Contents

1 Routine/Function Prologues

1.0.1 getevecfv (Source File: getevecfv.f90)

INTERFACE:

```
Subroutine getevecfv (vpl, vgpl, evecfv)
```

USES:

```
Use modmain
Use modinput
Use modmpi
```

DESCRIPTION:

The file where the (first-variational) eigenvectors are stored is `EVECFV.OUT`. It is a direct-access binary file, the record length of which can be determined with the help of the array sizes and data type information. One record of this file corresponds to one k-point in the irreducible Brillouin zone and has the following structure

k_{lat}	N_{mat}	N_{stfv}	N_{spfv}	Φ

The following table explains the parts of the record in more detail

name	type	shape	description
k_{lat}	real(8)	3	k-point in lattice coordinates
N_{mat}	integer	1	(L)APW basis size including local orbitals (maximum over k-points)
N_{stfv}	integer	1	number of (first-variational) states (without core states)
N_{spfv}	integer	1	first-variational spins (2 for spin-spirals, 1 otherwise)
Φ	complex(8)	$N_{\mathrm{mat}} \times N_{\mathrm{stfv}} \times N_{\mathrm{spfv}}$	(first-variational) eigenvector array

REVISION HISTORY:

```
Created Feburary 2007 (JKD)
Fixed transformation error, October 2007 (JKD, Anton Kozhevnikov)
Documentation added, Dec 2009 (S. Sagmeister)
Fixed l.o. rotation, June 2010 (A. Kozhevnikov)
```

1.0.2 getngkmax (Source File: getngkmax.f90)

INTERFACE:

```
Subroutine getngkmax
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Determines the largest number of $\mathbf{G}+\mathbf{k}$-vectors with length less than `gkmax` over all the k-points and stores it in the global variable `ngkmax`. This variable is used for allocating arrays.

REVISION HISTORY:

```
Created October 2004 (JKD)
```

1.0.3 init1 (Source File: init1.f90)

INTERFACE:

```
Subroutine init1
```

USES:

```
      Use modinput
      Use modmain
#ifdef TETRA
      Use modtetra
#endif
#ifdef XS
      Use modxs
#endif
```

DESCRIPTION:

Generates the k-point set and then allocates and initialises global variables which depend on the k-point set.

REVISION HISTORY:

```
Created January 2004 (JKD)
```

1.0.4 zpotcoul (Source File: zpotcoul.f90)

INTERFACE:

```
Subroutine zpotcoul (nr, nrmax, ld, r, igp0, gpc, jlgpr, ylmgp, sfacgp, &
& zn, zrhomt, zrhoir, zvclmt, zvclir, zrho0)
      Use modinput
```

USES:

```
      Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
nr     : number of radial points for each species (in,integer(nspecies))
nrmax  : maximum nr over all species (in,integer)
ld     : leading dimension of r (in,integer)
r      : radial mesh for each species (in,real(ld,nspecies))
igp0   : index of the shortest G+p-vector (in,integer)
gpc    : G+p-vector lengths (in,real(ngvec))
jlgpr  : spherical Bessel functions for evergy G+p-vector and muffin-tin
         radius (in,real(0:lmaxvr+npsden+1,ngvec,nspecies))
ylmgp  : spherical harmonics of the G+p-vectors (in,complex(lmmaxvr,ngvec))
sfacgp : structure factors of the G+p-vectors (in,complex(ngvec,natmtot))
zn     : nuclear charges at the atomic centers (in,real(nspecies))
zrhomt : muffin-tin charge density (in,complex(lmmaxvr,nrmax,natmtot))
zrhoir : interstitial charge density (in,complex(ngrtot))
zvclmt : muffin-tin Coulomb potential (out,complex(lmmaxvr,nrmax,natmtot))
zvclir : interstitial Coulomb potential (out,complex(ngrtot))
zrho0  : G+p=0 term of the pseudocharge density (out,complex)
```

DESCRIPTION:

Calculates the Coulomb potential of a complex charge density by solving Poisson's equation. First, the multipole moments of the muffin-tin charge are determined for the jth atom of the ith species by

$$q^{\rm MT}_{ij;lm} = \int_0^{R_i} r^{l+2}\rho_{ij;lm}(r)dr + z_{ij}Y_{00}\,\delta_{l,0}\ ,$$

where R_i is the muffin-tin radius and z_{ij} is a point charge located at the atom center (usually the nuclear charge, which should be taken as **negative**). Next, the multipole moments of the continuation of the interstitial density, $\rho^{\rm I}$, into the muffin-tin are found with

$$q^{\rm I}_{ij;lm} = 4\pi i^l R_i^{l+3}\sum_{\mathbf{G}}\frac{j_{l+1}(GR_i)}{GR_i}\rho^{\rm I}(\mathbf{G})\exp(i\mathbf{G}\cdot\mathbf{r}_{ij})Y^*_{lm}(\hat{\mathbf{G}}),$$

remembering that

$$\lim_{x\to 0}\frac{j_{l+n}(x)}{x^n} = \frac{1}{(2n+1)!!}\delta_{l,0}$$

should be used for the case $\mathbf{G}=0$. A pseudocharge is now constructed which is equal to the real density in the interstitial region and whose multipoles are the difference between the real and interstitial muffin-tin multipoles. This pseudocharge density is smooth in the sense that it can be expanded in terms of the finite set of $\mathbf{G}$-vectors. In each muffin-tin the pseudocharge has the form

$$\rho^{\rm P}_{ij}(\mathbf{r}) = \rho^{\rm I}(\mathbf{r}-\mathbf{r}_{ij}) + \sum_{lm}\rho^{\rm P}_{ij;lm}\frac{1}{R_i^{l+3}}\left(\frac{r}{R_i}\right)^l\left(1-\frac{r^2}{R_i^2}\right)^{N_i}Y_{lm}(\hat{\mathbf{r}})$$

where

$$\rho^{\rm P}_{ij;lm} = \frac{(2l+2N_i+3)!!}{2_i^N N_i!(2l+1)!!}\left(q^{\rm MT}_{ij;lm}-q^{\rm I}_{ij;lm}\right)$$

and $N_i \approx \frac{1}{2} R_i G_{\max}$ is generally a good choice. The pseudocharge in reciprocal-space is given by

$$\rho^{\rm P}(\mathbf{G}) = \rho^{\rm I}(\mathbf{G}) + \sum_{ij;lm} 2^{N_i} N_i! \frac{4\pi(-i)^l}{\Omega R_i^l} \frac{j_{l+N_i+1}(GR_i)}{(GR_i)^{N_i+1}} \rho^{\rm P}_{ij;lm} \exp(-i\mathbf{G}\cdot\mathbf{r}_{ij}) Y_{lm}(\hat{\mathbf{G}})$$

which may be used for solving Poisson's equation directly

$$V^{\rm P}(\mathbf{G}) = \begin{cases} 4\pi \frac{\rho^{\rm P}(\mathbf{G})}{G^2} & G > 0 \\ 0 & G = 0 \end{cases}.$$

The usual Green's function approach is then employed to determine the potential in the muffin-tin sphere due to charge in the sphere. In other words

$$V^{\rm MT}_{ij;lm}(r) = \frac{4\pi}{2l+1} \left(\frac{1}{r^{l+1}} \int_0^r \rho^{\rm MT}_{ij;lm}(r') r'^{l+2} dr' + r^l \int_r^{R_i} \frac{\rho^{\rm MT}_{ij;lm}(r')}{r'^{l-1}} dr' \right) + \frac{1}{Y_{00}} \frac{z_{ij}}{r} \delta_{l,0}$$

where the last term is the monopole arising from the point charge. All that remains is to add the homogenous solution of Poisson's equation,

$$V^{\rm H}_{ij}(\mathbf{r}) = \sum_{lm} V^{\rm H}_{ij;lm} \left(\frac{r}{R_i} \right)^l Y_{lm}(\hat{\mathbf{r}}),$$

to the muffin-tin potential so that it is continuous at the muffin-tin boundary. Therefore the coefficients, $\rho^{\rm H}_{ij;lm}$, are given by

$$V^{\rm H}_{ij;lm} = 4\pi i^l \sum_{\mathbf{G}} j_l(Gr) V^{\rm P}(\mathbf{G}) \exp(i\mathbf{G}\cdot\mathbf{r}_{ij}) Y^*_{lm}(\hat{\mathbf{G}}) - V^{\rm MT}_{ij;lm}(R_i).$$

Finally note that the **G**-vectors passed to the routine can represent vectors with a non-zero offset, $\mathbf{G}+\mathbf{p}$ say, which is required for calculating Coulomb matrix elements.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

1.0.5 getevalsv (Source File: getevalsv.f90)

INTERFACE:

```
Subroutine getevalsv (vpl, evalsvp)
```

USES:

```
Use modmain
Use modinput
Use modmpi
```

DESCRIPTION:

The file where the (second-variational) eigenvalues are stored is `EVALSV.OUT`. It is a direct-access binary file, the record length of which can be determined with the help of the array sizes and data type information. One record of this file has the following structure

k_{lat}	N_{stsv}	E

The following table explains the parts of the record in more detail

name	type	shape	description
k_{lat}	real(8)	3	k-point in lattice coordinates
N_{stsv}	integer	1	number of (second-variational) states (without core states)
E	real(8)	N_{stsv}	(second-variational) eigenvalue array

REVISION HISTORY:

```
Documentation added, Dec 2009 (S. Sagmeister)
```

1.0.6 findsymlat (Source File: findsymlat.f90)

Subroutine findsymlat **USES:**

```
Use modinput
Use modmain
```

DESCRIPTION:

Finds the point group symmetries which leave the Bravais lattice invariant. Let A be the matrix consisting of the lattice vectors in columns, then

$$g = A^{\mathrm{T}} A$$

is the metric tensor. Any 3×3 matrix S with elements -1, 0 or 1 is a point group symmetry of the lattice if $\det(S)$ is -1 or 1, and

$$S^{\mathrm{T}} g S = g.$$

The first matrix in the set returned is the identity.

REVISION HISTORY:

```
Created January 2003 (JKD)
Removed arguments and simplified, April 2007 (JKD)
```

1.0.7 gencfun (Source File: gencfun.f90)

INTERFACE:

```
Subroutine gencfun
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Generates the smooth characteristic function. This is the function which is 0 within the muffin-tins and 1 in the intersitial region and is constructed from radial step function form-factors with $G < G_{\max}$. The form factors are given by

$$\tilde{\Theta}_i(G) = \begin{cases} \frac{4\pi R_i^3}{3\Omega} & G = 0 \\ \frac{4\pi R_i^3}{\Omega} \frac{j_1(GR_i)}{GR_i} & 0 < G \le G_{\max} \\ 0 & G > G_{\max} \end{cases},$$

where R_i is the muffin-tin radius of the ith species and Ω is the unit cell volume. Therefore the characteristic function in G-space is

$$\tilde{\Theta}(\mathbf{G}) = \delta_{G,0} - \sum_{ij} \exp(-i\mathbf{G} \cdot \mathbf{r}_{ij}) \tilde{\Theta}_i(G),$$

where $\mathbf{r}_{ij}$ is the position of the jth atom of the ith species.

