file contains all the subroutines needed
```

```
for atom-wise, grid-based subsetting. These subroutines
are called in the driver subroutines atgrid and mixgrid.

Note that in all the subroutines i refers to
the core (i=1), inner buffer (i=2) or outer buffer (i=3).
k is a counter for igsub, igsub is filled up with
atoms within the allowed range. kx is a counter for igsubx,
igsubx is filled up with atoms that are outside of the allowed
range (as defined by gridmin and gridmax).

The code is elaborate and long to save a lot of
if-statements. These if-statements verify if the atoms
are within certain boundaries (set by gridmin and gridmax).

with i the number of cells in the x-direction,
     j the number of cells in the y-direction,
     k the number of cells in the z-direction for a subsystem,

A crude-force method would need 6*i*j*k if-statements, while this
method only needs 2(i*j + i*k + j*k) if-statements; for a cube
a crude-force method would need 6*n*n*n if-statements, this
method 6*n*n .
```

### 2.15.1 subroutine getacorners

```
author Arjan van der Vaart
date July-August 1997

 See file details
```

### 2.15.2 subroutine getaribs

```
author Arjan van der Vaart
date July-August 1997

 See file details
```

### 2.15.3 subroutine getaplanes

```
author Arjan van der Vaart
date July-August 1997

 See file details
```

### 2.15.4 subroutine getacore

```
author Arjan van der Vaart
date July-August 1997

 See file details
```

For calls made by the subroutine, "getacore":

(See [getaat], page 9.)

### 2.15.5 subroutine getaat

```
author Arjan van der Vaart
date July-August 1997

 See file details
```

### 2.15.6 subroutine getainner

```
author Arjan van der Vaart
date July-August 1997

 See file details
```

For calls made by the subroutine, "getainner":

(See [getaat], page 9.)

## 2.16 gridrfnc.F90

```
author Arjan van der Vaart
date July-August 1997

 This file contains all the subroutines needed
 for residue-wise, grid-based subsetting. These subroutines
```

```
are called in the driver subroutines mixgrid and resgrid.

Note that in all the subroutines i refers to
the core (i=1), inner buffer (i=2) or outer buffer (i=3).
k is a counter for igsub, igsub is filled up with
atoms within the allowed range. kx is a counter for igsubx,
igsubx is filled up with atoms that are outside of the allowed
range (as defined by gridmin and gridmax).

The code is elaborate and long to save a lot of
if-statements. These if-statements verify if the residues
are within certain boundaries (set by gridmin and gridmax).

with i the number of cells in the x-direction,
     j the number of cells in the y-direction,
     k the number of cells in the z-direction for a subsystem,

A crude-force method would need 6*i*j*k if-statements, while this
method only needs 2(i*j + i*k + j*k) if-statements; for a cube
a crude-force method would need 6*n*n*n if-statements, this
method 6*n*n .
```

### 2.16.1 subroutine getrcorners

```
author Arjan van der Vaart
date July-August 1997

See file details
```

### 2.16.2 subroutine getrribs

```
author Arjan van der Vaart
date July-August 1997

See file details
```

### 2.16.3 subroutine getrplanes

```
author Arjan van der Vaart
date July-August 1997

 See file details
```

### 2.16.4 subroutine getrcore

```
author Arjan van der Vaart
date July-August 1997

 See file details
```

For calls made by the subroutine, "getrcore":

(See [getrat], page 11.)

### 2.16.5 subroutine getrat

```
author Arjan van der Vaart
date July-August 1997

 See file details
```

### 2.16.6 subroutine getrinner

```
author Arjan van der Vaart
date July-August 1997

 See file details
```

For calls made by the subroutine, "getrinner":

(See [getrat], page 11.)

## 2.17 mixgrid.F90

### 2.17.1 subroutine mixgrid

```
 Grid based subsetting

author Arjan van der Vaart
date July-August 1997
date January 1998 (pbc)
```

```
Subsetting is residue-based for core and atom based for
buffers.
```

For calls made by the subroutine, "mixgrid":

(See [setncell], page 18.)
(See [mkagrid], page 13.)
(See [mkrgrid], page 13.)
(See [getrcorners], page 10.)
(See [getrribs], page 10.)
(See [getrplanes], page 10.)
(See [getrcore], page 11.)
(See [cmbcore], page 3.)
(See [pgetacorners], page 14.)
(See [pgetaribs], page 14.)
(See [pgetaplanes], page 14.)
(See [pgetainner], page 15.)
(See [getacorners], page 8.)
(See [getaribs], page 8.)
(See [getaplanes], page 8.)
(See [getainner], page 9.)
(See [busort], page 200.)
(See [bsort], page 200.)

## 2.18 mkgrid.F90

```
author Arjan van der Vaart
date July 1997

 This file contains subroutines for making the grid
 for atom-wise and residue-wise subsetting.

|   1  + z   |   2 + z    |    3  + z  | .....  | ncell       |  y
|ncell+1+z   | ncell+2+z  | ncell+3+z  | ....   | ncell+ncell |
|            |            |            |        |  + z        |  |
|                                                              |  |
|                                                              |  v
|                                                              |
```

```
|                                                                  |
| (ncell-1)* | (ncell-1)* | (ncell-1)* | ..... | ncell*ncell |
| ncell+1+z  | ncell+2+z  | ncell+3+z  |       |    + z      |


with z=0 for the first layer (starting from cartesian z = 0),
     z=ncell*ncell for the second layer
     ...
     z=(ncell-1)*ncell*ncell for the last layer (maximum cartesian
                                                 z value)
```

### 2.18.1 subroutine mkrgrid

```
Makes grid for atom-wise and residue-wise subsetting.
```

### 2.18.2 subroutine mkagrid

```
author Arjan van der Vaart
date July 1997

Makes the grid for atom-wise and residue-wise subsetting.
```

## 2.19 pgridafnc.F90

```
author Arjan van der Vaart
date July-August 1997
date January 1998 (pbc)

This file contains all the subroutines needed
for atom-wise, grid-based subsetting with periodic boundary
conditions. These subroutines are called in the driver subroutines
atgrid and mixgrid.

Note that in all the subroutines i refers to
the core (i=1), inner buffer (i=2) or outer buffer (i=3).
k is a counter for igsub, igsub is filled up with
atoms within the allowed range. kx is a counter for igsubx,
```

```
igsubx is filled up with atoms that are outside of the allowed
range (as defined by gridmin and gridmax).
The code is elaborate and long to save a lot of
if-statements. These if-statements verify if the atoms
are within certain boundaries (set by gridmin and gridmax).
with i the number of cells in the x-direction,
     j the number of cells in the y-direction,
     k the number of cells in the z-direction for a subsystem,
A crude-force method would need 6*i*j*k if-statements, while this
method only needs 2(i*j + i*k + j*k) if-statements; for a cube
a crude-force method would need 6*n*n*n if-statements, this
method 6*n*n.
```

### 2.19.1 subroutine pgetacorners

```
author Arjan van der Vaart
date July-August 1997
date January 1998 (pbc)

 see file details
```

### 2.19.2 subroutine pgetaribs

```
author Arjan van der Vaart
date July-August 1997
date January 1998 (pbc)

 see file details
```

### 2.19.3 subroutine pgetaplanes

```
author Arjan van der Vaart
date July-August 1997
date January 1998 (pbc)

 see file details
```

### 2.19.4 subroutine pgetacore

```
author Arjan van der Vaart
date July-August 1997
date January 1998 (pbc)

 see file details
```

For calls made by the subroutine, "pgetacore":

(See [getaat], page 9.)

### 2.19.5 subroutine pgetaat

```
author Arjan van der Vaart
date July-August 1997
date January 1998 (pbc)

 see file details
```

### 2.19.6 subroutine pgetainner

```
author Arjan van der Vaart
date July-August 1997
date January 1998 (pbc)

 see file details
```

For calls made by the subroutine, "pgetainner":

(See [getaat], page 9.)

## 2.20 pgridrfnc.F90

```
author Arjan van der Vaart
date July-August 1997
date January 1998 (pbc)

 This file contains all the subroutines needed
 for residue-wise, grid-based subsetting with periodic boundary
```

```
conditions. These subroutines are called in the driver subroutines
mixgrid and resgrid.

note that in all the subroutines i refers to
the core (i=1), inner buffer (i=2) or outer buffer (i=3).
k is a counter for igsub, igsub is filled up with
atoms within the allowed range. kx is a counter for igsubx,
igsubx is filled up with atoms that are outside of the allowed
range (as defined by gridmin and gridmax).

The code is elaborate and long to save a lot of
if-statements. These if-statements verify if the residues
are within certain boundaries (set by gridmin and gridmax).

with i the number of cells in the x-direction,
     j the number of cells in the y-direction,
     k the number of cells in the z-direction for a subsystem,

a crude-force method would need 6*i*j*k if-statements, while this
method only needs 2(i*j + i*k + j*k) if-statements; for a cube
a crude-force method would need 6*n*n*n if-statements, this
method 6*n*n .
```

### 2.20.1 subroutine pgetrcorners

```
author Arjan van der Vaart
date July-August 1997
date January 1998 (pbc)

 see file details
```

### 2.20.2 subroutine pgetrribs

```
author Arjan van der Vaart
date July-August 1997
date January 1998 (pbc)

 see file details
```

### 2.20.3 subroutine pgetrplanes

```
author Arjan van der Vaart
date July-August 1997
date January 1998 (pbc)

 see file details
```

### 2.20.4 subroutine pgetrcore

```
author Arjan van der Vaart
date July-August 1997
date January 1998 (pbc)

 see file details
```

For calls made by the subroutine, "pgetrcore":

(See [getrat], page 11.)

### 2.20.5 subroutine pgetrat

```
author Arjan van der Vaart
date July-August 1997
date January 1998 (pbc)

 see file details
```

### 2.20.6 subroutine pgetrinner

```
author Arjan van der Vaart
date July-August 1997
date January 1998 (pbc)

 see file details
```

For calls made by the subroutine, "pgetrinner":

(See [getrat], page 11.)

## 2.21 resgrid.F90

### 2.21.1 subroutine resgrid

```
author Arjan van der Vaart
date July-August 1997
date January 1998 (pbc)

 Grid based subsetting

 subsetting is residue-based for core and buffers.
```

For calls made by the subroutine, "resgrid":

(See [setncell], page 18.)
(See [mkrgrid], page 13.)
(See [getrcorners], page 10.)
(See [getrribs], page 10.)
(See [getrplanes], page 10.)
(See [getrcore], page 11.)
(See [cmbcore], page 3.)
(See [pgetrcorners], page 16.)
(See [pgetrribs], page 16.)
(See [pgetrplanes], page 16.)
(See [pgetrinner], page 17.)
(See [busort], page 200.)
(See [getrinner], page 11.)
(See [bsort], page 200.)

## 2.22 setncell.F90

### 2.22.1 subroutine setncell

```
author Arjan van der Vaart
date August 1997

 sets parameters for grid-subsetting.
 called everytime a gridsubsetting is done,
 because the minimum and maximum coordinates
 can change in a NPT-MC-simulation inducing a
 change in the grid-subsetting parameters.
```

```
conditions for these parameters:

1. the core must be build from 2 or more grid-cells
   (in 1 dimension) in order to maximize the use of
   symmetry (a lot of if-statements can be avoided in
   the subsetting in this way)
2. the cell-length should not exceed the value of (core-overlap)
   (resolution error)
3. the total number of cells shouldn't exceed the preset
   number in divcon.dim. An adaptive correction scheme
   is used to prevent this error.

if dgbuff1 or dgbuff2 is smaller than rcell, more atoms
might be included in the buffer than would be expected (the actual
buffer region is <= rcell). A warning will be given in this
case.
```

## 2.23 sproc.F90

```
Subsystem processing routines.
```

### 2.23.1 subroutine sproc1

```
Does a variety of subsystem setup work prior to creating
pairlist.

sproc1 should be called whenever a new subsystem atom list is
created.  It should be called before attempting to create or
update the pairlist.  Sproc1 requires the subsystem information
nsub, iatom1, iatoms, iabuff.

BUG FIX: 6/29/98 -- S. DIXON.  When dummy atoms appear after the
first atom, the is1atm pointer wasn't being updated properly.
This occurred in the "do 500 iatm=1,natoms loop".  Now fixed.
7/26/99 fix also applies to sparkles
```

### 2.23.2 subroutine sproc2

```
sproc2 should be called after each pairlist update.  It requires
pairlist arrays ip1 and ipair and the subsystem arrays isub1
and isubs.

For each pair of atoms (iatm,jatm) in the pairlist, this routine
assigns the number of subsystems nshare in which both atoms appear:

k = pairlist entry for (iatm,jatm).
nshare(1,k) = number of times (iatm,jatm) appear with at least
one atom having non-buffer status, and the other
atom having non-buffer status or inner buffer layer
status.
nshare(2,k) = number of times (iatm,jatm) appear with any status.


nshare(1,k) keeps track of how many times a pair of atoms contribute
to the density matrix, while nshare(2,k) keeps track of contributions
to the 1-electron and fock matrices.
Should be called after each pairlist update, but only after
calling ijmake.
```

For calls made by the subroutine, "sproc2":

(See [ijfind], page 210.)

# 3 FREQ

## 3.1 freq.F90

```
author D. Suarez
date December, 1999

 The geometry and gradient are stored in
 the bfgs divcon common block.
```

### 3.1.1 subroutine freq

```
 A frequency calculation is carried out using the
 the geometry stored in cartesian coordinates
 in the xyzcrd common block

 If the belly option is used, the hessian
 matrix is calculated only for those atoms not
 included in the belly group.
```

For calls made by the subroutine, "freq":

(See [setopt], page 183.)
(See [rdnum], page 71.)
(See [setbox], page 119.)
(See [gensub], page 7.)
(See [mvcrd], page 212.)
(See [energy], page 90.)
(See [gcart], page 206.)
(See [wrtxyz], page 83.)
(See [wrthss], page 81.)
(See [matsym], page 21.)
(See [jacobi], page 197.)
(See [sort2], page 197.)
(See [dothermo], page 191.)

### 3.1.2 subroutine matsym

```
Matout prints the symmetric matrix a(n,n) on the iout file.
If iop .eq. 1 then matsym receives a square matrix with np
elements stored in a linear array.
If iop.eq. 2 then matsym receives only the np elements of the
lower triangular matrix a and they are stored in a supervector
array.
```

## 3.2 freq_module.F90

### 3.2.1 module freq_module

```
Frequency Variables
```

## 3.3 freqRdKeys.F90

### 3.3.1 subroutine freqrdkeys

```
Read frequency Keywords
   - freq
   - hess=calcfc
   - hess=read
   - hess=calcall
   - hess=calcdump
```

## 3.4 freqRdTail.F90

### 3.4.1 subroutine freqrdtail

```
Read Frequency Tail
   - hess
```

For calls made by the subroutine, "freqrdtail":

(See [rdhess], page 67.)

## 3.5 hsep.F90

### 3.5.1 subroutine hsep

```
This routine performs a hessian calculation.

This is calculated numerically by
double numerical differentiation.

An increment rinc=0.001 is a compromise
between accuracy and numerical problems

Hsep assumes that the proper geometry
has been stored  in dcoord (common bfgs)

The hessian is stored in the rfo common block.

Temporal scratch space is used in the
rfo common block as well
```

For calls made by the subroutine, "hsep":

(See [mvcrd], page 212.)
(See [setbox], page 119.)
(See [gensub], page 7.)
(See [energy], page 90.)
(See [gcart], page 206.)
(See [dograd], page 205.)
(See [prjfc], page 193.)
(See [wrthss], page 81.)

# 4 HAM

## 4.1 am1_module.F90

### 4.1.1 subroutine initialize_am1vars

```
Initialize AM1 Semiempirical Parameters
```

## 4.2 hamRdKeys.F90

### 4.2.1 subroutine hamrdkeys

```
Read Hamiltonian Keywords
   - mndo
   - am1
   - pm3
   - znb
   - pddg-pm3
   - mndod
```

For calls made by the subroutine, "hamrdkeys":

(See [initd], page 89.)

## 4.3 mndo_module.F90

### 4.3.1 subroutine initialize_mndovars

```
Initialize MNDO Semiempirical Parameters
```

## 4.4 mndod_module.F90

### 4.4.1 module mndod_module

```
MNDO/d Semiempirical Parameters
```

```
To view what this module contains:
```

(See [initialize_mndodvars], page 24.)

### 4.4.2 subroutine initialize_mndodvars [from module: mndod_module]

```
Initialize MNDO/d Semiempirical Parameters
```

## 4.5 pddgpm3_module.F90

### 4.5.1 module pddgpm3_module

```
Initialize PDDG/PM3 Semiempirical Parameters

author Andrew Wollacott

To view what this module contains:
```
(See [initialize_pddgpm3vars], page 25.)

### 4.5.2 subroutine initialize_pddgpm3vars [from module: pddgpm3_module]

```
Initialize PDDG/PM3 Semiempirical Parameters

M.P. Repaksy, J Chandrasekhar, W.L. Jorgensen, 2002, J Comp Chem, 23(16)
1601-1622
```

## 4.6 pm3_module.F90

### 4.6.1 subroutine initialize_pm3vars

```
Initialize PM3 Semiempirical Parameters
```

# 5 INCLUDE

# 6 INT

## 6.1 aijm.F90

### 6.1.1 subroutine aijm

```
author W. Thiel.

 aij-values for evaluation of two-center two-electron integrals
 and of hybrid contribution to dipole moment in mndo-d.
 definition see equation (7) of tca paper.

 Results are stored in array aij(6,5,lmz). conventions:
 First  index      1 ss, 2 sp, 3 pp, 4 sd, 5 pd, 6 dd.
 Second index      l+1 from definition of multipole.
 Third  index      atomic number.
```

### 6.1.2 function aijl

```
author W. Thiel.

 aij-values for evaluation of two-center two-electron integrals
 and of hybrid contribution to dipole moment in mndo-d.
 definition see equation (7) of tca paper.
```

## 6.2 attrecurse.F90

### 6.2.1 recursive function attrecurse

```
author Ed Brothers
date October 23, 2001

 The this is taken from the recursive relation found in Obara and Saika,
 J. Chem. Phys. 84 (7) 1986, 3963.

 If this is one of the auxillary integrals (s||s)^(m), assign value and
 return.
```

## 6.3 ddpo.F90

### 6.3.1 subroutine ddpo

```
Routine from W. Thiel.

Calculation of charge separations and additive terms used
to compute the two-center two-electron integrals in mndo/d.

Charge separations dl(6,lmz) from array aij computed in aijm.
additive terms     al(9,lmz) from function poij.
second index of dl and al    ss 1, sp 2,pp 8, pp 3, sd 4, pd 5, dd 6.
see equations (12)-(16) of tca paper for dl.
see equations (19)-(26) of tca paper for al.
special convention for atomic core: additive term al(9,ni)
used in the evaluation of the core-electron attractions and
core-core repulsions.

The hybridization contributions to the dipole moment contain
factors which depend on the charge separations as follows.
hyfsp = aij(2,2,ni)/sqrt(three)*2.5416d0
hyfpd = aij(5,2,ni)/sqrt(five) *2.5416d0
these factors are computed and saved in common block dipolmndo.
```

### 6.3.2 function poij

```
Routine from W. Thiel.

Determine additive terms rho=poij for two-center two-electron
integrals from the requirement that the appropriate one-center
two-electron integrals are reproduced.
```

## 6.4 diat.F90

### 6.4.1 subroutine diat

```
This routine calculates the electron-electron, electron-nucleus
and nucleus-nucleus interaction terms.  Herein is where the MNDO/d
atom-atom specific paramters hide.  These should probably be moved
at some point, or dealt with in a more general manner.
```

For calls made by the subroutine, "diat":

(See [overlp], page 39.)
(See [repuld], page 46.)
(See [ecda], page 29.)
(See [ecdb], page 29.)

## 6.5 ecd.F90

```
author Arjan van der Vaart
date March 2001.

 Here two routines are provided to calculate the core-electron
 attraction in the spd basis.
```

### 6.5.1 subroutine ecda

```
author Arjan van der Vaart
date March 2001.

 ecda calculates the interaction of an spd-electron of
 atom A (i) with the core of atom B (j)
```

### 6.5.2 subroutine ecdb

```
author Arjan van der Vaart
date March 2001

 ecdb calculates the interaction of an spd-electron of
 atom B (j) with the core of atom A (j)
```

## 6.6 electricfield.F90

### 6.6.1 real*8 function electricfld

```
author Ed Brothers
date May 28, 2002

 The purpose of this subroutine is to calculate the derivative of the
```

```
nuclear attraction integral with respect to the nuclear displacement
of the atom the electronic distribution is attracted to:

d/dXC (Integral over all space) Phi(mu) Phi(nu) 1/rC

The notation is the same used throughout: gtfs with orbital exponents a
and b on A and B with angular momentums defined by i,j,k (a's x, y
and z exponents, respectively) and ii,jj,k and kk on B with the core at
(Cx,Cy,Cz) with charge Z. m is the "order" of the integral which
arises from the recusion relationship. New to this are the idx, idy, and
idz terms which denote derivatives in the x y and z direction for the
C atom.

The this is taken from the recursive relation found in Obara and Saika,
J. Chem. Phys. 84 (7) 1986, 3963.

The first step is generating all the necessary auxillary integrals.
These are (0|1/rc|0)^(m) = 2 Sqrt (g/Pi) (0||0) Fm(g(Rpc)^2)
The values of m range from 0 to i+j+k+ii+jj+kk+2. This is exactly the
same as in the attraction code, and is necessary as we will be calling
that code eventually.
```

For calls made by the function, "electricfld":

(See [fmt], page 31.)

## 6.7 erecursive.F90

### 6.7.1 recursive function elctfldrecurse

```
author Ed Brothers
date May 29, 2002.

The this is taken from the recursive relation found in Obara and Saika,
J. Chem. Phys. 84 (7) 1986, 3963.

Check to see if the integral has become just the nuclear attraction
integral.
```

## 6.8 FmT.F90

### 6.8.1 subroutine fmt

```
author Ed Brothers
date October 29, 2001
```

## 6.9 inighd.F90

### 6.9.1 subroutine inighd

```
Routine from W. Thiel.

One-center two-electron integrals for spd-basis.
```

For calls made by the subroutine, "inighd":

(See [scprm], page 31.)

### 6.9.2 subroutine scprm

```
Routine from W. Thiel.
One-center two-electron integrals (slater-condon parameters) for spd.
```

### 6.9.3 function rsc

```
Routine from W. Thiel.

Calculate the radial part of one-center two-electron integrals
(slater-condon parameter)

K     - Type of integral ,   can be equal to 0,1,2,3,4 in spd-basis
NA,NB -Principle quantum number of ao,corresponding electron 1
EA,EB -Exponents of ao,corresponding electron 1
NC,ND -Principle quantum number of ao,corresponding electron 2
EC,ED -Exponents of ao,corresponding electron 2
```

### 6.9.4 function xlogfac

```
Routine from W. Thiel.

xlogfac=log(dble(fact(n-1)))
```

## 6.10 int_module.F90

### 6.10.1 module int_module

```
Integral Variables

 To view what this module contains:
```

(See [initialize_intvars], page 32.)

### 6.10.2 subroutine initialize_intvars [from module: int_module]

```
Initialize integral variables.
```

## 6.11 intdd.F90

### 6.11.1 subroutine intdd

```
author Arjan van der Vaart
date December 2000

 This routine calculates the electron repulsion integrals
 Here only integrals of the type ( x1 x2 | y1 y2 ) are
 calculated, where x1 or x2 (or both) is a d orbital
 and y1 or y2 (or both) is a d orbital.
```

For calls made by the subroutine, "intdd":

(See [localdd], page 36.)
(See [intdd1], page 33.)
(See [intdd2], page 33.)
(See [intdd3], page 33.)
(See [intdd4], page 34.)
(See [intdd5], page 34.)
(See [intdd6], page 34.)

## 6.12 intdd1.F90

### 6.12.1 subroutine intdd1

```
author Arjan van der Vaart
date December 2000

 This routine calculates the electron repulsion integrals
 Here only integrals of the type ( x1 x2 | y1 y2 ) are
 calculated, where x1 or x2 (or both) is a d orbital
 and y1 or y2 (or both) is a d orbital.

 Continued from intdd.f
```

## 6.13 intdd2.F90

### 6.13.1 subroutine intdd2

```
author Arjan van der Vaart
date December 2000

 This routine calculates the electron repulsion integrals
 Here only integrals of the type ( x1 x2 | y1 y2 ) are
 calculated, where x1 or x2 (or both) is a d orbital
 and y1 or y2 (or both) is a d orbital.
 Continued from intdd1.f
```

## 6.14 intdd3.F90

### 6.14.1 subroutine intdd3

```
author Arjan van der Vaart
date December 2000

 This routine calculates the electron repulsion integrals
 Here only integrals of the type ( x1 x2 | y1 y2 ) are
 calculated, where x1 or x2 (or both) is a d orbital
 and y1 or y2 (or both) is a d orbital.

 Continued from intdd2.f
```

## 6.15 intdd4.F90

### 6.15.1 subroutine intdd4

```
author Arjan van der Vaart
date December 2000

 This routine calculates the electron repulsion integrals
 Here only integrals of the type ( x1 x2 | y1 y2 ) are
 calculated, where x1 or x2 (or both) is a d orbital
 and y1 or y2 (or both) is a d orbital.

 Continued from intdd3.f
```

## 6.16 intdd5.F90

### 6.16.1 subroutine intdd5

```
author Arjan van der Vaart
date December 2000

 This routine calculates the electron repulsion integrals
 Here only integrals of the type ( x1 x2 | y1 y2 ) are
 calculated, where x1 or x2 (or both) is a d orbital
 and y1 or y2 (or both) is a d orbital.

 Continued from intdd4.f
```

## 6.17 intdd6.F90

### 6.17.1 subroutine intdd6

```
author Arjan van der Vaart
date December 2000
 This routine calculates the electron repulsion integrals
 Here only integrals of the type ( x1 x2 | y1 y2 ) are
 calculated, where x1 or x2 (or both) is a d orbital
 and y1 or y2 (or both) is a d orbital.

 Continued from intdd5.f
```

## 6.18 intdp.F90

### 6.18.1 subroutine intdp

```
author Arjan van der Vaart
date December 2000

 This routine calculates the electron repulsion integrals
 Here only integrals of the type ( x1 x2 | y1 y2 ) are
 calculated, where x1 or x2 (or both) is a d orbital
 and y1 or y2 (or both) is a p orbital.
```

For calls made by the subroutine, "intdp":

(See [localdp], page 37.)

## 6.19 intds.F90

### 6.19.1 subroutine intds

```
author Arjan van der Vaart
date December 2000

 This routine calculates the electron repulsion integrals
 Here only integrals of the type ( x1 x2 | y1 y2 ) are
 calculated, where x1 or x2 (or both) is a d orbital
 and y1 and y2 are d orbitals.
```

For calls made by the subroutine, "intds":

(See [localds], page 37.)

## 6.20 intpd.F90

### 6.20.1 subroutine intpd

```
author Arjan van der Vaart
date December 2000

 This routine calculates the electron repulsion integrals
 Here only integrals of the type ( x1 x2 | y1 y2 ) are
 calculated, where x1 or x2 (or both) is a p orbital
 and y1 or y2 (or both) is a d orbital.
```

For calls made by the subroutine, "intpd":

(See [localpd], page 37.)

## 6.21 intRdKeys.F90

### 6.21.1 subroutine intrdkeys

```
Read Integral Keywords
```

For calls made by the subroutine, "intrdkeys":

(See [rdnum], page 71.)

## 6.22 intsd.F90

### 6.22.1 subroutine intsd

```
author Arjan van der Vaart
date December 2000

 This routine calculates the electron repulsion integrals
 Here only integrals of the type ( x1 x2 | y1 y2 ) are
 calculated, where x1 and x2 are s orbitals
 and y1 or y2 (or both) is a d orbital.
```

For calls made by the subroutine, "intsd":

(See [localsd], page 38.)

## 6.23 localdd.F90

### 6.23.1 subroutine localdd

```
author Arjan van der Vaart
date December 2000.

