VALENCE

Generated by Doxygen 1.8.14

Contents

Chapter 1

Variational Subspace Valence Bond (VSVB) method

"The variational subspace valence bond method", G. D. Fletcher, J. Chem. Phys., 142, 134112 (2015).
'Orbital Basis Set' (OBS) version recomputes the super-contracted integrals as needed or stores them in aggregate memory

Chapter 2

Modules Index

2.1 Modules List

Here is a list of all modules with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Module Documentation

4.1 densitywork Module Reference

Variables

- integer, dimension(2) dme_b
- integer, dimension(2) dme_k
- integer, dimension(:, :, :), allocatable pair_sc
- real(dp), dimension(:), allocatable coeff_sc
- integer, dimension(:), allocatable bra_a

  holds the OLCAO
- integer, dimension(:), allocatable bra_b

  holds the OLCAO
- integer, dimension(:), allocatable ket_a

  same as bra_a, but with the ket
- integer, dimension(:), allocatable ket_b

  same as bra_b, but with the ket
- integer, dimension(:), allocatable bexch
- integer, dimension(:), allocatable kexch
- integer, dimension(:), allocatable bra
- integer, dimension(:), allocatable ket
- real(dp), dimension(:, :), allocatable wdet

  overlap integrals between all spin orbitals
- real(dp), dimension(:, :), allocatable abra_npair

  holds the alpha part of the overlap matrix, stored as the overlap of the OLCAO associated with each alpha spin-coupled e- and every other spin orbital (npair, nelec)
- real(dp), dimension(:, :), allocatable bbra_npair

  same as abra_npair, but for beta e-
- real(dp), dimension(:, :), allocatable abra_docc_un

  same as abra_npair, but for the overlap of the OLCAO associated with each non-spin coupled alpha e- (ndocc+unpaired) and all spin-coupled e- (ndocc+nunpd, npair∗2)
- real(dp), dimension(:, :), allocatable bbra_docc_un

  same as bbra_docc_un but for the beta e-
- real(dp), dimension(:, :), allocatable aket

  holds the alpha part of the matrix to take the determinant of for the density
- real(dp), dimension(:, :), allocatable bket

same as aket, but the beta part

- real(dp), dimension(:, :), allocatable aket_docc_un

  holds the docc and unpaired (non-spin coupled) part of aket this never changes during the run, and is copied back into aket for determinant

- real(dp), dimension(:, :), allocatable bket_docc_un

  same as aket_docc_un, but for bket

- integer padded_size
- integer, dimension(:), allocatable ipvt

  holds the pivots for the call to dgetrf

4.1.1 Variable Documentation

4.1.1.1 abra_docc_un

```
real(dp), dimension( :  , :  ), allocatable densitywork::abra_docc_un
```

same as abra_npair, but for the overlap of the OLCAO associated with each non-spin coupled alpha e- (ndocc+unpaired) and all spin-coupled e- (ndocc+nunpd, npair∗2)

Referenced by build_abket(), calculate_vsvb_energy(), and set_up_unpaired_docc().

4.1.1.2 abra_npair

```
real(dp), dimension( :  , :  ), allocatable densitywork::abra_npair
```

holds the alpha part of the overlap matrix, stored as the overlap of the OLCAO associated with each alpha spin-coupled e- and every other spin orbital (npair, nelec)

Referenced by build_abket(), calculate_vsvb_energy(), and dbra().

4.1.1.3 aket

```
real(dp), dimension( :  , :  ), allocatable densitywork::aket
```

holds the alpha part of the matrix to take the determinant of for the density

Referenced by build_abket(), calculate_vsvb_energy(), and det().

4.1.1.4 aket_docc_un

```
real(dp), dimension( :  , :  ), allocatable densitywork::aket_docc_un
```

holds the docc and unpaired (non-spin coupled) part of aket this never changes during the run, and is copied back into aket for determinant

Referenced by build_abket(), calculate_vsvb_energy(), and set_up_unpaired_docc().

4.1.1.5 bbra_docc_un

```
real(dp), dimension( :  , :  ), allocatable densitywork::bbra_docc_un
```

same as bbra_docc_un but for the beta e-

Referenced by build_abket(), calculate_vsvb_energy(), and set_up_unpaired_docc().

4.1.1.6 bbra_npair

```
real(dp), dimension( :  , :  ), allocatable densitywork::bbra_npair
```

same as abra_npair, but for beta e-

Referenced by build_abket(), calculate_vsvb_energy(), and dbra().

4.1.1.7 bexch

```
integer, dimension( :  ), allocatable densitywork::bexch
```

Referenced by calculate_vsvb_energy(), and dbra().

4.1.1.8 bket

```
real(dp), dimension( :  , :  ), allocatable densitywork::bket
```

same as aket, but the beta part

Referenced by build_abket(), calculate_vsvb_energy(), and det().

4.1.1.9 bket_docc_un

```
real(dp), dimension( :  , :  ), allocatable densitywork::bket_docc_un
```

same as aket_docc_un, but for bket

Referenced by build_abket(), calculate_vsvb_energy(), and set_up_unpaired_docc().

4.1.1.10 bra

```
integer, dimension( :  ), allocatable densitywork::bra
```

Referenced by calculate_vsvb_energy(), demgs_opt(), first_order_opt(), guess_energy(), schwarz_ints(), spin_↩ opt(), vsvb_energy(), and wfndet().

4.1.1.11 bra_a

```
integer, dimension( :  ), allocatable densitywork::bra_a
```

holds the OLCAO

Referenced by calculate_vsvb_energy(), dbra(), det(), and set_up_unpaired_docc().

4.1.1.12 bra_b

```
integer, dimension( :  ), allocatable densitywork::bra_b
```

holds the OLCAO

Referenced by calculate_vsvb_energy(), dbra(), det(), and set_up_unpaired_docc().

4.1.1.13 coeff_sc

```
real(dp), dimension( :  ), allocatable densitywork::coeff_sc
```

Referenced by valence_finit::deallocate_input(), density(), valence_init::read_allocate_input(), spin_opt(), xm::xm↩ _input(), and xm::xm_output().

4.1.1.14 dme_b

```
integer, dimension(2) densitywork::dme_b
```

Referenced by det(), and vsvb_energy().

4.1.1.15 dme_k

```
integer, dimension(2) densitywork::dme_k
```

Referenced by det(), and vsvb_energy().

4.1.1.16 ipvt

```
integer, dimension( : ), allocatable densitywork::ipvt
```

holds the pivots for the call to dgetrf

Referenced by calculate_vsvb_energy(), and det().

4.1.1.17 ket

```
integer, dimension( : ), allocatable densitywork::ket
```

Referenced by calculate_vsvb_energy(), demgs_opt(), first_order_opt(), guess_energy(), schwarz_ints(), spin_↩ opt(), vsvb_energy(), and wfndet().

4.1.1.18 ket_a

```
integer, dimension( : ), allocatable densitywork::ket_a
```

same as bra_a, but with the ket

Referenced by build_abket(), calculate_vsvb_energy(), det(), dket(), and set_up_unpaired_docc().

4.1.1.19 ket_b

```
integer, dimension( : ), allocatable densitywork::ket_b
```

same as bra_b, but with the ket

Referenced by build_abket(), calculate_vsvb_energy(), det(), dket(), and set_up_unpaired_docc().

4.1.1.20 kexch

```
integer, dimension( : ), allocatable densitywork::kexch
```

Referenced by calculate_vsvb_energy(), and dket().

4.1.1.21 padded_size

```
integer densitywork::padded_size
```

Referenced by calculate_vsvb_energy(), and det().

4.1.1.22 pair_sc

```
integer, dimension( : , : , : ), allocatable densitywork::pair_sc
```

Referenced by dbra(), valence_finit::deallocate_input(), dket(), valence_init::read_allocate_input(), xm::xm_input(), and xm::xm_output().

4.1.1.23 wdet

```
real(dp), dimension( : , : ), allocatable densitywork::wdet
```

overlap integrals between all spin orbitals

Referenced by calculate_vsvb_energy(), dbra(), demgs_opt(), first_order_opt(), set_up_unpaired_docc(), and wfndet().

4.2 integrals Module Reference

Functions/Subroutines

- integer function shell_size (l_mom)

Variables

- integer nelec
- integer norbs
- integer nalpha
- integer nbeta
- integer hdim
- integer nstore
- integer max_obs

  largest possible number of basis functions in a OLCAO, based on the atoms specified in the OLCAO expansion
- integer natom

  The number of atoms/point charges in the geometry.
- integer npair

  Number of spin coupled electron/orbital PAIRS.
- integer ndocc

  Number of double-occupied (DOCC) orbitals.
- integer totlen

  Total length of the orbital weight list in wavefunction.
- integer nunpd

  Number of unpaired electrons/orbitals.
- integer ndf

  Number of derived basis functions (LCAO-type)
- integer nxorb

  Number of orbital excitations.
- integer natom_t

  The number of atom types.
- integer nspinc

  Number of spin couplings.
- integer nang

  Highest angular momentum in the basis set.
- integer xpmax

  Length of the largest orbital expansion.
- integer num_sh

  Number of unique atomic basis set shells.
- integer num_pr

  Number of unique atomic basis set primitives.
- integer nset
- integer mxctr
- integer ntol_e_min
- integer ntol_e_max
- integer max_iter
- integer, dimension(:, :), allocatable orbset
- integer, dimension(:), allocatable atom_t
- real(dp), dimension(:, :), allocatable coords
- integer, dimension(:), allocatable map_atom2shell

  holds indexing for the basis function shells for a given the atom type.
- integer, dimension(:), allocatable num_shell_atom

  holds the number of basis function shells for a given atom type
- integer, dimension(:), allocatable map_shell2prim

  holds indexing into the start of the primitive arrays for a given the total shell index.
- integer, dimension(:), allocatable ang_mom

  holds the shell angular momentum for a given shell index

- real(dp), dimension(:), allocatable nuc_charge
- real(dp), dimension(:), allocatable exponent

  holds the exponent for a primitive gaussian
- real(dp), dimension(:), allocatable con_coeff

  holds the contraction coefficient for a primitive gaussian
- integer, dimension(:), allocatable orbas_atnum

  holds the number of atoms whose basis sets make up a given OLCAO index
- integer, dimension(:, :), allocatable orbas_atset

  holds the overall atom index from the input file list for a given atom index in a OLCAO expansion and the OLCAO index
- integer, dimension(:), allocatable map_orbs

  holds the starting index for the set of basis functions that a given OLCAO is expanded in in the total OLCAO wavefunction (based just on how the orbitals are listed in order in the input file) can be used to index coeff() to loop through the basis function coefficients for a given orbital
- integer, dimension(:), allocatable xpset

  for a given index of a function in the OLCAO wavefunction, returns the index of the basis function in the list of all basis functions associated with the atoms that the OLCAO is expanded over
- integer, dimension(:), allocatable xorb
- integer, dimension(:), allocatable root
- real(dp), dimension(:), allocatable coeff

  holds the coefficient of a basis function in an OLCAO for a given basis function index in the total OLCAO wavefunction list (where the list is just based on the order the orbitals are listed in the input file)
- integer, dimension(:, :), allocatable nxyz

  powers of x,y,z coordinates for s, p, d shells in order
- real(dp), dimension(:), allocatable angn

  pnym(ij) = ashl(i) ∗ 1/(ashl(power of x)) ∗ 1/(ashl(power of y)) ∗ 1/(ashl((power of z)) for each primitive in each angular momentum i (where power of x + power of y + power of z = i). index ij walks over all primitives–1 for s, 3 for p, etc.
- real(dp), dimension(:), allocatable ashl

  sqrt∗((2∗i-1)!!) for each given angular momentum i=0,nang
- real(dp), dimension(:), allocatable ashi

  1/ashl(i) for each given angular momentum i=0,nang
- real(dp), dimension(:), allocatable dij
- real(dp), dimension(:), allocatable coeffi
- real(dp), dimension(:), allocatable coeffj
- integer, dimension(:, :), allocatable atom_ndf
- integer, dimension(:), allocatable ndf2orb
- integer, dimension(:), allocatable xpnew
- real(dp), dimension(:), allocatable dkl
- real(dp), dimension(:), allocatable coeffk
- real(dp), dimension(:), allocatable coeffl
- real(dp), dimension(:), allocatable schwarz
- real(dp), dimension(:), allocatable eribuf
- real(dp) ctol
- real(dp) dtol
- real(dp) itol
- logical spinopt
- logical store_eri
- logical eri_stored
- logical dem_gs
- real(dp) sint
- real(dp) hint
- real(dp) gint
- real(dp) enucrep
- real(dp), dimension(:, :), allocatable ham
- real(dp), dimension(:, :), allocatable ovl
- real(dp) ptbnmax
- real(dp) feather

4.2.1 Function/Subroutine Documentation

4.2.1.1 shell_size()

```
integer function integrals::shell_size (
            integer l_mom )
```

Referenced by calculate_vsvb_energy(), first_order_opt(), minimize_energy(), ndf2obs(), and setangn().

4.2.2 Variable Documentation

4.2.2.1 ang_mom

```
integer, dimension( : ), allocatable integrals::ang_mom
```

holds the shell angular momentum for a given shell index

Referenced by calculate_vsvb_energy(), valence_finit::deallocate_input(), first_order_opt(), minimize_energy(), ndf2obs(), valence_init::read_allocate_input(), and xm::xm_input().