REVISION HISTORY:

```
Created January 2003 (JKD)
```

1.0.8 create_plotlablels (Source File: mod_plotlabels.f90)

INTERFACE:

```
function create_plotlablels(title,filename,dimension) result (thisplotlabels)
```

DESCRIPTION:

This function creates a plotlabels type and initializes the axis descriptions

INPUT/OUTPUT PARAMETERS:

```
title     :  string for plot title (in, character(512))
dimension :  number of dimensions. e.g 1d plot has one dimension but 2 axes (in, integer)
create_plotlablels : pointer to plotlabels type  (out, type(plotlabels))
```

REVISION HISTORY:

```
Created April 2007 (JKD)
```

1.0.9 create_plotlablels (Source File: mod_plotlabels.f90)

INTERFACE:

```
subroutine set_plotlabel_axis(plotlabels_,axis,label,latexunit,graceunit)
```

DESCRIPTION:

This subroutine sets the axis labels for dinmension axis

INPUT/OUTPUT PARAMETERS:

```
plotlabels_  :  plotlabels type of which an axis schould be labeled (inout, type(plotlabel
axis :  number of the axis to set (in,integer)
label: (in,character*)
unit : (in,character*)
```

REVISION HISTORY:

```
Created April 2007 (JKD)
```

1.0.10 writeiad (Source File: writeiad.f90)

INTERFACE:

```
Subroutine writeiad (topt)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
topt : if .true. then the filename will be {\tt IADIST_OPT.OUT}, otherwise
       {\tt IADIST.OUT} (in,logical)
```

DESCRIPTION:

Outputs the interatomic distances to file.

REVISION HISTORY:

```
Created May 2005 (JKD)
```

1.0.11 writeeval (Source File: writeeval.f90)

INTERFACE:

```
Subroutine writeeval
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Outputs the second-variational eigenvalues and occupation numbers to the file `EIGVAL.OUT`.

REVISION HISTORY:

```
Created June 2003 (JKD)
```

1.0.12 rhoplot (Source File: rhoplot.f90)

INTERFACE:

```
Subroutine rhoplot
```

USES:

```
Use modinput
Use modmain
use modplotlabels
```

DESCRIPTION:

Outputs the charge density and the charge density gradients (modulus) read in from `STATE.OUT`, for 1D, 2D or 3D plotting.

REVISION HISTORY:

```
Created June 2003 (JKD)
Density gradients are added, March 2011 (DIN)
```

1.0.13 writelinen (Source File: writelinen.f90)

INTERFACE:

```
Subroutine writelinen
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Writes the linearisation energies for all APW and local-orbital functions to the file `LINENGY.OUT`.

REVISION HISTORY:

```
Created February 2004 (JKD)
```

1.0.14 fsmfield (Source File: fsmfield.f90)

INTERFACE:

```
Subroutine fsmfield
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Updates the effective magnetic field, $\mathbf{B}_{\rm FSM}$, required for fixing the spin moment to a given value, $\boldsymbol{\mu}_{\rm FSM}$. This is done by adding a vector to the field which is proportional to the difference between the moment calculated in the ith self-consistent loop and the required moment:

$$\mathbf{B}_{\rm FSM}^{i+1}=\mathbf{B}_{\rm FSM}^{i}+\lambda\left(\boldsymbol{\mu}^{i}-\boldsymbol{\mu}_{\rm FSM}\right),$$

where λ is a scaling factor.

REVISION HISTORY:

```
Created March 2005 (JKD)
```

1.0.15 plot2d (Source File: plot2d.f90)

INTERFACE:

```
Subroutine plot2d (labels, nf, lmax, ld, rfmt, rfir, plotdef)
```

USES:

```
      Use modinput
 use mod_muffin_tin
  use mod_atoms
  use mod_Gvector
      Use FoX_wxml
      use modmpi
  use modplotlabels
  use mod_plotting
```

INPUT/OUTPUT PARAMETERS:

```
fname : plot file name character(len=*)
nf    : number of functions (in,integer)
lmax  : maximum angular momentum (in,integer)
ld    : leading dimension (in,integer)
rfmt  : real muffin-tin function (in,real(ld,nrmtmax,natmtot,nf))
rfir  : real intersitial function (in,real(ngrtot,nf))
plotdef:type(plot2d) defines plot region
```

DESCRIPTION:

Produces a 2D plot of the real functions contained in arrays `rfmt` and `rfir` on the parallelogram defined by the corner vertices in the global array `vclp2d`. See routine `rfarray`.

REVISION HISTORY:

`Created June 2003 (JKD)`

1.0.16 rhonorm (Source File: rhonorm.f90)

INTERFACE:

```
Subroutine rhonorm
```

USES:

```
Use modmain
```

DESCRIPTION:

Loss of precision of the calculated total charge can result because the muffin-tin density is computed on a set of (θ, ϕ) points and then transformed to a spherical harmonic representation. This routine adds a constant to the density so that the total charge is correct. If the error in total charge exceeds a certain tolerance then a warning is issued.

REVISION HISTORY:

```
Created April 2003 (JKD)
Changed from rescaling to adding, September 2006 (JKD)
```

1.0.17 writegeom (Source File: writegeom.f90)

INTERFACE:

```
Subroutine writegeom (topt)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
topt : if .true. then the filename will be {\tt GEOMETRY_OPT.OUT}, otherwise
       {\tt GEOMETRY.OUT} (in,logical)
```

DESCRIPTION:

Outputs the lattice vectors and atomic positions to file, in a format which may be then used directly in `exciting.in`.

REVISION HISTORY:

`Created January 2004 (JKD)`

1.0.18 packeff (Source File: packeff.f90)

INTERFACE:

```
Subroutine packeff (tpack, n, nu)
```

USES:

```
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
tpack : .true. for packing, .false. for unpacking (in,logical)
n     : total number of real values stored (out,integer)
nu    : packed potential (inout,real(*))
```

DESCRIPTION:

Packs/unpacks the muffin-tin and interstitial parts of the effective potential and magnetic field into/from the single array `nu`. This array can then be passed directly to the mixing routine. See routine `rfpack`.

REVISION HISTORY:

```
Created June 2003 (JKD)
```

1.0.19 getoccsv (Source File: getoccsv.f90)

INTERFACE:

```
Subroutine getoccsv (vpl, occsvp)
```

USES:

```
Use modmain
Use modinput
Use modmpi
```

DESCRIPTION:

The file where the (second-variational) occupation numbers are stored is `OCCSV.OUT`. The maximum occupancies for spin-unpolarized systems is 2, whereas for spin-polarized systems it is 1. It is a direct-access binary file, the record length of which can be determined with the help of the array sizes and data type information. One record of this file has the following structure

k_{lat}	N_{stsv}	o

The following table explains the parts of the record in more detail

name	type	shape	description
k_{lat}	real(8)	3	k-point in lattice coordinates
N_{stsv}	integer	1	number of (second-variational) states (without core states)
o	real(8)	N_{stsv}	(second-variational) occupation number array

REVISION HISTORY:

```
Documentation added, Dec 2009 (S. Sagmeister)
```

1.0.20 writefermi (Source File: writefermi.f90)

INTERFACE:

```
Subroutine writefermi
```

USES:

```
Use modmain
```

DESCRIPTION:

Writes the Fermi energy to the file `EFERMI.OUT`.

REVISION HISTORY:

```
Created March 2005 (JKD)
```

1.0.21 potplot (Source File: potplot.f90)

INTERFACE:

```
Subroutine potplot
```

USES:

```
Use modinput
Use modmain
use modplotlabels
```

DESCRIPTION:

Outputs the exchange, correlation and Coulomb potentials, read in from `STATE.OUT`, for 1D, 2D or 3D plotting.

REVISION HISTORY:

```
Created June 2003 (JKD)
```

1.0.22 genapwfr (Source File: genapwfr.f90)

INTERFACE:

```
Subroutine genapwfr
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Generates the APW radial functions. This is done by integrating the scalar relativistic Schrödinger equation (or its energy deriatives) at the current linearisation energies using the spherical part of the effective potential. The number of radial functions at each l-value is given by the variable `apword` (at the muffin-tin boundary, the APW functions have continuous derivatives up to order `apword` $-$ 1). Within each l, these functions are orthonormalised with the Gram-Schmidt method. The radial Hamiltonian is applied to the orthonormalised functions and the results are stored in the global array `apwfr`.

REVISION HISTORY:

```
Created March 2003 (JKD)
```

1.0.23 writeefg (Source File: writeefg.f90)

INTERFACE:

```
Subroutine writeefg
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Computes the electric field gradient (EFG) tensor for each atom, α, and writes it to the file `EFG.OUT` along with its eigenvalues. The EFG is defined by

$$V_{ij}^{\alpha} \equiv \left.\frac{\partial^2 V_{\rm C}'(\mathbf{r})}{\partial \mathbf{r}_i \partial \mathbf{r}_j}\right|_{\mathbf{r}=\mathbf{r}_{\alpha}},$$

where $V_{\rm C}'$ is the Coulomb potential with the $l = m = 0$ component removed in each muffin-tin. The derivatives are computed explicitly using the routine `gradrfmt`.

REVISION HISTORY:

```
Created May 2004 (JKD)
Fixed serious problem, November 2006 (JKD)
```

1.0.24 energy (Source File: energy.f90)

INTERFACE:

```
Subroutine energy
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Computes the total energy and its individual contributions. The kinetic energy is given by

$$T_{\rm s} = \sum_i n_i \epsilon_i - \int \rho(\mathbf{r})[v_{\rm C}(\mathbf{r}) + v_{\rm xc}(\mathbf{r})]d\mathbf{r} - \int \mathbf{m}(\mathbf{r}) \cdot (\mathbf{B}_{\rm xc}(\mathbf{r}) + \mathbf{B}_{\rm ext}(\mathbf{r}))d\mathbf{r},$$

where n_i are the occupancies and ϵ_i are the eigenvalues of both the core and valence states; ρ is the density; $\mathbf{m}$ is the magnetisation density; $v_{\rm C}$ is the Coulomb potential; $v_{\rm xc}$ and $\mathbf{B}_{\rm xc}$ are the exchange-correlation potential and effective magnetic field, respectively; and $\mathbf{B}_{\rm ext}$ is the external magnetic field. The Hartree, electron-nuclear and nuclear-nuclear electrostatic energies are combined into the Coulomb energy:

$$\begin{aligned} E_{\rm C} &= E_{\rm H} + E_{\rm en} + E_{\rm nn} \\ &= \frac{1}{2} V_{\rm C} + E_{\rm Mad}, \end{aligned}$$

where

$$V_{\rm C} = \int \rho(\mathbf{r}) v_{\rm C}(\mathbf{r}) d\mathbf{r}$$

is the Coulomb potential energy. The Madelung energy is given by

$$E_{\rm Mad} = \frac{1}{2} \sum_\alpha z_\alpha R_\alpha,$$

where

$$R_\alpha = \lim_{r \to 0} \left(v^{\rm C}_{\alpha;00}(r) Y_{00} + \frac{z_\alpha}{r} \right)$$

for atom α, with $v^{\rm C}_{\alpha;00}$ being the $l = 0$ component of the spherical harmonic expansion of $v_{\rm C}$ in the muffin-tin, and z_α is the nuclear charge. Using the nuclear-nuclear energy determined at the start of the calculation, the electron-nuclear and Hartree energies can be isolated with

$$E_{\rm en} = 2 \left(E_{\rm Mad} - E_{\rm nn} \right)$$

and

$$E_{\rm H} = \frac{1}{2} (E_{\rm C} - E_{\rm en}).$$

Finally, the total energy is

$$E = T_{\rm s} + E_{\rm C} + E_{\rm xc},$$

where $E_{\rm xc}$ is obtained either by integrating the exchange-correlation energy density, or in the case of exact exchange, the explicit calculation of the Fock exchange integral. The energy from the external magnetic fields in the muffin-tins, `bfcmt`, is always removed from the total since these fields are non-physical: their field lines do not close. The energy of the physical external field, `bfieldc`, is also not included in the total because this field, like those in the muffin-tins, is used for breaking spin symmetry and taken to be infintesimal. If this field is intended to be finite, then the associated energy, `engybext`, should be added to the total by hand. See `potxc`, `exxengy` and related subroutines.