 Calculation of unique, non-zero multipoles and unique
 non-zero integrals in the local frame.
 Here only integrals of the type ( x1 x2 | y1 y2 ) are
 calculated, where x1 or x2 (or both) is a d orbital
 and y1 or y2 (or both) is a d orbital.

 A complete list of the non-unique, non-zero integrals is provided
 at the bottom of this file (localdd).
```

## 6.24 localdp.F90

### 6.24.1 subroutine localdp

```
author Arjan van der Vaart
date December 2000.

 Calculation of unique, non-zero multipoles and unique
 non-zero integrals in the local frame.
 Here only integrals of the type ( x1 x2 | y1 y2 ) are
 calculated, where x1 or x2 (or both) is a d orbital
 and y1 or y2 (or both) is a p orbital.

 A complete list of the non-unique, non-zero integrals is provided
 at the bottom of the file localdd.
```

## 6.25 localds.F90

### 6.25.1 subroutine localds

```
author Arjan van der Vaart
date December 2000.

 Calculation of unique, non-zero multipoles and unique
 non-zero integrals in the local frame.
 Here only integrals of the type ( x1 x2 | y1 y2 ) are
 calculated, where x1 or x2 (or both) is a d orbital
 and y1 and y2 are s orbitals.

 A complete list of the non-unique, non-zero integrals is provided
 at the bottom of the file localdd.
```

## 6.26 localpd.F90

### 6.26.1 subroutine localpd

```
author Arjan van der Vaart
date December 2000.

 Calculation of unique, non-zero multipoles and unique
 non-zero integrals in the local frame.
```

```
Here only integrals of the type ( x1 x2 | y1 y2 ) are
calculated, where x1 or x2 (or both) is a p orbital
and y1 or y2 (or both) is a d orbital.

A complete list of the non-unique, non-zero integrals is provided
at the bottom of the file localdd.
```

## 6.27 localsd.F90

### 6.27.1 subroutine localsd

```
author Arjan van der Vaart
date December 2000.

Calculation of unique, non-zero multipoles and unique
non-zero integrals in the local frame.
Here only integrals of the type ( x1 x2 | y1 y2 ) are
calculated, where x1 and x2 are s orbitals
and y1 or y2 (or both) is a d orbital.

A complete list of the non-unique, non-zero integrals is provided
at the bottom of the file localdd.
```

## 6.28 oned.F90

### 6.28.1 subroutine oned

```
One center, 2 electron d integral code.
At some point getting rid of iss would probably be good
```

## 6.29 onedu.F90

### 6.29.1 subroutine onedu

```
One center, 2 electron d integral code.
At some point getting rid of 1 would probably be good
Unrestricted
```

## 6.30 overlap.F90

### 6.30.1 real*8 function overlap1

```
author Ed Brothers
date  October 3, 2001

 The purpose of this subroutine is to calculate the overlap between
 two normalized gaussians. i,j and k are the x,y,
 and z exponents for the gaussian with exponent a, and ii,jj, and kk
 have the same order for b.
```

### 6.30.2 real*8 function ssoverlap

```
author Ed Brothers
date October 3, 2001

 Calculates the overlap between two s functions
```

## 6.31 overlp.F90

### 6.31.1 subroutine overlp

```
 Routine to calculate diatomic overlaps for the s-p-d shells of
 of atoms a and b.
 Returns s, p, and d overlaps in sabdia.

 Input variables:

   ia,ib     = Atomic numbers for atoms a and b.

   norbsa,
   norbsb    = Numbers of atomic orbitals centered on atoms a and b.
   npqa,npqb = Principal quantum numbers.
   xisa,xisb = Orbital exponents for s-type atomic orbitals.
   xipa,xipb = Orbital exponents for p-type atomic orbitals.
   xida,xidb = Orbital exponents for d-type atomic orbitals.
   xa,ya,za,
   xb,yb,zb  = Cartesian coordinates (angstroms).
   rab       = Interatomic distance (angstroms).
```

```
Returned:
  sabdia = Diatomic overlap matrix for atoms a and b.  The format of
  sabdia is shown below.  For example, sabdia(6,4) would
  contain the overlap between the dxz orbital of atom a and
  the pz orbital of atom b.
Format of sabdia:
  S(B) PX(B) PY(B) PZ(B) DZZ(B) DXZ(B) DYZ(B) DXX-YY(B) DXY(B)
  S(A)      |                    .
  PX(A)     |                    .
  PY(A)     |                    .
  PZ(A)     |                    .
  DZZ(A)    |                    .
  DXZ(A)    | . . . . . . . . . .(6,4) = <DXZ(A)|PZ(B)>
  DYZ(A)    |
  DXX-YY(A) |
  DXY(A)    |
Utilizes the following additional subprograms:
  - sab
  - slimit
  - rtrans

Or, when Talman's algorithm is used:
  - sabt
  - calcb
  - calca
  - calcd
  - calcf
  - calcg
  - calcphi
  - calc3j
```

For calls made by the subroutine, "overlp":

(See [slimit], page 41.)
(See [rtrans], page 42.)

### 6.31.2 subroutine abintg

```
author Steve Dixon
date 1999

 Computes a-type and b-type integrals.

 BUG FIX: 11/30/99 -- S. DIXON.  The array bfactr was previously
 declared as BFACTR(0:7,0:15), with the assumption that n would
 never be larger than 7.  However, n can be as large as the sum
 of the two principal quantum numbers, so the overlap calculation
 can fail when things like bromine and iodine are present.  The
 problem was rectified by increasing the limits to BFACTR(0:15,0:15)
 and computing all 256 terms.
```

### 6.31.3 subroutine slimit

```
 Sets up array limits to take advantage of sparsity of matrices
 used in diatomic overlap calculation.  The ultimate goal is
 to compute c'*sabloc*c using the minimum number of floating
 point operations.  the degree of sparsity depends depends on
 how many atomic orbitals are on each atom.  for norbsa.ge.norbsb,
 the combinations are coded according to the variable nab:

   NORBSA  NORBSB  NAB
   1       1     1
   4       1     2
   4       4     3
   9       1     4
   9       4     5
   9       9     6

 Once the value of nab is established, the sparsity variables are
 defined:

 NIJSC(NAB) = The number of nonzero elements in sabloc*c.


  FOR IJ=1,NIJSC(NAB)
```

```
ISC(NAB,IJ) = Row number of ijth nonzero element of SABLOC*C

JSC(NAB,IJ) = Column number of ijth nonzero element of sabloc*c

NKIJSC(NAB,IJ) = Number multiplications required to compute the
                 (ISC(NAB,IJ),JSC(NAB,IJ)) entry of SABLOC*C

KIJSC(NAB,IJ,K) = INner index for elements required to compute
                  the (ISC(NAB,IJ),JSC(NAB,IJ)) entry of
                  SABLOC*C, I.E.,

                        I=ISC(NAB,IJ)
                        J=JSC(NAB,IJ)
                        SCIJ = 0.0
                        DO K=1,NKIJSC(NAB,IJ)
                          KINDX = KIJSC(NAB,IJ,K)
                          SCIJ = SCIJ + SABLOC(I,KINDX)*C(KINDX,J)
                        CONTINUE
                        (I,J) ELMENT OF SABLOC*C = SCIJ


NKC(NAB,J) = The number of nonzero elements in column J OF C.
(KC(NAB,J,K), K=1,NKC(NAB,J)) = List of row numbers in column J
of C that have nonzero entries.
```

### 6.31.4 subroutine rtrans

```
Computes a matrix c that transforms local frame overlaps sabloc
to the molecular frame overlaps sabdia:

  SABDIA = C'*SABLOC*C.

C will be either 4 BY 4 OR 9 BY 9, depending on whether NORBS=4
or NORBS=9.
```

For calls made by the subroutine, "rtrans":

(See [rmatrx], page 42.)

### 6.31.5 subroutine rmatrx

```
Constructs a rotation matrix rot that aligns the z axis with the
vector (XB-XA,YB-YA,ZB-ZA).
```

### 6.31.6 real*8 function sab

```
Portable overlap function.  Computes the overlap integral between
slater-type orbitals a and b by transformation to prolate spheroidal
coordinates.  In terms of cartesian coordinates, this means that
the xyz coordinate systems of centers a and b are mirror images,
with the z-axis of a pointing toward b and vice versa.

Handles overlap for orbitals of type S, P, D, F, ..., etc.

Input:

  NCENTR = The number of centers involved (ncentr=2 means orbitals
  A and B are centered on different atoms, NCENTR=1 means
  they are centered on the same atom.  default=2).
  NA,NB = Principal quantum numbers for the orbitals.
  LA,LB = Angular momentum quantum numbers for the orbitals.
  XIA,XIB = Exponents for the radial terms in the sto.
  EXHLA = SQRT(0.5)*XIA**(NA+0.5)
  EXHLB = SQRT(0.5)*XIB**(NA+0.5)
  NEWEXP = .true. if either xia or xib has changed from previous
  call for a given atom, .false. otherwise.
  M = Magnetic quantum number (must be the same for both orbitals).
  RAB = Distance between the atomic centers in atomic units
  (atomic units = angtroms/0.52917706)
  Ken Changed back to original MOPAC value:
  (ATOMIC UNITS = ANGTROMS/0.529167)
  RNAB = RAB**(NA+NB+1)

Returned:

  SAB = Overlap between orbitals A and B.
```

```
    STO'S ARE OF THE FORM:

      STO(N,L,M,XI,R,THETA,PHI) = NORM*R**(N-1)*EXP(-XI*R)*Y(L,M,THETA,PHI);

    WHERE,

      N,L,M = Quantum numbers
      XI = Orbital exponent
      R,THETA,PHI = Spherical polar coordinates
      NORM = Radial function normalization factor
      Y(L,M,THETA,PHI) = Normalized, complex spherical harmonic function.


   reference J. P. P. Stewart, j. comp. aided mol. design, 4, 1-105, 1990.

    The final formula for the overlap
    appears on page 28 of the reference article.
    One correction was necessary for the exponent iexp -- see code for
    details.

   author Steve L. Dixon
   date October 1991.
```

For calls made by the function, "sab":

(See [abintg], page 40.)

## 6.32 repul.F90

### 6.32.1 subroutine repul

```
   author Steve Dixon

    Routine to calculate repulsion integrals for a pair of atoms i,j.

    Input:

    IAI,IAJ     = Atomic numbers for atoms i and j.

    NORBSI,
    NORBSJ,     = Numbers of atomic orbitals centered on atoms i and j.
```

```
AI0,AI1,AI2,
AJ0,AJ1,AJ2 = Monopole, dipole, and quadrupole additive terms.

DI1,DI2,
DJ1,DJ2,    = Dipole and quadrupole charge separations.

XI,YI,ZI,
XJ,YJ,ZJ,   = Cartesian coordinates (Angstroms).

RIJ         = Interatomic distance (Angstroms).


Returned:

REPIJ = 45X45 Array of repulsion integrals.
The first 10x10 elements are ORGANIZED ACCORDING TO:

S_S  S_PX  S_PY  S_PZ  PX_PX  PX_PY  PX_PZ  PY_PY  PY_PZ  PZ_PZ
S_PX   |                  .
S_PY   |                  .
S_PZ   |                  .
PX_PX  |                  .
PX_PY  |.................(6,4) = (PX(I)*PY(I)|S(J)*PZ(J))
PX_PZ  |
PY_PY  |
PY_PZ  |
PZ_PZ  |

The rest contain integrals involving d-orbitals.
```

For calls made by the subroutine, "repul":

(See [rrep], page 45.)

### 6.32.2 subroutine rrep

```
Routine to calculate repulsion integrals for a pair of atoms i,j.
```

## 6.33 repuld.F90

### 6.33.1 subroutine repuld

author Arjan van der Vaart
date December 2000

Calculation of the semiempirical two-electron repulsion
integrals in the spd basis.

The (k l | m n) integrals will be stored in repij. The indexing
of this array is as follows:
(k l | m n) = repij(i,j).

| k l | i | k l | i |
|---|---|---|---|
| s_s | 1. | py_dx2y2 | 24. |
| s_px | 2. | py_dxy | 25. |
| s_py | 3. | pz_dz2 | 26. |
| s_pz | 4. | pz_dxz | 27. |
| px_px | 5. | pz_dyz | 28. |
| px_py | 6. | pz_dx2y2 | 29. |
| px_pz | 7. | pz_dxy | 30. |
| py_py | 8. | dz2_dz2 | 31. |
| py_pz | 9. | dz2_dxz | 32. |
| pz_pz | 10. | dz2_dyz | 33. |
| s_dz2 | 11. | dz2_dx2y2 | 34. |
| s_dxz | 12. | dz2_dxy | 35. |
| s_dyz | 13. | dxz_dxz | 36. |
| s_dx2y2 | 14. | dxz_dyz | 37. |
| s_dxy | 15. | dxz_dx2y2 | 38. |
| px_dz2 | 16. | dxz_dxy | 39. |
| px_dxz | 17. | dyz_dyz | 40. |
| px_dyz | 18. | dyz_dx2y2 | 41. |
| px_dx2y2 | 19. | dyz_dxy | 42. |
| px_dxy | 20. | dx2y2_dx2y2 | 43. |
| py_dz2 | 21. | dx2y2_dxy | 44. |
| py_dxz | 22. | dxy_dxy | 45. |
| py_dyz | 23. | | |

For calls made by the subroutine, "repuld":

(See [repul], page 44.)
(See [setdorb], page 53.)
(See [intsd], page 36.)
(See [intpd], page 35.)
(See [intds], page 35.)
(See [intdp], page 35.)
(See [intdd], page 32.)

## 6.34 sabt.F90

```
author Arjan van der Vaart
date May 2000
reference J.D. Talman, Phys. Rev. A. 48 (1993) 243-249

 This file contains all the routines to calculate the
 overlap integral between two Slater functions.
 This integral is calculated by Talman's algorithm:

 Note that two implementations of this algorithm are provided
 in this file. The second implementation, at the end of the file,
 is the general implementation. This will work for ANY quantum
 numbers, i.e. for any <n1 l1 m1 | n2 l2 m2>. This implementation
 is given here for completeness and for future reference, but
 is effectively commented out here by use of directives (#if 0).

 The first implementation is the one that divcon will use. It's
 an implementation specific for (l1<=2)&&(l2<=2) , i.e. for
 a spd-basis. Some speed could be gained here by precalculating some
 factors, instead of calculating them on the fly (as is done
 in the general implementation).

 Note that both algorithms are highly optimized: no time is
 wasted in calculating zero contributions of any kind.

 Notes:
 1. This algorithm is only used when the directive STEWART_INTEGRALS
    in divcon.dim is inactivated.
 2. In contrast to Stewart's algoritm (function sab), this algorithm
    works with Angstrom and inverse Angstrom units for R and the
```

orbital exponents. The appropriate conversions have been
made in overlp.F, fock.F etc. etc. (look for the STEWART_INTEGRALS
directive).
3. Note the reversal of labels compared to Stewart's method:
<n1 l1 m1 | n2 l2 m2>Stewart == <n2 l2 m2|n1 l1 m1>Talman
(this is due to the different definition of the displacement vector,
but does not impose any difficulty).
4. Like Stewart's algorithm, hydrogen-hydrogen overlap is calculated
by a simple expression instead of the general algorithm
5. When the principal quantum number is extended beyond 7 (i.e. maxn in
divcon.dim is larger than 7), the neg1 arrays (corresponding to
(-1)**n ) should be extended as well. The pow2 (= 2**n), fact (= n
and fact1 (= 1/n

Specific modifications of the spd code versus the general code:
1. Instead of explicitly calculating the 3J coefficient by use of
the calc3j function, the array threej with preset values is
used. Threej contains all possible 3J coefficients for
(l1<=2)&&(l2<=2) (up to d orbitals).
2. Instead of explicitly calculating the function Dlkp by use of the
calcd function, the array ds2 with preset values is used. Ds2
contains all possible values for Dlkp for (l1<=2)&&(l2<=2)
(up to d orbitals).
3. Instead of explicitly calculating Fia,lab,L' = J3 Fia,lab,L
the values of Fia,lab,L' are stored in the array fia for
(l1<=2)&&(l2<=2) (up to d orbitals).

This code is specific for spd basis

### 6.34.1 subroutine calca

author Arjan van der Vaart
date May 2000.
reference J.D. Talman, Phys. Rev. A ('93) 48 243-249

This routine calculates the coefficients Aia':

$$\left| \begin{matrix} l1 & l2 & ia \end{matrix} \right| \qquad 2\,ia + 1$$

```
                  |  0   0   0 |          4 Pi

where Aia is given by Equation 2.27 of J.D. Talman,
Phys. Rev. A ('93) 48 243-249 :
Aia = Sum(lab,l) Fia,lab,l * Qlab,l

    and | l1  l2  ia |
        |            |
        |  0   0   0 |  is the 3-j coefficient.

Note that in the following we will abbreviate the product
of the 3-j coefficient with the square root by J3:

     | l1  l2  ia |        2 ia + 1
     |  0   0   0 |          4 Pi


Aia' will be stored in array aia, the indices of this array
will indicate the value of ia:
A'0 = aia(0), A'1 = aia(1) etc.

The J3 term will be absorbed in Fia,lab,L:

Fia,lab,L'
    = J3 Fia,lab,L
    = (-1)**(l1+lab+(l1+l2+ia)/2)
      * (2l+1)(2ia+1)
      * sqrt((2l1+1)(2l2+1)(ia+l1+l2+1)
      * Phi[l2,l;lab] Phi[l,ia;l1-lab]


There are three 3-j conditions that Fia,lab,l' = 0.
(1) ia + l1 + l2 is odd
(2) lab + l2 + l is odd
(3) l + l1-lab + ia is odd

To determine which ia,lab,l values will give a non-zero Fia,lab,l' ,
we make use of the following tables, in which "e" is even,
```

```
and "o" is odd:

ia   l2=e  l2=o     (table 1)
l1=e   e    o
l1=o   o    e

l+lab   l2=e  l2=o
l1=e     e    o
l1=o     e    o

l-lab   l2=e  l2=o  (making use of table 1)
l1=e     e    o
l1=o     e    o

We see that l+lab and l-lab have to be even when l2 is even,
and have to be odd when l2 is odd.
The parity of l+lab and l-lab is identical:

l+lab   lab=e  lab=o   l-lab   lab=e  lab
l=e       e     o      l=e       e     o
l=o       o     e      l=o       o     e

In other words, when l2 is even, allowed combinations for l and lab
are: (l=e,lab=e) and (l=o,lab=0) and when l2 is odd (l=o,lab=e) and
(l=e,lab=o).

Second, the 3-j conditions require:
(4) |l2-l1| <= ia <= l1+l2
(5) |l2-lab| <= l <= l2+lab
(6) |l-l1+lab| <= ia <= l+l1-lab

rewriting (5) gives:
l2-l <= lab <= l2+l AND lab >= l-l2
condition (6) can be rewritten as:
l1-l-ia <= lab <= l1-l+ia AND lab <= l+l1-ia

There are four 6-j conditions for lab and ia:
(7) |lab| <= lab <= 2l1-lab
```

```
(8) |l-l2| <= lab <= l+l2
(9) |l2-l1| <= ia <= l2+l1
(10) |l-l1+lab| <= ia <= l+l1-lab

condition (7) is always met, condition (9) is identical
to (4), and condition (10) is identical to (6).

condition (5), (6) and (8) and realizing that
0 <= lab <= l1, we now have for lab:
max(abs(l-l2),l1-l-ia,0) <= lab <= min(l+l2,l1-l+ia,l+l1-ia,l1)

For ia, we have (condition (9)):
abs(l2-l1) <= ia <= l1+l2


Note that u=(a+b)*r/2
          v=(b-a)*r/2
          xr=r**(n1+n2+1)
```

For calls made by the subroutine, "calca":

(See [calcf], page 52.)
(See [calcg], page 52.)

### 6.34.2 subroutine calcb

```
here we will calculate all Bm,n,mu as given by
Equation 2.13 of J.D. Talman, Phys. Rev. A ('93) 48 243-249
Note that the corresponding Equation 8.2.8 of Patchovskii's
thesis contains an error (the prefactor should be
(-1)**(n-mu+i) instead of (-1)**(n-mu+1)).

Instead of calculating (-1)**(N-mu+i) explicitly, we
will make use of the parity of N and mu. We will
first perform the summation on i with the same parity
as (N+mu) and then perform the summation on i with a
different parity then (N+mu).

We will use the number ipx to track the parity of the
number of interest. If ipx is 0, the number is even,
```

```
if ipx is 1, the number is odd.

reference J.D. Talman, Phys. Rev. A ('93) 48 243-249
author Arjan van der Vaart
date May 2000.
```

### 6.34.3 subroutine calcf

```
Calculates Fs(u) = s
as given by Equation 8.2.11 of Patchkovskii's thesis.

Fs(u) will be put into the array fs(s), for s=0..is
i.e. F0(u) = fs(0); F1(u) = fs(1); Fs(u) = fs(is)

author Arjan van der Vaart
date May 2000
```

### 6.34.4 subroutine calcg

```
Here we calculate Gs(u) as given by Equation 8.2.10 of
Patchovskii's thesis. See also J. D. Talman, Phys. Rev. A
48 (1993) 243-249 and W. Hierse, P.M. Oppeneer,
Int. J. Quantum Chem. 52 (1994) 1249-1265 for the original
treatments.

We will calculate Gs for all s=0..is, these will be put
in the array gs where the index indicates the s value,
i.e. G0(u)=gs(0); G1(u)=gs(1) etc.

Recursion will be used, we will use the Gs+1 recursion
to obtain G1..G|u| and the Gs-1 recursion for
G|u|+1..Gis-1

          Gs+1                    Gs-1
                      |u|

for the Gs+1 recursion we will use G0(u) as starting value
for the recursion. For the Gs-1 recursion, we will use
Gis(u) as starting value.
```

```
author Arjan van der Vaart
date May 2000
reference Hierse and Oppeneer, Int. J. Quantum Chem. 52 (1994) 1249-1265
reference J. D. Talman, Phys. Rev. A 48 (1993) 243-249
```

### 6.34.5 real*8 function sabt

```
author Arjan van der Vaart
date May 2000.

 This function returns the overlap integral. Most work
 is done in other routines however. Note that the B
 coefficients have already been calculated in setup;
 these only need to be calculated once.
 ncentr is the number of centers involved
 n1 and n2 are the principal quantum numbers for the atoms
 l1 and l2 are the angular momentum quantum numbers
 a and b are the exponents for the radial terms in the slater functions
 exha and exhb are the normalization constants:
 exha equals sqrt( (2*a)**(2*n1+1)/(2*n1)
 exhb equals sqrt( (2*b)**(2*n2+1)/(2*n2)
 m is the magnetic quantum number
 r is the distance between atom a and b
 rn equals r**(n1+n2+1)

 Note that neg1 should be expanded when maxn in divcon.dim is increased.
```

For calls made by the function, "sabt":

(See [calca], page 48.)

## 6.35 setdorb.F90

### 6.35.1 subroutine setdorb

```
author Arjan van der Vaart
date December 2000

 This routine initializes expressions that are
 frequently used in the calculation of the
 electron repulsion integrals in the spd basis.
```

## 6.36 slow.F90

### 6.36.1 subroutine slow

For calls made by the subroutine, "slow":

(See [slowinit], page 55.)
(See [det3], page 54.)
(See [det5], page 54.)
(See [slowint], page 54.)

### 6.36.2 subroutine setint

### 6.36.3 subroutine dfunc

### 6.36.4 subroutine slowint

For calls made by the subroutine, "slowint":

(See [setint], page 54.)
(See [dfunc], page 54.)

### 6.36.5 subroutine det3

```
very slow routine for determinant of 3x3 matrix
```

### 6.36.6 subroutine det5

```
very very slow routine for determinant of 5x5 matrix
```

For calls made by the subroutine, "det5":

(See [det3], page 54.)

## 6.37 slowi.F90

### 6.37.1 subroutine slowinit

# 7 IO

## 7.1 getversion.F90

### 7.1.1 subroutine getversion

```
Assign DivCon version and release numbers as well as
the start/expiration dates for the license manager.

 strtDate(1), endDate(1)  - year
 strtDate(2), endDate(2)  - month
 strtDate(3), endDate(3)  - day

Warning: Never assign the date, Feb 29th, as the starting or the
         expiration date for the license.
```

## 7.2 header.F90

### 7.2.1 subroutine header

```
Writes header to main output file. Note that DivCon version and
release numbers as well as start/end dates of license manager
are assigned at the routine getversion.F
```

For calls made by the subroutine, "header":

(See [getversion], page 56.)

## 7.3 ioRdKeys.F90

### 7.3.1 subroutine iordkeys

```
Read Input/Output Keywords
```

For calls made by the subroutine, "iordkeys":

(See [rdnum], page 71.)

## 7.4 ioRdTail.F90

### 7.4.1 subroutine iordtail

```
Read io Related Tail
```

For calls made by the subroutine, "iordtail":

(See [upcase], page 216.)
(See [rdguess], page 64.)
(See [rdprtvec], page 73.)

## 7.5 printen.F90

### 7.5.1 subroutine printen

```
Prints information about MC and PME jobs.
Also has some checks for close contacts.
```

For calls made by the subroutine, "printen":

(See [wrttimmc], page 82.)

## 7.6 printpar.F90

### 7.6.1 subroutine printpar

```
Prints information on atom parameters
```

For calls made by the subroutine, "printpar":

(See [bsort1], page 200.)

## 7.7 printsub.F90

### 7.7.1 subroutine printsub

```
Prints information about subsystems
```

## 7.8 rdbelly.F90

### 7.8.1 subroutine rdbelly

```
WARNING

If 'BELLY' and 'HESS'  present at the tail portion of divcon
input file, 'BELLY' info should precede 'HESS' info.


Reads in the atom intervals or residue intervals
which define the belly atoms (zero forces) to
be considered by the optimization routines

THE FOLLOWING FORMATS ARE POSSIBLE

BELLY
     ATOM(S)  1-100   150-255
300-350
END_BELLY

BELLY
     RESIDUE(S) 1-10  12-34 38
     RESIDUE(S) 40-45
END_BELLY

At the beginning of a line, if the word 'ATOM(S)' or 'RESIDUE(S)'
is not present, we assume that we are reading atom intervals to
define belly groups.

The error flag is set to one if any errors are encountered while
reading this information.
```

For calls made by the subroutine, "rdbelly":

(See [upcase], page 216.)
(See [wdjoin], page 216.)
(See [whatis2], page 216.)
(See [iatoi], page 72.)

## 7.9 rdbox.F90

### 7.9.1 subroutine rdbox

```
Reads in box dimensions for periodic boundary conditions.

For a box that is 15.0 by 20.0 by 25.0, the format of the
input would be:

BOX
    XBOX=15.0 YBOX=20.0 ZBOX=25.0
END_BOX

The error flag is set to one if any errors are encountered while
reading this information.
```

For calls made by the subroutine, "rdbox":

(See [upcase], page 216.)
(See [wdjoin], page 216.)
(See [rdnum], page 71.)

## 7.10 rdchemix.F90

### 7.10.1 subroutine rdchemix

```
Reads in some molecule informations used for only 'chemix'
and hence fortran side will skip this portion.
The following formats are possible
CHEMIX
COLLECTION  1       2-WATER-MOLECULES
  MOLECULE    1   1   1-WATER-MOLECULE-A
  MOLECULE    2   1   1-WATER-MOLECULE-B
END_CHEMIX
The error flag is set to one if any errors are encountered while
reading this information.
```

For calls made by the subroutine, "rdchemix":

(See [upcase], page 216.)

## 7.11 rdcluster.F90

### 7.11.1 subroutine rdcluster

```
Reads in parameters for cluster subsetting:

ncore  = number of residues in core of each subsystem.
dbuff1 = thickness of first (inner) buffer layer.
dbuff2 = thickness of second (outer) buffer layer.