4.2.2.2 angn

```
real(dp), dimension( : ), allocatable integrals::angn
```

pnym(ij) = ashl(i) $*$ 1/(ashl(power of x)) $*$ 1/(ashl(power of y)) $*$ 1/(ashl((power of z)) for each primitive in each angular momentum i (where power of x + power of y + power of z = i). index ij walks over all primitives–1 for s, 3 for p, etc.

Referenced by calculate_vsvb_energy(), minimize_energy(), and setangn().

4.2.2.3 ashi

```
real(dp), dimension( : ), allocatable integrals::ashi
```

1/ashl(i) for each given angular momentum i=0,nang

Referenced by calculate_vsvb_energy(), and setangn().

4.2.2.4 ashl

```
real(dp), dimension( : ), allocatable integrals::ashl
```

sqrt∗((2∗i-1)!!) for each given angular momentum i=0,nang

Referenced by calculate_vsvb_energy(), and setangn().

4.2.2.5 atom_ndf

```
integer, dimension( : , : ), allocatable integrals::atom_ndf
```

Referenced by calculate_vsvb_energy(), and ndf2obs().

4.2.2.6 atom_t

```
integer, dimension( : ), allocatable integrals::atom_t
```

Referenced by calculate_vsvb_energy(), valence_finit::deallocate_input(), first_order_opt(), minimize_energy(), ndf2obs(), valence_init::read_allocate_input(), and xm::xm_input().

4.2.2.7 coeff

```
real(dp), dimension( : ), allocatable integrals::coeff
```

holds the coefficient of a basis function in an OLCAO for a given basis function index in the total OLCAO wavefunction list (where the list is just based on the order the orbitals are listed in the input file)

Referenced by valence_finit::deallocate_input(), demgs_opt(), first_order_opt(), minimize_energy(), normal(), valence_init::read_allocate_input(), xm::xm_input(), and xm::xm_output().

4.2.2.8 coeffi

```
real(dp), dimension( : ), allocatable integrals::coeffi
```

Referenced by calculate_vsvb_energy(), and minimize_energy().

4.2.2.9 coeffj

```
real(dp), dimension( :  ), allocatable integrals::coeffj
```

Referenced by calculate_vsvb_energy(), and minimize_energy().

4.2.2.10 coeffk

```
real(dp), dimension( :  ), allocatable integrals::coeffk
```

Referenced by calculate_vsvb_energy(), and minimize_energy().

4.2.2.11 coeffl

```
real(dp), dimension( :  ), allocatable integrals::coeffl
```

Referenced by calculate_vsvb_energy(), and minimize_energy().

4.2.2.12 con_coeff

```
real(dp), dimension( :  ), allocatable integrals::con_coeff
```

holds the contraction coefficient for a primitive gaussian

Referenced by valence_finit::deallocate_input(), first_order_opt(), valence_init::read_allocate_input(), and xm::xm↩ _input().

4.2.2.13 coords

```
real(dp), dimension( :  , :  ), allocatable integrals::coords
```

Referenced by calcsurface(), calculate_vsvb_energy(), valence_finit::deallocate_input(), first_order_opt(), minimize_energy(), valence_init::read_allocate_input(), valence_init::valence_initialize(), and xm::xm_input().

4.2.2.14 ctol

```
real(dp) integrals::ctol
```

Referenced by valence_init::read_allocate_input().

4.2.2.15 dem_gs

```
logical integrals::dem_gs
```

Referenced by calculate_vsvb_energy(), minimize_energy(), and vsvb_energy().

4.2.2.16 dij

```
real(dp), dimension( :  ), allocatable integrals::dij
```

Referenced by calculate_vsvb_energy(), and minimize_energy().

4.2.2.17 dkl

```
real(dp), dimension( :  ), allocatable integrals::dkl
```

Referenced by calculate_vsvb_energy(), and minimize_energy().

4.2.2.18 dtol

```
real(dp) integrals::dtol
```

Referenced by det(), minimize_energy(), and valence_init::read_allocate_input().

4.2.2.19 enucrep

```
real(dp) integrals::enucrep
```

Referenced by calculate_vsvb_energy(), demgs_opt(), first_order_opt(), guess_energy(), and spin_opt().

4.2.2.20 eri_stored

```
logical integrals::eri_stored
```

Referenced by demgs_opt(), first_order_opt(), and vsvb_energy().

4.2.2.21 eribuf

```
real(dp), dimension( :  ), allocatable integrals::eribuf
```

Referenced by calculate_vsvb_energy(), and vsvb_energy().

4.2.2.22 exponent

```
real(dp), dimension( :  ), allocatable integrals::exponent
```

holds the exponent for a primitive gaussian

Referenced by valence_finit::deallocate_input(), first_order_opt(), valence_init::read_allocate_input(), and xm::xm↩
_input().

4.2.2.23 feather

```
real(dp) integrals::feather
```

Referenced by demgs_opt(), and xm::xm_input().

4.2.2.24 gint

```
real(dp) integrals::gint
```

Referenced by minimize_energy(), schwarz_ints(), and vsvb_energy().

4.2.2.25 ham

```
real(dp), dimension( :  , :  ), allocatable integrals::ham
```

Referenced by calculate_vsvb_energy(), density(), first_order_opt(), and spin_opt().

4.2.2.26 hdim

```
integer integrals::hdim
```

Referenced by calculate_vsvb_energy().

4.2.2.27 hint

```
real(dp) integrals::hint
```

Referenced by density(), minimize_energy(), and vsvb_energy().

4.2.2.28 itol

```
real(dp) integrals::itol
```

Referenced by valence_init::read_allocate_input(), and vsvb_energy().

4.2.2.29 map_atom2shell

```
integer, dimension( : ), allocatable integrals::map_atom2shell
```

holds indexing for the basis function shells for a given the atom type.

Referenced by calculate_vsvb_energy(), valence_finit::deallocate_input(), first_order_opt(), minimize_energy(), ndf2obs(), valence_init::read_allocate_input(), and xm::xm_input().

4.2.2.30 map_orbs

```
integer, dimension( : ), allocatable integrals::map_orbs
```

holds the starting index for the set of basis functions that a given OLCAO is expanded in in the total OLCAO wavefunction (based just on how the orbitals are listed in order in the input file) can be used to index coeff() to loop through the basis function coefficients for a given orbital

Referenced by valence_finit::deallocate_input(), demgs_opt(), first_order_opt(), minimize_energy(), ndf2obs(), normal(), valence_init::read_allocate_input(), xm::xm_input(), and xm::xm_output().

4.2.2.31 map_shell2prim

```
integer, dimension( : ), allocatable integrals::map_shell2prim
```

holds indexing into the start of the primitive arrays for a given the total shell index.

Referenced by valence_finit::deallocate_input(), first_order_opt(), valence_init::read_allocate_input(), and xm::xm↩
_input().

4.2.2.32 max_iter

```
integer integrals::max_iter
```

Referenced by calculate_vsvb_energy(), minimize_energy(), and valence_init::read_allocate_input().

4.2.2.33 max_obs

```
integer integrals::max_obs
```

largest possible number of basis functions in a OLCAO, based on the atoms specified in the OLCAO expansion

Referenced by calculate_vsvb_energy(), and minimize_energy().

4.2.2.34 mxctr

```
integer integrals::mxctr
```

Referenced by calculate_vsvb_energy(), valence_init::read_allocate_input(), and xm::xm_input().

4.2.2.35 nalpha

```
integer integrals::nalpha
```

Referenced by calculate_vsvb_energy(), det(), and valence_init::read_allocate_input().

4.2.2.36 nang

```
integer integrals::nang
```

Highest angular momentum in the basis set.

Referenced by calculate_vsvb_energy(), valence_init::read_allocate_input(), and setangn().

4.2.2.37 natom

```
integer integrals::natom
```

The number of atoms/point charges in the geometry.

Referenced by calcsurface(), calculate_vsvb_energy(), first_order_opt(), getn(), minimize_energy(), valence_init↩
::read_allocate_input(), valence_init::valence_initialize(), and xm::xm_input().

4.2.2.38 natom_t

```
integer integrals::natom_t
```

The number of atom types.

Referenced by first_order_opt(), valence_init::read_allocate_input(), and xm::xm_input().

4.2.2.39 nbeta

```
integer integrals::nbeta
```

Referenced by calculate_vsvb_energy(), det(), and valence_init::read_allocate_input().

4.2.2.40 ndf

```
integer integrals::ndf
```

Number of derived basis functions (LCAO-type)

Referenced by calculate_vsvb_energy(), valence_init::read_allocate_input(), xm::xm_input(), and xm::xm_output().

4.2.2.41 ndf2orb

```
integer, dimension( :  ), allocatable integrals::ndf2orb
```

Referenced by calculate_vsvb_energy(), and ndf2obs().

4.2.2.42 ndocc

```
integer integrals::ndocc
```

Number of double-occupied (DOCC) orbitals.

Referenced by build_abket(), calculate_vsvb_energy(), demgs_opt(), density(), density_sc(), first_order_opt(), guess_energy(), valence_init::read_allocate_input(), set_up_unpaired_docc(), spin_opt(), xm::xm_input(), and xm::xm_output().

4.2.2.43 nelec

```
integer integrals::nelec
```

Referenced by calculate_vsvb_energy(), dbra(), demgs_opt(), first_order_opt(), valence_init::read_allocate_input(), spin_opt(), vsvb_energy(), and wfndet().

4.2.2.44 norbs

```
integer integrals::norbs
```

Referenced by calculate_vsvb_energy(), first_order_opt(), minimize_energy(), valence_init::read_allocate_input(), and xm::xm_input().

4.2.2.45 npair

```
integer integrals::npair
```

Number of spin coupled electron/orbital PAIRS.

Referenced by build_abket(), calculate_vsvb_energy(), dbra(), demgs_opt(), density(), density_sc(), dket(), first↩_order_opt(), guess_energy(), valence_init::read_allocate_input(), set_up_unpaired_docc(), spin_opt(), vsvb_↩energy(), xm::xm_input(), and xm::xm_output().

4.2.2.46 nset

```
integer integrals::nset
```

Referenced by minimize_energy(), valence_init::read_allocate_input(), and xm::xm_input().

4.2.2.47 nspinc

```
integer integrals::nspinc
```

Number of spin couplings.

Referenced by calculate_vsvb_energy(), density(), minimize_energy(), valence_init::read_allocate_input(), spin_↩opt(), xm::xm_input(), and xm::xm_output().

4.2.2.48 nstore

```
integer integrals::nstore
```

Referenced by calculate_vsvb_energy(), and vsvb_energy().

4.2.2.49 ntol_e_max

```
integer integrals::ntol_e_max
```

Referenced by calculate_vsvb_energy(), minimize_energy(), and valence_init::read_allocate_input().

4.2.2.50 ntol_e_min

```
integer integrals::ntol_e_min
```

Referenced by minimize_energy(), and valence_init::read_allocate_input().

4.2.2.51 nuc_charge

```
real(dp), dimension( :  ), allocatable integrals::nuc_charge
```

Referenced by calculate_vsvb_energy(), valence_finit::deallocate_input(), minimize_energy(), valence_init::read↩
_allocate_input(), and xm::xm_input().

4.2.2.52 num_pr

```
integer integrals::num_pr
```

Number of unique atomic basis set primitives.

Referenced by valence_init::read_allocate_input(), and xm::xm_input().

4.2.2.53 num_sh

```
integer integrals::num_sh
```

Number of unique atomic basis set shells.

Referenced by valence_init::read_allocate_input(), and xm::xm_input().

4.2.2.54 num_shell_atom

```
integer, dimension( :  ), allocatable integrals::num_shell_atom
```

holds the number of basis function shells for a given atom type

Referenced by calculate_vsvb_energy(), valence_finit::deallocate_input(), first_order_opt(), minimize_energy(), ndf2obs(), valence_init::read_allocate_input(), and xm::xm_input().

4.2.2.55 nunpd

```
integer integrals::nunpd
```

Number of unpaired electrons/orbitals.

Referenced by build_abket(), calculate_vsvb_energy(), demgs_opt(), density(), density_sc(), first_order_opt(), guess_energy(), valence_init::read_allocate_input(), set_up_unpaired_docc(), spin_opt(), vsvb_energy(), xm::xm↩_input(), and xm::xm_output().

4.2.2.56 nxorb

```
integer integrals::nxorb
```

Number of orbital excitations.

Referenced by calculate_vsvb_energy(), first_order_opt(), valence_init::read_allocate_input(), xm::xm_input(), and xm::xm_output().

4.2.2.57 nxyz

```
integer, dimension( :  , :  ), allocatable integrals::nxyz
```

powers of x,y,z coordinates for s, p, d shells in order

Referenced by calculate_vsvb_energy(), and setangn().

4.2.2.58 orbas_atnum

```
integer, dimension( :  ), allocatable integrals::orbas_atnum
```

holds the number of atoms whose basis sets make up a given OLCAO index

Referenced by calculate_vsvb_energy(), valence_finit::deallocate_input(), first_order_opt(), minimize_energy(), ndf2obs(), valence_init::read_allocate_input(), xm::xm_input(), and xm::xm_output().