REVISION HISTORY:

`Created May 2003 (JKD)`

1.0.25 readfermi (Source File: readfermi.f90)

INTERFACE:

`Subroutine readfermi`

USES:

`Use modmain`

DESCRIPTION:

Reads the Fermi energy from the file `EFERMI.OUT`.

REVISION HISTORY:

`Created March 2005 (JKD)`

1.0.26 rtorat (Source File: modtetra.F90)

INTERFACE:

`Subroutine rtorat (eps, n, x, k, div)`

DESCRIPTION:

This subroutine factorizes the real coordinates of a vector $\mathbf{x}$. The output is an integer vector $\mathbf{k}$, such that

$$|x(i) - k(i)/\text{div}| < \text{eps}$$

for all $i = 1, \ldots, n$.

REVISION HISTORY:

`Created July 2008 by Sagmeister`

1.0.27 gentetlink (Source File: modtetra.F90)

INTERFACE:

```
Subroutine gentetlink (vpl, tqw, eps, bvec, ngridk, vkloff, nkpt, &
& nkptnr, vklnr, ikmapnr)
```

DESCRIPTION:

Generates an array connecting the tetrahedra of the **k**-point with the ones of the **k**+**q**-point. Interface routine referencing the `libbzint` library of Ricardo Gomez-Abal and Xinzheng Li.

REVISION HISTORY:

```
Created January 2008 (Sagmeister)
```

1.0.28 rfmtctof (Source File: rfmtctof.f90)

INTERFACE:

```
Subroutine rfmtctof (rfmt)
```

INPUT/OUTPUT PARAMETERS:

```
Use modinput
rfmt : real muffin-tin function (in,real(lmmaxvr,nrmtmax,natmtot))
```

DESCRIPTION:

Converts a real muffin-tin function from a coarse to a fine radial mesh by using cubic spline interpolation. See routines `rfinterp` and `spline`.

REVISION HISTORY:

```
Created October 2003 (JKD)
```

1.0.29 force (Source File: force.f90)

INTERFACE:

```
Subroutine force
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Computes the various contributions to the atomic forces. In principle, the force acting on a nucleus is simply the gradient at that site of the classical electrostatic potential from the other nuclei and the electronic density. This is a result of the Hellmann-Feynman theorem. However because the basis set is dependent on the nuclear coordinates and is not complete, the Hellman-Feynman force is inacurate and corrections to it are required. The first is the core correction which arises because the core wavefunctions were determined by neglecting the non-spherical parts of the effective potential v_s. Explicitly this is given by

$$\mathbf{F}^{\alpha}_{\rm core} = \int_{{\rm MT}_\alpha} v_{\rm s}(\mathbf{r}) \nabla \rho^{\alpha}_{\rm core}(\mathbf{r})\, d\mathbf{r}$$

for atom α. The second, which is the incomplete basis set (IBS) correction, is due to the position dependence of the APW functions, and is derived by considering the change in total energy if the eigenvector coefficients were fixed and the APW functions themselves were changed. This would result in changes to the first-variational Hamiltonian and overlap matrices given by

$$\delta H^{\alpha}_{\mathbf{G},\mathbf{G}'} = i(\mathbf{G}-\mathbf{G}')\left(H^{\alpha}_{\mathbf{G}+\mathbf{k},\mathbf{G}'+\mathbf{k}} - \frac{1}{2}(\mathbf{G}+\mathbf{k})\cdot(\mathbf{G}'+\mathbf{k})\tilde{\Theta}_{\alpha}(\mathbf{G}-\mathbf{G}')e^{-i(\mathbf{G}-\mathbf{G}')\cdot\mathbf{r}_{\alpha}}\right)$$
$$\delta O^{\alpha}_{\mathbf{G},\mathbf{G}'} = i(\mathbf{G}-\mathbf{G}')\left(O^{\alpha}_{\mathbf{G}+\mathbf{k},\mathbf{G}'+\mathbf{k}} - \tilde{\Theta}_{\alpha}(\mathbf{G}-\mathbf{G}')e^{-i(\mathbf{G}-\mathbf{G}')\cdot\mathbf{r}_{\alpha}}\right)$$

where both $\mathbf{G}$ and $\mathbf{G}'$ run over the APW indices; $\tilde{\Theta}_{\alpha}$ is the form factor of the smooth step function for muffin-tin α; and H^{α} and O^{α} are the muffin-tin Hamiltonian and overlap matrices, respectively. The APW-local-orbital part is given by

$$\delta H^{\alpha}_{\mathbf{G},\mathbf{G}'} = i(\mathbf{G}+\mathbf{k})H^{\alpha}_{\mathbf{G}+\mathbf{k},\mathbf{G}'+\mathbf{k}}$$
$$\delta O^{\alpha}_{\mathbf{G},\mathbf{G}'} = i(\mathbf{G}+\mathbf{k})O^{\alpha}_{\mathbf{G}+\mathbf{k},\mathbf{G}'+\mathbf{k}}$$

where $\mathbf{G}$ runs over the APW indices and $\mathbf{G}'$ runs over the local-orbital indices. There is no contribution from the local-orbital-local-orbital part of the matrices. We can now write the IBS correction in terms of the basis of first-variational states as

$$\mathbf{F}^{\alpha\mathbf{k}}_{ij} = \sum_{\mathbf{G},\mathbf{G}'} b^{i\mathbf{k}*}_{\mathbf{G}} b^{j\mathbf{k}}_{\mathbf{G}'} \left(\delta H^{\alpha}_{\mathbf{G},\mathbf{G}'} - \epsilon_j \delta O^{\alpha}_{\mathbf{G},\mathbf{G}'}\right),$$

where $b^{i\mathbf{k}}$ is the first-variational eigenvector. Finally, the $\mathbf{F}^{\alpha\mathbf{k}}_{ij}$ matrix elements can be multiplied by the second-variational coefficients, and contracted over all indices to obtain the IBS force:

$$\mathbf{F}^{\alpha}_{\rm IBS} = \sum_{\mathbf{k}} w_{\mathbf{k}} \sum_{l\sigma} n_{l\mathbf{k}} \sum_{ij} c^{l\mathbf{k}*}_{\sigma i} c^{l\mathbf{k}}_{\sigma j} \mathbf{F}^{\alpha\mathbf{k}}_{ij} + \int_{{\rm MT}_\alpha} v_{\rm s}(\mathbf{r}) \nabla \left[\rho(\mathbf{r}) - \rho^{\alpha}_{\rm core}(\mathbf{r})\right]\, d\mathbf{r},$$

where $c^{l\mathbf{k}}$ are the second-variational coefficients, $w_{\mathbf{k}}$ are the k-point weights, $n_{l\mathbf{k}}$ are the occupancies, and $v_{\rm s}$ is the Kohn-Sham effective potential. See routines `hmlaa`, `olpaa`, `hmlalo`, `olpalo`, `energy`, `seceqn` and `gencfun`.

REVISION HISTORY:

```
Created January 2004 (JKD)
Fixed problem with second-variational forces, May 2008 (JKD)
```

1.0.30 forcek (Source File: forcek.f90)

INTERFACE:

```
Subroutine forcek (ik, ffacg)
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Computes the **k**-dependent contribution to the incomplete basis set (IBS) force. See the calling routine `force` for a full description.

REVISION HISTORY:

```
Created June 2006 (JKD)
Updated for spin-spiral case, May 2007 (Francesco Cricchio and JKD)
```

1.0.31 rhoinit (Source File: rhoinit.f90)

INTERFACE:

```
Subroutine rhoinit
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Initialises the crystal charge density. Inside the muffin-tins it is set to the spherical atomic density. In the interstitial region it is taken to be constant such that the total charge is correct. Requires that the atomic densities have already been calculated.

REVISION HISTORY:

```
Created January 2003 (JKD)
```

1.0.32 genrmesh (Source File: genrmesh.f90)

INTERFACE:

```
Subroutine genrmesh
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Generates the coarse and fine radial meshes for each atomic species in the crystal. Also determines which points are in the inner part of the muffin-tin using the value of `fracinr`. For species i the radial mesh starts from the value $R_0 = r(1)$, hits the muffin-tin surface at $R_{\rm MT} = r(N)$, and ends at the effective infinity value $R_\infty = r(N_\infty)$. The number of points up to the effective infinity are determined by the number of points N up to the muffin-tin radius as well as by the smallest and largest mesh point and the muffin-tin radius, and is given by

$$N_\infty = \text{round}\left[\frac{(N-1)\ln(R_\infty/R_0)}{\ln(R_{\rm MT}/R_0)}\right] + 1.$$

The radial mesh points are finally defined by

$$r(j) = R_0\left(\frac{R_{\rm MT}}{R_0}\right)^{\frac{j-1}{N-1}},$$

for $j = 1, \ldots, N_\infty$.
Note: The number of mesh points initially defined in species file is adapted to be commensurate with the coarse mesh of step size `lradstep`

$$N = N^* - \quad \text{mod}\,(N^*, \texttt{lradstep}),$$

if N^* was the number of points defined in the species file. The number of mesh points $\tilde{N}$ of the coarse mesh $\tilde{r}(j)$ reads

$$\tilde{N} = \left\lfloor\frac{N-1}{\texttt{lradstep}}\right\rfloor + 1.$$

It is given by

$$\tilde{r}(j) = r([j-1] * \texttt{lradstep} + 1),$$

for $j = 1, \ldots, \tilde{N}$ and has the properties $\tilde{r}(1) = r(1) = R_0$ and $\tilde{r}(\tilde{N}) = r(N) = R_{\rm MT}$.
REVISION HISTORY:

```
Created September 2002 (JKD)
Revised and updated documentation, April 2011 (S. Sagmeister)
```

1.0.33 genpchgs (Source File: genpchgs.F90)

INTERFACE:

```
subroutine genpchgs(ik,evecfv,evecsv)
```

USES:

```
use modinput
use modmain
```

DESCRIPTION:

Generate partial charges for each state j, atom α and for each (lm) combination.