For a system with ncore=1, dbuff1=4.0 and dbuff2=3.0, the
format of the input would be:

CLUSTER
  NCORE=1 DBUFF1=4.0 DBUFF2=3.0
END_CLUSTER



For a system of 10 residues in which cores of 2 residues would
be build from residue 1, 3, 4, 5 and 7 and cores of 1 residue
would be build from residue 2, 6, 8, 9 and 10, with the first
buffer layer of 4.0 a and the second one of 2.0 a the input
would be:

CLUSTER
  NCORE=2 (1 3-5 7)
  NCORE=1 (2 6 8-10)
  DBUFF1=4.0 DBUFF2=2.0
END_CLUSTER

CLUSTSUB can also be used together with a split-fermi energy calculation,
for use in an energy decomposition study etc. the syntax would then be

CLUSTER
  NCORE=2 (1 3-5 7) [1]
  NCORE=1 (2 6 8-10) [0]
  DBUFF1=4.0 DBUFF2=2.0
END_CLUSTER
```

```
Meaning that the charge on residues 1 + 3 + 4 + 5 + 7 is +1, the
charge on residues 2 + 6 + 8 + 9 + 10 is zero. note that this is
only meaningfull when the keyword 'NO-OVERLAP' is specified.

The error flag is set to one if any errors are encountered while
reading this information.
```

For calls made by the subroutine, "rdcluster":

(See [upcase], page 216.)
(See [rdnum], page 71.)
(See [wdjoin], page 216.)
(See [whatis1i], page 216.)
(See [iatoimp], page 72.)
(See [bsort1], page 200.)
(See [whatis2], page 216.)
(See [iatoi], page 72.)

### 7.11.2 subroutine determinemaxsub

```
Determines the Global Variable maxsub using the information between
cluster and end_cluster
```

For calls made by the subroutine, "determinemaxsub":

(See [rsort], page 201.)

## 7.12 rdcomb.F90

### 7.12.1 subroutine rdcomb

```
Reads in parameters for combination subsetting

COMBSUB
  cluster 1-10
  resgrid 11-20 31-33 35 36
END_COMBSUB

etc. (any combination of cluster with resgrid, mixgrid or atgrid
can be specified)
```

```
The error flag is set to one if any errors are encountered while
reading this information.
```

For calls made by the subroutine, "rdcomb":

(See [upcase], page 216.)
(See [busort1], page 200.)
(See [wdjoin], page 216.)
(See [whatis2], page 216.)
(See [iatoi], page 72.)

## 7.13 rdcoord.F90

### 7.13.1 subroutine rdcoord

```
Read coordinate section of input file
```

For calls made by the subroutine, "rdcoord":

(See [opnfil], page 212.)
(See [rdpdb], page 73.)
(See [rdxyz], page 78.)
(See [rdrst], page 75.)
(See [rdint], page 67.)
(See [getxyz], page 208.)

## 7.14 rddmx.F90

### 7.14.1 subroutine rddmx

```
This routine initializes the density matrix from guess matrices

Atoms in the input file, whose density matrix elements are not
read in from dmx files ("missing" atoms), will be automatically
assigned diagonal matrix elements (off-diagonals will be set to zero).
Then, the diagonal elements of all non-frozen atoms will be scaled, to
have the proper number of electrons assigned to the density matrix.
However, if there is a long sequential list of "missing" atoms, e.g.
atoms 19 20 21 22 23 24 25 ... and the lenght of the sequence is
longer or equal to lseq (parameter below), the density matrix elements
of these atoms will not be scaled. This procedure will speedup
```

```
convergence.
```

For calls made by the subroutine, "rddmx":

(See [bovsort1], page 200.)
(See [rdnum], page 71.)

## 7.15 rddos.F90

### 7.15.1 subroutine rddos

```
author Arjan van der Vaart

Density of State analysis

DOS
  1-458 0.3
  558-960 0.2
END_DOS

This means that a density of state analysis will be performed
for the subsystems in which atoms 1-458 are, with sampling interval
of 0.3 eV, and also for the subsystems of atoms 558-960 with
sampling of 0.2 eV
```

For calls made by the subroutine, "rddos":

(See [upcase], page 216.)
(See [whatis2], page 216.)
(See [iatoi], page 72.)
(See [whatis1], page 216.)

## 7.16 rdgrid.F90

### 7.16.1 subroutine rdgrid

```
Reads in parameters for grid subsetting

GRID
    xcore=5.6 ycore=5.7 zcore=5.6 overlap=1.0 dbuff1=2.0 dbuff2=3.1
END_GRID
```

```
The error flag is set to one if any errors are encountered while
reading this information.
```

For calls made by the subroutine, "rdgrid":

(See [upcase], page 216.)
(See [wdjoin], page 216.)
(See [rdnum], page 71.)

## 7.17 rdgroup.F90

### 7.17.1 subroutine rdgroup

```
reads in parameters for groups

GROUP
 GROUP 1
   1-32 34
 GROUP 2
   33
 GROUP 3
   35-73
END_GROUP

the error flag is set to one if any errors are encountered while
reading this information.
```

For calls made by the subroutine, "rdgroup":

(See [upcase], page 216.)
(See [whatis2], page 216.)
(See [iatoi], page 72.)
(See [busort1], page 200.)

## 7.18 rdguess.F90

### 7.18.1 subroutine rdguess

```
Reads in the parameters for density matrix approximation.

In order to improve the usage of the DMX file together with its own
shift value, a couple of things are added.
 (a) The shift value will be allowed optionally for read/write next
to the number of atoms at the first line of DMX file
 (b) The description of GUESS parameter sections are expanded with
the words 'NOSHIFT' and 'SHIFT' (see below)
 (c) If the keyword 'SHIFT=...' is present in the keyword section of
Divcon input file, then all the shift values (if there is any)
in DMX files will be ignored regardless of the presense of the
word 'SHIFT' in GUESS parameters. On the other hand, if the
keyword 'SHIFT=...' is NOT present in the keyword section of
the input file but one of the DMX files in GUESS parameters is
associated with the word 'SHIFT', then the shift value read
from that DMX file will be used as a starting shift value for
the Divcon run, i.e., it is same as having the keyword 'SHIFT='
with that value.

Here are the possible GUESS paramters:

GUESS
  DNA.dmx F    NOSHIFT
  water1.dmx   SHIFT
  water2.dmx   NOSHIFT
END_GUESS

where the strings '(NO)SHIFT' are optional. At most one dmx file is
allowed to have SHIFT and the others must be with 'NOSHIFT' or none
which means 'NOSHIFT' by default

The total number of atoms should be equal to the ones in divcon.dim
sequential numbering is assumed, i.e. DNA.dmx contains atoms 1 - x,
water1.dmx atoms x+1 - y, and water2.dmx contains atoms y+1 - natoms

If the filename is followed by a 'F', then the subsystems with the atoms
```

```
of that file in the core, will be frozen, i.e. treated by the FDM
approximation.

GUESS
  1-1512 dna.dmx 3-1514 F       SHIFT
  1513-2931 water.dmx 1-1419    NOSHIFT
  2932-3001 salts.dmx 4-73
  3002-3010 dna.dmx 1515-1523 F NOSHIFT
END_GUESS

where the strings '(NO)SHIFT' are optional. At most one dmx file is
allowed to have SHIFT and the others must be with 'NOSHIFT' or none
which means 'NOSHIFT' by default

Density matrix elements for atoms 1-1512 will be filled with information
from atoms 3-1514 of dna.dmx and subsystems with these atoms in the core
will be frozen, i.e. treated by the FDM approximation. Density matrix
elements for atoms 1513-2931 will be filled with information from atoms
1-1419 of water.dmx and density matrix elements for atoms 2932-3001 will
be filled with information from atoms 4-73 of salts.dmx

It is imperative that all atomtypes (H, O, C, ...) correspond with each
other, i.e. in the example above atom #3 of the new file should have the
same atom type as atom #5 of dna.dmx. Note that this is up to the user,
divcon does not check this.

Note that no dashes '-' are allowed in the name of the dmx files
```

For calls made by the subroutine, "rdguess":

(See [upcase], page 216.)
(See [bovsort1], page 200.)
(See [whatis2], page 216.)
(See [iatoi], page 72.)
(See [rdword], page 77.)

## 7.19 rdhess.F90

### 7.19.1 subroutine rdhess

```
    WARNING

If 'BELLY' and 'HESS'  present at the tail portion of divcon
input file, 'BELLY' info should precede 'HESS' info.


Reads in initial hessian matrix for p-rfo optimizations

HESS
   READ(INPT,(5E16.8)) ( HESS(I,J),J=1,I),I=1,3*NATOMS)
END_HESS

IF ORIGINAL FC are in ATOMIC UNITS...THEN

HESS_AU
   READ(INPT,(5E16.8)) ( HESS(I,J),J=1,I),I=1,3*NATOMS)
END_HESS

The error flag is set to one if any errors are encountered while
reading this information.
```

For calls made by the subroutine, "rdhess":

(See [upcase], page 216.)

## 7.20 rdint.F90

### 7.20.1 subroutine rdint

```
Reads internal coordinates and related information from the main
input file.  This section of the input is terminated by the
delimiter 'END_COORD'.
```

For calls made by the subroutine, "rdint":

(See [upcase], page 216.)
(See [rdnext], page 70.)
(See [rdword], page 77.)

## 7.21 rdkeys.F90

### 7.21.1 subroutine rdkeys

```
Reads and echoes keywords and title, and extracts any numerical data
supplied via keywords.
```

For calls made by the subroutine, "rdkeys":

(See [opnfil], page 212.)
(See [opnpfil], page 212.)
(See [stripkeywd], page 183.)
(See [upcase], page 216.)
(See [wdjoin], page 216.)
(See [wrtdefaultkeys], page 79.)
(See [rdword], page 77.)
(See [mainrdkeys], page 90.)
(See [iordkeys], page 56.)
(See [hamrdkeys], page 24.)
(See [dcrdkeys], page 4.)
(See [intrdkeys], page 36.)
(See [chgrdkeys], page 1.)
(See [scfrdkeys], page 179.)
(See [minrdkeys], page 130.)
(See [freqrdkeys], page 22.)
(See [tsrdkeys], page 196.)
(See [mmrdkeys], page 134.)
(See [pbrdkeys], page 162.)
(See [toolsrdkeys], page 192.)
(See [nmrrdkeys], page 141.)
(See [mcrdkeys], page 115.)
(See [pmerdkeys], page 174.)
(See [paramrdkeys], page 147.)
(See [describedefaultkeys], page 79.)

## 7.22 rdlist.F90

### 7.22.1 subroutine rdlist

```
Reads in a list of atoms between the delimiters flag1 and flag2.
nval is the number of values stored.  Done is set to .true. if
the 'END_SUB' delimiter is encountered.  ierror is returned with
a value of 1 if any errors are encountered while reading or
processing the atoms.
```

For calls made by the subroutine, "rdlist":

(See [atmlst], page 180.)

## 7.23 rdneighb.F90

### 7.23.1 subroutine rdneighb

```
Reads in parameters for neighbors

NEIGHBORS
   NEIGHBORS=2 1
   NEIGHBORS=1 3
END_NEIGHBORS

Specifies the overlap of residues in the first and third cluster
groups. Residues in the first cluster group only overlap with
residues of the first cluster gorup if they are within 2 residues
away form each other. Residues in the third cluster group only
overlap with residues of the third cluster group if they are
within 1 residue away from each other.

the error flag is set to one if any errors are encountered while
reading this information.
```

For calls made by the subroutine, "rdneighb":

(See [upcase], page 216.)
(See [wdjoin], page 216.)
(See [bsort], page 200.)
(See [rdword], page 77.)
(See [rdinum], page 71.)

## 7.24 rdnext.F90

### 7.24.1 subroutine rdnext

```
Scans a character string for the next word, and tries to read in
the word as a double precision floating point number.

Input
string = character string containing number.
istart = starting point of search, i.e., string(istart:istart).

Returned
istart = the actual beginning of the numerical field.
istop  = the end of the numerical field.
value  = double precision number extracted from string(istart:istop).

ierror = error code: 0 --> successful read
                     1 --> error encountered
```

For calls made by the subroutine, "rdnext":

(See [rdword], page 77.)
(See [rdnum], page 71.)

## 7.25 rdnmr.F90

### 7.25.1 subroutine rdnmr

```
Reads in the atom intervals or residue intervals
which need for nmr calculation

The following formats are posible

NMR
   atom  1-100   150-255
END_NMR

NMR
   residue 1-10  12-34
   residue 40-45
```

```
END_NMR


The error flag is set to one if any errors are encountered while
reading this information.
```

For calls made by the subroutine, "rdnmr":

(See [upcase], page 216.)
(See [whatis2], page 216.)
(See [iatoi], page 72.)

## 7.26 rdnum.F90

```
Contains all the routines to convert strings to numbers
```

### 7.26.1 subroutine rdnum

```
Extracts a double precision floating point number from a character
string.  The field of search starts at string(istart:istart) or
after the first equals sign following the istart position.  The
number is returned in value.  if an error is encountered, ierror
is set to one.  This routine expects that there are no blank spaces
embedded anywhere within the numerical field.
```

For calls made by the subroutine, "rdnum":

(See [getnum], page 71.)

### 7.26.2 subroutine getnum

```
Get number from string
```

For calls made by the subroutine, "getnum":

(See [whole], page 71.)

### 7.26.3 subroutine whole

```
returns the whole number in the field string(ibeg:iend).  only
the numbers 0-9 are allowed to be present.
```

### 7.26.4 subroutine rdinum

```
Read number
```

For calls made by the subroutine, "rdinum":

(See [getinum], page 72.)

### 7.26.5 subroutine getinum

```
Get number
```

For calls made by the subroutine, "getinum":

(See [iwhole], page 72.)

### 7.26.6 subroutine iwhole

```
Returns the whole number in the field string(ibeg:iend).  only
the numbers 0-9 are allowed to be present.
```

### 7.26.7 subroutine iatoi

```
string to integer
```

For calls made by the subroutine, "iatoi":

(See [whatis2], page 216.)
(See [getinum], page 72.)

### 7.26.8 subroutine iatoimp

For calls made by the subroutine, "iatoimp":

(See [whatis1i], page 216.)
(See [getinum], page 72.)

## 7.27 rdpdb.F90

### 7.27.1 subroutine rdpdb

```
Reads coordinates in a pdb format
```

For calls made by the subroutine, "rdpdb":

(See [rdword], page 77.)
(See [rdinum], page 71.)
(See [upcase], page 216.)
(See [getnum], page 71.)

## 7.28 rdprtvec.F90

### 7.28.1 subroutine rdprtvec

```
author Arjan van der Vaart

Subroutine to selectively print eigenstates.

e.g.:

PRTVEC
   1-458 all
   558-960 -15.0 -10.0
   45-460  ef 10.0
END_PRTVEC

means that (line 1) all eigenstates for atom 1-458 will be printed,
and all eigenstates for atoms 558-960 that have energies between
-15.0 and -10.0 eV (line 2) and all eigenstates for atoms
45-460 with energies +- 10 eV round the Fermi energy (line 3).
```

For calls made by the subroutine, "rdprtvec":

(See [upcase], page 216.)
(See [whatis2], page 216.)
(See [iatoi], page 72.)
(See [whatis1], page 216.)

## 7.29 rdpush.F90

### 7.29.1 subroutine rdpush

```
Reads in parameters for 2 different subgroups that will be pushed
apart

PUSH
 PUSH 1
   1-32 34
 PUSH 2
   33 35-73
END_PUSH

the error flag is set to one if any errors are encountered while
reading this information.
```

For calls made by the subroutine, "rdpush":

(See [upcase], page 216.)
(See [whatis2], page 216.)
(See [iatoi], page 72.)
(See [busort1], page 200.)

## 7.30 rdrestraints.F90

### 7.30.1 subroutine rdrestraints

```
This subroutine reads in restrained groups

The following formats are possible:

RESTRAIN
   atom 1-100  150-255
END_RESTRAIN

RESTRAIN
   residue 1-10  12-34
END_RESTRAIN
```

```
The error flag is set to 1 if anything goes wrong
```

For calls made by the subroutine, "rdrestraints":

(See [upcase], page 216.)
(See [rdword], page 77.)
(See [whatis2], page 216.)
(See [iatoi], page 72.)

## 7.31 rdrst.F90

### 7.31.1 subroutine rdrst

```
Checks for the presence of a restart file and reads it in if
user has specified the 'restart' keywrd.  If a restart file
is present and the user has not specified 'RESTART' then the
user is informed of this fact and the error flag is set.  error
flag is also set if the user has specified restart and there is
no restart file.
```

## 7.32 rdsub.F90

### 7.32.1 subroutine rdsub

```
Reads in the atom lists for each subsystem in the structure.
the ith list is preceded by the card 'SUB I', and the last
subsystem should be followed by the card 'END_SUB'.  Within
each subsystem, a list of core atoms is provided, followed by
atom lists for the inner and outer buffer layers.

For example:

SUB 1
 CORE
   3 4 5 6
 BUFFER_1
   2 7
 BUFFER_2
   1 8
```

```
SUB 2
 CORE
   7 8 9 10
 BUFFER_1
   6 11
 BUFFER_2
   5 12
SUB 3
 CORE
   11/20
 BUFFER_1
   10 21
 BUFFER_2
   9 22
 .
 .
 .
END_SUB

Note that 11/20 indicates the range of atoms 11 through 20.
the buffer_1 and buffer_2 cards need to be present, even if
there are no buffer atoms for a particular subsystem.

Note also that dummy atoms and sparkles may be omitted from these
lists since they will not be placed in any subsystem.
```

For calls made by the subroutine, "rdsub":

(See [rdlist], page 68.)
(See [bsort], page 200.)

## 7.33 rdtail.F90

### 7.33.1 subroutine rdtail

```
Reads the tail of the input file
```

For calls made by the subroutine, "rdtail":

(See [opnfil], page 212.)
(See [upcase], page 216.)
(See [dcrdtail], page 5.)
(See [freqrdtail], page 22.)
(See [iordtail], page 57.)
(See [minrdtail], page 130.)
(See [mmrdtail], page 134.)
(See [nmrrdtail], page 141.)
(See [toolsrdtail], page 192.)

## 7.34 rdvdw.F90

### 7.34.1 subroutine rdvdw

```
author Kaushik Raha
date 12/29/03

 Reads in options for van der waal interaction calculation

 These options are
   PWD -- For pairwise interaction decomposition
   ION -- Define ions or atoms without any bonds
   CONNECT -- To define bonds and override default connect
   ATOMTYPE -- Define atomtype for any atom
   SIGEPSLN -- Assign sigma and epsilon values for atom

 This file also contain options for the molecular recognition
 solvation model.

 These options are
   PRESOLV -- For calculating solvation from divcon.pdb file
   SOLVNOW -- MRM solvation for current charges
```

## 7.35 rdword.F90

### 7.35.1 subroutine rdword

```
Locates the next word in string starting at string(istart:istart).
on return istart and istop will be updated and the word will be
in string(istart:istop).

If there are no more words in the string,
then both istart and istop will be returned with values of zero.
```

## 7.36 rdxyz.F90

### 7.36.1 subroutine rdxyz

```
Reads cartesian coordinates and related inforamtion from the main
input file.  This section of the input is terminated by the
delimiter 'END_COORD'.
```

For calls made by the subroutine, "rdxyz":

(See [upcase], page 216.)
(See [rdnext], page 70.)
(See [rdword], page 77.)

## 7.37 wrtch.F90

### 7.37.1 subroutine wrtch

```
Computes and writes out atomic charges during MC run
```

For calls made by the subroutine, "wrtch":

(See [atmchg], page 185.)

## 7.38 wrtchg.F90

### 7.38.1 subroutine wrtchg

```
WRITES OUT ATOMIC CHARGES TO UNIT IOUT.

The following table shows which charges are assigned to
```

```
which array (* is wildcard, valid for both T(rue) and F(false),
- is not calculated) [this routine won't be called when
mcsim is True]

no.|mcsim | pme | cm1 | cm2  | Mulliken | CM1    | CM2
 4    F      F     *     *      atchg     atchg2  atchg3
 5    F      T     F     F      atchg     atchg2  atchg3
 6    F      T     T     F      atchg2    atchg   atchg3
 7    F      T     F     T      atchg3    atchg2  atchg
```

## 7.39 wrtcoor.F90

### 7.39.1 subroutine wrtcoor

```
writes out some sort of coordinates to the MC trajectory file
```

## 7.40 wrtdefaultkeys.F90

### 7.40.1 subroutine wrtdefaultkeys

```
Write default keywords to the divcon output file. Anything updated
in the string 'defaultKeys' should be accounted for the routine
'describeDefaultKeys(...)' below which will be called from at the
end of the routine 'rdkeys(...)'.
```

For calls made by the subroutine, "wrtdefaultkeys":

(See [rdword], page 77.)

### 7.40.2 subroutine describedefaultkeys

```
Output (to a divcon output file) the descriptions for the default
keywords in the string 'defaultKeys'. Anything updated in the
string 'defaultKeys' in the routine 'wrtDefaultKeys' should be
accounted for this routine. Note also that the descriptions here
are somewhat different from the ones in the routine 'rdkeys'
```

For calls made by the subroutine, "describedefaultkeys":

(See [rdword], page 77.)

## 7.41 wrtdiv.F90

### 7.41.1 subroutine wrtdiv

```
Write divide-and-conquer information
```

For calls made by the subroutine, "wrtdiv":

(See [opnfil], page 212.)
(See [opnpfil], page 212.)

## 7.42 wrtdmx.F90

### 7.42.1 subroutine wrtdmx

```
author Arjan van der Vaart

Text file instead of binary, so easy portable between
different architectures
This should probably be changed to allow either.  Precision
loss happens in text files. -Ken
```

For calls made by the subroutine, "wrtdmx":

(See [opnfil], page 212.)
(See [opnpfil], page 212.)

## 7.43 wrtgrd.F90

### 7.43.1 subroutine wrtgrd

```
Writes gradient to unit iout.
```

### 7.43.2 subroutine wrtgrad

```
WRITES GRADIENT TO UNIT IGRD.
```

For calls made by the subroutine, "wrtgrad":

(See [g2int], page 205.)

## 7.44 wrthss.F90

### 7.44.1 subroutine wrthss

```
Writes hessian matrix to hessian file input_file.hss

input_file.hss can be used in restart calculations
```

## 7.45 wrtint.F90

### 7.45.1 subroutine wrtint

```
Writes out internal coordinates to unit iout.
```

## 7.46 wrtpdb.F90

### 7.46.1 subroutine wrtpdb

```
Writes out coordinates in PDB format
```

## 7.47 wrtqma.F90

### 7.47.1 subroutine wrtqma

```
Writes ouput for QM-QSAR
```

## 7.48 wrtrep.F90

### 7.48.1 subroutine wrtrep

```
Prints out the electron repulsion integrals in
readable format
```

## 7.49 wrtrst.F90

### 7.49.1 subroutine wrtrst

```
Writes latest coordinates to restart file input_file.rst and,
if the 'TRAJECTORY' keyword has been specified, coordinates
will be appended to input_file.trj.
```

For calls made by the subroutine, "wrtrst":

(See [bleng], page 198.)
(See [bndang], page 199.)
(See [dihedr], page 204.)

## 7.50 wrttim.F90

### 7.50.1 subroutine wrttims

```
Write timings for single point calculation
```

### 7.50.2 subroutine wrttimmc

```
Write timings for MC calculations
```

## 7.51 wrtvec.F90

### 7.51.1 subroutine wrtvec

```
Writes eigenvectors (MOs) to unit iout.
```

## 7.52 wrtvecsub.F90

### 7.52.1 subroutine wrtvecsubf

```
author Arjan van der Vaart

Writes eigenvectors (MOs) to unit iout.
for frozen calculation: eigenvectors can be printed at
end of SCF iteration
```

### 7.52.2 subroutine wrtvecsub

```
author Arjan van der Vaart
date 11/23/99
 Writes eigenvectors (MOs) to unit iout.
 For a DC calculation: one extra SCF iteration will be performed
 to print the eigenvectors during the last cycle

 parallelized this routine. The parallelization is not
 very elegant, but it works and is robust. Each node writes
 a temp file with the eigenvectors, which is added to input_file.out
 by the master. This is a much easier solution, especially
 when implemented in the already existing parallel divcon code,
 than a complete parallel I/O implementation. A parallel
 I/O implementation is way nicer of course, but might not
 be worth spending the development time (this routine
 is only called once, basically at the end of the calculation
 and does not add a lot of computation time).
 Anyway, other than the serial implementation, the subsystem
 eigenvectors are not printed in order. They are printed in
 the order of my_subs, so by node number. Although maybe not
 as convenient, it should not give any major problems for the
 end user. For the programmer it was indeed a lot more easier
 to realize parallelization this way.
```

For calls made by the subroutine, "wrtvecsub":

see system command "mpi_send"
see system command "flush"
see system command "mpi_recv"

## 7.53 wrtxyz.F90

### 7.53.1 subroutine wrtxyz

```
Writes out cartesian coordinates to unit iout.
```

# 8 MAIN

## 8.1 block.F90

### 8.1.1 subroutine initializedata

```
Initialize all global data
```

For calls made by the subroutine, "initializedata":

(See [initialize_mndodvars], page 24.)
(See [initialize_pddgpm3vars], page 25.)
(See [initialize_elementvars], page 89.)
(See [initialize_dcvars], page 4.)
(See [initialize_intvars], page 32.)
(See [initialize_scfvars], page 179.)
(See [initialize_nmrvars], page 139.)
(See [initialize_mndonmrvars], page 138.)
(See [initialize_paramvars], page 147.)
(See [initialize_errorvars], page 187.)
(See [initialize_mcvars], page 115.)
(See [initialize_mmvars], page 133.)
(See [initialize_pmevars], page 171.)

## 8.2 computeDimSizes.F90

### 8.2.1 subroutine computenatomsnresidues

```
Computes the number of atoms and residues
for memory allocation.
```

For calls made by the subroutine, "computenatomsnresidues":

(See [opnfil], page 212.)
(See [upcase], page 216.)

### 8.2.2 subroutine computemsorbs

```
Computes the maximum number of orbitals in a
subsystem for memory allocation.
```

### 8.2.3 subroutine computenorbs

```
Computes the number of orbitals
for memory allocation.
```

For calls made by the subroutine, "computenorbs":

(See [rdnum], page 71.)
(See [rdinum], page 71.)

### 8.2.4 subroutine computepeptides

```
Computes Maximum of peptide bonds
```

For calls made by the subroutine, "computepeptides":

(See [getpep], page 131.)

### 8.2.5 subroutine computemxatts

```
Computes Maxatts
```

### 8.2.6 subroutine computemaxatres

```
Computes Maxatres
```

## 8.3 denful.F90

### 8.3.1 subroutine denful

```
Computes the global density matrix (pdiag,pdiat) for the full system
in a calculation that does not involve separate subsystems.
```

For calls made by the subroutine, "denful":

see system command "date_and_time"
(See [ijfind], page 210.)
see system command "mpi_allreduce"

## 8.4 denfula.F90

### 8.4.1 subroutine denfula

```
Computes the global density matrix (pdiag,pdiat) for the full system
in a calculation that does not involve separate subsystems.
```

For calls made by the subroutine, "denfula":

(See [ijfind], page 210.)

## 8.5 denfulb.F90

### 8.5.1 subroutine denfulb

```
Computes the global density matrix (pdiag,pdiat) for the full system
in a calculation that does not involve separate subsystems.
```

For calls made by the subroutine, "denfulb":

(See [ijfind], page 210.)

## 8.6 densub.F90

### 8.6.1 subroutine densub

```
Adds in density matrix contribution from subsystem k.  It is
assumed that it will be faster to subtract electrons from a
saturated density matrix, which is just 2.0 times the identity
matrix.
```

For calls made by the subroutine, "densub":

(See [ijfind], page 210.)

## 8.7 densuba.F90

### 8.7.1 subroutine densuba

```
Adds in density matrix contribution from subsystem k.  It is
assumed that it will be faster to subtract electrons from a
saturated density matrix, which is just 2.0 times the identity
matrix.
```

For calls made by the subroutine, "densuba":

(See [ijfind], page 210.)

## 8.8 densubb.F90

### 8.8.1 subroutine densubb