4.2.2.59 orbas_atset

```
integer, dimension( :  , :  ), allocatable integrals::orbas_atset
```

holds the overall atom index from the input file list for a given atom index in a OLCAO expansion and the OLCAO index

Referenced by calculate_vsvb_energy(), valence_finit::deallocate_input(), first_order_opt(), minimize_energy(), ndf2obs(), valence_init::read_allocate_input(), xm::xm_input(), and xm::xm_output().

4.2.2.60 orbset

```
integer, dimension( :, :  ), allocatable integrals::orbset
```

Referenced by valence_finit::deallocate_input(), minimize_energy(), valence_init::read_allocate_input(), and xm↩ ::xm_input().

4.2.2.61 ovl

```
real(dp), dimension( :  , :  ), allocatable integrals::ovl
```

Referenced by calculate_vsvb_energy(), density(), first_order_opt(), and spin_opt().

4.2.2.62 ptbnmax

```
real(dp) integrals::ptbnmax
```

Referenced by calculate_vsvb_energy(), demgs_opt(), and xm::xm_input().

4.2.2.63 root

```
integer, dimension( :  ), allocatable integrals::root
```

Referenced by valence_finit::deallocate_input(), first_order_opt(), valence_init::read_allocate_input(), xm::xm_↩ input(), and xm::xm_output().

4.2.2.64 schwarz

```
real(dp), dimension( :  ), allocatable integrals::schwarz
```

Referenced by calculate_vsvb_energy(), schwarz_ints(), and vsvb_energy().

4.2.2.65 sint

```
real(dp) integrals::sint
```

Referenced by demgs_opt(), density(), first_order_opt(), minimize_energy(), normal(), vsvb_energy(), and wfndet().

4.2.2.66 spinopt

```
logical integrals::spinopt
```

Referenced by density(), valence_init::read_allocate_input(), and spin_opt().

4.2.2.67 store_eri

```
logical integrals::store_eri
```

Referenced by demgs_opt(), first_order_opt(), guess_energy(), spin_opt(), and vsvb_energy().

4.2.2.68 totlen

```
integer integrals::totlen
```

Total length of the orbital weight list in wavefunction.

Referenced by valence_init::read_allocate_input(), and xm::xm_input().

4.2.2.69 xorb

```
integer, dimension( :  ), allocatable integrals::xorb
```

Referenced by valence_finit::deallocate_input(), first_order_opt(), valence_init::read_allocate_input(), xm::xm_↩ input(), and xm::xm_output().

4.2.2.70 xpmax

```
integer integrals::xpmax
```

Length of the largest orbital expansion.

Referenced by calculate_vsvb_energy(), and valence_init::read_allocate_input().

4.2.2.71 xpnew

```
integer, dimension( :  ), allocatable integrals::xpnew
```

Referenced by calculate_vsvb_energy(), minimize_energy(), and ndf2obs().

4.2.2.72 xpset

```
integer, dimension( :  ), allocatable integrals::xpset
```

for a given index of a function in the OLCAO wavefunction, returns the index of the basis function in the list of all basis functions associated with the atoms that the OLCAO is expanded over

Referenced by valence_finit::deallocate_input(), first_order_opt(), minimize_energy(), ndf2obs(), valence_init↩ ::read_allocate_input(), xm::xm_input(), and xm::xm_output().

4.3 timing_flops Module Reference

Variables

- integer(8) flop
- integer(8) count_determinants
- real(dp) kernel_time
- real(dp) initial_time
- real(dp) guess_time
- real(dp) final_time

4.3.1 Variable Documentation

4.3.1.1 count_determinants

```
integer(8) timing_flops::count_determinants
```

Referenced by givdr(), xm::xm_end(), and xm::xm_propagate().

4.3.1.2 final_time

```
real(dp) timing_flops::final_time
```

Referenced by xm::xm_end().

4.3.1.3 flop

```
integer(8) timing_flops::flop
```

4.3.1.4 guess_time

```
real(dp) timing_flops::guess_time
```

Referenced by calculate_vsvb_energy(), and xm::xm_end().

4.3.1.5 initial_time

```
real(dp) timing_flops::initial_time
```

Referenced by xm::xm_end(), and xm::xm_propagate().

4.3.1.6 kernel_time

```
real(dp) timing_flops::kernel_time
```

Referenced by givdr(), xm::xm_end(), and xm::xm_propagate().

4.4 tools Module Reference

Functions/Subroutines

- subroutine, public angs2bohr (natom, coords)
- subroutine, public get_nuclear_repulsion_energy (natom, atom_t, nuc_charge, coords, nre)

Variables

- integer, parameter dp =kind(0.d0)

4.4.1 Function/Subroutine Documentation

4.4.1.1 angs2bohr()

```
subroutine, public tools::angs2bohr (
             integer, intent(in) natom,
             real(dp), dimension(:,:), intent(inout) coords )
```

Referenced by calcsurface(), calculate_vsvb_energy(), and valence_init::valence_initialize().

4.4.1.2 get_nuclear_repulsion_energy()

```
subroutine, public tools::get_nuclear_repulsion_energy (
             integer, intent(in) natom,
             integer, dimension(:), intent(in) atom_t,
             real(dp), dimension(:), intent(in) nuc_charge,
             real(dp), dimension(:,:), intent(in) coords,
             real(dp), intent(out) nre )
```

Referenced by calculate_vsvb_energy().

4.4.2 Variable Documentation

4.4.2.1 dp

```
integer, parameter tools::dp =kind(0.d0)
```

Referenced by build_abket(), calcsurface(), calculate_vsvb_energy(), dblfac(), dbra(), demgs_opt(), density(), density_sc(), det(), dket(), first_order_opt(), getn(), givdr(), givens(), givens_orig(), givens_single(), guess_↩energy(), init(), minimize_energy(), norm_prim(), valence_init::read_allocate_input(), schwarz_ints(), setangn(), spin_opt(), valence(), vsvb_energy(), wfndet(), xm::write_determinant(), xm::write_matrix(), xm::xm_abort(), xm↩::xm_dtriang8(), xm::xm_equalize(), xm::xm_input(), xm::xm_output(), xm::xm_print(), and xm::xm_propagate().

4.5 valence_finit Module Reference

Functions/Subroutines

- subroutine valence_finalize (comm)
- subroutine deallocate_input

4.5.1 Function/Subroutine Documentation

4.5.1.1 deallocate_input()

```
subroutine valence_finit::deallocate_input ( )
```

References integrals::ang_mom, integrals::atom_t, integrals::coeff, densitywork::coeff_sc, integrals::con_coeff, integrals::coords, integrals::exponent, integrals::map_atom2shell, integrals::map_orbs, integrals::map_shell2prim, integrals::nuc_charge, integrals::num_shell_atom, integrals::orbas_atnum, integrals::orbas_atset, integrals::orbset, densitywork::pair_sc, integrals::root, integrals::xorb, and integrals::xpset.

Referenced by valence_finalize().

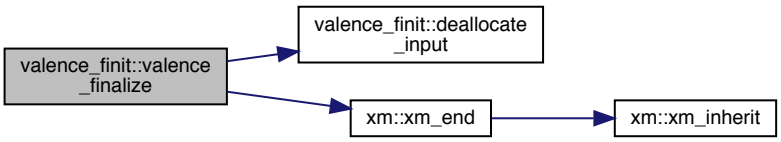
4.5.1.2 valence_finalize()

```
subroutine valence_finit::valence_finalize (
             integer, intent(in), optional comm )
```

References deallocate_input(), and xm::xm_end().

Referenced by finalize(), valence(), and valence_driver().

Here is the call graph for this function:



4.6 valence_init Module Reference

Functions/Subroutines

- subroutine valence_initialize (comm)
- subroutine read_allocate_input

4.6.1 Function/Subroutine Documentation

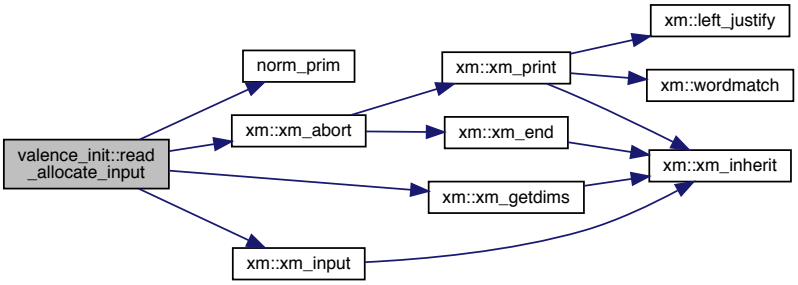
4.6.1.1 read_allocate_input()

```
subroutine valence_init::read_allocate_input ( )
```

References integrals::ang_mom, integrals::atom_t, integrals::coeff, densitywork::coeff_sc, integrals::con_coeff, integrals::coords, integrals::ctol, tools::dp, integrals::dtol, integrals::exponent, integrals::itol, integrals::map_↩ atom2shell, integrals::map_orbs, integrals::map_shell2prim, integrals::max_iter, integrals::mxctr, integrals::nalpha, integrals::nang, integrals::natom, integrals::natom_t, integrals::nbeta, integrals::ndf, integrals::ndocc, integrals↩ ::nelec, integrals::norbs, norm_prim(), integrals::npair, integrals::nset, integrals::nspinc, integrals::ntol_e_max, integrals::ntol_e_min, integrals::nuc_charge, integrals::num_pr, integrals::num_sh, integrals::num_shell_atom, integrals::nunpd, integrals::nxorb, integrals::orbas_atnum, integrals::orbas_atset, integrals::orbset, densitywork↩ ::pair_sc, integrals::root, integrals::spinopt, integrals::totlen, xm::xm_abort(), xm::xm_getdims(), xm::xm_input(), integrals::xorb, integrals::xpmax, and integrals::xpset.

Referenced by valence_initialize().

Here is the call graph for this function:



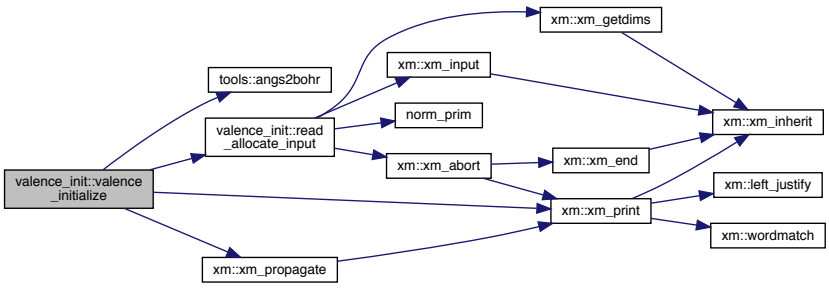
4.6.1.2 valence_initialize()

```
subroutine valence_init::valence_initialize (
            integer, intent(in), optional comm )
```

References tools::angs2bohr(), integrals::coords, integrals::natom, read_allocate_input(), xm::xm_print(), and xm↩ ::xm_propagate().

Referenced by init(), valence(), and valence_driver().

Here is the call graph for this function:



4.7 xm Module Reference

Functions/Subroutines

- subroutine xm_getdims (natom, natom_t, npair, nunpd, ndocc, totlen, xpmax, nspinc, num_sh, num_pr, nang, ndf, nset, nxorb, mxctr)
- subroutine xm_input (ntol_c, ntol_e_min_in, ntol_e_max_in, ntol_d, ntol_i, orbset_in, max_iter_in, mxctr_in)
- logical function wordmatch (word1, word2, nchar)
- subroutine xm_print (mode, message, ints, dbls)
- subroutine left_justify (sentence, outbuf, maxlen)
- subroutine xm_output (mode, energy, etol)
- subroutine xm_propagate (comm)
- subroutine xm_end (comm)
- subroutine xm_share (chtype, buff, len)
- subroutine xm_equalize (buff, datalen)
- subroutine xm_inherit (num_proc, myrank, master)
- subroutine xm_abort (error_message)
- subroutine xm_dtriang (ij, i, j)
- subroutine xm_dtriang8 (ij, i, j)
- subroutine write_matrix (adet, max_n, n, filename)
- subroutine write_determinant (d)

Variables

- integer valence_global_communicator
- integer nrank
- integer irank

4.7.1 Function/Subroutine Documentation

4.7.1.1 left_justify()

```
subroutine xm::left_justify (
             character(1), dimension(*) sentence,
             character(1), dimension(*) outbuf,
             integer maxlen )
```

Referenced by xm_print().

4.7.1.2 wordmatch()

```
logical function xm::wordmatch (
             character(1), dimension(*) word1,
             character(1), dimension(*) word2,
             integer nchar )
```

Referenced by xm_print().

4.7.1.3 write_determinant()

```
subroutine xm::write_determinant (
             real(dp), intent(in) d )
```

References tools::dp, and irank.

Referenced by givdr().

4.7.1.4 write_matrix()

```
subroutine xm::write_matrix (
             real(dp), dimension( max_n, * ), intent(in) adet,
             integer, intent(in) max_n,
             integer, intent(in) n,
             character(len=*), intent(in), optional filename )
```

References tools::dp, and irank.

Referenced by first_order_opt(), spin_opt(), and wfndet().