REVISION HISTORY:

```
Created 2010 (Sagmeister)
```

1.0.34 gensdmat (Source File: gensdmat.f90)

INTERFACE:

```
Subroutine gensdmat (evecsv, sdmat)
```

USES:

```
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
evecsv : second-variational eigenvectors (in,complex(nstsv,nstsv))
sdmat  : spin density matrices (out,complex(nspinor,nspinor,nstsv))
```

DESCRIPTION:

Computes the spin density matrices for a set of second-variational states.

REVISION HISTORY:

```
Created September 2008 (JKD)
```

1.0.35 mossbauer (Source File: mossbauer.f90)

INTERFACE:

```
Subroutine mossbauer
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Computes the contact charge density and contact magnetic hyperfine field for each atom and outputs the data to the file `MOSSBAUER.OUT`. The nuclear radius used for the contact quantities is approximated by the empirical formula $R_{\rm N} = 1.25Z^{1/3}$ fm, where Z is the atomic number.

REVISION HISTORY:

```
Created May 2004 (JKD)
```

1.0.36 seceqn (Source File: seceqn.f90)

Subroutine seceqn (ik, evalfv, evecfv, evecsv) **USES:**

```
Use modinput
Use modmain
Use modmpi
Use sclcontroll
Use diisinterfaces
```

INPUT/OUTPUT PARAMETERS:

```
ik     : k-point number (in,integer)
evalfv : first-variational eigenvalues (out,real(nstfv))
evecfv : first-variational eigenvectors (out,complex(nmatmax,nstfv))
evecsv : second-variational eigenvectors (out,complex(nstsv,nstsv))
```

DESCRIPTION:

Solves the first- and second-variational secular equations. See routines `match`, `seceqnfv`, `seceqnss` and `seceqnsv`.

REVISION HISTORY:

```
Created March 2004 (JKD)
```

1.0.37 symrfmt (Source File: symrfmt.f90)

INTERFACE:

```
Subroutine symrfmt (lrstp, is, rot, rfmt, srfmt)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
lrstp : radial step length (in,integer)
is    : species number (in,integer)
rot   : rotation matrix (in,real(3,3))
rfmt  : input muffin-tin function (in,real(lmmaxvr,nrmtmax))
srfmt : output muffin-tin function (out,real(lmmaxvr,nrmtmax))
```

DESCRIPTION:

Applies a symmetry operation (in the form of a rotation matrix) to a real muffin-tin function. The input function can also be the output function. See the routines `rtozflm` and `rotzflm`.

REVISION HISTORY:

```
Created May 2003 (JKD)
```

1.0.38 gencore (Source File: gencore.f90)

INTERFACE:

```
Subroutine gencore
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Computes the core radial wavefunctions, eigenvalues and densities. The radial Dirac equation is solved in the spherical part of the effective potential to which the atomic potential has been appended for $r > R^{\mathrm{MT}}$. In the case of spin-polarised calculations, the effective magnetic field is ignored.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

1.0.39 writeinfo (Source File: writeinfo.f90)

INTERFACE:

```
Subroutine writeinfo (fnum)
```

USES:

```
Use modinput
Use modmain
use modmpi, only: procs
#ifdef TETRA
Use modtetra
#endif
```

INPUT/OUTPUT PARAMETERS:

```
fnum : unit specifier for INFO.OUT file (in,integer)
```

DESCRIPTION:

Outputs basic information about the run to the file `INFO.OUT`. Does not close the file afterwards.

REVISION HISTORY:

```
Created January 2003 (JKD)
```

1.0.40 gndstate (Source File: gndstate.f90)

INTERFACE:

```
Subroutine gndstate
```

USES:

```
Use modinput
Use modmain
Use modmpi
Use scl_xml_out_Module
```

DESCRIPTION:

Computes the self-consistent Kohn-Sham ground-state. General information is written to the file `INFO.OUT`. First- and second-variational eigenvalues, eigenvectors and occupancies are written to the unformatted files `EVALFV.OUT`, `EVALSV.OUT`, `EVECFV.OUT`, `EVECSV.OUT` and `OCCSV.OUT`.

REVISION HISTORY:

```
Created October 2002 (JKD)
```

1.0.41 elfplot (Source File: elfplot.f90)

INTERFACE:

```
Subroutine elfplot
```

USES:

```
Use modinput
Use modmain
 use modplotlabels
```

DESCRIPTION:

Outputs the electron localisation function (ELF) for 1D, 2D or 3D plotting. The spin-averaged ELF is given by

$$f_{\rm ELF}(\mathbf{r}) = \frac{1}{1+[D(\mathbf{r})/D^0(\mathbf{r})]^2},$$

where

$$D(\mathbf{r}) = \frac{1}{2}\left(\tau(\mathbf{r}) - \frac{1}{4}\frac{[\nabla n(\mathbf{r})]^2}{n(\mathbf{r})}\right)$$

and

$$\tau(\mathbf{r}) = \sum_{i=1}^{N} |\nabla \Psi_i(\mathbf{r})|^2$$

is the spin-averaged kinetic energy density from the spinor wavefunctions. The function D^0 is the kinetic energy density for the homogeneous electron gas evaluated for $n(\mathbf{r})$:

$$D^0(\mathbf{r}) = \frac{3}{5}(6\pi^2)^{2/3}\left(\frac{n(\mathbf{r})}{2}\right)^{5/3}.$$

The ELF is useful for the topological classification of bonding. See for example T. Burnus, M. A. L. Marques and E. K. U. Gross [Phys. Rev. A 71, 10501 (2005)].

REVISION HISTORY:

```
Created September 2003 (JKD)
Fixed bug found by F. Wagner (JKD)
```

1.0.42 findsymcrys (Source File: findsymcrys.f90)

INTERFACE:

```
Subroutine findsymcrys
```

USES:

```
      Use modinput
      Use modmain
#ifdef XS
      Use modxs
#endif
```

DESCRIPTION:

Finds the complete set of symmetries which leave the crystal structure (including the magnetic fields) invariant. A crystal symmetry is of the form $\{\alpha_S|\alpha_R|\mathbf{t}\}$, where $\mathbf{t}$ is a translation vector, α_R is a spatial rotation operation and α_S is a global spin rotation. Note that the order of operations is important and defined to be from right to left, i.e. translation followed by spatial rotation followed by spin rotation. In the case of spin-orbit coupling $\alpha_S = \alpha_R$. In order to determine the translation vectors, the entire atomic basis is shifted so that the first atom in the smallest set of atoms of the same species is at the origin. Then all displacement vectors between atoms in this set are checked as possible symmetry translations. If the global variable `tshift` is set to `.false.` then the shift is not performed. See L. M. Sandratskii and P. G. Guletskii, *J. Phys. F: Met. Phys.* **16**, L43 (1986) and the routine `findsym`.

REVISION HISTORY:

```
Created April 2007 (JKD)
```

1.0.43 poteff (Source File: poteff.f90)

INTERFACE:

```
Subroutine poteff
```

USES:

```
      Use modmain
```

DESCRIPTION:

Computes the effective potential by adding together the Coulomb and exchange-correlation potentials. See routines `potcoul` and `potxc`.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

1.0.44 genshtmat (Source File: genshtmat.f90)

INTERFACE:

```
Subroutine genshtmat
```

USES:

```
      Use modinput
      Use modmain
#ifdef XS
      Use modxs
#endif
```

DESCRIPTION:

Generates the forward and backward spherical harmonic transformation (SHT) matrices using the spherical covering set produced by the routine `sphcover`. These matrices are used to transform a function between its (l, m)-expansion coefficients and its values at the (θ, ϕ) points on the sphere.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

1.0.45 writewiq2 (Source File: writewiq2.f90)

INTERFACE:

```
Subroutine writewiq2
```

USES:

```
Use modmain
```

DESCRIPTION:

Outputs the integrals of $1/q^2$ in the small parallelepiped around each q-point to the file `WIQ2.OUT`. Note that the integrals are calculated after the q-point has been mapped to the first Brillouin zone. See routine genwiq2.

REVISION HISTORY:

```
Created June 2005 (JKD)
```

1.0.46 checkmt (Source File: checkmt.f90)

INTERFACE:

```
Subroutine checkmt
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Checks for overlapping muffin-tins. If any muffin-tins are found to intersect the program is terminated with error.

REVISION HISTORY:

```
Created May 2003 (JKD)
```

1.0.47 readstate (Source File: readstate.f90)

INTERFACE:

```
Subroutine readstate
```

USES:

```
      Use modinput
      Use modmain
#ifdef XS
      Use modxs, Only: isreadstate0
#endif
```

DESCRIPTION:

Reads in the charge density and other relevant variables from the file `STATE.OUT`. Checks for version and parameter compatibility.

REVISION HISTORY:

```
Created May 2003 (JKD)
```

1.0.48 allatoms (Source File: allatoms.f90)

INTERFACE:

```
Subroutine allatoms
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Solves the Kohn-Sham-Dirac equations for each atom type in the solid and finds the self-consistent radial wavefunctions, eigenvalues, charge densities and potentials. The atomic densities can then be used to initialise the crystal densities, and the atomic self-consistent potentials can be appended to the muffin-tin potentials to solve for the core states. Note that, irrespective of the value of `xctype`, exchange-correlation functional type 3 is used. See also `atoms`, `rhoinit`, `gencore` and `modxcifc`.

REVISION HISTORY:

```
Created September 2002 (JKD)
Modified for GGA, June 2007 (JKD)
```

1.0.49 writegeometryxml (Source File: writegeometryxml.f90)

INTERFACE:

```
Subroutine writegeometryxml (topt)
```

USES:

```
Use modmain
Use modinput
Use modsp
Use modspdb
Use FoX_wxml
```

INPUT/OUTPUT PARAMETERS:

topt : if .true. then the filename will be {\ttgeometry_opt.xml}, otherwise {\tt geometry.xml} (in,logical)

DESCRIPTION:

Outputs the lattice vectors and atomic positions to file, in a format which may be then used directly in `exciting.in`.

REVISION HISTORY:

Created January 2004 (JKD)

1.0.50 nfftifc (Source File: nfftifc.f90)

INTERFACE:

Subroutine nfftifc (n)

INPUT/OUTPUT PARAMETERS:

n : required/avalable grid size (in,integer)

DESCRIPTION:

Interface to the grid requirements of the fast Fourier transform routine. Most routines restrict n to specific prime factorisations. This routine returns the next largest grid size allowed by the FFT routine.