```
Adds in density matrix contribution from subsystem k.  It is
assumed that it will be faster to subtract electrons from a
saturated density matrix, which is just 2.0 times the identity
matrix.
```

For calls made by the subroutine, "densubb":

(See [ijfind], page 210.)

## 8.9 densubfdm.F90

### 8.9.1 subroutine densubfdm

```
Adds in density matrix contribution from subsystem k.  It is
assumed that it will be faster to subtract electrons from a
saturated density matrix, which is just 2.0 times the identity
matrix.
```

For calls made by the subroutine, "densubfdm":

(See [ijfind], page 210.)

## 8.10 divcon.F90

### 8.10.1 program divcon

```
Main Program
```

For calls made by the program, "divcon":

(See [initializeglobalvariables], page 181.)
(See [initializesavevars], page 214.)
(See [readcommandline], page 180.)
(See [initializedata], page 84.)

see system command "mpi_init"
see system command "mpi_comm_size"
see system command "mpi_comm_rank"
(See [etimer], page 205.)
(See [setunit], page 215.)
(See [opnfil], page 212.)
(See [opnpfil], page 212.)
(See [rdkeys], page 68.)
(See [computenatomsnresidues], page 84.)
(See [allocatemxatomvars], page 121.)
(See [rdcoord], page 62.)
(See [rdtail], page 76.)
(See [allocatedcvars], page 120.)
(See [allocateguessvars], page 120.)
(See [allocatenmrvars1], page 121.)
(See [allocatebellyvars], page 121.)
(See [gensub], page 7.)
(See [allocatemslistvars], page 123.)
(See [computemaxatres], page 85.)
(See [computemsorbs], page 84.)
(See [allocatemsorbvars], page 123.)
(See [allocatemsvalvars], page 123.)
(See [allocatembpairvars], page 123.)
(See [allocatemxdiatvars], page 123.)
(See [allocatemxdiagvars], page 123.)
(See [allocatemaxrepvars], page 123.)
(See [allocatemxfdiavars], page 124.)
(See [allocatenmrvars2], page 121.)
(See [allocatevars], page 122.)
(See [computenorbs], page 84.)
(See [setup], page 183.)
(See [chkovlp], page 202.)
(See [nvtdriver], page 117.)
(See [nptdriver], page 117.)
(See [pmedriver], page 173.)
(See [mmdriver], page 133.)
(See [edriver], page 89.)
(See [pbdriver], page 160.)
(See [toolsdriver], page 191.)

(See [deallocatesavevars], page 214.)
(See [deallocatevars], page 124.)
see system command "mpi_finalize"

## 8.11 edriver.F90

### 8.11.1 subroutine edriver

```
Main Energy Driver
  Both for Single Points and Geometry Optimization
```

For calls made by the subroutine, "edriver":

(See [initialize_geooptvars], page 127.)
(See [geots], page 193.)
(See [deallocate_geooptvars], page 128.)
(See [geoopt], page 126.)
(See [setbox], page 119.)
(See [gensub], page 7.)
(See [energy], page 90.)
(See [freq], page 21.)
(See [atmchg], page 185.)
(See [push], page 188.)
(See [spinclean], page 215.)

## 8.12 element_module.F90

### 8.12.1 module element_module

```
Contains atomic information
```

```
To view what this module contains:
```

(See [initialize_elementvars], page 89.)
(See [initd], page 89.)

### 8.12.2 subroutine initialize_elementvars [from module: element_module]

```
Assign nonparametric elemental quantities.
```

### 8.12.3 subroutine initd [from module: element_module]

```
setup all the d orbitals for the mndo/d calculation.
s and p orbitals are setup in mndod.F90
```

## 8.13 energy.F90

### 8.13.1 subroutine energy

```
Driver routine to determine heat of formation eheat1.
```

For calls made by the subroutine, "energy":

(See [sproc1], page 19.)
(See [bpair], page 199.)
(See [glbpnt], page 181.)
(See [initp], page 177.)
(See [ijmake], page 209.)
(See [sproc2], page 19.)
(See [rcalc], page 214.)
(See [doscf], page 175.)
(See [dihedr], page 204.)
(See [bond_e], page 134.)
(See [angle_e], page 134.)
(See [tor_e], page 134.)

## 8.14 global_module.F90

### 8.14.1 module global_module

```
Container module for the lazy that don't want to include individual
modules
```

## 8.15 mainRdKeys.F90

### 8.15.1 subroutine mainrdkeys

```
Read Main Keywords
```

For calls made by the subroutine, "mainrdkeys":

(See [rdnum], page 71.)

## 8.16 mosub.F90

### 8.16.1 subroutine mosub

```
Determines the molecular orbitals for all subsystems.

goodf =  flag for a good estimate of the fermi energy.  When
the density matrix has just been initialized to
diagonal form, then we don't have a good estimate
of efermi.  In this case, two rounds of diagonalizations
will be necessary to determine the global density
matrix.

iter   = scf iteration number: 1,2,...

ndblscf= two diagonalization passes are always performed regardless
if iter <= ndblscf

pseudo = logical flag to carry out pseudo-diagonalization

dodble = logical flag to call lapack routine 'ssyevx' or 'dsyevx'
depending on its value when the routine 'diag' is not called

Returned error codes:

ierror = 0 --> okay
ierror = 1 --> (1) qr iteration in diag didn't converge, or
               (2) fermi energy in doferm didn't converge

Note: this routine uses the /work/ common block for temporary
storage.  anything in ff, evec, or ww below will be destroyed.
```

For calls made by the subroutine, "mosub":

(See [balance_bynorbs], page 136.)
(See [ijfind], page 210.)
see system command "date_and_time"
(See [diagp], page 99.)
(See [pdsyevd_driver], page 104.)
see system command "dsyevx"
see system command "ssyevx"
see system command "dsyevd"
see system command "dspev"
(See [diag], page 96.)
(See [denfula], page 86.)
(See [denfulb], page 86.)
(See [denful], page 85.)
(See [densuba], page 86.)
(See [densubb], page 87.)
(See [densub], page 86.)
(See [esqr], page 6.)
(See [wrtvecsub], page 82.)
(See [etimer], page 205.)
(See [doferm], page 5.)
see system command "mpi_allreduce"
(See [balance_bytime], page 136.)

## 8.17 mosubfdm.F90

### 8.17.1 subroutine mosubfdm

```
Determines the molecular orbitals for all subsystems.

goodf =  flag for a good estimate of the fermi energy.  When
the density matrix has just been initialized to
diagonal form, then we don't have a good estimate
of efermi.  In this case, two rounds of diagonalizations
will be necessary to determine the global density
matrix.

iter   = scf iteration number: 1,2,...

ndblscf= two diagonalization passes are always performed regardless
```

```
if iter <= ndblscf

pseudo = logical flag to carry out pseudo-diagonalization

dodble = logical flag to call lapack routine 'ssyevx' or 'dsyevx'
depending on its value when the routine 'diag' is not called

Returned error codes:

ierror = 0 --> okay
ierror = 1 --> (1) qr iteration in diag didn't converge, or
               (2) fermi energy in doferm didn't converge

Note: this routine uses the /work/ common block for temporary
storage.  anything in ff, evec, or ww below will be destroyed.
```

For calls made by the subroutine, "mosubfdm":

(See [ijfind], page 210.)
(See [diagp], page 99.)
(See [pdsyevx_driver], page 104.)
see system command "dsyevx"
see system command "ssyevx"
see system command "dsyevd"
see system command "dspev"
(See [diag], page 96.)
(See [esqrfdm], page 6.)
(See [doferm], page 5.)
(See [densubfdm], page 87.)

## 8.18 mosubfdmx.F90

### 8.18.1 subroutine mosubfdmx

```
Determines the molecular orbitals for all subsystems.

goodf =  flag for a good estimate of the fermi energy.  When
the density matrix has just been initialized to
diagonal form, then we don't have a good estimate
of efermi.  In this case, two rounds of diagonalizations
```

```
will be necessary to determine the global density
matrix.

iter   = SCF iteration number: 1,2,...

ndblscf= Two diagonalization passes are always performed regardless
if iter <= ndblscf

pseudo = logical flag to carry out pseudo-diagonalization

dodble = logical flag to call lapack routine 'ssyevx' or 'dsyevx'
depending on its value when the routine 'diag' is not called

returned error codes:

ierror = 0 --> okay
ierror = 1 --> (1) qr iteration in diag didn't converge, or
               (2) fermi energy in doferm didn't converge

note: this routine uses the /work/ common block for temporary
storage.  anything in ff, evec, or ww below will be destroyed.

parallellized by Arjan van der Vaart, 11/22/99
```

For calls made by the subroutine, "mosubfdmx":

(See [balance_bynorbs], page 136.)
(See [ijfind], page 210.)
(See [diagp], page 99.)
(See [pdsyevx_driver], page 104.)
see system command "dsyevx"
see system command "ssyevx"
see system command "dsyevd"
see system command "dspev"
(See [diag], page 96.)
(See [opnpfil1], page 212.)
(See [opnfil], page 212.)
(See [denful], page 85.)
(See [densub], page 86.)

(See [esqr], page 6.)
(See [wrtvecsub], page 82.)
(See [doferm], page 5.)
see system command "mpi_allreduce"

# 9 MATH

## 9.1 diag.F90

### 9.1.1 subroutine dummy

```
Dummy subroutine for CRAY
```

### 9.1.2 subroutine diag

```
author S. L. DIXON
date OCT., 1991.

 Driver routine for diagonalization of the real, symmetric,
 matrix a.

 VARIABLES REQUIRED:

 NDIM = ORDER OF THE MATRIX A (I.E., A IS NDIM BY NDIM);

 A = REAL SYMMETRIC MATRIX TO BE DIAGONALIZED.  ONLY THE LOWER
 HALF OF A NEED BE FILLED WHEN CALLING.  A IS DESTROYED BY
 THIS ROUTINE.

 NEVEC1 = THE NUMBER OF EIGENVECTORS REQUIRED;

 TOLERA = TOLERANCE FACTOR USED IN THE QR ITERATION TO DETERMINE
 WHEN OFF-DIAGONAL ENTRIES ARE ESSENTIALLY ZERO.  (DEFAULT
 IS 1.0D-8).

 V = 3 BY NDIM WORKSPACE.


 VARIABLES RETURNED:

 EVAL1 = EIGENVALUES OF A (SORTED IN INCREASING ALGEBRAIC VALUE);

 IDEGEN1 = DEGENERACIES (NUMBER OF TIMES REPEATED) FOR EIGENVALUES;

 EVEC1 = EIGENVECTORS OF A (IN COLUMNS OF EVEC1);
```

```
ERROR CODES:  IERROR=0 - SUCCESSFUL CALCULATION.
IERROR=1 - NO CONVERGENCE IN QR ITERATION.
```

For calls made by the subroutine, "diag":

(See [tridi], page 97.)
(See [degen], page 97.)
(See [eigvec], page 98.)

### 9.1.3 subroutine tridi

```
TRIDIAGONALIZES A REAL, SYMMETRIC MATRIX A BY THE METHOD OF
HOUSEHOLDER (J. H. WILKINSON, THE COMPUTER JOURNAL, VOL. 3,
P. 23 (1960)).  NDIM IS THE ORDER OF A.  THE DIAGONAL AND
SUBDIAGONAL OF A ARE OVERWRITTEN WITH THE TRIDIAGONALIZED
VERSION OF A.  THE VECTORS USED IN EACH HOUSEHOLDER
TRANSFORMATION ARE STORED ABOVE THE DIAGONAL IN THE FIRST
NDIM-2 ROWS OF A.  THE BETAHS ARE RETURNED BELOW THE SUBDIAGONAL
OF A.  V IS A WORKSPACE ARRAY.
PROGRAMMED BY S. L. DIXON, OCT., 1991.
```

### 9.1.4 subroutine eigval

```
QR ROUTINE FOR THE DETERMINATION OF ALL THE EIGENVALUES
OF THE NDIM BY NDIM SYMMETRIC, TRIDIAGONAL MATRIX A.
INPUT:
NDIM   = SIZE OF MATRIX A.
A      = NDIM BY NDIM SYMMETRIC TRIDIAGONAL MATRIX.
BETAH   = 3 BY NDIM WORKSPACE.
TOLERA = SMALL NUMBER USED TO DETERMINE WHEN OFF-DIAGONAL
ELEMENTS ARE ESSENTIALLY ZERO.
ANORM  = ABSOLUTE COLUMN NORM OF TRIDIAGONAL MATRIX A.
RETURNED:
EVAL1   = EIGENVALUES OF A IN ASCENDING ORDER.
IERROR = 1 IF QR ITERATION DID NOT CONVERGE; 0 OTHERWISE.
PROGRAMMED BY S. L. DIXON.
```

### 9.1.5 subroutine degen

```
DETERMINES DEGENERACIES OF THE EIGENVALUES.
INPUT:
NDIM   = SIZE OF MATRIX BEING DIAGONALIZED.
EVAL1   = SORTED EIGENVALUES (INCREASING VALUE).
```

```
TOLERA = SAME TOLERANCE USED TO DETERMINE EIGENVALUES.
ANORM  = ABSOLUTE COLUMN NORM OF TRIDIAGONAL MATRIX.
RETURNED:
IDEGEN1 = DEGENERACIES OF EIGENVALUES.
```

### 9.1.6 subroutine eigvec

```
INVERSE ITERATION ROUTINE FOR EIGENVECTOR DETERMINATION.
CALCULATES THE EIGENVECTORS OF AN NDIM BY NDIM SYMMETRIC,
TRIDIAGONAL MATRIX A.
INPUT:
NDIM   = SIZE OF MATRIX A.
NEVEC1  = NUMBER OF EIGENVECTORS REQUIRED.
A      = NDIM BY NDIM TRIDIAGONAL MATRIX.
TOLERA = SAME TOLERANCE USED TO DETERMINE EIGENVALUES.
ANORM  = ABSOLUTE COLUMN NORM OF TRIDIAGONAL MATRIX A.
EVAL1   = SORTED EIGENVALUES OF A.
IDEGEN1 = DEGENERACIES OF EIGENVALUES.
RETURNED:
EVEC1   = EIGENVECTORS OF TRIDIAGONAL MATRIX (IN COLUMNS).
PROGRAMMED BY S. L. DIXON, OCT., 1991.
```

For calls made by the subroutine, "eigvec":

(See [yrandom], page 98.)
(See [orthog], page 98.)

### 9.1.7 subroutine orthog

```
CONSTRUCTS A SET OF ORTHONORMAL VECTORS FROM THE NVECT LINEARLY
INDEPENDENT, NORMALIZED VECTORS IN THE ARRAY VECT.  THE VECTORS
SHOULD BE STORED COLUMNWISE, STARTING IN COLUMN JSTART.  VECT IS
OVERWRITTEN WITH THE ORTHONORMAL SET.  ALL VECTORS ARE NDIM BY 1.
ORTH IS RETURNED WITH A VALUE OF .TRUE. IF THE SET WAS LINEARLY
INDEPENDENT AND .FALSE. OTHERWISE.
PROGRAMMED BY S. L. DIXON.
```

### 9.1.8 subroutine yrandom

```
GENERATES INTEGER*4 AND DOUBLE PRECISION RANDOM NUMBERS ON
THE INTERVALS:
0 < IX < (2**31)-1
AND      0.0 < X < 1.0
ON THE FIRST CALL IX SHOULD SATISFY THE TOP INEQUALITY.
NUMBERS ARE GENERATED USING THE RELATION,
IX = IX*IC (MODULO (2**31)-1),  WHERE IC = 7**5
```

## 9.2 diagp.F90

### 9.2.1 subroutine diagp

```
PSEUDO-DIAGONALIZATION ROUTINE FOR THE MATRIX F.  THIS ROUTINE
IS USED WHEN EIGENVALUES IN SCF CALCULATION HAVE SUFFICIENTLY
CONVERGED.  THEIR VALUES ARE USED TO APPROXIMATELY ELIMINATE
SIGNIFICANT OFF-DIAGONAL ELEMENTS IN THE OCCUPIED-VIRTUAL
INTERSECTION OF THE MO'S WITH F.  SEE J.J.P. STEWART, C. CSASZAR,
AND P. PULAY, J. COMP. CHEM, VOL. 3, 227-228 (1982).
NORBS = THE TOTAL NUMBER OF MOLECULAR ORBITALS.
NOCC = THE NUMBER OF OCCUPIED MOLECULAR ORBITALS.
F = MATRIX TO BE PSEUDODIAGONALIZED.  THE STORAGE IS LINEAR
AND REQUIRES AT LEAST NORBS*NORBS SLOTS.  THE 2-D TO 1-D
CORRESPONDENCE IS (I,J) --> (I+(J-1)*NORBS).  ONLY THE
LOWER TRIANGLE (I.GE.J) IS NEEDED AND ONLY THE UPPER
TRIANGLE IS ALTERED.
E = EIGENVALUES OF F.  REQUIRES NORBS SLOTS.
FOV = TEMPORARY STORAGE FOR PARTIALLY DIAGONALIZED VERSION OF F.
REQUIRES AS MUCH STORAGE AS F.
V = TEMPORARY STORAGE.  REQUIRES NORBS SLOTS.
C = EIGENVECTORS FOR F.  STORAGE IS THE SAME AS F.  THIS ROUTINE
ROTATES THE EIGENVECTORS TO HELP ELIMINATE THE OCCUPIED-
VIRTUAL INTERSECTION.
```

## 9.3 fourier.F90

```
Fourier transformations
```

### 9.3.1 subroutine rlft3

```
USES fourn
```

For calls made by the subroutine, "rlft3":

(See [fourn], page 99.)

### 9.3.2 subroutine fourn

```
Fourier transform the matrix DATA.  This is a really bad name for
a matrix in Fortran.  Fix this at some point.
```

## 9.4 lsolve.F90

### 9.4.1 subroutine lsolve

```
USES GAUSSIAN ELIMINATION WITH ROW PIVOTING TO SOLVE THE ORDER N
LINEAR SYSTEM:
A*X = B.
W IS A WORK VECTOR OF LENGTH N, AND THRESH IS A THRESHOLD FOR
ZERO PIVOTAL ELEMENTS OF A.
ERROR CODES:  IERROR = 0 - SOLUTION FOUND SUCCESSFULLY;
IERROR = 1 - ZERO PIVOT ENCOUNTERED, SINGULAR MATRIX.
```

## 9.5 math.F90

### 9.5.1 subroutine dgedi

```
dgedi computes the determinant and inverse of a matrix
using the factors computed by dgeco or dgefa.

On entry:

A       Double precision(lda, n)
the output from dgeco or dgefa.

LDA     Integer
the leading dimension of the array  a.

N       Integer
the order of the matrix  a .

IPVT    Integer(n)
the pivot vector from dgeco or dgefa.

WORK    Double precision(n)
work vector.  contents destroyed.

Job     Integer
= 11   Both determinant and inverse.
= 01   Inverse only.
= 10   Determinant only.
```

```
On return:

A       Inverse of original matrix if requested.
otherwise unchanged.

DET     Double precision(2)
Determinant of original matrix if requested.
otherwise not referenced.
determinant = det(1) * 10.0**det(2)
with  1.0 .le. abs(det(1)) .lt. 10.0
or  det(1) .eq. 0.0 .

Error condition:

A division by zero will occur if the input factor contains
a zero on the diagonal and the inverse is requested.
It will not occur if the subroutines are called correctly
and if dgeco has set rcond .gt. 0.0 or dgefa has set
info .eq. 0 .

LINPACK. THIS VERSION DATED 08/14/78 .
CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LAB.

Subroutines and functions:

blas daxpy,dscal,dswap
fortran abs,mod

Compute determinant
```

For calls made by the subroutine, "dgedi":

(See [dscal], page 103.)
(See [daxpy], page 103.)

### 9.5.2 subroutine dgefa

```
Dgefa factors a double precision matrix by gaussian elimination.

dgefa is usually called by dgeco, but it can be called
directly with a saving in time if  rcond  is not needed.
(time for dgeco) = (1 + 9/n)*(time for dgefa) .

On entry:

A       Double precision(lda, n)
the matrix to be factored.

LDA     Integer
the leading dimension of the array  a.

N       Integer
the order of the matrix  a.

on return:

A       An upper triangular matrix and the multipliers
which were used to obtain it.
The factorization can be written  A = L*U  where
L  is a product of permutation and unit lower
triangular matrices and  U  is upper triangular.

IPVT    Integer(N)
An integer vector of pivot indices.

INFO    Integer
= 0  Normal value.
= K  If  U(K,K) .EQ. 0.0 .  This is not an error
condition for this subroutine, but it does
indicate that dgesl or dgedi will divide by zero
if called.  Use  rcond  in dgeco for a reliable
indication of singularity.
```

```
LINPACK. THIS VERSION DATED 08/14/78 .
CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LAB.

Subroutines and functions

Blas daxpy,dscal,idamax

Gaussian elimination with partial pivoting
```

For calls made by the subroutine, "dgefa":

(See [dscal], page 103.)
(See [daxpy], page 103.)

### 9.5.3 subroutine daxpy

```
CONSTANT TIMES A VECTOR PLUS A VECTOR.
DY(I) = DY(I) + DA * DX(I)
USES UNROLLED LOOPS FOR INCREMENTS EQUAL TO ONE.
JACK DONGARRA, LINPACK, 3/11/78.
```

### 9.5.4 subroutine dscal

```
Scales a vector by a constant.
DX(I) = DA * DX(I)
Uses unrolled loops for increment equal to one.
JACK DONGARRA, LINPACK, 3/11/78.
```

### 9.5.5 subroutine dswap

```
Interchanges two vectors.
DX(I) <==> DY(I)
Uses unrolled loops for increments equal one.
JACK DONGARRA, LINPACK, 3/11/78.
```

### 9.5.6 integer function idamax

```
Finds the index of element having max. absolute value.
JACK DONGARRA, LINPACK, 3/11/78.
```

### 9.5.7 real*8 function ddot

```
Forms the dot product of two vectors.
DOT = DX(I) * DY(I)
Uses unrolled loops for increments equal to one.
JACK DONGARRA, LINPACK, 3/11/78.
```

## 9.6 math_module.F90

### 9.6.1 module math_module

This module contains math parameters

## 9.7 normalization.F90

### 9.7.1 real*8 function xnorm

The purpose of this function is to calculate the normalization
constant of a gtf with exponent a and x,y,z exponents of i,j,k

According to Mathematica, the simplied normalization constant is:

$$\frac{2^{3/4+(i+j+k)/2}\, a^{3/4+(i+j+k)/2}}{\text{Sqrt}[\text{Gamma}[-+i]]^{1}\ \text{Sqrt}[\text{Gamma}[-+j]]^{1}\ \text{Sqrt}[\text{Gamma}[-+k]]^{1}}$$

author Ed Brothers
date October 2001

## 9.8 pdsyevd_driver.F90

### 9.8.1 subroutine pdsyevd_driver

## 9.9 pdsyevx_driver.F90

### 9.9.1 subroutine pdsyevx_driver

## 9.10 zheev.F90

### 9.10.1 subroutine zheev_dummy

### 9.10.2 subroutine sgemv

### 9.10.3 subroutine sgbmv

### 9.10.4 subroutine ssymv

### 9.10.5 subroutine ssbmv

### 9.10.6 subroutine sspmv

### 9.10.7 subroutine strmv

### 9.10.8 subroutine stbmv

### 9.10.9 subroutine stpmv

### 9.10.10 subroutine strsv

### 9.10.11 subroutine stbsv

### 9.10.12 subroutine stpsv

### 9.10.13 subroutine sger

### 9.10.14 subroutine ssyr

### 9.10.15 subroutine sspr

### 9.10.16 subroutine ssyr2

### 9.10.17 subroutine sspr2

### 9.10.18 subroutine dlamc1

### 9.10.19 subroutine dlamc2

For calls made by the subroutine, "dlamc2":

(See [dlamc1], page 106.)
(See [dlamc4], page 106.)
(See [dlamc5], page 106.)

### 9.10.20 subroutine dlamc4

### 9.10.21 subroutine dlamc5

### 9.10.22 subroutine xerbla

### 9.10.23 subroutine zaxpy

### 9.10.24 subroutine zcopy

### 9.10.25 subroutine zdscal

### 9.10.26 subroutine zgemm

For calls made by the subroutine, "zgemm":

(See [xerbla], page 106.)

### 9.10.27 subroutine zgemv

For calls made by the subroutine, "zgemv":

(See [xerbla], page 106.)

### 9.10.28 subroutine zgerc

For calls made by the subroutine, "zgerc":

(See [xerbla], page 106.)

### 9.10.29 subroutine zhemv

For calls made by the subroutine, "zhemv":

(See [xerbla], page 106.)

### 9.10.30 subroutine zher2

For calls made by the subroutine, "zher2":

(See [xerbla], page 106.)

### 9.10.31 subroutine zher2k

For calls made by the subroutine, "zher2k":

(See [xerbla], page 106.)

### 9.10.32 subroutine zscal

### 9.10.33 subroutine zswap

### 9.10.34 subroutine ztrmm

For calls made by the subroutine, "ztrmm":

(See [xerbla], page 106.)

### 9.10.35 subroutine ztrmv

For calls made by the subroutine, "ztrmv":

(See [xerbla], page 106.)

### 9.10.36 subroutine zheev

For calls made by the subroutine, "zheev":

(See [xerbla], page 106.)
(See [zlascl], page 109.)
(See [zhetrd], page 108.)
(See [dsterf], page 112.)
(See [zungtr], page 111.)
(See [zsteqr], page 110.)
(See [dscal], page 103.)

### 9.10.37 subroutine zhetd2

For calls made by the subroutine, "zhetd2":

(See [xerbla], page 106.)
(See [zlarfg], page 109.)
(See [zhemv], page 107.)
(See [zaxpy], page 106.)
(See [zher2], page 107.)

### 9.10.38 subroutine zhetrd

For calls made by the subroutine, "zhetrd":

(See [xerbla], page 106.)
(See [zlatrd], page 109.)
(See [zher2k], page 107.)
(See [zhetd2], page 108.)

### 9.10.39 subroutine zlacgv

### 9.10.40 subroutine zlarf

For calls made by the subroutine, "zlarf":

(See [zgemv], page 106.)
(See [zgerc], page 106.)

### 9.10.41 subroutine zlarfb

For calls made by the subroutine, "zlarfb":

(See [zcopy], page 106.)
(See [zlacgv], page 108.)
(See [ztrmm], page 107.)
(See [zgemm], page 106.)

### 9.10.42 subroutine zlarfg

For calls made by the subroutine, "zlarfg":

(See [zdscal], page 106.)
(See [zscal], page 107.)

### 9.10.43 subroutine zlarft

For calls made by the subroutine, "zlarft":

(See [zgemv], page 106.)
(See [zlacgv], page 108.)
(See [ztrmv], page 107.)

### 9.10.44 subroutine zlascl

For calls made by the subroutine, "zlascl":

(See [xerbla], page 106.)

### 9.10.45 subroutine zlaset

### 9.10.46 subroutine zlasr

For calls made by the subroutine, "zlasr":

(See [xerbla], page 106.)

### 9.10.47 subroutine zlassq

### 9.10.48 subroutine zlatrd

For calls made by the subroutine, "zlatrd":

(See [zlacgv], page 108.)
(See [zgemv], page 106.)
(See [zlarfg], page 109.)
(See [zhemv], page 107.)
(See [zscal], page 107.)
(See [zaxpy], page 106.)

### 9.10.49 subroutine zsteqr

For calls made by the subroutine, "zsteqr":

(See [xerbla], page 106.)
(See [zlaset], page 109.)
(See [dlascl], page 111.)
(See [dlaev2], page 111.)
(See [zlasr], page 109.)
(See [dlae2], page 111.)
(See [dlartg], page 111.)
(See [dlasrt], page 111.)
(See [zswap], page 107.)

### 9.10.50 subroutine zung2l

For calls made by the subroutine, "zung2l":

(See [xerbla], page 106.)
(See [zlarf], page 108.)
(See [zscal], page 107.)

### 9.10.51 subroutine zung2r

For calls made by the subroutine, "zung2r":

(See [xerbla], page 106.)
(See [zlarf], page 108.)
(See [zscal], page 107.)

### 9.10.52 subroutine zungql

For calls made by the subroutine, "zungql":

(See [xerbla], page 106.)
(See [zung2l], page 110.)
(See [zlarft], page 109.)
(See [zlarfb], page 108.)

### 9.10.53 subroutine zungqr

For calls made by the subroutine, "zungqr":

(See [xerbla], page 106.)
(See [zung2r], page 110.)
(See [zlarft], page 109.)
(See [zlarfb], page 108.)

### 9.10.54 subroutine zungtr

For calls made by the subroutine, "zungtr":

(See [xerbla], page 106.)
(See [zungql], page 110.)
(See [zungqr], page 111.)

### 9.10.55 subroutine dladiv

### 9.10.56 subroutine dlae2

### 9.10.57 subroutine dlaev2

### 9.10.58 subroutine dlartg

### 9.10.59 subroutine dlascl

For calls made by the subroutine, "dlascl":

(See [xerbla], page 106.)

### 9.10.60 subroutine dlasrt

For calls made by the subroutine, "dlasrt":

(See [xerbla], page 106.)

### 9.10.61 subroutine dlassq

### 9.10.62 subroutine dsterf

For calls made by the subroutine, "dsterf":

(See [xerbla], page 106.)
(See [dlascl], page 111.)
(See [dlae2], page 111.)
(See [dlasrt], page 111.)

### 9.10.63 logical function lsame

### 9.10.64 real*8 function dcabs1

### 9.10.65 real*8 function dlamch

For calls made by the function, "dlamch":

(See [dlamc2], page 106.)

### 9.10.66 real*8 function dlamc3

### 9.10.67 real*8 function dznrm2

### 9.10.68 real*8 function zlanhe

For calls made by the function, "zlanhe":

(See [zlassq], page 109.)

### 9.10.69 real*8 function dlanst

For calls made by the function, "dlanst":

(See [dlassq], page 112.)

### 9.10.70 real*8 function dlapy2

### 9.10.71 real*8 function dlapy3

### 9.10.72 integer function ieeeck

### 9.10.73 integer function ilaenv

# 10 MC

## 10.1 boxmove.F90

### 10.1.1 subroutine boxmove