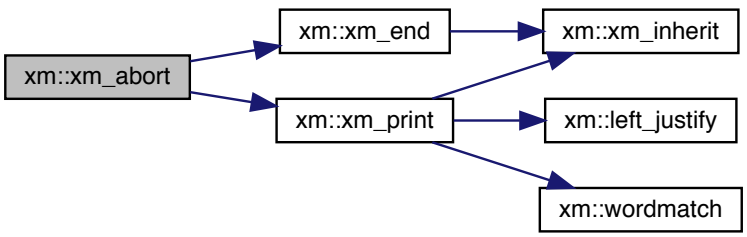
4.7.1.5 xm_abort()

```
subroutine xm::xm_abort (
             character(*) error_message )
```

References tools::dp, xm_end(), and xm_print().

Referenced by first_order_opt(), valence_init::read_allocate_input(), and spin_opt().

Here is the call graph for this function:



4.7.1.6 xm_dtriang()

```
subroutine xm::xm_dtriang (
             integer ij,
             integer i,
             integer j )
```

Referenced by vsvb_energy().

4.7.1.7 xm_dtriang8()

```
subroutine xm::xm_dtriang8 (
             integer(8) ij,
             integer i,
             integer j )
```

References tools::dp.

Referenced by vsvb_energy().

4.7.1.8 xm_end()

```
subroutine xm::xm_end (
             integer, intent(in), optional comm )
```

References timing_flops::count_determinants, timing_flops::final_time, timing_flops::guess_time, timing_flops::initial_time, timing_flops::kernel_time, nrank, valence_global_communicator, and xm_inherit().

Referenced by calcsurface(), valence_finit::valence_finalize(), and xm_abort().

Here is the call graph for this function:

4.7.1.9 xm_equalize()

```
subroutine xm::xm_equalize (
            real(dp), dimension(*) buff,
            integer datalen )
```

References tools::dp, and valence_global_communicator.

Referenced by demgs_opt(), first_order_opt(), guess_energy(), schwarz_ints(), spin_opt(), and wfndet().

4.7.1.10 xm_getdims()

```
subroutine xm::xm_getdims (
            integer natom,
            integer natom_t,
            integer npair,
            integer nunpd,
            integer ndocc,
            integer totlen,
            integer xpmax,
            integer nspinc,
            integer num_sh,
            integer num_pr,
            integer nang,
            integer ndf,
            integer nset,
            integer nxorb,
            integer mxctr )
```

References valence_global_communicator, and xm_inherit().

Referenced by valence_init::read_allocate_input().

Here is the call graph for this function:

4.7.1.11 xm_inherit()

```
subroutine xm::xm_inherit (
             integer num_proc,
             integer myrank,
             integer master )
```

References valence_global_communicator.

Referenced by xm_end(), xm_getdims(), xm_input(), xm_output(), and xm_print().

4.7.1.12 xm_input()

```
subroutine xm::xm_input (
             integer ntol_c,
             integer ntol_e_min_in,
             integer ntol_e_max_in,
             integer ntol_d,
             integer ntol_i,
             integer, dimension( 2, *) orbset_in,
             integer max_iter_in,
             integer mxctr_in )
```

References integrals::ang_mom, integrals::atom_t, integrals::coeff, densitywork::coeff_sc, integrals::con_coeff, integrals::coords, tools::dp, integrals::exponent, integrals::feather, integrals::map_atom2shell, integrals::map_↩ orbs, integrals::map_shell2prim, integrals::mxctr, integrals::natom, integrals::natom_t, integrals::ndf, integrals↩ ::ndocc, integrals::norbs, integrals::npair, integrals::nset, integrals::nspinc, integrals::nuc_charge, integrals::num↩ _pr, integrals::num_sh, integrals::num_shell_atom, integrals::nunpd, integrals::nxorb, integrals::orbas_atnum, integrals::orbas_atset, integrals::orbset, densitywork::pair_sc, integrals::ptbnmax, integrals::root, integrals::totlen, valence_global_communicator, xm_inherit(), integrals::xorb, and integrals::xpset.

Referenced by valence_init::read_allocate_input().

Here is the call graph for this function:

4.7.1.13 xm_output()

```
subroutine xm::xm_output (
             character(*) mode,
             real(dp) energy,
             real(dp) etol )
```

References integrals::coeff, densitywork::coeff_sc, tools::dp, integrals::map_orbs, integrals::ndf, integrals::ndocc, integrals::npair, integrals::nspinc, integrals::nunpd, integrals::nxorb, integrals::orbas_atnum, integrals::orbas_atset, densitywork::pair_sc, integrals::root, xm_inherit(), integrals::xorb, and integrals::xpset.

Referenced by calculate_vsvb_energy(), demgs_opt(), and minimize_energy().

Here is the call graph for this function:

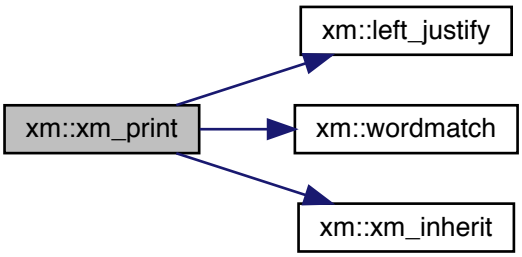
4.7.1.14 xm_print()

```
subroutine xm::xm_print (
             character(*) mode,
             character(*) message,
             integer, dimension(*), intent(in), optional ints,
             real(dp), dimension(*), intent(in), optional dbls )
```

References tools::dp, left_justify(), wordmatch(), and xm_inherit().

Referenced by calculate_vsvb_energy(), demgs_opt(), first_order_opt(), minimize_energy(), spin_opt(), valence_↩ init::valence_initialize(), xm_abort(), and xm_propagate().

Here is the call graph for this function:



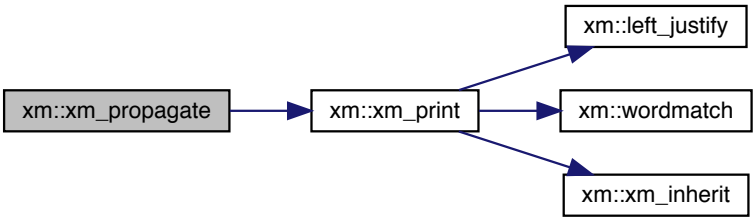
4.7.1.15 xm_propagate()

```
subroutine xm::xm_propagate (
            integer, intent(in), optional comm )
```

References timing_flops::count_determinants, tools::dp, timing_flops::initial_time, irank, timing_flops::kernel_time, nrank, valence_global_communicator, and xm_print().

Referenced by valence_init::valence_initialize().

Here is the call graph for this function:



4.7.1.16 xm_share()

```
subroutine xm::xm_share (
            character(1) chtype,
            integer, dimension(*) buff,
            integer len )
```

References valence_global_communicator.

4.7.2 Variable Documentation

4.7.2.1 irank

```
integer xm::irank
```

Referenced by schwarz_ints(), vsvb_energy(), wfndet(), write_determinant(), write_matrix(), and xm_propagate().

4.7.2.2 nrank

```
integer xm::nrank
```

Referenced by calculate_vsvb_energy(), schwarz_ints(), vsvb_energy(), wfndet(), xm_end(), and xm_propagate().

4.7.2.3 valence_global_communicator

```
integer xm::valence_global_communicator
```

Referenced by xm_end(), xm_equalize(), xm_getdims(), xm_inherit(), xm_input(), xm_propagate(), and xm_share().

Chapter 5

File Documentation

5.1 src/givens.F90 File Reference

Functions/Subroutines

- subroutine givens (a, lda, n, tol)
- subroutine givens_single (a, lda, n, tol)
- subroutine givens_orig (a, lda, nord, tol)

5.1.1 Function/Subroutine Documentation

5.1.1.1 givens()

```
subroutine givens (
             real(dp), dimension( lda, *) a,
             integer lda,
             integer n,
             real(dp) tol )
```

References tools::dp.

Referenced by givdr().

5.1.1.2 givens_orig()

```
subroutine givens_orig (
             real(dp), dimension( lda, *) a,
             integer lda,
             integer nord,
             real(dp) tol )
```

References tools::dp.

5.1.1.3 givens_single()

```
subroutine givens_single (
             real(dp), dimension( lda, *) a,
             integer lda,
             integer n,
             real(dp) tol )
```

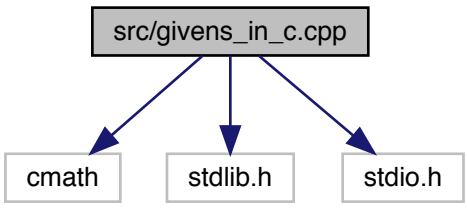
References tools::dp.

Referenced by givdr().

5.2 src/givens_in_c.cpp File Reference

```
#include <cmath>
#include <stdlib.h>
#include <stdio.h>
```

Include dependency graph for givens_in_c.cpp:



Functions

- void givensc_ (double *a, int *lda_in, int *n_in, double *tol_in)

5.2.1 Function Documentation

5.2.1.1 givensc_()

```
void givensc_ (
             double * a,
             int * lda_in,
             int * n_in,
             double * tol_in )
```

5.3 src/moduledensity.F90 File Reference

Modules

- module densitywork

Variables

- integer, dimension(2) densitywork::dme_b
- integer, dimension(2) densitywork::dme_k
- integer, dimension(:, :, :), allocatable densitywork::pair_sc
- real(dp), dimension(:), allocatable densitywork::coeff_sc
- integer, dimension(:), allocatable densitywork::bra_a

  holds the OLCAO
- integer, dimension(:), allocatable densitywork::bra_b

  holds the OLCAO
- integer, dimension(:), allocatable densitywork::ket_a

  same as bra_a, but with the ket
- integer, dimension(:), allocatable densitywork::ket_b

  same as bra_b, but with the ket
- integer, dimension(:), allocatable densitywork::bexch
- integer, dimension(:), allocatable densitywork::kexch
- integer, dimension(:), allocatable densitywork::bra
- integer, dimension(:), allocatable densitywork::ket
- real(dp), dimension(:, :), allocatable densitywork::wdet

  overlap integrals between all spin orbitals
- real(dp), dimension(:, :), allocatable densitywork::abra_npair

  holds the alpha part of the overlap matrix, stored as the overlap of the OLCAO associated with each alpha spin-coupled e- and every other spin orbital (npair, nelec)
- real(dp), dimension(:, :), allocatable densitywork::bbra_npair

  same as abra_npair, but for beta e-
- real(dp), dimension(:, :), allocatable densitywork::abra_docc_un

  same as abra_npair, but for the overlap of the OLCAO associated with each non-spin coupled alpha e- (ndocc+unpaired) and all spin-coupled e- (ndocc+nunpd, npair∗2)
- real(dp), dimension(:, :), allocatable densitywork::bbra_docc_un

  same as bbra_docc_un but for the beta e-
- real(dp), dimension(:, :), allocatable densitywork::aket

  holds the alpha part of the matrix to take the determinant of for the density
- real(dp), dimension(:, :), allocatable densitywork::bket

  same as aket, but the beta part
- real(dp), dimension(:, :), allocatable densitywork::aket_docc_un

  holds the docc and unpaired (non-spin coupled) part of aket this never changes during the run, and is copied back into aket for determinant
- real(dp), dimension(:, :), allocatable densitywork::bket_docc_un

  same as aket_docc_un, but for bket
- integer densitywork::padded_size
- integer, dimension(:), allocatable densitywork::ipvt

  holds the pivots for the call to dgetrf

5.4 src/moduleintegrals.F90 File Reference

Modules

- module integrals

Functions/Subroutines

- integer function integrals::shell_size (l_mom)

Variables

- integer integrals::nelec
- integer integrals::norbs
- integer integrals::nalpha
- integer integrals::nbeta
- integer integrals::hdim
- integer integrals::nstore
- integer integrals::max_obs

  largest possible number of basis functions in a OLCAO, based on the atoms specified in the OLCAO expansion
- integer integrals::natom

  The number of atoms/point charges in the geometry.
- integer integrals::npair

  Number of spin coupled electron/orbital PAIRS.
- integer integrals::ndocc

  Number of double-occupied (DOCC) orbitals.
- integer integrals::totlen

  Total length of the orbital weight list in wavefunction.
- integer integrals::nunpd

  Number of unpaired electrons/orbitals.
- integer integrals::ndf

  Number of derived basis functions (LCAO-type)
- integer integrals::nxorb

  Number of orbital excitations.
- integer integrals::natom_t

  The number of atom types.
- integer integrals::nspinc

  Number of spin couplings.
- integer integrals::nang

  Highest angular momentum in the basis set.
- integer integrals::xpmax

  Length of the largest orbital expansion.
- integer integrals::num_sh

  Number of unique atomic basis set shells.
- integer integrals::num_pr

  Number of unique atomic basis set primitives.
- integer integrals::nset
- integer integrals::mxctr
- integer integrals::ntol_e_min
- integer integrals::ntol_e_max