REVISION HISTORY:

Created October 2002 (JKD)

1.0.51 genidxlo (Source File: genidxlo.f90)

INTERFACE:

Subroutine genidxlo

USES:

Use modmain

DESCRIPTION:

Generates an index array which maps the local-orbitals in each atom to their locations in the overlap or Hamiltonian matrices. Also finds the total number of local-orbitals.

REVISION HISTORY:

Created June 2003 (JKD)

1.0.52 zfmtinp (Source File: zfmtinp.f90)

INTERFACE:

```
Complex (8) Function zfmtinp (tsh, lmax, nr, r, ld, zfmt1, zfmt2)
```

INPUT/OUTPUT PARAMETERS:

```
tsh   : .true. if the functions are in spherical harmonics (in,logical)
lmax  : maximum angular momentum
nr    : number of radial mesh points (in,integer)
r     : radial mesh (in,real(nr))
ld    : leading dimension (in,integer)
zfmt1 : first complex muffin-tin function in spherical harmonics/
        coordinates (in,complex(ld,nr))
zfmt2 : second complex muffin-tin function in spherical harmonics/
        coordinates (in,complex(ld,nr))
```

DESCRIPTION:

Calculates the inner product of two complex fuctions in the muffin-tin. In other words, given two complex functions of the form

$$f(\mathbf{r}) = \sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} f_{lm}(r) Y_{lm}(\hat{\mathbf{r}}),$$

the function returns

$$I = \sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} \int f_{lm}^{1*}(r) f_{lm}^{2}(r) r^2 \, dr \; .$$

Note that if `tsh` is `.false.` the functions are in spherical coordinates rather than spherical harmonics. In this case I is multiplied by $4\pi/(l_{\max}+1)^2$.

REVISION HISTORY:

```
Created November 2003 (Sharma)
```

1.0.53 zfftifc (Source File: zfftifc.f90)

INTERFACE:

```
Subroutine zfftifc (nd, n, sgn, z)
```

INPUT/OUTPUT PARAMETERS:

```
nd   : number of dimensions (in,integer)
n    : grid sizes (in,integer(nd))
sgn  : FFT direction, -1: forward; 1: backward (in,integer)
z    : array to transform (inout,complex(n(1)*n(2)*...*n(nd)))
```

DESCRIPTION:

Interface to the double-precision complex fast Fourier transform routine. This is to allow machine-optimised routines to be used without affecting the rest of the code. See routine `nfftifc`.

REVISION HISTORY:

```
Created October 2002 (JKD)
```

1.0.54 potcoul (Source File: potcoul.f90)

INTERFACE:

```
Subroutine potcoul
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Calculates the Coulomb potential of the real charge density stored in the global variables `rhomt` and `rhoir` by solving Poisson's equation. These variables are converted to complex representations and passed to the routine `zpotcoul`.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

1.0.55 init0 (Source File: init0.f90)

INTERFACE:

```
Subroutine init0
```

USES:

```
Use modinput
Use modmain
Use modxcifc
#ifdef XS
Use modxs
#endif
```

DESCRIPTION:

Performs basic consistency checks as well as allocating and initialising global variables not dependent on the k-point set.

REVISION HISTORY:

```
Created January 2004 (JKD)
```

1.0.56 portstate (Source File: portstate.F90)

INTERFACE:

```
Subroutine portstate (act)
```

USES:

```
Use ioarray
use mod_misc, only: refversion_gitstate
```

DESCRIPTION:

Toggle file format of `STATE.OUT`. If tb2a is true an ASCII file with the name `STATE.xml` is generated and the data from `STATE.OUT` is transferred. If tb2a is false the conversion goes in the other direction. Based upon the routines `readstate` and `writestate`.

REVISION HISTORY:

```
Created 2007 (Sagmeister)
```

1.0.57 genveffig (Source File: genveffig.f90)

INTERFACE:

```
Subroutine genveffig
```

USES:

```
Use modmain
```

DESCRIPTION:

Generates the Fourier transform of the effective potential in the intersitial region. The potential is first multiplied by the characteristic function which zeros it in the muffin-tins. See routine `gencfun`.

REVISION HISTORY:

```
Created January 2004 (JKD)
```

1.0.58 charge (Source File: charge.f90)

INTERFACE:

```
Subroutine charge
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Computes the muffin-tin, interstitial and total charges by integrating the density.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

1.0.59 genppts (Source File: genppts.f90)

INTERFACE:

```
Subroutine genppts (reducep, tfbz, ngridp, boxl, nppt, ipmap, ivp, vpl, &
& vpc, wppt)
```

USES:

```
      Use modinput
      Use modmain
#ifdef XS
      Use modxs
#endif
```

INPUT/OUTPUT PARAMETERS:

```
reducep : .true. if p-point set is to be reduced (in,logical)
tfbz    : .true. if vpl and vpc should be mapped to the first Brillouin
          zone (in,logical)
ngridp  : p-point grid size (in,integer(3))
boxl    : corners of box containing p-points in lattice coordinates, the
          first vector is the origin (in,real(3,4))
nppt    : total number of p-points (out,integer)
ipmap   : map from integer grid to p-point index
          (out,integer(0:ngridp(1)-1,0:ngridp(2)-1,0:ngridp(3)-1))
ivp     : integer coordinates of the p-points
          (out,integer(3,ngridp(1)*ngridp(2)*ngridp(3)))
vpl     : lattice coordinates of each p-point
          (out,real(3,ngridp(1)*ngridp(2)*ngridp(3)))
vpc     : Cartesian coordinates of each p-point
          (out,real(3,ngridp(1)*ngridp(2)*ngridp(3)))
wppt    : weights of each p-point (out,real(ngridp(1)*ngridp(2)*ngridp(3)))
```

DESCRIPTION:

This routine is used for generating k-point or q-point sets. Since these are stored in global arrays, the points passed to this and other routines are referred to as p-points. If `reducep` is `.true.` the set is reduced with the spatial part of the crystal symmetries. In lattice coordinates, the $\mathbf{p}$ vectors are given by

$$\mathbf{p} = \begin{pmatrix} \mathbf{B}_2 - \mathbf{B}_1 & \mathbf{B}_3 - \mathbf{B}_1 & \mathbf{B}_4 - \mathbf{B}_1 \end{pmatrix} \begin{pmatrix} i_1/n_1 \\ i_2/n_2 \\ i_3/n_3 \end{pmatrix} + \mathbf{B}_1$$

where i_j runs from 0 to $n_j - 1$, and the $\mathbf{B}$ vectors define the corners of a box with $\mathbf{B}_1$ as the origin. If `tfbz` is `.true.` then the vectors `vpl` (and `vpc`) are mapped to the first Brillouin zone. If `tfbz` is `.false.` and `reducep` is `.true.` then the coordinates of `vpl` are mapped to the $[0, 1)$ interval. The p-point weights are stored in `wppt` and the array `ipmap` contains the map from the integer coordinates to the reduced index.

REVISION HISTORY:

```
Created August 2002 (JKD)
Updated April 2007 (JKD)
Modifications for excited states, November 2007 (Sagmeister)
Added mapping to the first Brillouin zone, September 2008 (JKD)
```

1.0.60 readinput (Source File: setdefault.f90)

INTERFACE:

```
Subroutine setdefault
```

USES:

```
      Use modinput
      Use modmain
#ifdef TETRA
      Use modtetra
#endif
#ifdef XS
      Use modmpi, Only: rank
      Use modxs
#endif
      Use sclcontroll
```

DESCRIPTION:

Sets default values for the input parameters.

REVISION HISTORY:

```
Created September 2002 (JKD)
Additional parmeters for excited states and tetrahedron method
  2004-2008 (Sagmeister)
```

1.0.61 seceqnfv (Source File: seceqnfv.f90)

INTERFACE:

```
Subroutine seceqnfv (nmatp, ngp, igpig, vgpc, apwalm, evalfv, evecfv)
```

USES:

```
Use modinput
Use modmain
Use modfvsystem
```

INPUT/OUTPUT PARAMETERS:

```
nmatp  : order of overlap and Hamiltonian matrices (in,integer)
ngp    : number of G+k-vectors for augmented plane waves (in,integer)
igpig  : index from G+k-vectors to G-vectors (in,integer(ngkmax))
vgpc   : G+k-vectors in Cartesian coordinates (in,real(3,ngkmax))
apwalm : APW matching coefficients
         (in,complex(ngkmax,apwordmax,lmmaxapw,natmtot))
evalfv : first-variational eigenvalues (out,real(nstfv))
evecfv : first-variational eigenvectors (out,complex(nmatmax,nstfv))
```

DESCRIPTION:

Solves the secular equation,

$$(H - \epsilon O)b = 0,$$

for the all the first-variational states of the input k-point.

REVISION HISTORY:

```
Created March 2004 (JKD)
```

1.0.62 potxc (Source File: potxc.f90)

INTERFACE:

```
subroutine potxc
```

USES:

```
use modmain
use modxcifc
```

DESCRIPTION:

Computes the exchange-correlation potential and energy density. In the muffin-tin, the density is transformed from spherical harmonic coefficients ρ_{lm} to spherical coordinates (θ, ϕ) with a backward spherical harmonic transformation (SHT). Once calculated, the exchange-correlation potential and energy density are transformed with a forward SHT.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

1.0.63 symrfir (Source File: symrfir.f90)

Subroutine symrfir (ngv, rfir) **USES:**

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
ngv  : number of G-vectors to be used for the Fourier space rotation
       (in,integer)
rfir : real intersitial function (inout,real(ngrtot))
```

DESCRIPTION:

Symmetrises a real scalar interstitial function. The function is first Fourier transformed to G-space, and then averaged over each symmetry by rotating the Fourier coefficients and multiplying them by a phase factor corresponding to the symmetry translation.

REVISION HISTORY:

```
Created July 2007 (JKD)
```

1.0.64 zfinp (Source File: zfinp.f90)

INTERFACE:

```
Complex (8) Function zfinp (tsh, zfmt1, zfmt2, zfir1, zfir2)
```

USES:

```
Use modmain
Use modinput
```

INPUT/OUTPUT PARAMETERS:

```
tsh   : .true. if the muffin-tin functions are in spherical harmonics
        (in,logical)
zfmt1 : first complex function in spherical harmonics/coordinates for all
        muffin-tins (in,complex(lmmaxvr,nrcmtmax,natmtot))
zfmt2 : second complex function in spherical harmonics/coordinates for all
        muffin-tins (in,complex(lmmaxvr,nrcmtmax,natmtot))
zfir1 : first complex interstitial function in real-space
        (in,complex(ngrtot))
zfir2 : second complex interstitial function in real-space
        (in,complex(ngrtot))
```

DESCRIPTION:

Calculates the inner product of two complex fuctions over the entire unit cell. The muffin-tin functions should be stored on the coarse radial grid and have angular momentum cut-off `lmaxvr`. In the intersitial region, the integrand is multiplied with the characteristic function, to remove the contribution from the muffin-tin. See routines `zfmtinp` and `gencfun`.

REVISION HISTORY:

```
Created July 2004 (Sharma)
```

1.0.65 gensfacgp (Source File: gensfacgp.f90)

INTERFACE:

```
Subroutine gensfacgp (ngp, vgpc, ld, sfacgp)
```

USES:

```
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
ngp    : number of G+p-vectors (in,integer)
vgpc   : G+p-vectors in Cartesian coordinates (in,real(3,*))
ld     : leading dimension (in,integer)
sfacgp : structure factors of G+p-vectors (out,complex(ld,natmtot))
```

DESCRIPTION:

Generates the atomic structure factors for a set of $\mathbf{G}+\mathbf{p}$-vectors:

$$S_\alpha(\mathbf{G}+\mathbf{p}) = \exp(i(\mathbf{G}+\mathbf{p})\cdot\mathbf{r}_\alpha),$$

where $\mathbf{r}_\alpha$ is the position of atom α.

REVISION HISTORY:

```
Created January 2003 (JKD)
```

1.0.66 zpotclmt (Source File: zpotclmt.f90)

INTERFACE:

```
Subroutine zpotclmt (ptnucl, lmax, nr, r, zn, ld, zrhomt, zvclmt)
```

INPUT/OUTPUT PARAMETERS:

```
ptnucl : .true. if the nucleus is a point particle (in,logical)
lmax   : maximum angular momentum (in,integer)
nr     : number of radial mesh points (in,integer)
r      : radial mesh (in,real(nr))
zn     : nuclear charge at the atomic center (in,real)
ld     : leading dimension (in,integer)
zrhomt : muffin-tin charge density (in,complex(ld,nr))
zvclmt : muffin-tin Coulomb potential (out,complex(ld,nr))
```

DESCRIPTION:

Solves the Poisson equation for the charge density contained in an isolated muffin-tin using the Green's function approach. In other words, the spherical harmonic expansion of the Coulomb potential, V_{lm}, is obtained from the density expansion, ρ_{lm}, by

$$V_{lm}(r) = \frac{4\pi}{2l+1}\left(\frac{1}{r^{l+1}}\int_0^r \rho_{lm}(r')r'^{l+2}dr' + r^l\int_r^R \frac{\rho_{lm}(r')}{r'^{l-1}}dr'\right) + \frac{1}{Y_{00}}\frac{z}{r}\delta_{l,0}$$

where the last term is the monopole arising from the point charge z, and R is the muffin-tin radius.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

1.0.67 wavefmt (Source File: wavefmt.f90)

INTERFACE:

```
Subroutine wavefmt (lrstp, lmax, is, ia, ngp, apwalm, evecfv, ld, wfmt)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
lrstp  : radial step length (in,integer)
lmax   : maximum angular momentum required (in,integer)
is     : species number (in,integer)
ia     : atom number (in,integer)
ngp    : number of G+p-vectors (in,integer)
apwalm : APW matching coefficients
         (in,complex(ngkmax,apwordmax,lmmaxapw,natmtot))
evecfv : first-variational eigenvector (in,complex(nmatmax))
ld     : leading dimension (in,integer)
wfmt   : muffin-tin wavefunction (out,complex(ld,*))
```

DESCRIPTION:

Calculates the first-variational wavefunction in the muffin-tin in terms of a spherical harmonic expansion. For atom α and a particular k-point $\mathbf{p}$, the r-dependent (l,m)-coefficients of the wavefunction for the ith state are given by

$$\Phi^{i\mathbf{p}}_{\alpha lm}(r) = \sum_{\mathbf{G}} b^{i\mathbf{p}}_{\mathbf{G}} \sum_{j=1}^{M^\alpha_l} A^\alpha_{jlm}(\mathbf{G}+\mathbf{p})u^\alpha_{jl}(r) + \sum_{j=1}^{N^\alpha} b^{i\mathbf{p}}_{(\alpha,j,m)} v^\alpha_j(r)\delta_{l,l_j},$$

where $b^{i\mathbf{p}}$ is the ith eigenvector returned from routine `seceqn`; $A^\alpha_{jlm}(\mathbf{G}+\mathbf{p})$ is the matching coefficient; M^α_l is the order of the APW; u^α_{jl} is the APW radial function; N^α is the number of

local-orbitals; v_j^{α} is the jth local-orbital radial function; and (α, j, m) is a compound index for the location of the local-orbital in the eigenvector. See routines `genapwfr`, `genlofr`, `match` and `seceqn`.

REVISION HISTORY:

```
Created April 2003 (JKD)
Fixed description, October 2004 (C. Brouder)
Removed argument ist, November 2006 (JKD)
```

1.0.68 wavefmt_add (Source File: wavefmt.f90)

INTERFACE:

```
Subroutine wavefmt_add (nr, ld, wfmt, a, b, lrstp, fr)
```

INPUT/OUTPUT PARAMETERS:

```
nr     : number of radial mesh points (in,integer)
ld     : leading dimension (in,integer)
wfmt   : complex muffin-tin wavefunction passed in as a real array
         (inout,real(2*ld,*))
a      : real part of complex constant (in,real)
b      : imaginary part of complex constant (in,real)
lrstp  : radial step length (in,integer)
fr     : real radial function (in,real(lrstp,*))
```

DESCRIPTION:

Adds a real function times a complex constant to a complex muffin-tin wavefunction as efficiently as possible. See routine `wavefmt`.

REVISION HISTORY:

```
Created December 2006 (JKD)
```

1.0.69 linengy (Source File: linengy.f90)

INTERFACE:

```
Subroutine linengy
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Calculates the new linearisation energies for both the APW and local-orbital radial functions. See the routine `findband`.

REVISION HISTORY:

```
Created May 2003 (JKD)
```

1.0.70 plot1d (Source File: plot1d.f90)

INTERFACE:

```
Subroutine plot1d (labels, nf, lmax, ld, rfmt, rfir, plotdef)
```

USES:

```
Use modinput
use modmpi
Use FoX_wxml
use mod_muffin_tin
use mod_atoms
use mod_Gvector
use modplotlabels
use mod_plotting
```

INPUT/OUTPUT PARAMETERS:

```
lables : plot labels (character*)
nf     : number of functions (in,integer)
lmax   : maximum angular momentum (in,integer)
ld     : leading dimension (in,integer)
rfmt   : real muffin-tin function (in,real(ld,nrmtmax,natmtot,nf))
rfir   : real intersitial function (in,real(ngrtot,nf))
plotdef: type(plot1d) defining plotregion
```

DESCRIPTION:

Produces a 1D plot of the real functions contained in arrays `rfmt` and `rfir` along the lines connecting the vertices in the global array `vvlp1d`. See routine `rfarray`.

REVISION HISTORY:

```
Created June 2003 (JKD)
```

1.0.71 findprim (Source File: findprim.f90)

INTERFACE:

```
Subroutine findprim
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

This routine finds the smallest primitive cell which produces the same crystal structure as the conventional cell. This is done by searching through all the vectors which connect atomic positions and finding those which leave the crystal structure invariant. Of these, the three shortest which produce a non-zero unit cell volume are chosen.

REVISION HISTORY:

```
Created April 2007 (JKD)
```

1.0.72 gengpvec (Source File: gengpvec.f90)

INTERFACE:

```
Subroutine gengpvec (vpl, vpc, ngp, igpig, vgpl, vgpc, gpc, tpgpc)
```

USES:

```
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
vpl   : p-point vector in lattice coordinates (in,real(3))
vpc   : p-point vector in Cartesian coordinates (in,real(3))
ngp   : number of G+p-vectors returned (out,integer)
igpig : index from G+p-vectors to G-vectors (out,integer(ngkmax))
vgpl  : G+p-vectors in lattice coordinates (out,real(3,ngkmax))
vgpc  : G+p-vectors in Cartesian coordinates (out,real(3,ngkmax))
gpc   : length of G+p-vectors (out,real(ngkmax))
tpgpc : (theta, phi) coordinates of G+p-vectors (out,real(2,ngkmax))
```

DESCRIPTION:

Generates a set of $\mathbf{G}+\mathbf{p}$-vectors for the input p-point with length less than `gkmax`. These are used as the plane waves in the APW functions. Also computes the spherical coordinates of each vector.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

1.0.73 match (Source File: match.f90)

INTERFACE:

```
Subroutine match (ngp, gpc, tpgpc, sfacgp, apwalm)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
ngp    : number of G+p-vectors (in,integer)
gpc    : length of G+p-vectors (in,real(ngkmax))
tpgpc  : (theta, phi) coordinates of G+p-vectors (in,real(2,ngkmax))
sfacgp : structure factors of G+p-vectors (in,complex(ngkmax,natmtot))
apwalm : APW matching coefficients
         (out,complex(ngkmax,apwordmax,lmmaxapw,natmtot))
```

DESCRIPTION:

Computes the $(\mathbf{G}+\mathbf{p})$-dependent matching coefficients for the APW basis functions. Inside muffin-tin α, the APW functions are given by

$$\phi^{\alpha}_{\mathbf{G}+\mathbf{p}}(\mathbf{r}) = \sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} \sum_{j=1}^{M_l^{\alpha}} A^{\alpha}_{jlm}(\mathbf{G}+\mathbf{p}) u^{\alpha}_{jl}(r) Y_{lm}(\hat{\mathbf{r}}),$$

where $A^{\alpha}_{jlm}(\mathbf{G}+\mathbf{p})$ is the matching coefficient, M_l^{α} is the order of the APW and u^{α}_{jl} is the radial function. In the interstitial region, an APW function is a plane wave, $\exp(i(\mathbf{G}+\mathbf{p})\cdot\mathbf{r})/\sqrt{\Omega}$, where Ω is the unit cell volume. Ensuring continuity up to the $(M_l^{\alpha}-1)$th derivative across the muffin-tin boundary therefore requires that the matching coefficients satisfy

$$\sum_{j=1}^{M_l^{\alpha}} D_{ij} A^{\alpha}_{jlm}(\mathbf{G}+\mathbf{p}) = b_i \ ,$$

where

$$D_{ij} = \left. \frac{d^{i-1} u^{\alpha}_{jl}(r)}{dr^{i-1}} \right|_{r=R_{\alpha}}$$

and

$$b_i = \frac{4\pi i^l}{\sqrt{\Omega}} |\mathbf{G}+\mathbf{p}|^{i-1} j_l^{(i-1)}(|\mathbf{G}+\mathbf{p}|R_{\alpha}) \exp(i(\mathbf{G}+\mathbf{p})\cdot\mathbf{r}_{\alpha}) Y^{*}_{lm}(\widehat{\mathbf{G}+\mathbf{p}}),$$

with $\mathbf{r}_{\alpha}$ the atomic position and R_{α} the muffin-tin radius. See routine `wavefmt`.

REVISION HISTORY:

```
Created April 2003 (JKD)
Fixed documentation, June 2006 (JKD)
```

1.0.74 chgdist (Source File: chgdist.F90)

INTERFACE:

```
Subroutine chgdist
```

USES:

```
use modmain
```

DESCRIPTION:

Calculated the charge distance between two charge densities of the current and of the last iteration according to the expression

$$\Delta Q = \int_{\Omega} \mathrm{d}^3 r [\rho^{(n)}(\mathbf{r}) - \rho^{(n-1)}(\mathbf{r})].$$

Based on the routine `charge`.

REVISION HISTORY:

```
Created 2010 (Sagmeister)
```

1.0.75 rhovalk (Source File: rhovalk.f90)

INTERFACE:

```
Subroutine rhovalk (ik, evecfv, evecsv)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
ik     : k-point number (in,integer)
evecfv : first-variational eigenvectors (in,complex(nmatmax,nstfv,nspnfv))
evecsv : second-variational eigenvectors (in,complex(nstsv,nstsv))
```

DESCRIPTION:

Generates the partial valence charge density from the eigenvectors at k-point `ik`. In the muffin-tin region, the wavefunction is obtained in terms of its (l, m)-components from both the APW and local-orbital functions. Using a backward spherical harmonic transform (SHT), the wavefunction is converted to real-space and the density obtained from its modulus squared. This density is then transformed with a forward SHT and accumulated in the global variable `rhomt`. A similar proccess is used for the intersitial density in which the wavefunction in real-space is obtained from a Fourier transform of the sum of APW functions. The interstitial density is added to the global array `rhoir`. See routines `wavefmt`, `genshtmat` and `seceqn`.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

1.0.76 gridsize (Source File: gridsize.f90)

INTERFACE:

```
Subroutine gridsize
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Finds the **G**-vector grid which completely contains the vectors with $G < G_{\max}$ and is compatible with the FFT routine. The optimal sizes are given by

$$n_i = \frac{G_{\max}|\mathbf{a}_i|}{\pi} + 1,$$

where $\mathbf{a}_i$ is the ith lattice vector.

REVISION HISTORY:

```
Created July 2003 (JKD)
```

1.0.77 gengvec (Source File: gengvec.f90)

INTERFACE:

```
Subroutine gengvec
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Generates a set of **G**-vectors used for the Fourier transform of the charge density and potential and sorts them according to length. Integers corresponding to the vectors in lattice coordinates are stored, as well as the map from these integer coordinates to the **G**-vector index. A map from the **G**-vector set to the standard FFT array structure is also generated. Finally, the number of **G**-vectors with magnitude less than `gmaxvr` is determined.

REVISION HISTORY:

```
Created October 2002 (JKD)
Increased number of G-vectors to ngrtot, July 2007 (JKD)
```

1.0.78 atom (Source File: atom.f90)

INTERFACE:

```
Subroutine atom (ptnucl, zn, nst, n, l, k, occ, xctype, xcgrad, np, nr, &
& r, eval, rho, vr, rwf)
```

USES:

```
Use modxcifc
```

INPUT/OUTPUT PARAMETERS:

```
ptnucl : .true. if the nucleus is a point particle (in,logical)
zn     : nuclear charge (in,real)
nst    : number of states to solve for (in,integer)
n      : priciple quantum number of each state (in,integer(nst))
l      : quantum number l of each state (in,integer(nst))
k      : quantum number k (l or l+1) of each state (in,integer(nst))
occ    : occupancy of each state (inout,real(nst))
xctype : exchange-correlation type (in,integer)
xcgrad : 1 for GGA functional, 0 otherwise (in,integer)
np     : order of predictor-corrector polynomial (in,integer)
nr     : number of radial mesh points (in,integer)
r      : radial mesh (in,real(nr))
eval   : eigenvalue without rest-mass energy for each state (out,real(nst))
rho    : charge density (out,real(nr))
vr     : self-constistent potential (out,real(nr))
rwf    : major and minor components of radial wavefunctions for each state
         (out,real(nr,2,nst))
```

DESCRIPTION:

Solves the Dirac-Kohn-Sham equations for an atom using the exchange-correlation functional `xctype` and returns the self-consistent radial wavefunctions, eigenvalues, charge densities and potentials. The variable `np` defines the order of polynomial used for performing numerical integration. Requires the exchange-correlation interface routine `xcifc`.

REVISION HISTORY:

```
Created September 2002 (JKD)
Fixed s.c. convergence problem, October 2003 (JKD)
Added support for GGA functionals, June 2006 (JKD)
```

1.0.79 addrhocr (Source File: addrhocr.f90)

INTERFACE:

```
Subroutine addrhocr
```

USES:

```
Use modmain
```

DESCRIPTION:

Adds the core density to the muffin-tin and interstitial densities. A uniform background density is added in the interstitial region to take into account leakage of core charge from the muffin-tin spheres.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

1.0.80 plot3d (Source File: plot3d.f90)

INTERFACE:

```
Subroutine plot3d (plotlabels3d, nf, lmax, ld, rfmt, rfir, plotdef)
```

USES:

```
use modplotlabels
   Use modinput
   use mod_muffin_tin
   use mod_atoms
   use mod_Gvector
   Use FoX_wxml
   use modmpi
```

INPUT/OUTPUT PARAMETERS:

```
plotlabels : plot file number (in,integer)
nf   : number of functions (in,integer)
lmax : maximum angular momentum (in,integer)
ld   : leading dimension (in,integer)
rfmt : real muffin-tin function (in,real(ld,nrmtmax,natmtot,nf))
rfir : real intersitial function (in,real(ngrtot,nf))
```

DESCRIPTION:

Produces a 3D plot of the real functions contained in arrays `rfmt` and `rfir` in the parallelepiped defined by the corner vertices in the global array `vclp3d`. See routine `rfarray`.

REVISION HISTORY:

```
Created June 2003 (JKD)
Modified, October 2008 (F. Bultmark, F. Cricchio, L. Nordstrom)
Modified, February 2011 (D. Nabok)
```

1.0.81 symrvfir (Source File: symrvfir.f90)

Subroutine symrvfir (ngv, rvfir) **USES:**

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
ngv   : number of G-vectors to be used for the Fourier space rotation
        (in,integer)
rvfir : real interstitial vector function (inout,real(ngrtot,ndmag))
```

DESCRIPTION:

Symmetrises a real interstitial vector function. See routines `symrvf` and `symrfir` for details.
REVISION HISTORY:

```
Created July 2007 (JKD)
```

1.0.82 moment (Source File: moment.f90)

INTERFACE:

```
Subroutine moment
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Computes the muffin-tin, interstitial and total moments by integrating the magnetisation.
REVISION HISTORY:

```
Created January 2005 (JKD)
```

1.0.83 rfinp (Source File: rfinp.f90)

INTERFACE:

```
Function rfinp (lrstp, rfmt1, rfmt2, rfir1, rfir2)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
lrstp : radial step length (in,integer)
rfmt1 : first function in real spherical harmonics for all muffin-tins
        (in,real(lmmaxvr,nrmtmax,natmtot))
rfmt2 : second function in real spherical harmonics for all muffin-tins
        (in,real(lmmaxvr,nrmtmax,natmtot))
rfir1 : first real interstitial function in real-space (in,real(ngrtot))
rfir2 : second real interstitial function in real-space (in,real(ngrtot))
```

DESCRIPTION:

Calculates the inner product of two real fuctions over the entire unit cell. The input muffin-tin functions should have angular momentum cut-off `lmaxvr`. In the intersitial region, the integrand is multiplied with the characteristic function, $\tilde{\Theta}(\mathbf{r})$, to remove the contribution from the muffin-tin. See routines `rfmtinp` and `gencfun`.

REVISION HISTORY:

```
Created July 2004 (JKD)
```

1.0.84 reciplat (Source File: reciplat.f90)

INTERFACE:

```
Subroutine reciplat
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Generates the reciprocal lattice vectors from the real-space lattice vectors

$$\mathbf{b}_1 = \frac{2\pi}{s}(\mathbf{a}_2 \times \mathbf{a}_3)$$
$$\mathbf{b}_2 = \frac{2\pi}{s}(\mathbf{a}_3 \times \mathbf{a}_1)$$
$$\mathbf{b}_3 = \frac{2\pi}{s}(\mathbf{a}_1 \times \mathbf{a}_2)$$

and finds the unit cell volume $\Omega = |s|$, where $s = \mathbf{a}_1 \cdot (\mathbf{a}_2 \times \mathbf{a}_3)$.

REVISION HISTORY:

```
Created September 2002 (JKD)
```

1.0.85 findsym (Source File: findsym.f90)

INTERFACE:

```
Subroutine findsym (apl1, apl2, nsym, lspl, lspn, iea)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
apl1 : first set of atomic positions in lattice coordinates
       (in,real(3,maxatoms,maxspecies))
apl2 : second set of atomic positions in lattice coordinates
       (in,real(3,maxatoms,maxspecies))
nsym : number of symmetries (out,integer)
lspl : spatial rotation element in lattice point group for each symmetry
       (out,integer(48))
lspn : spin rotation element in lattice point group for each symmetry
       (out,integer(48))
iea  : equivalent atom index for each symmetry
       (out,integer(iea(natmmax,nspecies,48))
```

DESCRIPTION:

Finds the symmetries which rotate one set of atomic positions into another. Both sets of positions differ only by a translation vector and have the same muffin-tin magnetic fields (stored in the global array `bfcmt`). Any symmetry element consists of a spatial rotation of the atomic position vectors followed by a global magnetic rotation: $\{\alpha_S|\alpha_R\}$. In the case of spin-orbit coupling $\alpha_S = \alpha_R$. The symmetries are returned as indices of elements in the Bravais lattice point group. An index to equivalent atoms is stored in the array `iea`.

REVISION HISTORY:

```
Created April 2007 (JKD)
Fixed use of proper rotations for spin, February 2008 (L. Nordstrom)
```

1.0.86 symvect (Source File: symvect.f90)

INTERFACE:

```
Subroutine symvect (tspin, vc)
```

USES:

```
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
tspin : .true. if the global spin rotations should be used instead of the
        spatial rotations (in,logical)
vc    : vectors in Cartesian coordinates for all atoms (in,real(3,natmtot))
```

DESCRIPTION:

Symmetrises a 3-vector at each atomic site by rotating and averaging over equivalent atoms. Depending on `tspin`, either the spatial or spin part of the crystal symmetries are used.

REVISION HISTORY:

```
Created June 2004 (JKD)
Modified for spin rotations, May 2008 (JKD)
```

1.0.87 symrvf (Source File: symrvf.f90)

INTERFACE:

```
Subroutine symrvf (lrstp, rvfmt, rvfir)
```

USES:

```
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
lrstp : radial step length (in,integer)
rvfmt : real muffin-tin vector field
        (in,real(lmmaxvr,nrmtmax,natmtot,ndmag))
rvfir : real interstitial vector field
        (in,real(ngrtot,ndmag))
```

DESCRIPTION:

Symmetrises a vector field defined over the entire unit cell using the full set of crystal symmetries. If a particular symmetry involves rotating atom 1 into atom 2, then the spatial and spin rotations of that symmetry are applied to the vector field in atom 2 (expressed in spherical harmonic coefficients), which is then added to the field in atom 1. This is repeated for all symmetry operations. The fully symmetrised field in atom 1 is then rotated and copied to atom 2. Symmetrisation of the interstitial part of the field is performed by `symrvfir`. See also `symrfmt` and `findsym`.