```
Does the MC box move for a NPT ensemble
```

## 10.2 boxstep.F90

### 10.2.1 subroutine boxstep

```
acceptance / rejection criteria after boxmove
*****NOTE: this version assumes that every molecule is a residue
```

## 10.3 dovir.F90

### 10.3.1 subroutine dovir

For calls made by the subroutine, "dovir":

(See [etimer], page 205.)

## 10.4 gcartmc.F90

### 10.4.1 subroutine gcartmc

```
SUBROUTINE TO COMPUTE CARTESIAN ENERGY GRADIENT USING VARIATIONAL
FINITE-DIFFERENCE DERIVATIVES.  RETURNS GRADIENT IN KCAL/ANGSTROM.
SEE SUBROUTINE FOCK FOR MORE EXTENSIVE COMMENTS ON ENERGY TERMS.
ONLY INTERMOLECULAR TERMS ARE INCLUDED: INTRAMOLECULAR TERMS
ARE SET TO ZERO
(ASSUMED IS THAT EVERY RESIDUE CORRESPONDS TO ONE MOLECULE
AND VICE VERSA)
```

For calls made by the subroutine, "gcartmc":

(See [etimer], page 205.)
(See [ijfind], page 210.)
(See [pbcxyz], page 213.)
(See [diat], page 28.)

## 10.5 getrforc.F90

### 10.5.1 subroutine getrforc

```
Calculate the force on all residues for a MC calculation
```

## 10.6 mc_module.F90

### 10.6.1 module mc_module

```
Monte Carlo Variables
```

```
To view what this module contains:
```
(See [initialize_mcvars], page 115.)

### 10.6.2 subroutine initialize_mcvars [from module: mc_module]

```
Initialize Monte Carlo Variables
```

## 10.7 mccopy.F90

### 10.7.1 subroutine mccopynew

```
called if new configuration is accepted
```

### 10.7.2 subroutine mccopyold

```
MC routine that is called if new configuration is rejected
```

## 10.8 mcRdKeys.F90

### 10.8.1 subroutine mcrdkeys

```
Read Monte Carlo Keywords
```

For calls made by the subroutine, "mcrdkeys":

(See [rdinum], page 71.)
(See [rdnum], page 71.)

## 10.9 mcsetup.F90

### 10.9.1 subroutine mcsetup

```
Setup MC systems for an MC run.
```

For calls made by the subroutine, "mcsetup":

(See [stripkeymc], page 183.)
(See [opnfil], page 212.)
(See [opnpfil], page 212.)
(See [setup], page 183.)
(See [setbox], page 119.)
(See [setopt], page 183.)
(See [pme_setup], page 173.)
(See [pme_calcb], page 163.)
(See [pme_calctheta], page 164.)
(See [gensub], page 7.)

### 10.9.2 subroutine mcupdate

```
Update MC systems for a MC caculation
```

For calls made by the subroutine, "mcupdate":

(See [gensub], page 7.)

### 10.9.3 subroutine mcclose

```
Write out the MC information from the run
```

For calls made by the subroutine, "mcclose":

(See [wrtdiv], page 80.)

## 10.10 nptdriver.F90

### 10.10.1 subroutine nptdriver

```
Monte Carlo Simulation
```

For calls made by the subroutine, "nptdriver":

(See [mcsetup], page 116.)
(See [energy], page 90.)
(See [atmchg], page 185.)
(See [gcartmc], page 114.)
(See [pme_directmc], page 167.)
(See [pme_recip], page 172.)
(See [pme_recipmc], page 172.)
(See [dovir], page 114.)
(See [wrtcoor], page 79.)
(See [wrtch], page 78.)
(See [printen], page 57.)
(See [mccopynew], page 115.)
(See [mcupdate], page 116.)
(See [submove], page 119.)
(See [resmove], page 118.)
(See [resstep], page 118.)
(See [mccopyold], page 115.)
(See [boxmove], page 114.)
(See [pme_calctheta], page 164.)
(See [boxstep], page 114.)
(See [wrtdiv], page 80.)
(See [mcclose], page 116.)

## 10.11 nvtdriver.F90

### 10.11.1 subroutine nvtdriver

```
Monte Carlo Simulation
```

For calls made by the subroutine, "nvtdriver":

(See [mcsetup], page 116.)
(See [energy], page 90.)

(See [atmchg], page 185.)
(See [gcartmc], page 114.)
(See [pme_directmc], page 167.)
(See [pme_recip], page 172.)
(See [pme_recipmc], page 172.)
(See [dovir], page 114.)
(See [wrtcoor], page 79.)
(See [wrtch], page 78.)
(See [printen], page 57.)
(See [mccopynew], page 115.)
(See [mcupdate], page 116.)
(See [submove], page 119.)
(See [resmove], page 118.)
(See [resstep], page 118.)
(See [mccopyold], page 115.)
(See [wrtdiv], page 80.)
(See [mcclose], page 116.)

## 10.12 resmove.F90

### 10.12.1 subroutine resmove

```
Does the mc move for the residues
```

## 10.13 resstep.F90

### 10.13.1 subroutine resstep

```
acceptance / rejection criteria for MC NVT ensemble
```

## 10.14 rottr.F90

### 10.14.1 subroutine rottr

```
Rotate in a random fashion around an interal molecular axis
```

For calls made by the subroutine, "rottr":

(See [av], page 198.)

## 10.15 setbox.F90

### 10.15.1 subroutine setbox

```
In the following "local" refers to residue i
xyztot -> local vector sum of all coordinates
xyzmin -> global minimum
xyzmax -> global maximum
gcmin  -> global minimum coordinate geometric center residues
gcmax  -> global maximum coordinate geometric center residues
Calculate geometric center and gyration radius of first residue
C   Because center of mass is needed for virial calc, we
C   calculate COM instead of geom. center
-- Arjan --
Setbox is needed for standard calculations if
PME and/or PBC and/or MC are/is used. The same
is true for D&C calculations; in addition setbox
is then needed for some more obscure subsetting
methods.
By the way, standard or D&C has *nothing* to do
with the irpnt array. This array is filled when
the "RESIDUE" keyword is active; which can be
used in standard, as well in D&C calculations.
It just does some extra bookkeeping, which is
absolutely necessary for D&C, and handy for
standard calculations.
A good fix would be to always activate the
"RESIDUE" keyword; in a next version we should
hard-code this keyword, so that it's always active.
```

## 10.16 submove.F90

### 10.16.1 subroutine submove

```
Does the MC move for the residues
```

For calls made by the subroutine, "submove":

(See [rottr], page 118.)

# 11 MEM

## 11.1 allocateDcVars.F90

### 11.1.1 subroutine allocatedcvars

```
Allocates arrays used for DivCon
```

## 11.2 allocateFreqVars.F90

### 11.2.1 subroutine allocatefreqvars

```
Allocates Freq arrays used for DivCon
```

## 11.3 allocateGuessVars.F90

### 11.3.1 subroutine allocateguessvars

```
Allocates GUESS arrays used for DivCon
```

## 11.4 allocateIntVars.F90

### 11.4.1 subroutine allocateintvars

```
Allocates integral arrays used for DivCon
```

## 11.5 allocateMcVars.F90

### 11.5.1 subroutine allocatemcvars

```
Allocates MC arrays used for DivCon
```

## 11.6 allocateMinVars.F90

### 11.6.1 subroutine allocateminvars

```
Allocates Min arrays used for DivCon
```

### 11.6.2 subroutine allocatebellyvars

```
Allocates belly arrays used for DivCon
```

## 11.7 allocateMmVars.F90

### 11.7.1 subroutine allocatemmvars

```
Allocates MM arrays used for DivCon
```

## 11.8 allocateMxAtomVars.F90

### 11.8.1 subroutine allocatemxatomvars

```
ALlocate arrays that depend on mxatom
```

## 11.9 allocateNmrVars.F90

### 11.9.1 subroutine allocatenmrvars1

```
Allocates NMR arrays used in DivCon
```

### 11.9.2 subroutine allocatenmrvars2

```
Allocates NMR arrays used in DivCon
```

## 11.10 allocateParamVars.F90

### 11.10.1 subroutine allocateparamvars

```
Allocates Param arrays used in DivCon
```

## 11.11 allocatePmeVars.F90

### 11.11.1 subroutine allocatepmevars

```
Allocates PME arrays used in DivCon
```

## 11.12 allocateScrfVars.F90

### 11.12.1 subroutine allocatescrfvars

```
Allocates Scrf arrays
```

## 11.13 allocateToolsVars.F90

### 11.13.1 subroutine allocatetoolsvars

```
Allocates Tools arrays
```

## 11.14 allocatevars.F90

### 11.14.1 subroutine allocatevars

```
Allocate all arrays
```

For calls made by the subroutine, "allocatevars":

(See [computemxatts], page 85.)
(See [allocatefreqvars], page 120.)
(See [allocateintvars], page 120.)
(See [allocatemcvars], page 120.)
(See [allocateminvars], page 121.)
(See [computepeptides], page 85.)

(See [allocatemmvars], page 121.)
(See [allocateparamvars], page 122.)
(See [allocatepmevars], page 122.)
(See [allocatescrfvars], page 122.)
(See [allocatetoolsvars], page 122.)

### 11.14.2 subroutine allocatemslistvars

```
Allocate mslist arrays used for DivCon
```

### 11.14.3 subroutine allocatemsorbvars

```
Allocate msorb arrays used for DivCon
```

### 11.14.4 subroutine allocatemsvalvars

```
Allocate msval arrays used for DivCon
```

### 11.14.5 subroutine allocatembpairvars

```
Allocate mbpair arrays used for DivCon
```

For calls made by the subroutine, "allocatembpairvars":

(See [sproc1], page 19.)
(See [bpair], page 199.)

### 11.14.6 subroutine allocatemxdiatvars

```
Allocate mxdiat arrays used for DivCon
```

For calls made by the subroutine, "allocatemxdiatvars":

(See [glbpnt], page 181.)

### 11.14.7 subroutine allocatemxdiagvars

```
Allocate mxdiag arrays used for DivCon
```

### 11.14.8 subroutine allocatemaxrepvars

```
Allocate maxrep arrays used for DivCon
```

### 11.14.9 subroutine allocatemxfdiavars

```
Allocate mxfdiag/mxfdiat arrays used for DivCon
```

## 11.15 deallocatevars.F90

### 11.15.1 subroutine deallocatevars

```
Deallocate all arrays
```

### 11.15.2 subroutine deallocatenmrvars

```
Deallocate NMR arrays
```

# 12 MIN

## 12.1 bsrch.F90

### 12.1.1 subroutine bsrch

```
Does a binary search to find a lower energy in the direction direct.
Parameters are modified according to step*direct, (step/2)*direct,
(step/4)*direct, etc., Until a lower total energy is found.
The step size that lowers the total energy is returned in alpha.
The corresponding total energy is returned in eheat1.
```

For calls made by the subroutine, "bsrch":

(See [getcrd], page 207.)
(See [setbox], page 119.)
(See [gensub], page 7.)
(See [energy], page 90.)
(See [energy_restraints], page 135.)

## 12.2 diislb.F90

### 12.2.1 subroutine diislb

```
we include stuff from LBFGS to perform a -H*G DIIS technique
using DIVCON
```

For calls made by the subroutine, "diislb":

(See [daxpy], page 103.)
(See [lsolve2], page 125.)

### 12.2.2 subroutine lsolve2

```
Uses gaussian elimination with row pivoting to solve the order n
linear system:

a*x = b.

w is a work vector of length n, and thresh is a threshold for
zero pivotal elements of a.
```

```
Error codes:  ierror = 0 - solution found successfully;
              ierror = 1 - zero pivot encountered, singular matrix.
```

## 12.3 diisub.F90

### 12.3.1 subroutine diisub

```
diisub
```

For calls made by the subroutine, "diisub":

(See [savset], page 126.)
(See [lsolve], page 100.)
(See [getcrd], page 207.)

### 12.3.2 subroutine savset

```
Stores coordinates (zmat or xyz), gradient (grad1), and energy (eheat1)
in postion istore of crdset, grdset, and eset, respectively.
```

## 12.4 geoopt.F90

### 12.4.1 subroutine geoopt

```
Geometry optimization routine.
```

For calls made by the subroutine, "geoopt":

(See [allocate_geooptvars], page 128.)
(See [etimer], page 205.)
(See [rdnum], page 71.)
(See [setopt], page 183.)
(See [setbox], page 119.)
(See [gensub], page 7.)
(See [energy], page 90.)
(See [initialize_divpbvars], page 156.)
(See [gcart], page 206.)
(See [set_restraints], page 135.)

(See [dograd], page 205.)
(See [wrtrst], page 82.)
(See [wrtpdb], page 81.)
(See [wrtdmx], page 80.)
(See [freq], page 21.)
(See [flabel], page 127.)
(See [lbfgs], page 128.)
(See [mvcrd], page 212.)
(See [getdir], page 128.)
(See [linmin], page 129.)
(See [bsrch], page 125.)
(See [diisub], page 126.)
(See [grad_restraints], page 135.)

### 12.4.2 subroutine flabel

```
**** NO COMMENTS ****
```

## 12.5 geoopt_module.F90

### 12.5.1 module geoopt_module

```
Working arrays which will be used in geometry optimization process

Target arrays, wdummy and w, are used in the following GEOOPT
subroutines:
bsrch.F, diisub.F, freq.F, geoopt.F, lbfgs.F, linmin.F, getdir.F

Variablbes, Ncopy, JFOLLOW, INDX, RTRUST, are used in the file:
tsqna.F

Variables, CRDSET(MAXPAR,MXSTOR),GRDSET(MAXPAR,MXSTOR),
SHFSET(MAXPAR,MXSTOR),DTEST(MAXPAR),GTEST1(MAXPAR),ESET(MXSTOR),
are used in the file:
diisub.F and diislb.F

  To view what this module contains:
```
(See [initialize_geooptvars], page 127.)
(See [allocate_geooptvars], page 128.)
(See [deallocate_geooptvars], page 128.)

### 12.5.2 subroutine initialize_geooptvars [from module: geoopt_module]

```
Initialize variables which play the role of save attribute.
```

### 12.5.3 subroutine allocate_geooptvars [from module: geoopt_module]

```
Take care of dynamic working arrays in optimization subroutines:
bsrch.F, diisub.F, geoopt.F, lbfgs.F, linmin.F, getdir.F
```

### 12.5.4 subroutine deallocate_geooptvars [from module: geoopt_module]

```
Take care of dynamic working arrays in optimization subroutines:
bsrch.F, diisub.F, freq.F, geoopt.F, lbfgs.F, linmin.F, getdir.F
```

## 12.6 getdir.F90

### 12.6.1 subroutine getdir

```
Determines search direction for optimization.
Supports Steepest Descent, Conjugate-Gradient,
and BFGS.
```

For calls made by the subroutine, "getdir":

(See [getcrd], page 207.)
(See [setbox], page 119.)
(See [gensub], page 7.)
(See [energy], page 90.)
(See [gcart], page 206.)
(See [set_restraints], page 135.)
(See [dograd], page 205.)

## 12.7 lbfgs.F90

### 12.7.1 subroutine lbfgs

```
No Comments Present
```

For calls made by the subroutine, "lbfgs":

(See [daxpy], page 103.)
(See [mcsrch], page 129.)
(See [diislb], page 125.)

### 12.7.2 subroutine lb1

```
No Comments Present
```

### 12.7.3 subroutine mcsrch

```
No Comments Present
```

For calls made by the subroutine, "mcsrch":

(See [mcstep], page 129.)

### 12.7.4 subroutine mcstep

```
No Comments Present
```

## 12.8 linmin.F90

### 12.8.1 subroutine linmin

```
Carries out a line minimization in the direction of the vector
direct.  The energy from a forward half step is combined with
the reference gradient to construct a parabola in the coordinate
alpha*direct.  The predicted minimum coordinate is returned
in alpha, and the cooresponding energy in eheat1.
If the diis flag is turned on, then we will return after the
initial step if the energy drops.  If the energy increases,
however, then the full line minimization will proceed, and diis
will be set to .false. to tell the calling routine not to use
the diis procedure this cycle.
```

For calls made by the subroutine, "linmin":

(See [getcrd], page 207.)
(See [setbox], page 119.)
(See [gensub], page 7.)
(See [energy], page 90.)
(See [energy_restraints], page 135.)

## 12.9 min_module.F90

### 12.9.1 module min_module

```
min variables
```

## 12.10 minRdKeys.F90

### 12.10.1 subroutine minrdkeys

```
Read min Related Keywords
```

For calls made by the subroutine, "minrdkeys":

(See [rdnum], page 71.)

## 12.11 minRdTail.F90

### 12.11.1 subroutine minrdtail

```
Read min Related Tail
```

For calls made by the subroutine, "minrdtail":

(See [rdbelly], page 58.)

# 13 MM

## 13.1 getmm.F90

### 13.1.1 subroutine getmm

```
Get MM
```

## 13.2 getpep.F90

### 13.2.1 subroutine getpep

```
Locates and stores atoms involved in each peptide bond.  The
Arrangement H-N-C-O is located on the basis of interatomic
distances.  the atoms H-N-C-O are stored in iabc, as well as
X-N-C-O, where X is the other atom bonded to the nitrogen:

                      X
                     /
                H--N
                     \
                      C=O
                     /

The total number of quadruples stored in iabc is n2pep.
iabc and n2pep are returned via common block /peptid/

error flag is set to 1 if storage limits are exceeded.
```

## 13.3 ljglobals.F90

### 13.3.1 module ljglobals

```
Lennard-Jones Globals
```

## 13.4 ljint.F90

### 13.4.1 subroutine ljint

```
author Kaushik Raha
date 12/05/03

 This subroutine will calculate lennard jones energy and
 pairwise interaction based on lj energy
```

### 13.4.2 subroutine psm

```
author Kaushik Raha

 Phenomenonological Solvation Model (PSM) or moelcular recognition model
 adapted from VERKHIVER ET. AL., J.MOL.RECOG. 12 (1999) 371
 Interaction energy einter is calculated as
 Einter = Edr + Ees + Esol
 Edr is the dispersion/repulsion energy calculated as
 EDR = AIJ/(RIJ^6+DEL^6)^2 - BIJ/(RIJ^6+DEL^6)
 Here RIJ is the distance between atoms I AND J
 DEL Is the soft core value repulsion for dr interaction
 DEL = 2.75 Ao. AIJ and BIJ depend on epslon and sigma from amber
 Ees is the electrostatic energy calculated as
 EES = 332.0*QI*QJ/D(RIJ^6+DEL^6)^1/3
 DEL = 1.75 Ao. and QI and QJ are charges from qm calculation
```

### 13.4.3 subroutine veccross

```
 Vector Cross
```

### 13.4.4 subroutine vecdiff

```
 Vector Diff
```

### 13.4.5 real*8 function angle

```
Angle
```

For calls made by the function, "angle":

(See [vecdiff], page 132.)

### 13.4.6 real*8 function dihedral

```
Dihedral
```

For calls made by the function, "dihedral":

(See [vecdiff], page 132.)
(See [veccross], page 132.)

## 13.5 mm_module.F90

### 13.5.1 module mm_module

```
This module cantains variables concerning the Molecular Mechanics
corrections added to the SE methods.
```

```
To view what this module contains:
```

(See [initialize_mmvars], page 133.)

### 13.5.2 subroutine initialize_mmvars [from module: mm_module]

```
Initialize MM variables
```

## 13.6 mmDriver.F90

### 13.6.1 subroutine mmdriver

```
MM Driver
```

For calls made by the subroutine, "mmdriver":

(See [ljint], page 132.)
(See [psm], page 132.)

## 13.7 mmRdKeys.F90

### 13.7.1 subroutine mmrdkeys

```
Read MM Keywords
```

For calls made by the subroutine, "mmrdkeys":

(See [rdnum], page 71.)

## 13.8 mmRdTail.F90

### 13.8.1 subroutine mmrdtail

```
Read mm Related Tail
```

For calls made by the subroutine, "mmrdtail":

(See [rdvdw], page 77.)
(See [rdrestraints], page 74.)

## 13.9 mmroutines.F90

### 13.9.1 subroutine bond_e

```
Calculate Bond Energy
```

### 13.9.2 subroutine angle_e

```
Calculate Angle Energy
```

For calls made by the subroutine, "angle_e":

(See [bndang], page 199.)

### 13.9.3 subroutine tor_e

```
Calculate Torsion Energy
```

## 13.10 restraint.F90

### 13.10.1 subroutine set_restraints

```
This subroutine adds harmonic restraints for minimizations
Andrew Wollacott, Dec 04, 2004
```

### 13.10.2 subroutine energy_restraints

```
harmonic potential of the form:  E = K * R^2
where K = force constant
R = distance from original position.

E   = K * R^2
R^2 = (Xi - Xio)^2 + (Yi - Yio)^2 + (Zi - Zio)^2

dE/dR = 2KR
dR/dX = 2(Xi - Xio) * (1/2R)
dE/dx = 2K * (Xi - Xio)
=>
dE/dy = 2K * (Yi - Yio)
dE/dz = 2K * (Zi - Zio)

F(i) = F(i) - dE/dx

The basis of this code was taken from the sander program of amber6


harmonic restraints only setup for cartesian coordinates
```

### 13.10.3 subroutine grad_restraints

# 14 MPI

## 14.1 balance.F90

```
author Jim Vincent

 Replace these two routines with just second sort by time
 routine but for first pass fill in time_subs with
 the orbital counts instead of actual times
 and just use that to balance by
```

### 14.1.1 subroutine balance_bynorbs

```
Load balance MPI calculation by number of orbitals
```

For calls made by the subroutine, "balance_bynorbs":

(See [bsort], page 200.)

### 14.1.2 subroutine balance_bytime

```
Load balance MPI calculation by number of orbitals
```

For calls made by the subroutine, "balance_bytime":

(See [rsort], page 201.)

### 14.1.3 subroutine mpi_error_check

```
Check a few possible errors that would only happen if running
parallel version
```

# 15 NMR

## 15.1 assignbasis.F90

### 15.1.1 subroutine assignbasis

```
author Ed Brothers
date 02/18/02

 Assigns basis
```

## 15.2 chshift.F90

### 15.2.1 subroutine chshift

```
 Chemical Shift
```

For calls made by the subroutine, "chshift":

(See [rdinum], page 71.)
(See [diam], page 137.)
(See [paramg], page 137.)
(See [ijfind], page 210.)
(See [apscreen], page 138.)
see system command "ssyevx"
see system command "dsyevx"
see system command "dspev"
(See [diag], page 96.)
(See [deallocatenmrvars], page 124.)

### 15.2.2 subroutine diam

```
author Bing Wang
date 09/08/2002

 This routine calculates the diamagnetic integrals
```

### 15.2.3 subroutine paramg

```
author Bing Wang
date 09/18/2002

 this routine calculates the paramagnetic integrals
```

### 15.2.4 subroutine diam1

```
author Bing Wang
date 09/16/2002

 This routine calculates the paramagnetic integrals
```

### 15.2.5 subroutine diam2

```
author Bing Wang
date 09/16/2002

 This routine calculates the paramagnetic integrals
```

### 15.2.6 subroutine apscreen

```
author Bing Wang
date 09/18/2002

 This routine estimates the NMR integrals
```

## 15.3 mndonmr_module.F90

### 15.3.1 module mndonmr_module

```
 MNDO/NMR Semiempirical parameters

To view what this module contains:
```
(See [initialize_mndonmrvars], page 138.)

### 15.3.2 subroutine initialize_mndonmrvars [from module: mndonmr_module]

```
Initialize MNDO/NMR Parameters
```

## 15.4 nmr_module.F90

### 15.4.1 module nmr_module

```
NMR module
```

```
To view what this module contains:
```

(See [initialize_nmrvars], page 139.)

### 15.4.2 subroutine initialize_nmrvars [from module: nmr_module]

```
Initialize NMR variables
```

## 15.5 nmrdenful.F90

### 15.5.1 subroutine nmrdenful

```
Computes the global density matrix (pdiag,pdiat) for the full system
in a calculation that does not involve separate subsystems.
```

For calls made by the subroutine, "nmrdenful":

(See [ijfind], page 210.)

## 15.6 nmrdensub.F90

### 15.6.1 subroutine nmrdensub

```
Adds in density matrix contribution from subsystem k.  It is
assumed that it will be faster to subtract electrons from a
saturated density matrix, which is just 2.0 times the identity
matrix.
```

For calls made by the subroutine, "nmrdensub":

(See [ijfind], page 210.)

## 15.7 nmrdmx.F90

### 15.7.1 subroutine nmrdmx

```
Determines the molecular orbitals for all subsystems.

goodf =  flag for a good estimate of the fermi energy.  when
the density matrix has just been initialized to
diagonal form, then we don't have a good estimate
of efermi.  in this case, two rounds of diagonalizations
will be necessary to determine the global density
matrix.

iter   = scf iteration number: 1,2,...

ndblscf= two diagonalization passes are always performed regardless
if iter <= ndblscf

pseudo = logical flag do carry out pseudo-diagonalization

returned error codes:

ierror = 0 --> okay
ierror = 1 --> (1) qr iteration in diag didn't converge, or
               (2) fermi energy in doferm didn't converge

note: this routine uses the /work/ common block for temporary
storage.  anything in ff, evec, or ww below will be destroyed.
```

For calls made by the subroutine, "nmrdmx":

(See [balance_bynorbs], page 136.)
(See [ijfind], page 210.)
(See [diagp], page 99.)
see system command "ssyevx"
(See [zheev], page 107.)
see system command "zhpev"
(See [nmrdenful], page 139.)
(See [nmrdensub], page 139.)

(See [nmresqr], page 141.)
(See [wrtvecsub], page 82.)
(See [etimer], page 205.)
(See [doferm], page 5.)
see system command "mpi_allreduce"

## 15.8 nmresqr.F90

### 15.8.1 subroutine nmresqr

```
Determines the coefficients evecsq for subsystem k:

evecsq(l0+l) = d11*evec1(1,l)**2 + d22*evec1(2,l)**2 + . . .

Here l0 is just a global pointer for subsystem k, and dii is
the normalization factor that accounts for subsystem overlap.