- integer integrals::max_iter
- integer, dimension(:, :), allocatable integrals::orbset
- integer, dimension(:), allocatable integrals::atom_t
- real(dp), dimension(:, :), allocatable integrals::coords
- integer, dimension(:), allocatable integrals::map_atom2shell

  holds indexing for the basis function shells for a given the atom type.
- integer, dimension(:), allocatable integrals::num_shell_atom

  holds the number of basis function shells for a given atom type
- integer, dimension(:), allocatable integrals::map_shell2prim

  holds indexing into the start of the primitive arrays for a given the total shell index.
- integer, dimension(:), allocatable integrals::ang_mom

  holds the shell angular momentum for a given shell index
- real(dp), dimension(:), allocatable integrals::nuc_charge
- real(dp), dimension(:), allocatable integrals::exponent

  holds the exponent for a primitive gaussian
- real(dp), dimension(:), allocatable integrals::con_coeff

  holds the contraction coefficient for a primitive gaussian
- integer, dimension(:), allocatable integrals::orbas_atnum

  holds the number of atoms whose basis sets make up a given OLCAO index
- integer, dimension(:, :), allocatable integrals::orbas_atset

  holds the overall atom index from the input file list for a given atom index in a OLCAO expansion and the OLCAO index
- integer, dimension(:), allocatable integrals::map_orbs

  holds the starting index for the set of basis functions that a given OLCAO is expanded in in the total OLCAO wavefunction (based just on how the orbitals are listed in order in the input file) can be used to index coeff() to loop through the basis function coefficients for a given orbital
- integer, dimension(:), allocatable integrals::xpset

  for a given index of a function in the OLCAO wavefunction, returns the index of the basis function in the list of all basis functions associated with the atoms that the OLCAO is expanded over
- integer, dimension(:), allocatable integrals::xorb
- integer, dimension(:), allocatable integrals::root
- real(dp), dimension(:), allocatable integrals::coeff

  holds the coefficient of a basis function in an OLCAO for a given basis function index in the total OLCAO wavefunction list (where the list is just based on the order the orbitals are listed in the input file)
- integer, dimension(:, :), allocatable integrals::nxyz

  powers of x,y,z coordinates for s, p, d shells in order
- real(dp), dimension(:), allocatable integrals::angn

  pnym(ij) = ashl(i) ∗ 1/(ashl(power of x)) ∗ 1/(ashl(power of y)) ∗ 1/(ashl((power of z)) for each primitive in each angular momentum i (where power of x + power of y + power of z = i). index ij walks over all primitives–1 for s, 3 for p, etc.
- real(dp), dimension(:), allocatable integrals::ashl

  sqrt∗((2∗i-1)!!) for each given angular momentum i=0,nang
- real(dp), dimension(:), allocatable integrals::ashi

  1/ashl(i) for each given angular momentum i=0,nang
- real(dp), dimension(:), allocatable integrals::dij
- real(dp), dimension(:), allocatable integrals::coeffi
- real(dp), dimension(:), allocatable integrals::coeffj
- integer, dimension(:, :), allocatable integrals::atom_ndf
- integer, dimension(:), allocatable integrals::ndf2orb
- integer, dimension(:), allocatable integrals::xpnew
- real(dp), dimension(:), allocatable integrals::dkl
- real(dp), dimension(:), allocatable integrals::coeffk
- real(dp), dimension(:), allocatable integrals::coeffl
- real(dp), dimension(:), allocatable integrals::schwarz

- real(dp), dimension(:), allocatable integrals::eribuf
- real(dp) integrals::ctol
- real(dp) integrals::dtol
- real(dp) integrals::itol
- logical integrals::spinopt
- logical integrals::store_eri
- logical integrals::eri_stored
- logical integrals::dem_gs
- real(dp) integrals::sint
- real(dp) integrals::hint
- real(dp) integrals::gint
- real(dp) integrals::enucrep
- real(dp), dimension(:, :), allocatable integrals::ham
- real(dp), dimension(:, :), allocatable integrals::ovl
- real(dp) integrals::ptbnmax
- real(dp) integrals::feather

5.5 src/moduletools.F90 File Reference

Modules

- module tools

Functions/Subroutines

- subroutine, public tools::angs2bohr (natom, coords)
- subroutine, public tools::get_nuclear_repulsion_energy (natom, atom_t, nuc_charge, coords, nre)

Variables

- integer, parameter tools::dp =kind(0.d0)

5.6 src/modulevalence_simint.F90 File Reference

5.7 src/timing_flops.F90 File Reference

Modules

- module timing_flops

Variables

- integer(8) timing_flops::flop
- integer(8) timing_flops::count_determinants
- real(dp) timing_flops::kernel_time
- real(dp) timing_flops::initial_time
- real(dp) timing_flops::guess_time
- real(dp) timing_flops::final_time

5.8 src/valence.F90 File Reference

Functions/Subroutines

- subroutine valence (energy)
- subroutine calculate_vsvb_energy (energy)
- subroutine guess_energy (energy)

  compute energy for system, without optimization
- subroutine demgs_opt (iorb, num_iter, cumulx, energy, etol, coefflock)
- subroutine first_order_opt (iorb, w, eig, v1, v2, energy)
- subroutine spin_opt (w, eig, v1, v2, energy)
- subroutine vsvb_energy (iorb, num_non_docc, num_spatial_orbs, energy, wfnorm, spinav, ijkl_symmetry_↩ is_enabled)

  compute energy integral and normalization integral for system
- subroutine wfndet

  computes the overlap integrals and stores in densitywork::wdet
- subroutine schwarz_ints (num_spatial_orbs, num_non_docc)

  compute integrals used in schwarz screening
- subroutine density (nord, erep_density, exchanged_erep_density, erep_int, exchanged_erep_int, calc_dens, calc_exchange_dens)

  computes the density for the given spin orbitals in dme_[bk]
- subroutine density_sc (nord, isc, jsc, erep_density, exchanged_erep_density, calc_dens, calc_exchange_↩ dens)

  computes the density for the given spin orbitals in dme_[bk] and between the isc'th and jsc'th spin coupling
- subroutine dbra (nord, nexb, nexk, dima, dimb, erep_density, exchanged_erep_density, isc, jsc, calc_dens, calc_exchange_dens)

  for the isc, jsc spin coupling pair, sets up mapping matrices to generate the 2^{N_p} density terms in the bra, and for each, calls dket (which the generates 2^{N_p} density terms for permutations in the ket)
- subroutine dket (nord, nexk, dima, dimb, erep_density, exchanged_erep_density, isc, jsc, calc_dens, calc_↩ exchange_dens)

  for the isc, jsc spin coupling pair, and the mapping already set up in bra_[ab], this routine sets up mapping matrices to generate the 2^{N_p} density terms in the ket, and calls det() to calculate the density.
- subroutine det (dima, dimb, nord, density, exchanged_density, calc_dens, calc_exchange_dens)

  perform spin integration and calculate density determinant
- subroutine givdr (max_n, n, adet, tol, d, ipvt)

  driver to calculate a determinant via givens rotations
- subroutine normal (ist, ind)
- subroutine ndf2obs (iorb, indf)
- subroutine norm_prim (ang_mom, con_length, exponent, con_coeff)

  normalize a shell of primitive GTO functions
- real(dp) function dblfac (n)
- subroutine cartesian (nang, nxyz)

  fills array nxyz with powers of x,y,z coordinates of primitive functions in order:
- subroutine setangn

  fills arrays with coefficients of primitive cartesian GTOs
- integer function indx (i, j)
- subroutine set_up_unpaired_docc

  Sets up the bra_[ab], ket_[ab], [ab]bra arrays for the docc and unpaired electrons since these arrays never change during a run. the goal is to set up the bra_[ab] and ket_[ab] contain indexes mapping alpha (beta) electrons to their spin functions indices in the bra and ket part of the wavefunction.
- subroutine build_abket (dima, dimb)

  builds the [ab]ket matrix using the overlaps in [ab]bra and the configurarion in ket_[ab], [ab]ket is formed, so the determinant can be calculated, and the density element computed

- subroutine check_spin_and_locate (spin_orb_in_bra, spin_orb_in_ket, bra_alpha, bra_beta, ket_alpha, ket_beta, nalpha, nbeta, orb_in_bra_alpha_index, orb_in_ket_alpha_index, orb_in_bra_beta_index, orb_in_ket_beta_index, both_are_alpha_spin, both_are_beta_spin)

  checks the spin of the given two spin orbitals (one in bra and one in ket) and returns logicals denoting the spin as well as locations of the spin orbitals in the bra and ket submatrices ([ab]ket of the full overlap matrix
- subroutine minimize_energy (energy, w, eig, v1, v2, coefflock, int_out, dbl_out)

  minimize energy

5.8.1 Function/Subroutine Documentation

5.8.1.1 build_abket()

```
subroutine build_abket (
             integer dima,
             integer dimb )
```

builds the [ab]ket matrix using the overlaps in [ab]bra and the configurarion in ket_[ab], [ab]ket is formed, so the determinant can be calculated, and the density element computed

Parameters

dima,dimb	[in]: number of alpha and beta electrons

References densitywork::abra_docc_un, densitywork::abra_npair, densitywork::aket, densitywork::aket_docc_un, densitywork::bbra_docc_un, densitywork::bbra_npair, densitywork::bket, densitywork::bket_docc_un, tools::dp, densitywork::ket_a, densitywork::ket_b, integrals::ndocc, integrals::npair, and integrals::nunpd.

Referenced by density(), and dket().

5.8.1.2 calculate_vsvb_energy()

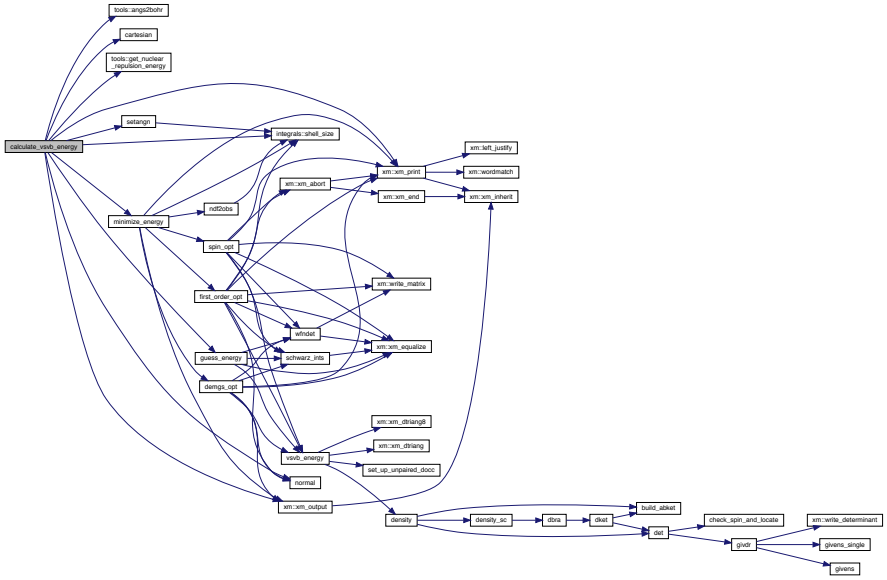
```
subroutine calculate_vsvb_energy (
             real(dp) energy )
```

References densitywork::abra_docc_un, densitywork::abra_npair, densitywork::aket, densitywork::aket_docc_un, integrals::ang_mom, integrals::angn, tools::angs2bohr(), integrals::ashi, integrals::ashl, integrals::atom_ndf, integrals::atom_t, densitywork::bbra_docc_un, densitywork::bbra_npair, densitywork::bexch, densitywork::bket, densitywork::bket_docc_un, densitywork::bra, densitywork::bra_a, densitywork::bra_b, cartesian(), integrals::coeffi, integrals::coeffj, integrals::coeffk, integrals::coeffl, integrals::coords, integrals::dem_gs, integrals::dij, integrals::dkl, tools::dp, integrals::enucrep, integrals::eribuf, tools::get_nuclear_repulsion_energy(), guess_energy(), timing_flops::guess_time, integrals::ham, integrals::hdim, densitywork::ipvt, densitywork::ket, densitywork::ket_a, densitywork::ket_b, densitywork::kexch, integrals::map_atom2shell, integrals::max_iter, integrals::max_obs, minimize_energy(), integrals::mxctr, integrals::nalpha, integrals::nang, integrals::natom, integrals::nbeta, integrals::ndf, integrals::ndf2orb, integrals::ndocc, integrals::nelec, integrals::norbs, normal(), integrals::npair, xm::nrank, integrals::nspinc, integrals::nstore, integrals::ntol_e_max, integrals::nuc_charge, integrals::num_shell_atom, integrals::nunpd, integrals::nxorb, integrals::nxyz, integrals::orbas_atnum, integrals::orbas_atset,

integrals::ovl, densitywork::padded_size, integrals::ptbnmax, integrals::schwarz, setangn(), integrals::shell_size(), densitywork::wdet, xm::xm_output(), xm::xm_print(), integrals::xpmax, and integrals::xpnew.

Referenced by calcsurface(), valence(), and valence_driver().

Here is the call graph for this function:



5.8.1.3 cartesian()

```
subroutine cartesian (
             integer nang,
             integer, dimension(3,*) nxyz )
```

fills array nxyz with powers of x,y,z coordinates of primitive functions in order:

nxyz(1-3, 1) = 0 (since x, y, z powers are 0 for the first primitive s shell.

nxyz(1-3, 2-4) = 1,0,0 then 0,1,0, then 0,0,1 since the power of x,y,z are each one for the 2-4 primitive functions

Parameters

nang	[in]: highest angular momentum in the basis set
nxyz	[in]: output array, filled

Referenced by calculate_vsvb_energy().