Once all of the entries of evecsq have been determined, then
the number of electrons can be computed for any set of fermi
occupation numbers:

no. of electrons = fermi(1)*evecsq(1) + fermi(2)*evecsq(2) + . . .
```

## 15.9 nmrRdKeys.F90

### 15.9.1 subroutine nmrrdkeys

```
Read NMR Keywords
```

For calls made by the subroutine, "nmrrdkeys":
(See [rdinum], page 71.)

## 15.10 nmrRdTail.F90

### 15.10.1 subroutine nmrrdtail

```
Read nmr Related Tail
```

For calls made by the subroutine, "nmrrdtail":
(See [rdnmr], page 70.)

## 15.11 oethccs.F90

### 15.11.1 real*8 function oethccs1

```
author Bing Wang
date August 14, 2002

 Calculates [rbrc-rbarcb]/rc3 type one-electron three-center
 integrals for NMR chemical shieldings

 Arguments
   a       -  the orbital exponent for A
   b       -  the orbital exponent for B
   i,j,k   -  the angular momentum indexes for A
   ii,jj,kk-  the angular momentum indexes for B
   Ax,Ay,Az-  the coordinates for A
   Bx,By,Bz-  the coordinates for B
   Cx,Cy,Cz-  the coordinates for C
   ia,ib   -  the directions for integrals

 The this is taken from the recursive relation found in Obara and Saika,
 J. Chem. Phys. 84 (7) 1986, 3963.
```

### 15.11.2 real*8 function oethccs2

```
 Calculates Lcia/rc3 type one-electron three-center integral for
 NMR chemical shieldings

author Bing Wang
date August 14, 2002

 ARGUMENTS
   a       -  the orbital exponent for A
   b       -  the orbital exponent for B
   i,j,k   -  the angular momentum indexes for A
   ii,jj,kk-  the angular momentum indexes for B
   Ax,Ay,Az-  the coordinates for A
   Bx,By,Bz-  the coordinates for B
```

```
  Cx,Cy,Cz-  the coordinates for C
  ia,ib   -  the directions for integrals

 The this is taken from the recursive relation found in Obara and Saika,
 J. Chem. Phys. 84 (7) 1986, 3963.
```

### 15.11.3 real*8 function oethccs3

```
 Calculates (RABra)Lci/rc3 one-electron three-center integrals for
 NMR chemical shieldings

author Bing Wang
date August 14, 2002

 ARGUMENTS
  a       -  the orbital exponent for A
  b       -  the orbital exponent for B
  i,j,k   -  the angular momentum indexes for A
  ii,jj,kk-  the angular momentum indexes for B
  Ax,Ay,Az-  the coordinates for A
  Bx,By,Bz-  the coordinates for B
  Cx,Cy,Cz-  the coordinates for C
  ia,ib   -  the directions for integrals

 The this is taken from the recursive relation found in Obara and Saika,
 J. Chem. Phys. 84 (7) 1986, 3963.
```

### 15.11.4 real*8 function oethccs4

```
author Bing Wang
date August 14, 2002

 Calculates (QAB)Lci/rc3 one-electron three-center integrals for
 NMR chemical shieldings

 ARGUMENTS
  a       -  the orbital exponent for A
  b       -  the orbital exponent for B
```

```
  i,j,k   -  the angular momentum indexes for A
  ii,jj,kk-  the angular momentum indexes for B
  Ax,Ay,Az-  the coordinates for A
  Bx,By,Bz-  the coordinates for B
  Cx,Cy,Cz-  the coordinates for C
  ia,ib   -  the directions for integrals

 The this is taken from the recursive relation found in Obara and Saika,
 J. Chem. Phys. 84 (7) 1986, 3963.
```

### 15.11.5 real*8 function oethccs5

```
author Bing Wang
date August 14, 2002

 Calculates Lcia type one-electron two-center integral for
 NMR chemical shieldings

 ARGUMENTS
  a       -  the orbital exponent for A
  b       -  the orbital exponent for B
  i,j,k   -  the angular momentum indexes for A
  ii,jj,kk-  the angular momentum indexes for B
  Ax,Ay,Az-  the coordinates for A
  Bx,By,Bz-  the coordinates for B
  Cx,Cy,Cz-  the coordinates for C
  ia,ib   -  the directions for integrals

 The this is taken from the recursive relation found in Obara and Saika,
 J. Chem. Phys. 84 (7) 1986, 3963.
```

### 15.11.6 real*8 function oethccs6

```
author Bing Wang
date August 14, 2002

 Calculates dipole moment type one-electron two-center integral for
```

```
NMR chemical shieldings

ARGUMENTS
  a       -  the orbital exponent for A
  b       -  the orbital exponent for B
  i,j,k   -  the angular momentum indexes for A
  ii,jj,kk-  the angular momentum indexes for B
  Ax,Ay,Az-  the coordinates for A
  Bx,By,Bz-  the coordinates for B
  Cx,Cy,Cz-  the coordinates for C
  ia,ib   -  the directions for integrals

The this is taken from the recursive relation found in Obara and Saika,
J. Chem. Phys. 84 (7) 1986, 3963.
```

# 16 PARAM

## 16.1 coppnt.F90

### 16.1.1 subroutine coppnt

```
author Ed Brothers
date July 1999

 The pupose of this subprogram is to vector of semiempirical
 parameters to be the parameters in use, and recalculate the
 derived parameters.
```

For calls made by the subroutine, "coppnt":

(See [recalc], page 149.)

## 16.2 error.F90

### 16.2.1 subroutine error

```
author Ed Brothers
date August 1998

 The purpose of this routine is to calculate the error of the calculated
 parameters of this molecule when compared to literature or "correct"
 values.  While this has little value as a stand alone, it is necessary
 as a portion of the parameterization code.
```

For calls made by the subroutine, "error":

(See [dipole], page 187.)

## 16.3 loop.F90

### 16.3.1 subroutine loop

```
author Ed Brothers
date August 1998

 The pupose of this subprogram is to loop thought multiple input files
```

```
and perform divcon calculations on each of them.  Due to the nature
of this process, the entire file is a large ioop.

Please note that this was devised with standard calculations in mind.
If other types of calculations are desired, retrofitting additional
code may be necessary.  Also, testing and developement was not performed
on systems with residue pointers, thus there may be difficulties there.
```

For calls made by the subroutine, "loop":

(See [etimer], page 205.)
(See [rdkeys], page 68.)
see system command "rdkey2"
(See [edriver], page 89.)

## 16.4 param_module.F90

### 16.4.1 module param_module

```
SE Parameter variables

 To view what this module contains:
```

(See [initialize_paramvars], page 147.)

### 16.4.2 subroutine initialize_paramvars [from module: param_module]

```
Initializes SE Parameter variables.
```

## 16.5 paramRdKeys.F90

### 16.5.1 subroutine paramrdkeys

```
Read SE Parameter Keywords
   - testpoint
   - powell
   - ga
   - external
   - notnamed=mndo
   - notnamed=am1
```

```
- notnamed=pm3
- notnamed=mndod
- prtpar
```

For calls made by the subroutine, "paramrdkeys":

(See [tstpnt], page 149.)
(See [rdinum], page 71.)
(See [parop], page 148.)
(See [parop2], page 148.)
(See [rdnum], page 71.)
(See [loop], page 146.)

## 16.6 parop.F90

### 16.6.1 subroutine parop

```
author Ed Brothers
date July 1999

 The pupose of this subprogram is to optimize semi-empirical parameters
 for elements of the user's choice using a POWELL linear optimizer.
 This will also include the bisection search for 1D minimization.
```

For calls made by the subroutine, "parop":

(See [upcase], page 216.)
(See [rdnum], page 71.)
(See [getpar], page 207.)
(See [rdword], page 77.)
(See [loop], page 146.)
(See [coppnt], page 146.)

## 16.7 parop2.F90

### 16.7.1 subroutine parop2

```
author Ed Brothers
date December 1998

 The pupose of this subprogram is to optimize semi-empirical parameters
```

```
for elements of the user's choice using a genetic algorithm.
```

For calls made by the subroutine, "parop2":

(See [upcase], page 216.)
(See [rdnum], page 71.)
(See [getpar], page 207.)
(See [rdword], page 77.)
(See [recalc], page 149.)
(See [loop], page 146.)
(See [rsort], page 201.)

## 16.8 recalc.F90

### 16.8.1 subroutine recalc

```
author Ed Brothers
date November 1998
date June 1999 (modified)
reference J. Comp.-Aid. Mol. Des., vol. 4, 1990, 1-45.

 The pupose of this subprogram is to calculate the new values of the DL
 and AL parameters based on values of the Slater exponents and the 2 center
 1 electron integrals.  This is necessary as Slater expnt, etc are moved
 in the param'tion procedure.  Note this process is iterative, and hence
 slightly expensive. A later version will only call this when necessary
 rather than on each parameterization loop call.

 Please see JJP Stewart's paper which contains easy explanations of these
 parameters.

 The first thing to be calculated is the DL terms.
```

## 16.9 tstpnt.F90

### 16.9.1 subroutine tstpnt

```
author Ed Brothers
date July 1999

 The pupose of this subprogram is to test GA results.
```

For calls made by the subroutine, "tstpnt":

(See [upcase], page 216.)
(See [rdnum], page 71.)
(See [getpar], page 207.)
(See [rdword], page 77.)
(See [loop], page 146.)

# 17 PB

## 17.1 divpb.F90

### 17.1.1 subroutine divpb

```
divpb is the main subroutine of the Poisson-Boltzmann solver using
Finite Difference Method.

The subroutine is supposed to be called from divcon for serveral
times. Atom names, coordinates and partial charges will be passed
from DIVCON. Surface charges together with their coordinates,
reaction field energy and surface area will be passed back to
DIVCON after the PB calculation.
```

For calls made by the subroutine, "divpb":

(See [dpinit], page 154.)
(See [dpgetmol], page 154.)
(See [dpchargegrid], page 152.)
(See [dpmakesurface], page 156.)
(See [dpaccelerate], page 151.)
(See [dploop], page 155.)
(See [dpcollectsurfcharge], page 152.)

## 17.2 dpaccelerate.F90

### 17.2.1 subroutine dpaccelerate

```
dpAccelerate prepares arrays needed to accelerate the relaxation
of phi matrix.

bModified: define whether the coordinates of solute is changed
           from last divpb run. If not, only charge acc array is updated.
rSpectral: The spectral needed for phi relaxation in dploop. Only
           calculated the first time the solute coordiantes is changed.

The basic idea of accelerating is the epslions are largely uniform
in the grid space, also the charged grids are small amount. So the
```

```
PB equation on most grid points are just laplace equation. We just
calculate the laplace equation for all the grid points and then add
the effects caused by non-uniformed epsilon at dielectric boundary
and by solute charges. When there is salt in solution, things turn
to be a little bit complex. The laplace equation always has a term
of kappa - the inverse of deby length. Normally half the grids in
salt and half not. So it is no gain in speed to calculate all as if
they are in/out salt and "patch" the other half. So a full size
array is set up for kappa term of all grid points. This array will
be filled and used only when there is salt to save time (by using
laplace + boundary patch + charge patch) for mostly often met pure
solution situation.
```

## 17.3 dpchargegrid.F90

### 17.3.1 subroutine dpchargegrid

```
first allocate maximum charge grid array = 8 x gnMolAtomNum
now only save the list of charged grid, no longer using the
gnGridNum x gnGridNum x gnGridNum array to save "0" on most
grid points. This can save a lot memory.
```

## 17.4 dpcollectsurfcharge.F90

### 17.4.1 subroutine dpcollectsurfcharge

```
dpCollectSurfCharge will generate solvate-accessible surface
points and calculate the charges on these points based on the
relaxed phi on surrounding grid points.

grMolarea and graAtarea stores the molecular surface area of
the solute and each atom. They will be returned to divcon,
mainly for Ed's new non-polar parameterization use.

Since this is the last subroutine in regular PB calculation,
the non-uniform grid (aka. finesurf) codes are called here
after the finish of the coarser grid calculation.
```

For calls made by the subroutine, "dpcollectsurfcharge":

(See [dpmakesurface], page 156.)
(See [dpfinesurfacc], page 153.)
(See [dpfinesurfrelax], page 153.)

## 17.5 dpfinesurfacc.F90

### 17.5.1 subroutine dpfinesurfacc

```
dpFineSurfAcc does pretty much the same thing as dpAccelerate,
just on the finer grid space.

bModified: defined whether the coordinates of solute is changed
           after last divpb run.
rSpectral: the spectral of finer grid, will be used in dpFineSurfRelax.
```

## 17.6 dpfinesurfprep.F90

### 17.6.1 subroutine dpfinesurfprep

```
dpFineSurfPrep is to generate a finer grid around the dielectric
boundary and allocate all the arrays will be used later, like the
the new g.u. atom coordinates, charged grid points, and epsilons
on the new grid. Also redefine the grid boundary.
```

For calls made by the subroutine, "dpfinesurfprep":

(See [dpsetepsilon], page 157.)

## 17.7 dpfinesurfrelax.F90

### 17.7.1 subroutine dpfinesurfrelax

```
dpFineSurfRelax is to relax the finer grid around the dielectric
boundary using the arrays processed in dpFineSurfAcc.
This subroutine consists of two parts:
      1st, interpolate the phi's at the double grid points from
           the relaxed coarser original grid points if not almost
           converged.
```

```
        2nd, loop through all the points (for 20 cycles, and then
             local points only) and relax the phi's at the double
             grid points, or local points only if almost converged.

bAlmostConv: boolean variable set in dploop, which shows how close
             the QM/PB SCRF calculation to convergence.
rSpectral & rDblSpec: both coarser & finer grid spectrals are needed
             for faster PB relaxation on finer grid.
```

## 17.8 dpgetmol.F90

### 17.8.1 subroutine dpgetmol

```
dpGetMol get atom charges, coordinates, brief info from DIVCON
radii information got from array gsaRadAtomName and graRadAtomRadii
If the solute's structure is not modified, update charge only.
Otherwise all information are updated.
```

## 17.9 dpinit.F90

```
Keywords & radii initialization for divpb
```

### 17.9.1 subroutine dpinit

```
dpInit is used to perform basic initialization for divpb, including
keywords reading and radii file reading. Now the keywords reading
is integrated in the divcon's keyword reading routine. Radii file
reading ability is kept for flexibility, says easily scale radii to
do parameterization.
```

### 17.9.2 subroutine dpreadparam

```
dpReadParam is an obsolete subroutine used to reading parameter
for divpb from a parameter file. No longer used.
```

For calls made by the subroutine, "dpreadparam":

(See [stripkeywd], page 183.)
(See [upcase], page 216.)
(See [wdjoin], page 216.)
(See [rdword], page 77.)
(See [rdnum], page 71.)

### 17.9.3 subroutine dpreadradii

```
dpReadRadii read atom radii from file defined in dpparameter.h
Although the radii information now stored in dpmodule, this
subroutine is kept for possible future radii parameterization.
```

## 17.10 dploop.F90

### 17.10.1 subroutine dploop

```
dploop do the real Finite Difference Poisson-Boltzmann relaxation
of electronic potential (phi) on all grid points. Here the over-
relaxation method is used. The rOmega is the relaxation factor,
which is calculated from rSpectral.

rSpectral: the spectral obtained from dpAccelerate, used to caculate
           the over-relaxation factor
bAlmostConv: Compare current relaxation loops with the first time
             relaxation loop number. If the relaxation is close to
             convergence, this flag is set to TRUE for later use.
```

## 17.11 dpmakegrid.F90

### 17.11.1 subroutine dpmakegrid

```
dpMakeGrid generate 3D grids for FDM Possion-Boltzmann solution
according to the span of molecule in space and parameters, like
grGridPerAng, grPerFill etc.
```

## 17.12 dpmakesurface.F90

### 17.12.1 subroutine dpmakesurface

```
dpMakeSurface call pb_mds routine to generate molecular surface (not
SAS, but atom sphere + reentrant area) and extended van der waals
surface (ri+rp). The molecular surface is used to determine position
of surface charge. The extended vdw surface is for epsilon assignment.

nType: the type of surface to make - currently either MS_SURF or
       VDW_SURF, which are defined in dpparameters.h
```

For calls made by the subroutine, "dpmakesurface":

(See [pb_mds], page 158.)

## 17.13 dpmodule.F90

```
Contains all global variables used in divpb.
```

### 17.13.1 module divpb_private

```
Interface for DIVPB, any subroute outside of DIVPB need to use this
module: "use divpb_interface"



Private global data for DIVPB use only
```

```
To view what this module contains:
```

(See [initialize_divpbvars], page 156.)
(See [deallocate_divpbvars], page 156.)

### 17.13.2 subroutine initialize_divpbvars [from module: divpb_private]

```
Initialize variables (used in DivPB calculation) which need to be
fresh ones for multiple runs of divcon jobs (NOT divpb jobs).
```

### 17.13.3 subroutine deallocate_divpbvars [from module: divpb_private]

Deallocate dynamic arrays used in DivPB calculation.

## 17.14 dpsetepsilon.F90

### 17.14.1 subroutine dpsetepsilon

dpSetEpsilon decide whether each grid-line-middle-point belongs to outside or inside of the molecule and thus assign different epsilon value to the points. First set every point within globe (R=Ri+Rp) of each atom as "inside". Then put prob on each surface point (this surface inculde the prob's radius) and reset those points falling into prob globe to "outside". Thus the reentrent area get "inside" epsilon value.

## 17.15 dpsetphi.F90

### 17.15.1 subroutine dpsetphi

dpSetPhi set the initial phi for both boundary grid points and all the grid points which will be relaxed later. The boundary condition is set by counting all the atomic charge of the solute.

## 17.16 nonpolar2.F90

### 17.16.1 subroutine nonpolar2

New Gnp using atom type based parameters. Defaultly use the total solute surface and Gnp = a + bS. This new method use a set of b for different atom types

gnp2: the return value of new Gnp

## 17.17 pb_mds.F90

### 17.17.1 subroutine pb_mds

```
Molecular dot surface computation subroutine
```

For calls made by the subroutine, "pb_mds":

(See [pb_cross], page 159.)
(See [pb_subcir], page 159.)
(See [pb_normal], page 159.)
(See [pb_subarc], page 159.)
(See [pb_putpnt], page 158.)
(See [pb_normal], page 159.)
(See [pb_subarc], page 159.)
(See [pb_putpnt], page 158.)

## 17.18 pb_putpnt.F90

### 17.18.1 subroutine pb_putpnt

```
Calculate points on each arc according to the pre-set density.
The actual point coordinates collection is done by pb_spt.
```

For calls made by the subroutine, "pb_putpnt":

(See [pb_spt], page 158.)
(See [pb_normal], page 159.)

## 17.19 pb_spt.F90

### 17.19.1 subroutine pb_spt

```
Transfer surface point coordinates/area/outnormal to divcon arrays
the divcon arrays storing vdw & MS surface points will grow automaticly
to contain all surface points. so there is no longer need of MAXSCHG
```

## 17.20 pb_subdiv.F90

```
Here is a collection of those subdivision routines used by pb_mds.
```

### 17.20.1 subroutine pb_subdiv

```
Divide the arc/circle into segments based on the pre-set density
```

### 17.20.2 subroutine pb_subcir

```
Make a full circle.
```

For calls made by the subroutine, "pb_subcir":

(See [pb_normal], page 159.)
(See [pb_cross], page 159.)
(See [pb_subdiv], page 159.)

### 17.20.3 subroutine pb_subarc

```
Make an arc.
```

For calls made by the subroutine, "pb_subarc":

(See [pb_cross], page 159.)
(See [pb_subdiv], page 159.)

## 17.21 pb_vector.F90

```
Here is a collection of those vector calculations used by pb_mds.
```

### 17.21.1 subroutine pb_cross

```
Calculate the cross product between two vectors.
```

### 17.21.2 subroutine pb_normal

```
Calculate the outward normal at point x on the surface.
```

### 17.21.3 real function pb_dis

```
Calculate the norm between two vectors.
```

### 17.21.4 real function pb_dis2

```
Calculate the square norm between two vectors.
```

### 17.21.5 real function pb_disptl

```
Calculate the distance from a point to a line.
```

### 17.21.6 real function pb_dot

```
Calculate the dot product between two vectors.
```

### 17.21.7 real function pb_triple

```
Calculate the cross product between vectors c and d
and the dot product between the cd and e vectors.
```

For calls made by the function, "pb_triple":

(See [pb_cross], page 159.)

## 17.22 pbDriver.F90

### 17.22.1 subroutine pbdriver

```
pbDriver is detached from original edriver.
```

For calls made by the subroutine, "pbdriver":

(See [setprtvec], page 215.)
(See [wrtvec], page 82.)
(See [wrtvecsubf], page 82.)
(See [pbsolver], page 162.)
(See [energy], page 90.)

(See [etimer], page 205.)
(See [deallocate_divpbvars], page 156.)
(See [nonpolar2], page 157.)

## 17.23 pbfock.F90

### 17.23.1 subroutine pbfock

```
pbfock does:
1. Transfer surface charges from the Poisson-Boltzmann solver
2. Calculates the reaction field contribution to the Fock matrix
2a. surface charge-core interaction energies
2b. surface charge-electron interaction (1 electron integrals)

NOTE: In semiempirical methods core-core and core-electron energies
are calculated using 2 electron integrals. This is because the
'penetration' integrals where removed (CNDO/2), consequently there is
no genuine 1 electron integrals (except for 'atomic' integrals). The
use of 2 electron integral to calculate surface charge interaction
with the core and electron requires the assignment of chemical ident-
tity to surface charges. Thus, the surface charges are assumed to be
hydrogen cores. Thus integrals of <ss|ss>, <ss|sp> and <ss|pp'> are
used to evaluate the matrix elements due to  surface charges.
Because 2 electron 2 center integrals (calculated using multipole
approx) are parameterized to give 2 electron 1 center integral in
zero distance limit, they are basically inapropriate for calculating
matrix elements involving surface charges, but because all formulas
used in semiempirical calculations are unbalanced, the use of correct
formulas for surface charge - core/electron interaction does not
guaranty good computational results. Consequently, the 2 electron
integral formulas were also used to calculate the fock matrix elems
due to the surface charges.
These matrix elements are calculated only at the begining of the SCF
```

For calls made by the subroutine, "pbfock":

(See [pbsolver], page 162.)
(See [diat], page 28.)
see system command "mpi_allreduce"

## 17.24 pbRdKeys.F90

### 17.24.1 subroutine pbrdkeys

```
Read PB Keywords
```

For calls made by the subroutine, "pbrdkeys":

(See [rdnum], page 71.)

## 17.25 pbsolver.F90

### 17.25.1 subroutine pbsolver

```
The pbsolver does:
  1. Write surface charges to file, backups density marix and atom
     charges files.
  2. Pass on atom info, coordinates, charges to divpb subroutine
  3. Calls divpb which solves Poisson-Boltzmann equation
     and returns a set of surface charges which are used to modify
     the Fock matrix and redo the SCF calculations

NOTE: the surface charges are passed to  subroutine pbfock
which calculates the reaction field contribution to Fock matrix
```

For calls made by the subroutine, "pbsolver":

(See [etimer], page 205.)
(See [atmchg], page 185.)
(See [divpb], page 151.)
see system command "mpi_bcast"

# 18 PME

## 18.1 pme_calcb.F90

### 18.1.1 subroutine pme_calcb

```
this routine calculates |1/bi|**2, i=x,y,z as given by [Eq.4.4]
of Essmann et.al., J.Chem.Phys. 103 8577
written by Arjan van der Vaart
```

For calls made by the subroutine, "pme_calcb":

see system command "mkbspline"

## 18.2 pme_calcq.F90

### 18.2.1 subroutine pme_calcq

```
author Arjan van der Vaart
date Dec. 1997
reference J.Chem.Phys. 103, 8577, 1995

 This routine calculates Q as defined in Eq.4.6 of
 J.Chem.Phys. 103 ('95) 8577.

 compiler directives:
 *) MEMORY_OVERLAP         if defined, the pme variables will
                           share memory with the eigenvectors /
                           eigenvalues

 QPME_, a wrapped-around real array, will be filled
          with the values of Q.

 Note that the performance of this algorithm is very high
 by demanding that nspline <= K_i, i=1,2,3 and by
 shifting the scaled fractional coordinates upme such that
 0 <= upme_i <= K_i, i=1,2,3.

 Q needs to be recalculated whenever the coordinates or
 charges change.
```

For calls made by the subroutine, "pme_calcq":

see system command "mbspline1"

## 18.3 pme_calcqmc.F90

### 18.3.1 subroutine pme_calcqmc

```
this routine calculates Q as defined in Eq.4.6 of
J.Chem.Phys. 103 ('95) 8577 of residue ir.
compiler directives:
*) MEMORY_OVERLAP         if defined, the pme variables will
.                         share memory with the eigenvectors /
.                         eigenvalues
QPME_, a wrapped-around real array, will be filled
.        with the values of Q.
Written by Arjan van der Vaart, Dec. '97
Note that the performance of this algorithm is very high
by demanding that nspline <= K_i, i=1,2,3 and by
shifting the scaled fractional coordinates upme such that
0 <= upme_i <= K_i, i=1,2,3.
Q needs to be recalculated whenever the coordinates or
charges change.
```

For calls made by the subroutine, "pme_calcqmc":

see system command "mbspline1"

## 18.4 pme_calctheta.F90

### 18.4.1 subroutine pme_calctheta

```
this routine calculates the array thetapme = F(C.B)
as defined in J.Chem.Phys. 103 ('95) 8577
thetapme is needed in the calculation of the reciprocal
energy:
.   Erec = 0.5 Sum{m} [ Q Conv(thetapme . Q) ]    [Eq.4.7]
thetapme is stored in wrapped-around order.
this routine only needs to be called once at the start of
the simulation if k1pme,k2pme,k3pme are not dependent on
the boxdimension (gridsize varies).
Written by Arjan van der Vaart, Oct. '97
```

## 18.5 pme_derec.F90

### 18.5.1 subroutine pme_derec

```
this routine calculates the derivative of the reciprocal energy
to the coordinates.
in order to save memory, dQ(k1,k2,k3)/dr is not calculated in
pme_calcq, but calculated here where Conv(theta,Q) is known.
Written by Arjan van der Vaart, Dec. '97
parameters:
*) derec, a wrapped-around real array, will be filled with
.         the derivative dErec/dr
Note that the performance of this algorithm is very high
by demanding that nspline <= K_i, i=1,2,3 and by
shifting the scaled fractional coordinates upme such that
0 <= upme_i <= K_i, i=1,2,3.
derec needs to be recalculated whenever the coordinates or
charges change.
note that the convolution Conv(theta,Q) is stored in QPMEC_
```

For calls made by the subroutine, "pme_derec":

(See [mbspline], page 168.)

## 18.6 pme_derecmc.F90

### 18.6.1 subroutine pme_derecmc

```
author Arjan van der Vaart
date Dec. '97

 this routine calculates the intramolecular contribution to
 the derivative of the reciprocal energy to the coordinates
 or residue ir and subtracts this term from derec for all
 residues in ir's group.

 in order to save memory, dQ(k1,k2,k3)/dr is not calculated in
 pme_calcq, but calculated here where Conv(theta,Q) is known.

 parameters:
  *) ir:    residue number
  *) igr:   group number
  *) n:     offset for groupmembers
```

```
 *) derec: the derivative of the energy to the coordinates (kcal/A)
 *) ch:    charges

Note that the performance of this algorithm is very high
by demanding that nspline <= K_i, i=1,2,3 and by
shifting the scaled fractional coordinates upme such that
0 <= upme_i <= K_i, i=1,2,3.

note that the convolution Conv(theta,Q) is stored in QPMEC_
```

For calls made by the subroutine, "pme_derecmc":

(See [mbspline], page 168.)

## 18.7 pme_direct.F90

### 18.7.1 subroutine pme_direct

```
This routine returns the direct energy, the self energy and
the short-range (classical) Coulomb energy and gradients.

Error Function ( erf(x)= 1-erfc(x)) { erfc(x) is obtained from the
polynomial expression}. The term expar2 is the expotential appearing
in the Integral appearing in the exact expression for erf.
{ erf(x) = (2/sqrt(pi))*Integ(exp(-t*t)dt)}
The polynomial expression for erfc(x) makes use of the same exponental.
However, the derivative is obtained from the actual expression, not the
polynomial.

d(erf(ax)/x) = -erf(ax) + (2/sqrt(pi)) {ax*exp(-ax*ax)}
dx                                x*x
```

For calls made by the subroutine, "pme_direct":

(See [etimer], page 205.)
(See [pbcxyz], page 213.)

## 18.8 pme_directmc.F90

### 18.8.1 subroutine pme_directmc

```
This routine returns the direct energy, the self energy and
the short-range (classical) Coulomb energy and gradient.
it will skip all the intramolecular terms to the gradient.

Error Function ( erf(x)= 1-erfc(x)) { erfc(x) is obtained from the
polynomial expression}. The term expar2 is the expotential appearing
in the Integral appearing in the exact expression for erf.
{ erf(x) = (2/sqrt(pi))*Integ(exp(-t*t)dt)}
The polynomial expression for erfc(x) makes use of the same exponental.
However, the derivative is obtained from the actual expression, not the
polynomial.
d(erf(ax)/x) = -erf(ax) + (2/sqrt(pi)) {ax*exp(-ax*ax)}
dx                                x*x
```

For calls made by the subroutine, "pme_directmc":

(See [etimer], page 205.)
(See [pbcxyz], page 213.)

## 18.9 pme_erec.F90

### 18.9.1 subroutine pme_erec

```
author Arjan van der Vaart
date Dec. 1997

 This routine calculates the reciprocal energy erec and
 the derivative of the reciprocal energy to the coordinates.

 In order to save memory, dQ(k1,k2,k3)/dr is not stored, but
 calculated on the fly. In doing so, Q (Eq.4.6, J.Chem.Phys. 103
 ('95) 8577) can be recalculated very cheaply, which again
 saves memory by not having to store Q (note that the array
 containing Q is overwritten with the value of Conv(theta,Q) in
 pme_recip).
```

```
compiler directives:
*) MEMORY_OVERLAP          if defined, the pme variables will
                           share memory with the eigenvectors /
                           eigenvalues

parameters:
  *) erec:  the reciprocal energy (eV)
  *) derec: the derivative of the reciprocal energy to the
            coordinates (kcal/A)

Note that the performance of this algorithm is very high
by demanding that nspline <= K_i, i=1,2,3 and by
shifting the scaled fractional coordinates upme such that
0 <= upme_i <= K_i, i=1,2,3.
```

For calls made by the subroutine, "pme_erec":

(See [mbspline], page 168.)

### 18.9.2 subroutine mbspline

```
author Arjan van der Vaart
date Oct. 1997

 Massive bspline algorithm
 returns the Cardinal B-spline Mn(w) and
 the derivative of the Cardinal B-spline at point w
 (d Mn(w)/dw) for w=u+int(n-u), u+int(n-u-1), .. , u, u-1,
                   u-2, ... , u-n
 in the x, y and z-direction.
 [note that n=nspline is the order of the B-Spline]

 parameters:
 *) iat is the atom number. This is needed for the scaled fractional
    coordinate defined by
    u(i) = K(i)*a(i)*r(i), i=1,3
      with K integer representing the grid size
           a the reciprocal box vector
           r the (Cartesian) coordinate of the particle
```

```
*) xmn will contain the Cardinal B-spline of order n in for
   the x coordinate;
     xmn(1) = Mn(u+int(n-u))
     xmn(int(u)+int(n-u)+1) = Mn(u-n)
   xmn should be n+int(u)+int(n-u)+1-2 <= 2n-1 elements long,
   this extra length of smn is used as workspace
*) ymn, zmn see xmn
*) dxmn contains the derivatives dMn(w)/dw in the x direction;
     dxmn(1) = dMn(u+int(n-u))/d(u+int(n-u))
     dxmn(int(u)+int(n-u)+1) = dMn(u-n)/d(u-n)
   dxmn should be int(u)+int(n-u)+1 <= n+1 elements long.
   analogous for dymn and dzmn

note that nspline (= n) is the order of the B-spline,
nspline > 2, nspline must be even
```

## 18.10 pme_erecmc.F90

### 18.10.1 subroutine pme_erecmc

```
author Arjan van der Vaart
date Dec. '97

 This routine calculates the intramolecular contribution to
 the derivative of the reciprocal energy to the coordinates
 or residue ir and subtracts this term from derec.

 in order to save memory, dQ(k1,k2,k3)/dr is not stored, but
 calculated on the fly. In doing so, Q (Eq.4.6, J.Chem.Phys. 103
 ('95) 8577) can be recalculated very cheaply, which again
 saves memory by not having to store Q (note that the array
 containing Q is overwritten with the value of Conv(theta,Q) in
 pme_recip).

 the reciprocal energies of the groupmolecules (gerec) are not
 calculated (to save time), but can be easily retrieved by removing
 the 'ccc' comments in this routine (pme_erecmc) and mbsplinemc.
```

```
compiler directives:
*) MEMORY_OVERLAP          if defined, the pme variables will
                           share memory with the eigenvectors /
                           eigenvalues

parameters:
*) ir:    residue number
*) igr:   group number
*) n:     offset for groupmembers
*) derec: the derivative of the energy to the coordinates (kcal/A)
*) ch:    charges

Note that the performance of this algorithm is very high
by demanding that nspline <= K_i, i=1,2,3 and by
shifting the scaled fractional coordinates upme such that
0 <= upme_i <= K_i, i=1,2,3.
```

For calls made by the subroutine, "pme_erecmc":

(See [mbsplinemc], page 170.)

### 18.10.2 subroutine mbsplinemc

```
author Arjan van der Vaart
date Oct. 1997
 Massive bspline algorithm
 returns the derivative of the Cardinal B-spline at point w
 (d Mn(w)/dw) for w=u+int(n-u), u+int(n-u-1), .. , u, u-1,
 .                  u-2, ... , u-n
 in the x, y and z-direction.
 [note that n=nspline is the order of the B-Spline]


 parameters:
 *) iat is the atom number. This is needed for the scaled fractional
    coordinate defined by
    u(i) = K(i)*a(i)*r(i), i=1,3
      with K integer representing the grid size
           a the reciprocal box vector
```

```
            r the (Cartesian) coordinate of the particle
   *) xmn will contain the Cardinal B-spline of order n-1 in for
      the x coordinate;
        xmn(1) = Mn(u+int(n-u))
        xmn(int(u)+int(n-u)+1) = Mn(u-n)
      xmn should be n+int(u)+int(n-u)+1-2 <= 2n-1 elements long.

      note that xmn is used as workspace, it's contents aren't used
      in pme_erecmc

      by removing the 'ccc' comments, xmn will contain the Cardinal
      B-spline of order n (needed to calculate energies in pme_erecmc).
   *) ymn, zmn see xmn
   *) dxmn contains the derivatives dMn(w)/dw in the x direction;
        dxmn(1) = dMn(u+int(n-u))/d(u+int(n-u))
        dxmn(int(u)+int(n-u)+1) = dMn(u-n)/d(u-n)
      dxmn should be int(u)+int(n-u)+1 <= n+1 elements long.
      analogous for dymn and dzmn

   note that nspline (= n) is the order of the B-spline,
   nspline > 2, nspline must be even
```

## 18.11 pme_module.F90

### 18.11.1 module pme_module

```
   PME variables

  To view what this module contains:
```

(See [initialize_pmevars], page 171.)

### 18.11.2 subroutine initialize_pmevars [from module: pme_module]

```
   Initialize PME variables
```

## 18.12 pme_recip.F90

### 18.12.1 subroutine pme_recip

```
author Arjan van der Vaart
date Dec. '97

 This routine calculates the reciprocal enery and derivatives
 by the PME method of J.Chem.Phys. 103 ('95) 8577.

 parameters:
  *) erec            the reciprocal energy
  *) derec           the derivative of erec to the coordinates

 compiler directives:
 *) MEMORY_OVERLAP         if defined, the pme variables will
                           share memory with the eigenvectors /
                           eigenvalues
```

For calls made by the subroutine, "pme_recip":

(See [etimer], page 205.)
(See [pme_calcq], page 163.)
(See [fourn], page 99.)
(See [pme_derec], page 165.)

## 18.13 pme_recipmc.F90

### 18.13.1 subroutine pme_recipmc

```
author Arjan van der Vaart
date Dec. '97

 This routine calculates the reciprocal enery and derivatives
 by the PME method of J.Chem.Phys. 103 ('95) 8577.

 parameters:
  *) erec            the reciprocal energy
  *) derec           the derivative of erec to the coordinates

 compiler directives:
```

```
*) MEMORY_OVERLAP          if defined, the pme variables will
                           share memory with the eigenvectors /
                           eigenvalues
```

For calls made by the subroutine, "pme_recipmc":

(See [etimer], page 205.)
(See [pme_recip], page 172.)
(See [pme_calcqmc], page 164.)
(See [fourn], page 99.)
(See [pme_derecmc], page 165.)

## 18.14 pme_setup.F90

### 18.14.1 subroutine pme_setup

```
Setup for PME
```

## 18.15 pmeDriver.F90

### 18.15.1 subroutine pmedriver

```
Driver function for PME Calculations
```

For calls made by the subroutine, "pmedriver":

(See [setup], page 183.)
(See [setbox], page 119.)
(See [push], page 188.)
(See [pme_setup], page 173.)
(See [pme_calcb], page 163.)
(See [pme_calctheta], page 164.)
(See [gensub], page 7.)
(See [energy], page 90.)
(See [atmchg], page 185.)
(See [gcartmc], page 114.)
(See [gcart], page 206.)
(See [pme_directmc], page 167.)
(See [pme_direct], page 166.)
(See [pme_recip], page 172.)
(See [pme_recipmc], page 172.)

## 18.16 pmeRdKeys.F90

### 18.16.1 subroutine pmerdkeys

```
Read keywords associated with PME
```

For calls made by the subroutine, "pmerdkeys":

(See [rdnum], page 71.)
(See [rdinum], page 71.)

# 19 SCF

## 19.1 doscf.F90

### 19.1.1 subroutine doscf

```
Driver routine to carry out the iterative self-consistent field
calculation.  Energies are returned in eV.

eelect1 = electronic energy
ecore1  = core-core repulsions
etot1   = eelect1 + ecore1
```

For calls made by the subroutine, "doscf":

(See [rdnum], page 71.)
(See [rdnum], page 71.)
(See [etimer], page 205.)
(See [initp], page 177.)
(See [ijmake], page 209.)
(See [sproc2], page 19.)
(See [setprtvec], page 215.)
(See [focku], page 177.)
(See [fock], page 176.)
(See [escf], page 176.)
(See [diffp_prt_stat], page 203.)
(See [diffp_prt_hist], page 204.)
(See [fmix], page 176.)
(See [fshift], page 7.)
(See [mosubfdm], page 92.)
(See [mosubfdmx], page 93.)
(See [nmrdmx], page 140.)
(See [mosub], page 91.)
(See [diffp_stat], page 203.)
(See [pmixa], page 178.)
(See [pmixb], page 178.)
(See [pmix], page 178.)
(See [wrtdmx], page 80.)

## 19.2 escf.F90

### 19.2.1 subroutine escf

```
Determines the electronic energy for the current scf iteration.
```

For calls made by the subroutine, "escf":

(See [etimer], page 205.)

## 19.3 fmix.F90

### 19.3.1 subroutine fmix

```
Fock matrix mixing scheme to accelerate scf convergence.

Call after fock matrix is constructed, after energy evaluation,
before call to mosub.
```

For calls made by the subroutine, "fmix":

(See [decalc], page 176.)

### 19.3.2 subroutine decalc

```
Fock matrix mixing scheme to accelerate scf convergence.
```

## 19.4 fock.F90

### 19.4.1 subroutine fock

```
BUG FIX: 7/26/99 -- S. Dixon.  When sparkles appear after the first
atom, then the ijbond pointer can misidentify entries in the bonded
pairlist.  this was fixed by making sure that atoms i and j are both
valid candidates for the pairlist before using the ijbond pointer.
```

For calls made by the subroutine, "fock":

(See [pbfock], page 161.)
(See [oned], page 38.)
(See [pbcxyz], page 213.)

(See [diat], page 28.)
(See [diam1], page 138.)
(See [diam2], page 138.)
see system command "mpi_allreduce"

## 19.5 fock_uhf.F90

### 19.5.1 subroutine focku

```
author Bing Wang
author Steve Dixon

 Assembles the Fock matrix for Unrestricted calculations
```

For calls made by the subroutine, "focku":

(See [pbfock], page 161.)
(See [onedu], page 38.)
(See [pbcxyz], page 213.)
(See [diat], page 28.)
see system command "mpi_allreduce"

## 19.6 initp.F90

### 19.6.1 subroutine initp

```
author Steve Dixon

 Initializes global density matrix.  This routine should be called
 after each pairlist update and global pointer update.

 If init=0 then the density matrix will be fully initialized to
 diagonal form.  Otherwise, the old density matrix will be used
 to build the new one.

 Note that this routine uses old pairlist information stored in
 common /pairij/ to extract diatomic blocks from the old density
 matrix.  common /pairij/ reverts back to its normal purpose with
 the next call to ijmake.
```

For calls made by the subroutine, "initp":

(See [rddmx], page 62.)

## 19.7 pmix.F90

### 19.7.1 subroutine pmix

```
author Steve Dixon
reference P. Badziag AND F. Solms, j. Comput. Chem., 12, 233-236, 1988

 Uses the Improved Iteration Scheme (IIS) to accelerate convergence
 on the density matrix.  For details see reference.

 The iteration counter corresponds to the scf cycle, i.e., iter=1
 for the first iteration, iter=2 for the second, etc.  This differs
 from Badziag and Solms convention where iter=0 corresponds to the
 first cycle.
```

## 19.8 pmixa.F90

### 19.8.1 subroutine pmixa

```
reference P. Badziag AND F. Solms, j. Comput. Chem., 12, 233-236, 1988

 Uses the Improved Iteration Scheme (IIS) to accelerate convergence
 on the density matrix.  For details see reference.

 The iteration counter corresponds to the scf cycle, i.e., iter=1
 for the first iteration, iter=2 for the second, etc.  This differs
 from Badziag and Solms convention where iter=0 corresponds to the
 first cycle.
```

## 19.9 pmixb.F90

### 19.9.1 subroutine pmixb

```
reference P. Badziag AND F. Solms, j. Comput. Chem., 12, 233-236, 1988

 Uses the Improved Iteration Scheme (IIS) to accelerate convergence
 on the density matrix.  For details see reference.

 The iteration counter corresponds to the scf cycle, i.e., iter=1
 for the first iteration, iter=2 for the second, etc.  This differs
 from Badziag and Solms convention where iter=0 corresponds to the
```

```
first cycle.
```

## 19.10 scf_module.F90

### 19.10.1 module scf_module

```
This module cantains variables for the SCF procedure.
```

```
To view what this module contains:
```
(See [initialize_scfvars], page 179.)

### 19.10.2 subroutine initialize_scfvars [from module: scf_module]

```
Initialize scf variables.
```

## 19.11 scfRdKeys.F90

### 19.11.1 subroutine scfrdkeys

```
date August 2005

Reads SCF keywords:
 - ECRIT
 - DCRIT
 - 1SCF
 - DESCF
 - DPSCF
 - FULLSCF
 - MAXIT
 - UHF
 - IMULT
 - DOUBLE
```

For calls made by the subroutine, "scfrdkeys":

(See [rdnum], page 71.)
(See [rdinum], page 71.)

# 20 SETUP

## 20.1 assign.F90

### 20.1.1 subroutine assign

```
Assign Hamiltonian values to internal arrays.
```

## 20.2 atmlist.F90

### 20.2.1 subroutine atmlst

```
Extracts atom numbers from the character string card and stores them
in ilist.  nstore is returned with the number of non-dummy,
non-sparkle atoms stored.  If more than maxval values (including
dummies and sparkles) are present, then ierror is set to 1.
ierror is also flagged if the list of integers in card is not
specified correctly.
```

For calls made by the subroutine, "atmlst":

(See [wdjoin], page 216.)
(See [rdnum], page 71.)

## 20.3 command.F90

### 20.3.1 subroutine initializefname

```
This subroutine initializes the arrays fname, iunit, fstat, and ext to
the default values.  Subsequently these may be changed based upon what
readCommandLine sees.

ASSIGN UNIT NUMBERS, FILENAMES, AND STATUS FOR FILES THAT MAY BE OPENED.
note that fname is the basename for the filename for parallel I/O
ext is the extension used for parallel I/O
```

### 20.3.2 subroutine readcommandline

```
This subroutine reads in the command line arguments and from that sets
file names or alternately leaves things as they were.
Functionality here was previously in the updatefname subroutine
```

For calls made by the subroutine, "readcommandline":

(See [initializefname], page 180.)
see system command "getarg"

## 20.4 glbpnt.F90

### 20.4.1 subroutine glbpnt

```
Sets up pointers to access diatomic and diagonal blocks of
global matrices.  Should be called after subsystems are defined
and the bonded atom pairlist is created.

iimat(k) --> start of diagonal block in hdiag, pdiag, and fdiag
             for atom k (only needs to be determined once).

ijmat(i) --> start of diatomic blocks in hdiat, pdiat, and
             fdiat for bonded atom pairlist entry i.

ijrep(i) --> start of diatomic repulsions in eerep for full
             pairlist entry i (done only once, and only if
             it's not a direct calculation).

Error flag is set to 1 if storage limits for global matrices
would be exceeded.
```

## 20.5 initializeVars.F90

### 20.5.1 subroutine initializeglobalvariables

```
Initializes global variables
```

## 20.6 intpr.F90

### 20.6.1 subroutine intpr

```
author Arjan van der Vaart
date May 1998

 Routines to handle external parameters.

 input format:

   DATA NAME(...) / ... /
   NAME(...) / .... /
   NAME(...) = ....

 Comment is preceded by a 'C' in the first column.

 The external file does not have to contain all the
 parameters, so we need to initialize them (done by
 the NOTNAMED=.. keyword).
```

For calls made by the subroutine, "intpr":

(See [upcase], page 216.)
(See [getvalue], page 182.)
(See [assign], page 180.)

### 20.6.2 subroutine getvalue

```
author Arjan van der Vaart
date May 1998

 Gets value from external parameter file.
```

For calls made by the subroutine, "getvalue":

(See [whatis7], page 216.)
(See [getinum], page 72.)
(See [whatis1], page 216.)

## 20.7 setdef.F90

### 20.7.1 subroutine setdef

```
Assigns default settings for program parameters that have not been
specified in the keywords.
```

## 20.8 setopt.F90

### 20.8.1 subroutine setopt

```
Sets up the correspondence between the optimizable geometric
parameters and the z-matrix or xyz array.  This fills in the
information in common block /optmze/.
```

## 20.9 setup.F90

### 20.9.1 subroutine setup

```
Does all the preliminary setup to prepare the program for the
first energy evaluation.
```

For calls made by the subroutine, "setup":

(See [setdef], page 183.)
(See [intpr], page 182.)
(See [calcb], page 51.)

## 20.10 stripkeywd.F90

### 20.10.1 subroutine stripkeywd

```
all multiple whitespace in keywd will be converted
to one space, lstrip is the length of the stripped keyword
lkeywd is the length of keywd
add one space at the end
```

### 20.10.2 subroutine stripkeymc

```
strip dr, dangle, seed, betapme, dbox and mc out of keywords
(since these change during mc-run and need to be filled out by
wrtdiv)
```

For calls made by the subroutine, "stripkeymc":

(See [stripkeywd], page 183.)

# 21 TOOLS

## 21.1 atmchg.F90

### 21.1.1 subroutine atmchg

```
for non-MC simulations, CM1, CM2 and Mulliken charges
are calculated.
atchg is the charge array used in the PME calculations.
so it depends on the value of cm1 and cm2 if atchg will be filled
with Mulliken charges or CM1 or CM2 charges.
The following table shows which charges will be assigned to
which array (* is wildcard, valid for both T(rue) and F(false),
- is not calculated)
no.|mcsim | pme | cm1 | cm2  | Mulliken | CM1    | CM2
1   T       *     F     F      atchg        -       -
2   T       *     T     F        -        atchg     -
3   T       *     F     T        -          -      atchg
4   F       F     *     *      atchg      atchg2   atchg3
5   F       T     F     F      atchg      atchg2   atchg3
6   F       T     T     F      atchg2     atchg    atchg3
7   F       T     F     T      atchg3     atchg2   atchg
for now atchg refers to Mulliken charges, atchg2 to CM1 charges,
atchg3 to CM2 charges (swap this if needed at the end of the
routine)
.  there are two efficient ways to calculate the CM1 charges:
.  the faster one is a loop from iat=2,natoms with
.  an inner loop from jat=1,iat-1, calculate and store the
.  B(iat,jat) = B(jat, iat) coefficients and then
.  do another iat=2,natoms - jat=1,iat-1 loop for
.  the normalization using the stored B(iat,jat) = B(jat,iat)
.  coefficients.
.  The second slower one loops from iat=1,natoms in which the
.  inner loop jat=1,natoms calculates the B(iat,jat) coefficients.
.  After this jat-loop a partial normalization is performed.
.  Method one is faster because it only performs the
.  density matrix summation needed for the B(iat,jat) calculation
.  N^2/2 times (method 2 does this N^2 times),
.  but requires a lot more memory: N^2/2 elements to
```

```
.  store the B(iat,jat) coefficients. Method 2 only needs to
.  store the N intermediate B(iat,jat) coefficients within the
.  iat-loop (N is the number of atoms).
.  Here memory considerations lead me to implement the second
.  (slower) method.
.  CM2 charges can be cheaply obtained in the same sweep.
```

For calls made by the subroutine, "atmchg":

(See [ijfind], page 210.)
(See [getcm1a], page 186.)
(See [getcm2a], page 186.)
(See [calccm1a], page 186.)
(See [getcm1p], page 186.)
(See [getcm2p], page 186.)
(See [calccm1p], page 186.)

### 21.1.2 subroutine getcm1a

```
subroutine for CM1/AM1 charge model
```

### 21.1.3 subroutine calccm1a

```
subroutine for CM1/AM1 charge model
```

### 21.1.4 subroutine getcm1p

```
subroutine for CM1/PM3 charge model
```

### 21.1.5 subroutine calccm1p

```
subroutine for CM1/PM3 charge model
```

### 21.1.6 subroutine getcm2a

```
subroutine for CM2/AM1 charge model
```

### 21.1.7 subroutine getcm2p

```
subroutine for CM2/AM1 charge model
```

## 21.2 bcheck.F90

### 21.2.1 subroutine bcheck

```
DIAGNOSITIC TOOL FOR CHECKING BOND ORDERS BETWEEN PAIRS OF
ATOMS.  LOOKS FOR DATA IN THE FILE bcheck.dat.
```

For calls made by the subroutine, "bcheck":

(See [opnfil], page 212.)
(See [ijfind], page 210.)

## 21.3 dipole.F90

### 21.3.1 subroutine dipole

```
Dipole.F calculates the dipole of a molecule.  Th variables required by
it are all contained in divcon_old.h, thus none are explicitly passed.
Basically, it calculates the center of mass, moves the atom so the origin
residues at that center, then calculates the point charge contribution of
each atom to the dipole.  It then calculates the contribution of
hybridization to the the dipole, and finally includes all these values in
divcon output. For more information, please see McGlynn, et al.,
"Introduction to Applied Quantum Chemistry," 1972.
```

## 21.4 error_module.F90

### 21.4.1 module error_module

```
error parameters

To view what this module contains:
```

(See [initialize_errorvars], page 187.)

### 21.4.2 subroutine initialize_errorvars [from module: error_module]

```
Initialization error variables
```

## 21.5 homolumo.F90

### 21.5.1 subroutine homolumo

```
author  Arjan van der Vaart
date 11/18/99 parallel implementation
reference none

Parallel implementation. This code makes sure
that the homo-lumo gaps are printed in the right
order
```

For calls made by the subroutine, "homolumo":

see system command "mpi_send"
see system command "mpi_recv"

## 21.6 mkdos.F90

### 21.6.1 subroutine mkdos

```
author Arjan van der Vaart

reference Lee and Yang Phys. Rev. Lett. 1998, 90(22), 5011-5014.

 Generates Density of States
 this routine is not parallellized, since it
 only takes a minor amount of time
```

For calls made by the subroutine, "mkdos":

see system command "mpi_allreduce"
(See [gendos], page 188.)

### 21.6.2 subroutine gendos

```
 Generate Density of States
```

## 21.7 push.F90

### 21.7.1 subroutine push

```
 pushes the residues, as defined in the push groups, apart
```

## 21.8 pwdecomp.F90

### 21.8.1 subroutine pwdecomp

```
author Arjan van der Vaart
date November 2001
reference Raha, van der Vaart, Riley, Peters, Westerhoff, Kim,
 Merz J. Am. Chem. Soc. 2005, 127(18), 6583-6594.

This routine performs a pairwise energy decomposition
Etotal = Sum_A ( E_A + Sum_B<A (E_AB' + E_AB + Ecore_AB) )

This routine is directly modeled from Fock.F, so some extra
statements are in this code that will never be reached (for
example, the "if (iter1)" statements (iter1 is always false
```

```
when this routine is called). However, this should not impact
the performance. This routine has also not been parallelized.
Since this routine will always be called at the end of an SCF
cycle (when a converged density matrix has been obtained), some
arrays can be overwritten (to avoid the introduction of more
memory consuming arrays).

array    quantity
pijold   E_A and E_AB
fijold   E_AB'
ff       Ecore_AB
ww       rij
```

For calls made by the subroutine, "pwdecomp":

(See [zerof], page 189.)
(See [oned], page 38.)
(See [pbcxyz], page 213.)
(See [diat], page 28.)
(See [opnfil], page 212.)

### 21.8.2 subroutine zerof

```
author Arjan van der Vaart
date November 2001
reference Raha, van der Vaart, Riley, Peters, Westerhoff, Kim,
  Merz J. Am. Chem. Soc. 2005, 127(18), 6583-6594.
 Zeroes out the array FDIAG in pwdecomp.F90
```

### 21.8.3 function indw

```
author Arjan van der Vaart
date November 2001
reference Raha, van der Vaart, Riley, Peters, Westerhoff, Kim,
  Merz J. Am. Chem. Soc. 2005, 127(18), 6583-6594.
     returns the index of the triangular interaction matrix
     .   j->
     . i 1
     . | 2 3
     . v 4 5 6
     .   7 8 9 10 ...
```

## 21.9 rderror.F90

### 21.9.1 subroutine rderror

```
error
  heat=float
  ip=float
  dipole=float
  association=float
     file
     file
  end_association
  etotdiff=float
     file
     file
  end_etotdiff
end_error
```

For calls made by the subroutine, "rderror":

(See [initialize_errorvars], page 187.)
(See [upcase], page 216.)
(See [wdjoin], page 216.)
(See [rdnum], page 71.)

## 21.10 rotate.F90

### 21.10.1 subroutine rotate

```
author Ed Brothers
date July 1998

 The purpose of this segment is to calculate the energetic barrier
to rotation
 of a molecule.
```

For calls made by the subroutine, "rotate":

(See [atmlst], page 180.)
(See [zmake], page 217.)
(See [wrtint], page 81.)
(See [edriver], page 89.)

## 21.11 thermo.F90

### 21.11.1 subroutine dothermo

```
author Ed Brothers
date 01/22/03
```

```
Calculates thermodynamic quantities(ZPE,rotational energy, translational
energy, vibrational energy/enthalpy, internal energy/enthalpy) from
vibrational analyses.
```

For calls made by the subroutine, "dothermo":

(See [bndang], page 199.)

## 21.12 tools_module.F90

### 21.12.1 module tools_module

```
This module cantains variables for the tools directory
```

## 21.13 toolsDriver.F90

### 21.13.1 subroutine toolsdriver

```
The driver function for tools.
```

For calls made by the subroutine, "toolsdriver":

(See [setopt], page 183.)
(See [gcartmc], page 114.)
(See [gcart], page 206.)
(See [dograd], page 205.)
(See [setprtvec], page 215.)
(See [wrtvec], page 82.)
(See [wrtvecsubf], page 82.)
(See [wrtint], page 81.)
(See [wrtxyz], page 83.)
(See [zmake], page 217.)
(See [wrtqma], page 81.)
(See [psm], page 132.)

## 21.14 toolsRdKeys.F90

### 21.14.1 subroutine toolsrdkeys

```
Read tools Related Keywords
    - geocalc
    - error
    - qmalign
    - trajectory
    - homolumo
    - push
    - dos
    - rotate
    - thermo
    - dipole
    - ip
    - pwd
    - pwd_atom
    - pwd_residue
    - zmake
```

For calls made by the subroutine, "toolsrdkeys":

(See [rdnum], page 71.)

## 21.15 toolsRdTail.F90

### 21.15.1 subroutine toolsrdtail

```
Read tools Related Tail
   - dos
   - correct
```

For calls made by the subroutine, "toolsrdtail":

(See [rddos], page 63.)
(See [rderror], page 190.)

# 22 TS

## 22.1 geots.F90

### 22.1.1 subroutine geots

```
author Dimas Suarez
date November 1999
reference none

TS Geometry optimization driver
```

For calls made by the subroutine, "geots":

(See [rdnum], page 71.)
(See [setopt], page 183.)
(See [tsrfo], page 196.)
(See [tsqna], page 194.)
(See [wrtrst], page 82.)
(See [wrtpdb], page 81.)
(See [wrtdmx], page 80.)
(See [freq], page 21.)

## 22.2 prjfc.F90

### 22.2.1 subroutine prjfc

```
author Dimas Suarez
date November 1999
reference W.H.Miller,N.C.Handy,J.E. Adams, J. Chem. Phys. 1980, 72, 99-112.