5.8.1.4 check_spin_and_locate()

```
subroutine check_spin_and_locate (
             integer spin_orb_in_bra,
             integer spin_orb_in_ket,
             integer, dimension(*) bra_alpha,
             integer, dimension(*) bra_beta,
             integer, dimension(*) ket_alpha,
             integer, dimension(*) ket_beta,
             integer nalpha,
             integer nbeta,
             integer orb_in_bra_alpha_index,
             integer orb_in_ket_alpha_index,
             integer orb_in_bra_beta_index,
             integer orb_in_ket_beta_index,
             logical both_are_alpha_spin,
             logical both_are_beta_spin )
```

checks the spin of the given two spin orbitals (one in bra and one in ket) and returns logicals denoting the spin as well as locations of the spin orbitals in the bra and ket submatrices ([ab]ket of the full overlap matrix

this routine requires bra_[ab] and ket_[ab] to be filled properly.

Parameters

spin_orb_in_bra,spin_orb_in_ket	[in] : indexes for a spin orbital in the bra and another in the ket
bra_alpha,bra_beta,ket_alpha,ket_beta	[in] : mapping arrays that take in an alpha/beta electron and return the location of the electron in the full bra/ket wavefunction.
nalpha,nbeta	[in] : number of alpha and beta electrons
orb_in_bra_alpha_index	[out] : if spin_orb_in_bra has alpha spin, this holds the position of spin_orb_in_bra in the alpha submatrix of the overlap

Referenced by det().

5.8.1.5 dblfac()

```
real(dp) function dblfac (
             integer n )
```

References tools::dp.

5.8.1.6 dbra()

```
subroutine dbra (
             integer nord,
             integer nexb,
             integer nexk,
             integer dima,
```

```
            integer dimb,
            real(dp) erep_density,
            real(dp) exchanged_erep_density,
            integer isc,
            integer jsc,
            logical calc_dens,
            logical calc_exchange_dens )
```

for the isc, jsc spin coupling pair, sets up mapping matrices to generate the 2^{N_p} density terms in the bra, and for each, calls dket (which the generates 2^{N_p} density terms for permutations in the ket)

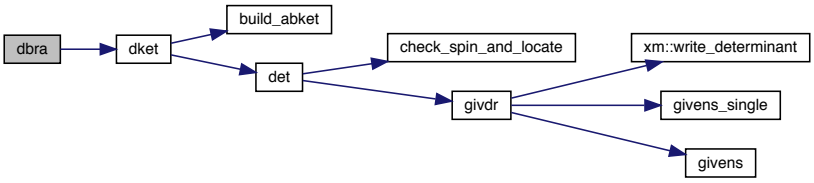
Parameters

nord	[in]: 1 if it's a 1e density, 2 if it's a 2e density
nexb,nexk	[in]: number of spin coupled pairs in the bra and ket
dima,dimb	[in]: number of alpha and beta electrons
erep_density,exchanged_erep_density	[in/out] density cofactors. if nord == 1, only erep_density has output
isc,jsc	[in]: spin couplings, only meaningful if nspinc>0
calc_dens,calc_exchange_dens	[in] : logical flags controlling whether or not to calculate the density or exchanged density. (so for nord == 1, calc_exchange_dens should be false)

References densitywork::abra_npair, densitywork::bbra_npair, densitywork::bexch, densitywork::bra_a, densitywork::bra_b, dket(), tools::dp, integrals::nelec, integrals::npair, densitywork::pair_sc, and densitywork::wdet.

Referenced by density_sc().

Here is the call graph for this function:



5.8.1.7 demgs_opt()

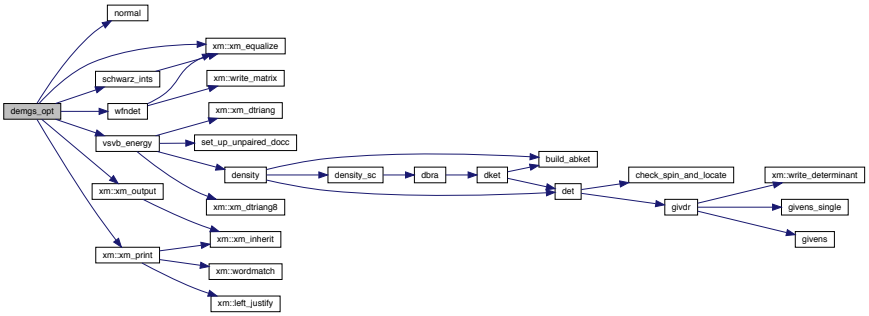
```
subroutine demgs_opt (
            integer iorb,
            integer num_iter,
            real(dp) cumulx,
            real(dp), dimension(1) energy,
            real(dp) etol,
            logical, dimension(*) coefflock )
```

References densitywork::bra, integrals::coeff, tools::dp, integrals::enucrep, integrals::eri_stored, integrals::feather, densitywork::ket, integrals::map_orbs, integrals::ndocc, integrals::nelec, normal(), integrals::npair, integrals::nunpd,

integrals::ptbnmax, schwarz_ints(), integrals::sint, integrals::store_eri, vsvb_energy(), densitywork::wdet, wfndet(), xm::xm_equalize(), xm::xm_output(), and xm::xm_print().

Referenced by minimize_energy().

Here is the call graph for this function:



5.8.1.8 density()

```
subroutine density (
             integer nord,
             real(dp) erep_density,
             real(dp) exchanged_erep_density,
             real(dp) erep_int,
             real(dp) exchanged_erep_int,
             logical calc_dens,
             logical calc_exchange_dens )
```

computes the density for the given spin orbitals in dme_[bk]

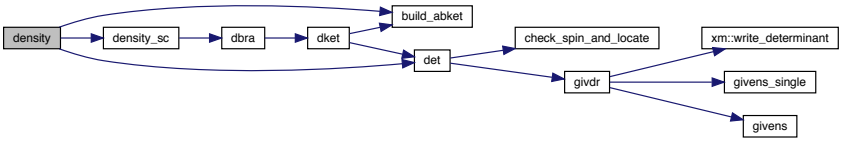
Parameters

nord	[in]: 1 if it's a 1e density, 2 if it's a 2e density
erep_density,exchanged_erep_density	[out]: density cofactors. if nord == 1, only erep_density has output
calc_dens,calc_exchange_dens	[in] : logical flags controlling whether or not to calculate the density or exchanged density. (so for nord == 1, calc_exchange_dens should be false)

References build_abket(), densitywork::coeff_sc, density_sc(), det(), tools::dp, integrals::ham, integrals::hint, integrals::ndocc, integrals::npair, integrals::nspinc, integrals::nunpd, integrals::ovl, integrals::sint, and integrals↩ ::spinopt.

Referenced by vsvb_energy().

Here is the call graph for this function:



5.8.1.9 density_sc()

```
subroutine density_sc (
             integer nord,
             integer isc,
             integer jsc,
             real(dp) erep_density,
             real(dp) exchanged_erep_density,
             logical calc_dens,
             logical calc_exchange_dens )
```

computes the density for the given spin orbitals in dme_[bk] and between the isc'th and jsc'th spin coupling

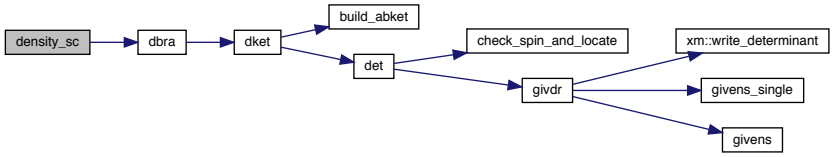
Parameters

nord	[in]: 1 if it's a 1e density, 2 if it's a 2e density
isc,jsc	[in]: spin couplings, only meaningful if nspinc>0
erep_density,exchanged_erep_density	[out]: density cofactors. if nord == 1, only erep_density has output
calc_dens,calc_exchange_dens	[in] : logical flags controlling whether or not to calculate the density or exchanged density. (so for nord == 1, calc_exchange_dens should be false)

References dbra(), tools::dp, integrals::ndocc, integrals::npair, and integrals::nunpd.

Referenced by density().

Here is the call graph for this function:



5.8.1.10 det()

```
subroutine det (
             integer dima,
             integer dimb,
             integer nord,
             real(dp) density,
             real(dp) exchanged_density,
             logical calc_dens,
             logical calc_exchange_dens )
```

perform spin integration and calculate density determinant

if nord == 1, returns first order cofactor d^1_{dme_b(1),dme_k(1)} w_{dme_b(1),dme_k(1)} in density

If nord == 2, returns second order cofactor d^2_{dme_b(1),dme_k(1),dme_b(2),dme_k(2)} w_{dme_b(1),dme_↩ b(2),dme_k(1),dme_k(2)} in density and returns second order cofactor d^2_{dme_b(1),dme_k(1),dme_b(2),dme↩ _k(2)} w_{dme_b(1),dme_b(2),dme_k(2),dme_k(1)} in exchanged_density.

(the densities are equivalent to cofactors, since dme_b(2) $<$ dme_b (1) and dme_k(2) $<$ dme_k(1))

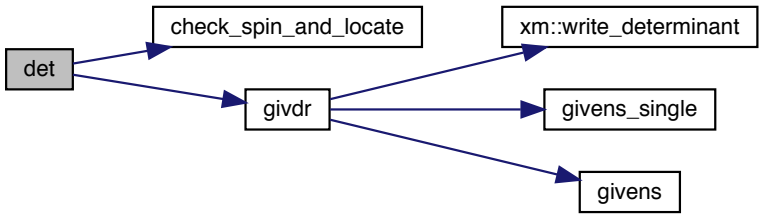
Parameters

dima,dimb	[in] : number of alpha and beta electrons
nord	[in] : 1 if it's a 1e density, 2 if it's a 2e density
density,exchanged_density	[in/out] : density cofactors if nord == 1, only density has output
calc_dens,calc_exchange_dens	[in] : logical flags controlling whether or not to calculate the density or exchanged density. (so for nord == 1, calc_exchange_dens should be false)

References densitywork::aket, densitywork::bket, densitywork::bra_a, densitywork::bra_b, check_spin_↩ and_locate(), densitywork::dme_b, densitywork::dme_k, tools::dp, integrals::dtol, givdr(), densitywork::ipvt, densitywork::ket_a, densitywork::ket_b, integrals::nalpha, integrals::nbeta, and densitywork::padded_size.

Referenced by density(), and dket().

Here is the call graph for this function:



5.8.1.11 dket()

```
subroutine dket (
             integer nord,
             integer nexk,
             integer dima,
             integer dimb,
             real(dp) erep_density,
             real(dp) exchanged_erep_density,
             integer isc,
             integer jsc,
             logical calc_dens,
             logical calc_exchange_dens )
```

for the isc, jsc spin coupling pair, and the mapping already set up in bra_[ab], this routine sets up mapping matrices to generate the 2^{N_p} density terms in the ket, and calls det() to calculate the density.

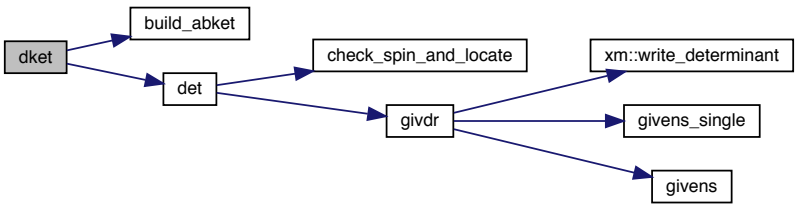
Parameters

nord	[in]: 1 if it's a 1e density, 2 if it's a 2e density
nexk	[in]: number of spin coupled pairs in the bra and ket
dima,dimb	[in]: number of alpha and beta electrons
erep_density,exchanged_erep_density	[in/out] density cofactors. if nord == 1, only erep_density has output
isc,jsc	[in]: spin couplings, only meaningful if nspinc>0
calc_dens,calc_exchange_dens	[in] : logical flags controlling whether or not to calculate the density or exchanged density. (so for nord == 1, calc_exchange_dens should be false)

References build_abket(), det(), tools::dp, densitywork::ket_a, densitywork::ket_b, densitywork::kexch, integrals::npair, and densitywork::pair_sc.

Referenced by dbra().

Here is the call graph for this function:



5.8.1.12 first_order_opt()

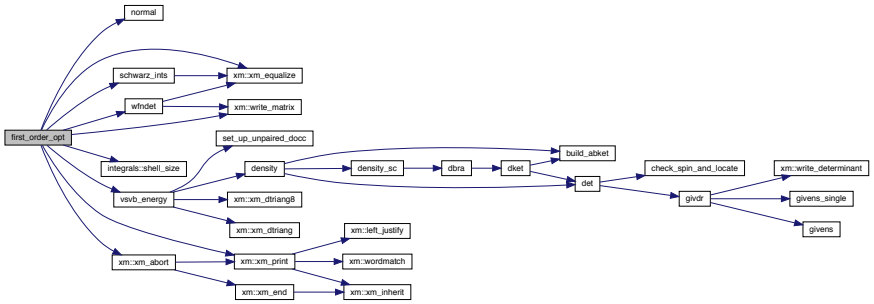
```
subroutine first_order_opt (
             integer iorb,
```

```
            real(dp), dimension( hdim, *) w,
            real(dp), dimension(*) eig,
            real(dp), dimension(*) v1,
            real(dp), dimension(*) v2,
            real(dp) energy )
```

References integrals::ang_mom, integrals::atom_t, densitywork::bra, integrals::coeff, integrals::con_coeff, integrals::coords, tools::dp, integrals::enucrep, integrals::eri_stored, integrals::exponent, integrals::ham, densitywork::ket, integrals::map_atom2shell, integrals::map_orbs, integrals::map_shell2prim, integrals::natom, integrals::natom_t, integrals::ndocc, integrals::nelec, integrals::norbs, normal(), integrals::npair, integrals::num_shell_atom, integrals::nunpd, integrals::nxorb, integrals::orbas_atnum, integrals::orbas_atset, integrals::ovl, integrals::root, schwarz_ints(), integrals::shell_size(), integrals::sint, integrals::store_eri, vsvb_energy(), densitywork::wdet, wfndet(), xm::write_matrix(), xm::xm_abort(), xm::xm_equalize(), xm::xm_print(), integrals::xorb, and integrals::xpset.