 This routine projects R-T modes from a Cartesian Hessian matrix in order to
 perform RFO optimizations using Cartesian coordinates.
```

For calls made by the subroutine, "prjfc":

(See [dgefa], page 101.)
(See [dgedi], page 100.)

## 22.3 septs.F90

### 22.3.1 subroutine septs

```
author Dimas Suarez
date November, 1999
reference none
 This routine performs an energy and gradient calculation on the geometry
 currently stored in the array DCOORD(BFGS COMMON BLOCK).
```

For calls made by the subroutine, "septs":

(See [setbox], page 119.)
(See [gensub], page 7.)
(See [mvcrd], page 212.)
(See [energy], page 90.)
(See [gcart], page 206.)
(See [dograd], page 205.)

## 22.4 ts_module.F90

### 22.4.1 module ts_module

```
 This module cantains variables concerning ts calculations.
```

## 22.5 tsqna.F90

### 22.5.1 subroutine tsqna

```
author D. Suarez
date December 1999

 This routine drives a TS-search using the Quasi-Newton
 Algorithm porposed in the following
 following reference : Theor. Chim. Acta vol 82, 189-205, 1992

 For details about the method see the above ref.

 This routine assumes that there exists a septs(x,g,e)
 subroutine that computes the gradient and energy
 for a given coordinate vector x. any details
 about the energy function, coordinate system,
```

```
dual-method stuff, and so on, will be managed
by septs not by ts

Similarly, a hsep subroutine is assumed to
calculate the hessian matrix for a given
geometry x
```

For calls made by the subroutine, "tsqna":

(See [etimer], page 205.)
(See [septs], page 194.)
(See [hsep], page 23.)
(See [prjfc], page 193.)
(See [wrtrst], page 82.)
(See [wrtpdb], page 81.)
(See [wrtdmx], page 80.)
(See [wrthss], page 81.)
(See [jacobi], page 197.)
(See [sort2], page 197.)
(See [stepnr], page 195.)

### 22.5.2 subroutine stepnr

```
This is an auxiliary function of the TSQNA
subroutines. It computes the norm of the
Newton-Raphson Step and the Step vector itself.
I/O runs through the RFO COMMON BLOCK
```

### 22.5.3 function vlambda

```
This auxiliary Function is employed in the determiantion
of the RFO parameter Lambda.

This is an auxiliary function of the TSQNA
subroutines. It computes the norm of the
Newton-Raphson Step and the Step vector itself.
I/O runs through the RFO COMMON BLOCK
```

### 22.5.4 function rootv

```
Root finding: bissection method
for the Vlambda function
```

## 22.6 tsRdKeys.F90

### 22.6.1 subroutine tsrdkeys

```
Read ts Keywords
   - opt=ts
   - opt=tsprfo
   - opt=tsqna
   - opt=tsnr
   - nomodefollow
```

## 22.7 tsrfo.F90

### 22.7.1 subroutine tsrfo

```
author D. Suarez
date September-December 1999

 Following the Baker's Recipe JCC vol 7 385 (1986)
 this routine drives a TS-search using the P-RFO
 method. For details about the method, see the cited
 reference.

 This routine assumes that there exists a septs(x,g,e)
 subroutine that computes the gradient and energy
 for a given coordinate vector x. Any details
 about the energy function, coordinate system,
 dual-method stuff, and so on, will be managed
 by septs not by ts

 Similarly, a hsep subroutine is assumed to
 calculate the hessian matrix for a given
```

```
geometry x
```

For calls made by the subroutine, "tsrfo":

(See [etimer], page 205.)
(See [septs], page 194.)
(See [hsep], page 23.)
(See [prjfc], page 193.)
(See [wrtrst], page 82.)
(See [wrtpdb], page 81.)
(See [wrtdmx], page 80.)
(See [wrthss], page 81.)
(See [jacobi], page 197.)
(See [sort2], page 197.)

### 22.7.2 subroutine jacobi

```
Computes all eigenvalues and eigenvectors of a real symmetric
matrix A, wich is of size N by N, stored in a physycal NP by NP
array. On output, elements of A above the diagonal are destroyed.
D returns the eigenvalues of A in its first N elements. V is a mat
with the same logical and physical dimensions as A whose columns
contain, on output, the normalized eigenvectors of A. NROT returns
the number of Jacobi rotations which were required.
```

### 22.7.3 subroutine sort2

```
Sorts an array ra of length n into ascending numerical order
using the heapsort algorithm, while making the corresponding
rearrangement of the array ib.
```

### 22.7.4 function vmax

### 22.7.5 function vrms

# 23 UTIL

## 23.1 av.F90

### 23.1.1 subroutine av

```
Calculates partial sums (sx), partial fluctuations (fx) and
absolute maximum so far (hx)
```

## 23.2 binloc_index.F90

### 23.2.1 subroutine binloc_index

```
Uses a binary search to locate the greatest index (index_g) such that

list(index_g - 1) < list(index_g) <= itarget
or
list(index_g - 1) <= list(index_g) < itarget

Where
 list    = sorted (increasing) list of integers to be searched
 istart  = starting index of search in list
 istop   = stopping index of search in list
 itarget = target value
 index_g = returned index of the list; if ierror /= 0, then
 returned value of index_g will be -999999999
 ierror  =  2   if istart > istop;
         =  1   if itarget > list(istop)
         = -1   if itarget < list(istart)
         =  0   otherwise
```

## 23.3 bleng.F90

### 23.3.1 subroutine bleng

```
Computes the bond length in angstroms between atoms i and ia.
```

## 23.4 bndang.F90

### 23.4.1 subroutine bndang

```
Computes the angle in radians formed by atoms i-ia-ib.
Returned in angle.
```

## 23.5 bpair.F90

### 23.5.1 subroutine bpair

```
Bonded atom pairlist generation routine.  Two atoms are stored
in the bonded atom pairlist if they ever appear together in the
same subsystem and they are separated by less than the user
parameter cutbond.

For i=2,3,...,natoms, the atoms that "bond" with i and which have
lower atom numbers than i are:

ipair(k), for k=ip1(i),ip1(i)+1,...,ip1(i+1)-1

If the maximum allowed number of pairs is exceeded, ierror is
returned with the atom number i at which the maximum was reached.
this routine should be called each time a new set of subsystems
is defined.

Note that the common block /pairij/ is used temporarily here to
store the old pairlist and pointers to diatomic blocks in the
old density matrix.  This information will be used in initp to
construct a new density matrix from the old one.
```

For calls made by the subroutine, "bpair":

(See [pbcxyz], page 213.)

## 23.6 bsort.F90

### 23.6.1 subroutine bsort

```
Heap sort routine for list of integer point numbers ia.
The integer values ib are carried along.  Sorts in order of
increasing value.

heap sort algorithm instead of bubble sort:
bubble sort = N*N, heap sort = N Log2(N)
```

### 23.6.2 subroutine bsort1

```
Heap sort routine for list of integer point numbers ia.
sorts in increasing order.

heap sort algorithm instead of bubble sort:
bubble sort = N*N, heap sort = N Log2(N)
```

### 23.6.3 subroutine busort

```
Heap sort routine for list of integer point numbers ia.
Sorts in order of increasing value, non-unique values
are removed from the list. Entries for which nia is .false.
are removed.

heap sort algorithm instead of bubble sort:
bubble sort = N*N, heap sort = N Log2(N)
```

### 23.6.4 subroutine busort1

```
Heap sort routine for list of integer point numbers ia.
Sorts in order of increasing value, non-unique values
are removed from the list.

heap sort algorithm instead of bubble sort:
bubble sort = N*N, heap sort = N Log2(N)
```

### 23.6.5 subroutine bovsort1

```
Returns list of unique and non-overlapping integers
used in rddmx.F to significantly decrease copy time
of density matrix later on

note that ia(i) contains the starting index,
          ia(i+1) the ending index, i is odd

so array 5 7 1 3 8 12 13 13 will be overwritten by 1 3 5 13

ierror will be set to one in case there were overlapping /
nonuniqe numbers, else ierror = 0
AvdV
```

### 23.6.6 subroutine rsort

```
Heap sort routine for list of floating point numbers ra.
The integer values ia are carried along.  Sorts in order of
increasing value.
```

### 23.6.7 subroutine rsort2

```
Heap sort routine for list of floating point numbers ra. the
integer values ia and ja are carried along. sorts in order of
increasing value.

iabs = 0  : sort purely without considering absolute vaules of ra
iabs = 1  : sort in order of increasing absolute values of ra
```

### 23.6.8 subroutine bsort2

```
Heap sort routine for list of integer point numbers ia.
The real values xb are carried along.  Sorts in order of
increasing value. Note that xb has two fields per ia field.

heap sort algorithm instead of bubble sort:
```

```
bubble sort = N*N, heap sort = N Log2(N)
```

### 23.6.9 subroutine bsort3

```
HEAP SORT ROUTINE FOR LIST OF INTEGERS KA. THE INTEGER VALUES IA
AND JA ARE CARRIED ALONG. SORTS IN ORDER OF INCREASING VALUE.

IABS = 0  : SORT PURELY WITHOUT CONSIDERING ABSOLUTE VAULES OF KA
IABS = 1  : SORT IN ORDER OF INCREASING ABSOLUTE VALUES OF KA
```

## 23.7 bspline.F90

## 23.8 calcgeo.F90

### 23.8.1 subroutine calcgeo

```
author Ed Brothers
date April 1999

 The purpose of this routine is to calculate geometric properties
 of interest such as distances, angles, and dihedrals.  If the error
 function is turned on the error for geometry is calculated here too.
```

For calls made by the subroutine, "calcgeo":

(See [wdjoin], page 216.)
(See [rdnum], page 71.)
(See [iwhole], page 72.)
(See [bndang], page 199.)
(See [dihedr], page 204.)

## 23.9 chkovlp.F90

### 23.9.1 subroutine chkovlp

```
Error handling for overlap.
```

## 23.10 diffp_stat.F90

### 23.10.1 subroutine diffp_stat

```
Analyzing statistics of DP (difference in DMX entries compared to
the previous ones). The interval [0, abs(DPMAX)] is equally
divided into 10 subintervals, and the number of DP values in each
subinterval is stored in the array ICOUNT. Also, the routine
stores some (NUM_TOP) large values of DP along with their
corresponding atom pairs into the arrays, DP_TOP, IATM_TOP, and
JATM_TOP. Here NUM_TOP is calculated by the formula

num_top = min(min(max(icount(10),10),500),num_big)

where NUM_BIG is the number of top half (50%) of absolute DMX
elements (DP values). In other words, stored are either top 10%
or at least top 10 (within top half) of large absoulte DP values
but at most 500 of them.

Also, the frequency of (iatm, jatm) pairs are accumulated/stored
in a two-dimensional array IHIST:
IHIST(1) = iatm
IHIST(2) = jatm
IHIST(3) = frequency of (iatm, jatm) pair
NUM_HIST = the total number of different pairs
MAX_HIST = (dynamically changing) size of the array IHIST

WWWWWWWWWWW        WARNING     WARNING     WARNING        WWWWWWWWWW

Since IHIST is a pointer dummy argument, this routine should be
interfaced in a calling routine, DOSCF, so that any changes in
the varible lists must be accounted in there too.
```

For calls made by the subroutine, "diffp_stat":

(See [rsort2], page 201.)

### 23.10.2 subroutine diffp_prt_stat

```
Data (DP values, i.e., difference in DMX entries compared to the
previous ones) are stored into arrays at the routine DIFFP_STAT.
This routine just reports (screen output) the statistics of them.
Atom symbols and residues corresponding to the atom pairs
(iatom_top, jatm_top) are figured out in this purpose too.
```

For calls made by the subroutine, "diffp_prt_stat":

(See [binloc_index], page 198.)

### 23.10.3 subroutine diffp_prt_hist

```
This routine just reports (screen output) the frequency statistics
stored in the array, IHIST, after sorting it by frequencies.
Frequencies means the number of each atom pairs appeared in the
reported statistics of DP through the routine, DIFFP_PRT_STAT.
Frequencies of atom pairs, (IHIST(1,:), IHIST(2,:)), are stored
in IHIST(3,:). Atom symbols and residues corresponding to that
atom pairs are figured out in this purpose too.
```

For calls made by the subroutine, "diffp_prt_hist":

(See [bsort3], page 202.)
(See [binloc_index], page 198.)

## 23.11 dihedr.F90

### 23.11.1 subroutine dihedr

```
Determines the I-IA-IB-IC dihedral angle in radians.  The
angle DIH is positive if IC is located clockwise from I when
viewing from IA through IB.
```

## 23.12 dograd.F90

### 23.12.1 subroutine dograd

```
Returns gradient and gradient norm in either internal or
cartesian coordinates.  Internal coordinates are selected
if the keywords 'XYZSPACE' or 'CARTESIAN' are not present.
energy units are kcal.  The error code corresponds to that
of subroutine g2int and getxyz.
```

For calls made by the subroutine, "dograd":

(See [g2int], page 205.)

## 23.13 etimer.F90

### 23.13.1 subroutine etimer

```
Returns the elapsed cpu time in seconds
```

### 23.13.2 real*8 function myclock

For calls made by the function, "myclock":

see system command "date_and_time"

## 23.14 g2int.F90

### 23.14.1 subroutine g2int

```
Converts cartesian gradient GXYZ to gradient in internal
coordinates.  GINT is returned in units of energy/angstrom
or energy/radian.  Gradient is transformed by means of a
finite-difference jacobian b,


/                                       \
|  dX(1)     dX(2)              dX(3N)   |
|  dQ(1)     dQ(1)              dQ(1)    |
|                                        |
|                                        |
```

```
|  dX(1)     dX(2)               dX(3N)  |
B  =    |  dQ(2)     dQ(2)               dQ(2)   |
|                                        |
|    .         .                  .     |
|    .         .                  .     |
|    .         .                  .     |
|                                        |
|  dX(1)     dX(2)               dX(3N)  |
| dQ(3N-6)  dQ(3N-6)            dQ(3N-6) |
\                                       /
```

```
Here X represents cartesian coordinates, and Q represents
internal coordinates.  Internal coordinates are given by
the matrix-vector product:  GINT =  B*GXYZ.  The array WXYZ
is work space to store the original cartesian coordinates.
IERROR is returned with the error code of subroutine GETXYZ.

Note that gint is returned with the full internal coordinate
gradient, without regard for which parameters have been flagged
for optimization.  This fact should be noted when doing a
geometry optimization.
```

For calls made by the subroutine, "g2int":

(See [getxyz], page 208.)

## 23.15 gcart.F90

### 23.15.1 subroutine gcart

```
Subroutine to compute cartesian energy gradient using variational
finite-difference derivatives.  Returns gradient in kcal/angstrom.
see subroutine FOCK for more extensive comments on energy terms.
```

For calls made by the subroutine, "gcart":

(See [etimer], page 205.)
(See [pbcxyz], page 213.)
(See [diat], page 28.)
(See [bond_e], page 134.)

(See [angle_e], page 134.)
(See [tor_e], page 134.)
(See [dihedr], page 204.)

## 23.16 getcrd.F90

### 23.16.1 subroutine getcrd

```
Changes the coordinates by STEP*DIRECT1.  If the operation is carried
out on internal coordinates, then the cartesian coordinates are
recalculated.  Since the cartesian and/or internal coordinates are
updated through the globel variables in the module. The array
Coord0 should not be xyz or zmat.
```

For calls made by the subroutine, "getcrd":

(See [getxyz], page 208.)

## 23.17 getpar.F90

### 23.17.1 subroutine getpar

```
Assigns semiempirical parameters based on which hamiltonian the
user has selected.
```

For calls made by the subroutine, "getpar":

(See [initialize_mndovars], page 24.)
(See [initialize_am1vars], page 24.)
(See [initialize_pm3vars], page 25.)
(See [aijm], page 27.)
(See [inighd], page 31.)
(See [ddpo], page 28.)

## 23.18 getpar2.F90

### 23.18.1 subroutine getpar2

```
This routine will only be called when PARAM_SPLIT_IS_ON
is defined, however, the following #ifdef is needed since
the paramx parameters are only defined when PARAM_SPLIT_IS_ON
is defined (to prevent compile errors)
```

For calls made by the subroutine, "getpar2":

(See [getpar], page 207.)

## 23.19 getxyz.F90

### 23.19.1 subroutine getxyz

```
author: S. Dixon

 computes cartesian coordinates from internal coordinates.

 INPUT:
 NATOMS1 = Number of atoms.
 ZMAT1   = Bond, angle, dihedral angle values.  Units should be
           angstroms, radians, radians, respectively.
 IZMAT1  = Reference atoms defining zmat1.
 IFIRST  = First atom for which cartesian coordinates are needed.
           this is normally equal to 1, but when computing jacobian
           matrix in gradient calculation, this number can be
           higher so that time is not wasted computing coordinates
           that do not change.

 RETURNED:
 XYZ1    = Cartesian coordinates.
 IERROR  = Error code: ierror=0 --> Successful calculation.
                       ierror=1 --> Three atoms in a straight line
                                    prevented accurate determination
                                    of cartesian coordinates.
```

## 23.20 icoord.F90

### 23.20.1 function icoord

```
 If the belly option is active, then
 ICOORD will project the coordinate
 I into the global set of coordinates.

 Note that I < 3 * number of non-belly atoms.
```

## 23.21 ijpair.F90

```
This file contains subroutines IJMAKE, IJFIND, and BINLOC.  They
allow fast location of an atom pair (iatm,jatm) in the pairlist
ipair.  After the pairlist has been created, ijmake should be
called to create three arrays that allow fast access to the pairlist.
after calling IJMAKE, subroutine IJFIND may be used to locate the
position of (iatm,jatm) in ipair.  BINLOC is a binary search
routine used by IJMAKE and IJFIND.
Name change variables in order to have the same variable names
everywhere:: oldname -> newname:
             ipmod1  -> ip1old
             ipack   -> ipold
             ipaddr  -> ijold
```

### 23.21.1 subroutine ijmake

```
Creates three pairlist-size arrays (IP1OLD,IPOLD,IJOLD) that
allow near direct access of the address of any atom pair (IATM,JATM)
stored in the pairlist array IPAIR.

After calling this routine, then given any atom pair (IATM,JATM)
that is known to be in the pairlist, the following procedure
will locate the address of that pair in the array IPAIR:

let NPAIRS = IP1(NATOMS+1)-1
    IJPACK = (IATM*(IATM-3))/2 + JATM + 1
     IJMOD = mod(IJPACK,NPAIRS)
      IMIN = IP1OLD(IJMOD)
      IMAX = IP1OLD(IJMOD+1)-1

Search IPOLD(I) over the range I=IMIN,IMAX.

When IPOLD(I).eq.IJPACK then IPMOD(I) will contain the position
in IPAIR where (IATM,JATM) is stored.

The search from IMIN to IMAX will span at most nratio entries, where
NRATIO = ((NATOMS*(NATOMS-1)/2)/NPAIRS
```

```
        = ratio of potential pairs to actual pairs stored.

  If a binary search is used, then the entry can be located in at
  most log2(NRATIO)+1 steps.
```

For calls made by the subroutine, "ijmake":

(See [binloc], page 210.)

### 23.21.2 subroutine ijfind

```
  Locates the atom pair (IATM,JATM) in the pairlist after the arrays
  IP1OLD, IPOLD, and IJOLD have been created by subroutine IJMAKE.

  INPUT:
  NPAIRS = Total number of stored pairs.

  IATM,
  JATM   = Atom pair to be located (iatm > jatm).

  RETURNED:
  IJADDR = Position in ipair where (iatm,jatm) is stored.

  If (IATM,JATM) is not stored, then IJADDR is returned with
  a value of zero.
```

For calls made by the subroutine, "ijfind":

(See [binloc], page 210.)

### 23.21.3 subroutine binloc

```
  Uses a binary search to locate a target element in a sorted list.

  LIST   = Sorted list of integers to be searched.
  ISTART = Starting point of search in list.
  ISTOP  = Stopping point of search in list.
  ITARGT = Value of target element sought in list.
  IFOUND = Returned location of itarget in list.  If not present, then
           ifound is returned with a value of 0.
```

## 23.22 mcweeny.F90

### 23.22.1 subroutine mcweeny

```
author Arjan van der Vaart
date 3/18/1999

 This subroutine performs the McWeeny transformation
 P' = 3P**2 - 2P**3

 The old density matrix (stored in pdiag and pdiat) will
 be overwritten with the transformed matrix. In the process
 piiold and pijold will be used as storage space; piiold
 and pijold will also be overwritten with the transformed
 matrix.

 The transformation is performed nmcw times.
```

For calls made by the subroutine, "mcweeny":

(See [ijfind], page 210.)
(See [getrow], page 211.)

### 23.22.2 subroutine getrow

```
author Arjan van der Vaart
date 3/18/1999

 This algorithm retrieves the rows of the density matrix
 for atom iat. The number of rows equals the number of
 orbitals for atom iat.
```

For calls made by the subroutine, "getrow":

(See [ijfind], page 210.)

## 23.23 mvcrd.F90

### 23.23.1 subroutine mvcrd

```
Copies the coordinates from coord0 into the xyz array
out on internal coordinates, then the cartesian coordinates are
recalculated.  Since the cartesian and/or internal coordinates are
updated through the common blocks /xyzcrd/ and /intcrd/, the array
coord0 should not be xyz or zmat.
```

For calls made by the subroutine, "mvcrd":

(See [getxyz], page 208.)

## 23.24 namemcf.F90

### 23.24.1 function namemcf

```
Returns basename<inti>.ext
e.g. namemcf('divcon',33,'in') = divcon33.in
```

## 23.25 opnfil.F90

### 23.25.1 subroutine opnfil

```
Routine to open file number ifile.  See block data routine for
unit numbers, file names, etc.  Error messages are written
to the main output file and to standard output.  This assumes
that the main output file is opened before any other file, and
that no problems are encountered opening it.
```

### 23.25.2 subroutine opnpfil

```
parallel I/O

works analogous to opnfil, but the filename is
generated and is fname(ifile)<myid>.ext(ifile)
```

### 23.25.3 subroutine opnpfil1

```
parallel I/O

used only in parallel version mosubfdmx
opnpfil can not be used there, since by default
all files of the slaves are set to /dev/null

open file file_name<myid>.fdmx
```

## 23.26 pbcxyz.F90

### 23.26.1 subroutine pbcxyz

```
Returns minimum image coordinates for atom j based on its
location relative to atom i (atomwise)

Note that in case of push, the values of gcres correspond to the
original (non-pushed) configuration. This way, the correct geometry
will be taken (since else dxij would always be larger than dhalf)
```

## 23.27 pbgcart.F90

### 23.27.1 subroutine pbgcart

```
    SUBROUTINE TO COMPUTE CARTESIAN REACTION FIELD ENERGY GRADIENT
    RETURNS GRADIENT IN KCAL/ANGSTROM.
```

## 23.28 random.F90

### 23.28.1 function xrandom

```
Random number generator, taken from Numerical Recipes
returns a random between 0.0 and 1.0
```

### 23.28.2 function irandom

```
Random number generator, adapted from Numerical Recipes
returns a random between 1.0 and imax
```

### 23.28.3 function grandom

```
Generates a random number from a gaussian distribution
with zero mean and variance var
```

## 23.29 rcalc.F90

### 23.29.1 subroutine rcalc

```
Computes and stores interatomic distances rpair for all atoms.
don't use if 'direct' keyword has been specified.

Returns with an error if any two atoms are separated by less
than 0.5 angstroms.
```

For calls made by the subroutine, "rcalc":

(See [pbcxyz], page 213.)

## 23.30 save_module.F90

### 23.30.1 module save_module

```
Save variables

  To view what this module contains:
```

(See [initializesavevars], page 214.)
(See [deallocatesavevars], page 214.)

### 23.30.2 subroutine initializesavevars [from module: save_module]

```
Initialize variables which play the role of save attribute in each
single subroutine.
```

### 23.30.3 subroutine deallocatesavevars [from module: save_module]

```
Deallocate dynamic arrays with 'save' statements.
```

## 23.31 setprtvec.F90

### 23.31.1 subroutine setprtvec

```
Function is used to print eigenvectors from D&C Calculations
```

For calls made by the subroutine, "setprtvec":

(See [bsort1], page 200.)
(See [bsort2], page 201.)

## 23.32 setunit.F90

### 23.32.1 subroutine setunit

```
Set the unit number for the different files
```

## 23.33 spinclean.F90

### 23.33.1 subroutine spinclean

```
Spin checking and cleaning routine adapted from Ed's code
```

## 23.34 strcatf.F90

### 23.34.1 subroutine strcatf

```
Append from app to str having nblank blank spaces between them.
Here trailing blank characters of str will be removed before
caternating.
```

## 23.35 upcase.F90

### 23.35.1 subroutine upcase

```
author Steve Dixon

Changes lower case characters in string(1:iend) to upper case.
```

## 23.36 wdjoin.F90

### 23.36.1 subroutine wdjoin

```
SEARCHES STRING(1:IEND) FOR WORDS THAT ARE SEPARATED BY THE
ONE-CHARACTER SYMBOL SEP, AND MAKES A CONTIGUOUS BLOCK OF
CHARACTERS.  FOR EXAMPLE, IF SEP IS AN EQUALS SIGN '=', THEN
THE SUBSTRING 'TMAX = 9999.0' WOULD BECOME 'TMAX=9999.0'
```

For calls made by the subroutine, "wdjoin":

(See [rdword], page 77.)

## 23.37 whatis.F90

### 23.37.1 subroutine whatis1

```
Function determines if a string is a float
```

### 23.37.2 subroutine whatis1i

```
Function determines if the string is an integer
```

### 23.37.3 subroutine whatis2

```
Function determines if the string is an integer
```

### 23.37.4 subroutine whatis7

```
Function determines if the string is a number, character
or white space
```

## 23.38 zmake.F90

### 23.38.1 subroutine zmake

```
author Ed Brothers
date June 1998

 The purpose of this subroutine is to return internal coordinates to
 the output file.  Conversion from cartesian to internal is not automatic,
 although various subprograms do it.  This is why there are so many calls
 in this subroutine.

 Note that the routine checks to see if the values already present, and,
 if so simply returns.
```

For calls made by the subroutine, "zmake":

(See [bndang], page 199.)
(See [dihedr], page 204.)

# 24 XRAY

# INDEX

## A



## B



## C



## D

## E



## F



## G

## H



## I



## J



## L



## M

## N



## O



## P

## R



## S

## T



## U



## V



## W



## X



## Y



## Z