Referenced by minimize_energy().

Here is the call graph for this function:



5.8.1.13 givdr()

```
subroutine givdr (
            integer max_n,
            integer n,
            real(dp), dimension( max_n, *) adet,
            real(dp) tol,
            real(dp) d,
            integer, dimension(*) ipvt )
```

driver to calculate a determinant via givens rotations

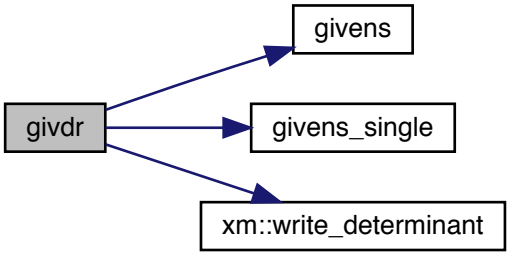
Parameters

max_n	[in]: sizing, in case memory doesn't match
n	[in]: size of adet
adet	[in]: matrix to calculate determinant of
tol	[in]: tolerance
d	[out]: determinant of adet

References timing_flops::count_determinants, tools::dp, givens(), givens_single(), timing_flops::kernel_time, and xm::write_determinant().

Referenced by det().

Here is the call graph for this function:



5.8.1.14 guess_energy()

```
subroutine guess_energy (
            real(dp), dimension(1) energy )
```

compute energy for system, without optimization

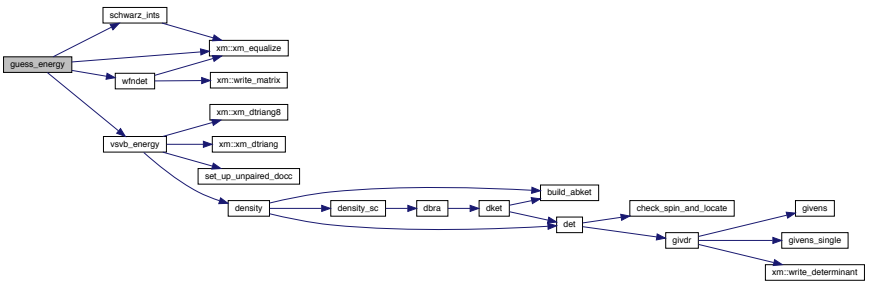
Parameters

energy	[out]: VSVB energy

References densitywork::bra, tools::dp, integrals::enucrep, densitywork::ket, integrals::ndocc, integrals::npair, integrals::nunpd, schwarz_ints(), integrals::store_eri, vsvb_energy(), wfndet(), and xm::xm_equalize().

Referenced by calculate_vsvb_energy().

Here is the call graph for this function:



5.8.1.15 indx()

```
integer function indx (
             integer i,
             integer j )
```

5.8.1.16 minimize_energy()

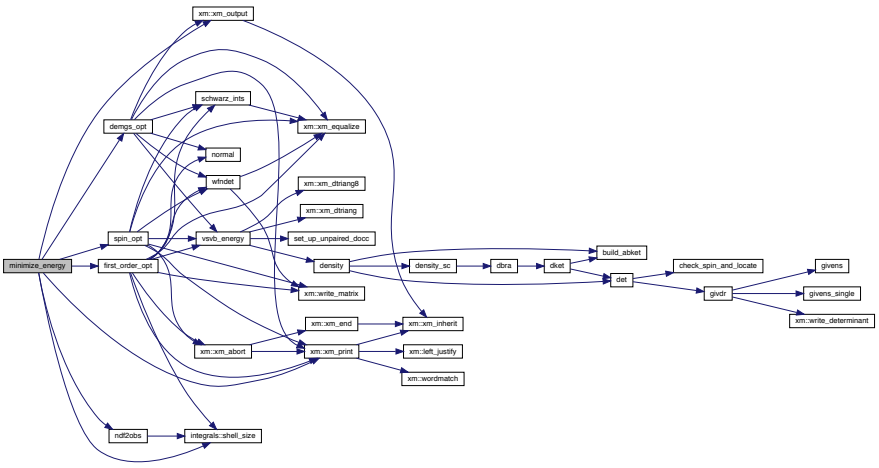
```
subroutine minimize_energy (
             real(dp) energy,
             real(dp), dimension( hdim, *) w,
             real(dp), dimension(*) eig,
             real(dp), dimension(*) v1,
             real(dp), dimension(*) v2,
             logical, dimension(*) coefflock,
             integer, dimension(2) int_out,
             real(dp), dimension(2) dbl_out )
```

minimize energy

References integrals::ang_mom, integrals::angn, integrals::atom_t, integrals::coeff, integrals::coeffi, integrals::coeffj, integrals::coeffk, integrals::coeffl, integrals::coords, integrals::dem_gs, demgs_opt(), integrals::dij, integrals::dkl, tools::dp, integrals::dtol, first_order_opt(), integrals::gint, integrals::hint, integrals::map_atom2shell, integrals::map_orbs, integrals::max_iter, integrals::max_obs, integrals::natom, ndf2obs(), integrals::norbs, integrals::nset, integrals::nspinc, integrals::ntol_e_max, integrals::ntol_e_min, integrals::nuc_charge, integrals::num_shell_atom, integrals::orbas_atnum, integrals::orbas_atset, integrals::orbset, integrals::shell_size(), integrals::sint, spin_opt(), xm::xm_output(), xm::xm_print(), integrals::xpnew, and integrals::xpset.

Referenced by calculate_vsvb_energy().

Here is the call graph for this function:



5.8.1.17 ndf2obs()

```
subroutine ndf2obs (
             integer iorb,
             integer indf )
```

References integrals::ang_mom, integrals::atom_ndf, integrals::atom_t, integrals::map_atom2shell, integrals::map_orbs, integrals::ndf2orb, integrals::num_shell_atom, integrals::orbas_atnum, integrals::orbas_atset, integrals::shell_size(), integrals::xpnew, and integrals::xpset.

Referenced by minimize_energy().

Here is the call graph for this function:

5.8.1.18 norm_prim()

```
subroutine norm_prim (
             integer ang_mom,
             integer con_length,
             real(dp), dimension(*) exponent,
             real(dp), dimension(*) con_coeff )
```

normalize a shell of primitive GTO functions

Parameters

ang_mom	[in] : angular momentum of the shell (and primitive GTOs)
con_length	[in] : number of primitives in the shell–dimension of exponent and con_coeff
exponent	[in] : array of exponents of primitive GTOs in the shell
con_coeff	[in/out] : array of coefficients of primitive GTOs in the shell

References tools::dp.

Referenced by valence_init::read_allocate_input().

5.8.1.19 normal()

```
subroutine normal (
             integer ist,
             integer ind )
```

References integrals::coeff, integrals::map_orbs, and integrals::sint.

Referenced by calculate_vsvb_energy(), demgs_opt(), and first_order_opt().

5.8.1.20 schwarz_ints()

```
subroutine schwarz_ints (
             integer num_spatial_orbs,
             integer num_non_docc )
```

compute integrals used in schwarz screening

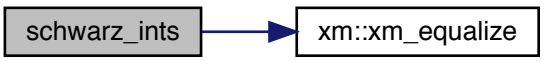
Parameters

num_spatial_orbs	[in]: number of "spatial" orbitals, not spin orbitals (2∗npair+nunpd+ndocc)
num_non_docc	[in]: number of non-docc orbitals (2∗npair+nunpd)

References densitywork::bra, tools::dp, integrals::gint, xm::irank, densitywork::ket, xm::nrank, integrals::schwarz, and xm::xm_equalize().

Referenced by demgs_opt(), first_order_opt(), guess_energy(), and spin_opt().

Here is the call graph for this function:



5.8.1.21 set_up_unpaired_docc()

```
subroutine set_up_unpaired_docc ( )
```

Sets up the bra_[ab], ket_[ab], [ab]bra arrays for the docc and unpaired electrons since these arrays never change during a run. the goal is to set up the bra_[ab] and ket_[ab] contain indexes mapping alpha (beta) electrons to their spin functions indices in the bra and ket part of the wavefunction.

References densitywork::abra_docc_un, densitywork::aket_docc_un, densitywork::bbra_docc_un, densitywork::bket_docc_un, densitywork::bra_a, densitywork::bra_b, densitywork::ket_a, densitywork::ket_b, integrals::ndocc, integrals::npair, integrals::nunpd, and densitywork::wdet.

Referenced by vsvb_energy().

5.8.1.22 setangn()

```
subroutine setangn ( )
```

fills arrays with coefficients of primitive cartesian GTOs

fills ashl(i) with sqrt∗((2∗i-1)!!) for each angular momentum i=0,nang

fills ashi(i) with 1/(ashl(i))

fills angn(ij) with ashl(i) ∗ 1/(ashl(power of x)) ∗ 1/(ashl(power of y)) ∗ 1/(ashl((power of z)) for each primitive in each angular momentum i,where (power of x) + (power of y) + (power of z) = i) and ij walks over all primitives–1 for s, 3 for p, etc.

References integrals::angn, integrals::ashi, integrals::ashl, tools::dp, integrals::nang, integrals::nxyz, and integrals::shell_size().

Referenced by calculate_vsvb_energy().

Here is the call graph for this function:

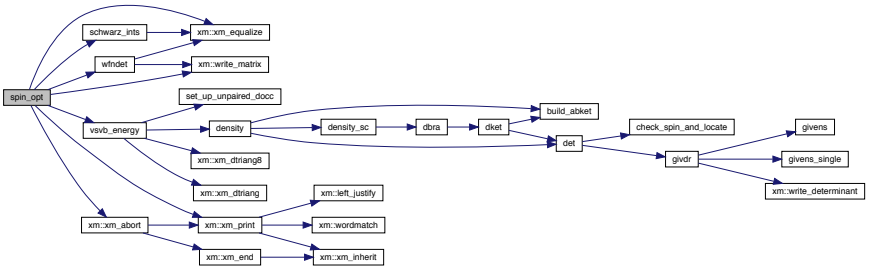
5.8.1.23 spin_opt()

```
subroutine spin_opt (
             real(dp), dimension( hdim, *) w,
             real(dp), dimension(*) eig,
             real(dp), dimension(*) v1,
             real(dp), dimension(*) v2,
             real(dp) energy )
```

References densitywork::bra, densitywork::coeff_sc, tools::dp, integrals::enucrep, integrals::ham, densitywork::ket, integrals::ndocc, integrals::nelec, integrals::npair, integrals::nspinc, integrals::nunpd, integrals::ovl, schwarz_ints(), integrals::spinopt, integrals::store_eri, vsvb_energy(), wfndet(), xm::write_matrix(), xm::xm_abort(), xm::xm_equalize(), and xm::xm_print().

Referenced by minimize_energy().

Here is the call graph for this function:

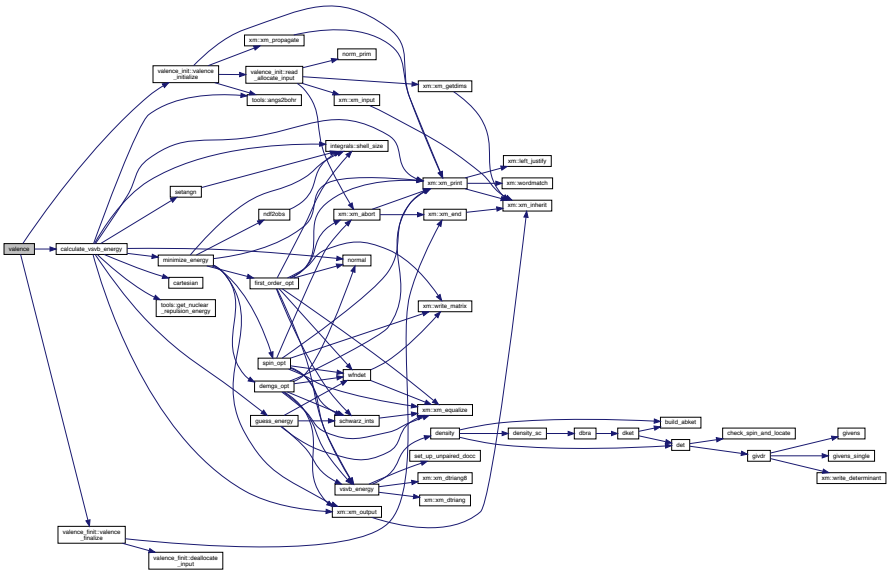


5.8.1.24 valence()

```
subroutine valence (
            real(dp) energy )
```

References calculate_vsvb_energy(), tools::dp, valence_finit::valence_finalize(), and valence_init::valence_↩ initialize().

Here is the call graph for this function:



5.8.1.25 vsvb_energy()

```
subroutine vsvb_energy (
             integer iorb,
             integer num_non_docc,
             integer num_spatial_orbs,
             real(dp) energy,
             real(dp) wfnorm,
             logical spinav,
             logical ijkl_symmetry_is_enabled )
```

compute energy integral and normalization integral for system

For nelec electrons, and a single determinant wavefunction (no spin couplings), the energy integral is $E = \sum_{ij}^{nelec}[d^1_{ij}w_{ij}h_{s,ij}]+\sum_i^{nelec}\sum_{j<i}\sum_k^{nelec}\sum_{l<k}[d^2_{ikjl}w_{ikjl}(< i(1)j(2)|k(1)l(2) >_s - < i(1)j(2)|l(1)k(2) >_s)]$

For N_p spin coupled pairs, and N_{sc} spin couplings, there are M_{sc} = N_{sc}^2 2^{2N_p} terms like the one above, which differ in spin functions:

$E = \sum_{ij}^{nelec}[(\sum_q^{M_{sc}} d^1_{q,ij}w_{q,ij})h_{s,ij}]+\sum_i^{nelec}\sum_{j<i}\sum_k^{nelec}\sum_{l<k}[(\sum_q^{M_{sc}} d^2_{q,ikjl}(w_{q,ikjl} < i(1)j(2)|k(1)l(2) >_s -w_{q,iljk} < i(1)j(2)|l(1)k(2) >_s)]$

The first term in the sum is the one electron term.

$h_{s,ij} =< i(1)|h_s|j(1) >$ where h is the standard one-electron electron kinetic and nuclei-electron attraction operator. the s subscript is meant to denote that this integral does not include spin integration (this is in the "w" variable) i,j are one-electron spin orbitals.

$d^1_{q,ij}$ is the first-order cofactor of the matrix of overlap integrals between the spin orbitals. That is, it is the determinant of the overlap matrix with row i and column j removed, multiplied by (-1)^{i+j} q denotes the term in the spin coupling/pairs expansion

$w_{q,ij}$ is the spin function integration, where the spin functions are alpha or beta, whichever are associated with spin orbital i(1) and j(1) q denotes the term in the spin coupling/pairs expansion

The second term in the sum is the two electron term.

$< i(1)j(2)|k(1)l(2) >_s$ is the electron-electron repulsion integral. i,j,k,l are one-electron spin orbitals. the s subscript is meant to denote that this integral does not include spin integration (this is in the "w" variable)

$d^2_{q,ikjl}$ is the second-order cofactor of the matrix of overlap integrals between the spin orbitals. That is, it is the determinant of the overlap matrix with row i, column k, row j, and column l removed, multiplied by (-1)^{i+j+k+l} q denotes the term in the spin coupling/pairs expansion

$w_{q,ijkl}$ is the spin function integration, where the spin functions are alpha or beta, whichever are associated with spin orbitals i(1),k(1),j(2),l(2). q denotes the term in the spin coupling/pairs expansion

The normalization integral is the determinant of the matrix of overlap integrals between spin orbitals. N = Determinant of spin orbital overlap matrix $= \sum_j^{nelec}[(\sum_q^{M_{sc}} d^1_{q,1j}w_{q,1j}) < 1(1)|j(1) >_s]$

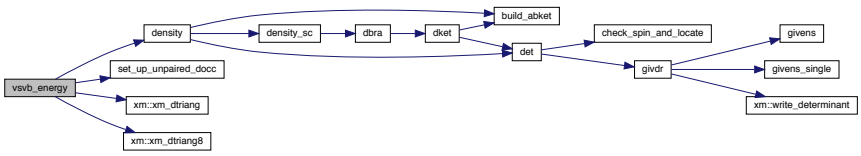
Parameters

iorb	[in]: perturbed orbital, if called from an optimization routine
num_non_docc	[in]: number of non-docc orbitals (2∗npair+nunpd)
num_spatial_orbs	[in]: number of "spatial" orbitals, not spin orbitals (2∗npair+nunpd+ndocc)
energy	[out]: VSVB energy
wfnorm	[out]: normalization term for the wavefunction
spinav	[in]: logical flag controlling whether or not to spin-average
ijkl_symmetry_is_enabled	[in]: logical flag controlling whether or not to to use kl < ij symmetry. this should be false for optimizations where bra != ket

References densitywork::bra, integrals::dem_gs, density(), densitywork::dme_b, densitywork::dme_k, tools↩ ::dp, integrals::eri_stored, integrals::eribuf, integrals::gint, integrals::hint, xm::irank, integrals::itol, densitywork↩ ::ket, integrals::nelec, integrals::npair, xm::nrank, integrals::nstore, integrals::nunpd, integrals::schwarz, set_up_↩ unpaired_docc(), integrals::sint, integrals::store_eri, xm::xm_dtriang(), and xm::xm_dtriang8().

Referenced by demgs_opt(), first_order_opt(), guess_energy(), and spin_opt().

Here is the call graph for this function:



5.8.1.26 wfndet()

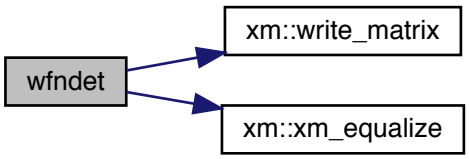
```
subroutine wfndet ( )
```

computes the overlap integrals and stores in densitywork::wdet

References densitywork::bra, tools::dp, xm::irank, densitywork::ket, integrals::nelec, xm::nrank, integrals::sint, densitywork::wdet, xm::write_matrix(), and xm::xm_equalize().

Referenced by demgs_opt(), first_order_opt(), guess_energy(), and spin_opt().

Here is the call graph for this function:



5.9 src/valence_api.F90 File Reference

Functions/Subroutines

- subroutine init (info)

  initializes VSVB info
- subroutine getn (n)

  returns the number of atoms
- subroutine calcsurface (x, v)

  returns the VSVB energy for given coordinates
- subroutine finalize

5.9.1 Function/Subroutine Documentation

5.9.1.1 calcsurface()

```
subroutine calcsurface (
             real(dp), dimension(*) x,
             real(dp) v )
```

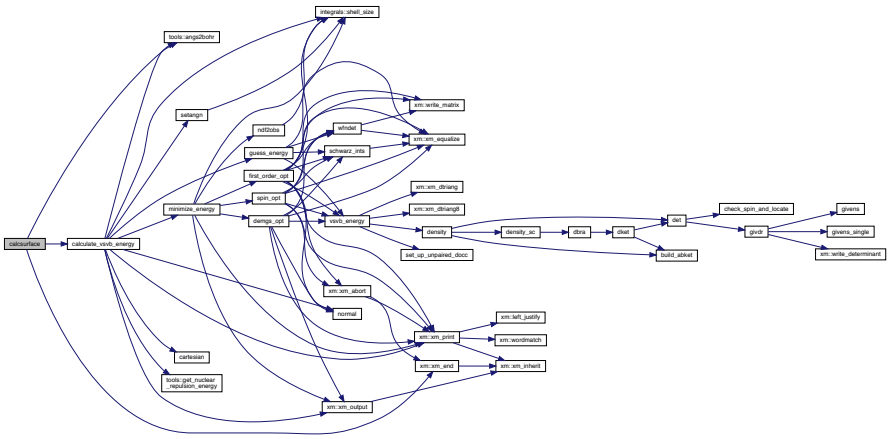
returns the VSVB energy for given coordinates

Parameters

x	[in] : cartesian coordinates to get the VSVB energy of (angstroms)
v	[out] : VSVB energy for input coordinates (cm^{-1})

References tools::angs2bohr(), calculate_vsvb_energy(), integrals::coords, tools::dp, integrals::natom, and xm↩ ::xm_end().

Here is the call graph for this function:

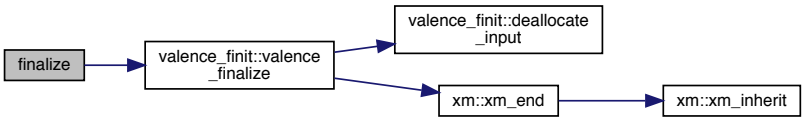


5.9.1.2 finalize()

```
subroutine finalize ( )
```

References valence_finit::valence_finalize().

Here is the call graph for this function:



5.9.1.3 getn()

```
subroutine getn (
             integer n )
```

returns the number of atoms

Parameters

n	[out] : number of atoms

References tools::dp, and integrals::natom.

5.9.1.4 init()

```
subroutine init (
             integer info )
```

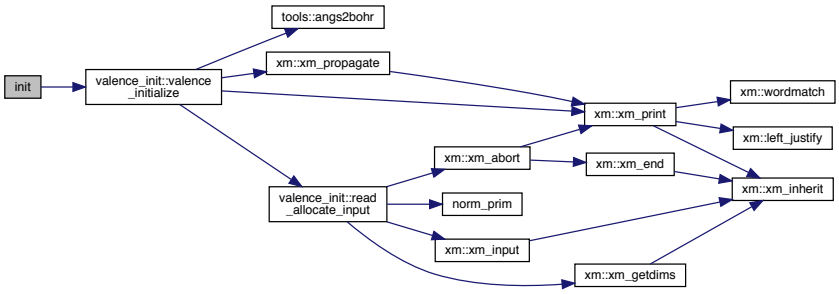
initializes VSVB info

Parameters

info	[out] : set to 0 if initialization occurs successfully

References tools::dp, and valence_init::valence_initialize().

Here is the call graph for this function:



5.10 src/valence_driver.F90 File Reference

Functions/Subroutines

- program valence_driver

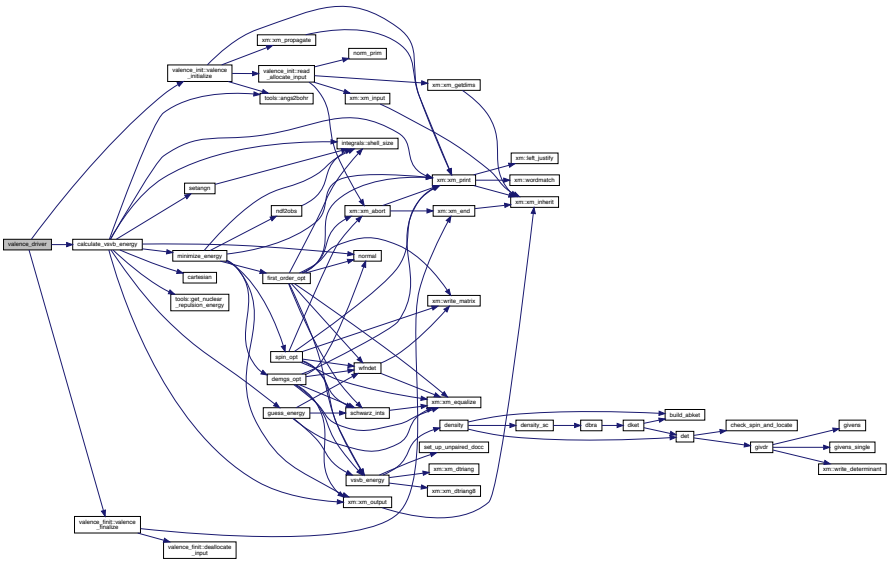
5.10.1 Function/Subroutine Documentation

5.10.1.1 valence_driver()

```
program valence_driver ( )
```

References calculate_vsvb_energy(), valence_finit::valence_finalize(), and valence_init::valence_initialize().

Here is the call graph for this function:



5.11 src/valence_finalize.F90 File Reference

Modules

- module valence_finit

Functions/Subroutines

- subroutine valence_finit::valence_finalize (comm)
- subroutine valence_finit::deallocate_input

5.12 src/valence_initialize.F90 File Reference

Modules

- module valence_init

Functions/Subroutines

- subroutine valence_init::valence_initialize (comm)
- subroutine valence_init::read_allocate_input

5.13 src/xm.F90 File Reference

Modules

- module xm

Functions/Subroutines

- subroutine xm::xm_getdims (natom, natom_t, npair, nunpd, ndocc, totlen, xpmax, nspinc, num_sh, num_pr, nang, ndf, nset, nxorb, mxctr)
- subroutine xm::xm_input (ntol_c, ntol_e_min_in, ntol_e_max_in, ntol_d, ntol_i, orbset_in, max_iter_in, mxctr_in)
- logical function xm::wordmatch (word1, word2, nchar)
- subroutine xm::xm_print (mode, message, ints, dbls)
- subroutine xm::left_justify (sentence, outbuf, maxlen)
- subroutine xm::xm_output (mode, energy, etol)
- subroutine xm::xm_propagate (comm)
- subroutine xm::xm_end (comm)
- subroutine xm::xm_share (chtype, buff, len)
- subroutine xm::xm_equalize (buff, datalen)
- subroutine xm::xm_inherit (num_proc, myrank, master)
- subroutine xm::xm_abort (error_message)
- subroutine xm::xm_dtriang (ij, i, j)
- subroutine xm::xm_dtriang8 (ij, i, j)
- subroutine xm::write_matrix (adet, max_n, n, filename)
- subroutine xm::write_determinant (d)

Variables

- integer xm::valence_global_communicator
- integer xm::nrank
- integer xm::irank