REVISION HISTORY:

```
Created May 2007 (JKD)
Fixed problem with improper rotations, February 2008 (L. Nordstrom,
 F. Bultmark and F. Cricchio)
```

1.0.88 dos (Source File: dos.f90)

INTERFACE:

```
Subroutine dos
```

USES:

```
Use modinput
Use modmain
Use FoX_wxml
```

DESCRIPTION:

Produces a total and partial density of states (DOS) for plotting. The total DOS is written to the file `TDOS.OUT` while the partial DOS is written to the file `PDOS_Sss_Aaaaa.OUT` for atom `aaaa` of species `ss`. In the case of the partial DOS, each symmetrised (l, m)-projection is written consecutively and separated by blank lines. If the global variable `lmirep` is `.true.`, then the density matrix from which the (l, m)-projections are obtained is first rotated into a irreducible representation basis, i.e. one that block diagonalises all the site symmetry matrices in the Y_{lm} basis. Eigenvalues of a quasi-random matrix in the Y_{lm} basis which has been symmetrised with the site symmetries are written to `ELMIREP.OUT`. This allows for identification of the irreducible representations of the site symmetries, for example e_g or t_{2g}, by the degeneracies of the eigenvalues. In the plot, spin-up is made positive and spin-down negative. See the routines `gendmat` and `brzint`.

REVISION HISTORY:

```
Created January 2004 (JKD)
```

1.0.89 genlofr (Source File: genlofr.f90)

INTERFACE:

```
Subroutine genlofr
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Generates the local-orbital radial functions. This is done by integrating the scalar relativistic Schrödinger equation (or its energy deriviatives) at the current linearisation energies using the spherical part of the effective potential. For each local-orbital, a linear combination of `lorbord` radial functions is constructed such that its radial derivatives up to order `lorbord`-1 are zero at the muffin-tin radius. This function is normalised and the radial Hamiltonian applied to it. The results are stored in the global array `lofr`.

REVISION HISTORY:

```
Created March 2003 (JKD)
```

1.0.90 vecplot (Source File: vecplot.f90)

INTERFACE:

```
Subroutine vecplot
```

DESCRIPTION:

Use modinput use modplotlabels Outputs a 2D or 3D vector field for plotting. The vector field can be the magnetisation vector field, $\mathbf{m}$; the exchange-correlation magnetic field, $\mathbf{B}_{\rm xc}$; or the electric field $\mathbf{E} \equiv -\nabla V_{\rm C}$. The magnetisation is obtained from the spin density matrix, $\rho_{\alpha\beta}$, by solving

$$\rho_{\alpha\beta}(\mathbf{r}) = \frac{1}{2}\left(n(\mathbf{r})\delta_{\alpha\beta} + \sigma \cdot \mathbf{m}(\mathbf{r})\right),$$

where $n \equiv {\rm tr}\,\rho_{\alpha\beta}$ is the total density. In the case of 2D plots, the magnetisation vectors are still 3D, but are in the coordinate system of the plane.

REVISION HISTORY:

```
Created August 2004 (JKD)
Included electric field plots, August 2006 (JKD)
```

1.0.91 genwfsv (Source File: genwfsv.f90)

INTERFACE:

```
Subroutine genwfsv (tocc, ngp, igpig, evalsvp, apwalm, evecfv, evecsv, &
& wfmt, wfir)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
tocc    : .true. if only occupied wavefunctions are required (in,logical)
ngp     : number of G+p-vectors (in,integer)
igpig   : index from G+p-vectors to G-vectors (in,integer(ngkmax))
evalsvp : second-variational eigenvalue for every state (in,real(nstsv))
apwalm  : APW matching coefficients
          (in,complex(ngkmax,apwordmax,lmmaxapw,natmtot))
evecfv  : first-variational eigenvectors (in,complex(nmatmax,nstfv))
evecsv  : second-variational eigenvectors (in,complex(nstsv,nstsv))
wfmt    : muffin-tin part of the wavefunctions for every state in spherical
          coordinates (out,complex(lmmaxvr,nrcmtmax,natmtot,nspinor,nstsv))
wfir    : interstitial part of the wavefunctions for every state
          (out,complex(ngrtot,nspinor,nstsv))
```

DESCRIPTION:

Calculates the second-variational spinor wavefunctions in both the muffin-tin and interstitial regions for every state of a particular k-point. The wavefunctions in both regions are stored on a real-space grid. A coarse radial mesh is assumed in the muffin-tins with with angular momentum cut-off of `lmaxvr`. If `tocc` is `.true.`, then only the occupied states (those below the Fermi energy) are calculated.

REVISION HISTORY:

```
Created November 2004 (Sharma)
```

1.0.92 bandstr (Source File: bandstr.f90)

INTERFACE:

```
Subroutine bandstr
```

USES:

```
Use modinput
Use modmain
Use FoX_wxml
```

DESCRIPTION:

Produces a band structure along the path in reciprocal-space which connects the vertices in the array `vvlp1d`. The band structure is obtained from the second-variational eigenvalues and is written to the file `BAND.OUT` with the Fermi energy set to zero. If required, band structures are plotted to files `BAND_Sss_Aaaaa.OUT` for atom `aaaa` of species `ss`, which include the band characters for each l component of that atom in columns 4 onwards. Column 3 contains the sum over l of the characters. Vertex location lines are written to `BANDLINES.OUT`.

REVISION HISTORY:

```
Created June 2003 (JKD)
```

1.0.93 writesym (Source File: writesym.f90)

INTERFACE:

```
Subroutine writesym
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Outputs the Bravais, crystal and site symmetry matrices to files `SYMLAT.OUT`, `SYMCRYS.OUT` and `SYMSITE.OUT`, respectively. Also writes out equivalent atoms and related crystal symmetries to `EQATOMS.OUT`.

REVISION HISTORY:

```
Created October 2002 (JKD)
```

1.0.94 rvfcross (Source File: rvfcross.f90)

INTERFACE:

```
Subroutine rvfcross (rvfmt1, rvfmt2, rvfir1, rvfir2, rvfmt3, rvfir3)
```

USES:

```
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
rvfmt1 : first input muffin-tin field (in,real(lmmaxvr,nrmtmax,natmtot,3))
rvfmt2 : second input muffin-tin field (in,real(lmmaxvr,nrmtmax,natmtot,3))
rvfir1 : first input interstitial field (in,real(ngrtot,3))
rvfir2 : second input interstitial field (in,real(ngrtot,3))
rvfmt3 : output muffin-tin field (out,real(lmmaxvr,nrmtmax,natmtot,3))
rvfir3 : output interstitial field (out,real(ngrtot,3))
```

DESCRIPTION:

Given two real vector fields, $\mathbf{f}_1$ and $\mathbf{f}_2$, defined over the entire unit cell, this routine computes the local cross product

$$\mathbf{f}_3(\mathbf{r}) \equiv \mathbf{f}_1(\mathbf{r}) \times \mathbf{f}_2(\mathbf{r}).$$

REVISION HISTORY:

```
Created February 2007 (JKD)
```

1.0.95 writestate (Source File: writestate.f90)

INTERFACE:

```
Subroutine writestate
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Writes the charge density, potentials and other relevant variables to the file `STATE.OUT`. Note to developers: changes to the way the variables are written should be mirrored in `readstate`.

REVISION HISTORY:

```
Created May 2003 (JKD)
```

1.0.96 writepchgs (Source File: writepchgs.F90)

INTERFACE:

```
subroutine writepchgs(fnum,lmax)
```

USES:

```
use modinput
use modmain
```

DESCRIPTION:

Write partial charges to file.

REVISION HISTORY:

```
Created 2010 (Sagmeister)
```

1.0.97 rotzflm (Source File: rotzflm.f90)

INTERFACE:

```
Subroutine rotzflm (rot, lmax, n, ld, zflm1, zflm2)
```

INPUT/OUTPUT PARAMETERS:

```
rot   : rotation matrix (in,real(3,3))
lmax  : maximum angular momentum (in,integer)
n     : number of functions to rotate (in,integer)
ld    : leading dimension (in,integer)
zflm1 : coefficients of complex spherical harmonic expansion for each
        function (in,complex(ld,n))
zflm2 : coefficients of rotated functions (out,complex(ld,n))
```

DESCRIPTION:

Rotates a set of functions

$$f_i(\mathbf{r}) = \sum_{lm} f^i_{lm} Y_{lm}(\hat{\mathbf{r}})$$

for all i, given the coefficients f^i_{lm} and a rotation matrix R. This is done by first the computing the Euler angles (α,β,γ) of R^{-1} (see routine `euler`) and then generating the rotation matrix for spherical harmonics, $D^l_{mm'}(\alpha,\beta,\gamma)$, with which

$$Y_{lm}(\theta',\phi')=\sum_{m'}D^l_{mm'}(\alpha,\beta,\gamma)Y_{lm'}(\theta,\phi),$$

where (θ',ϕ') are the angles (θ,ϕ) rotated by R. The matrix D is given explicitly by

$$D^l_{mm'}(\alpha,\beta,\gamma)=\sum_i\frac{(-1)^i\sqrt{(l+m)!(l-m)!(l+m')!(l-m')!}}{(l-m'-i)!(l+m-i)!i!(i+m'-m)!}\times\left(\cos\frac{\beta}{2}\right)^{2l+m-m'-2i}\left(\sin\frac{\beta}{2}\right)^{2i+m'-m}e^{-i(m\alpha+m'\gamma)},$$

where the sum runs over all i which make the factorial arguments non-negative. For improper rotations, i.e. those which are a combination of a rotation and inversion, the rotation is first made proper with $R\rightarrow -R$ and D is modified with $D^l_{mm'}\rightarrow(-1)^lD^l_{mm'}$.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

1.0.98 occupy (Source File: occupy.f90)

INTERFACE:

```
Subroutine occupy
```

USES:

```
      Use modinput
      Use modmain
#ifdef TETRAOCC_DOESNTWORK
      Use modtetra
#endif
```

DESCRIPTION:

Finds the Fermi energy and sets the occupation numbers for the second-variational states using the routine `fermi`.

REVISION HISTORY:

```
Created February 2004 (JKD)
Modifiactions for tetrahedron method, November 2007 (RGA alias
  Ricardo Gomez-Abal)
Modifications for tetrahedron method, 2007-2010 (Sagmeister)
```

1.0.99 genylmg (Source File: genylmg.f90)

INTERFACE:

```
Subroutine genylmg
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Generates a set of spherical harmonics, $Y_{lm}(\hat{\mathbf{G}})$, with angular momenta up to `lmaxvr` for the set of $\mathbf{G}$-vectors.

REVISION HISTORY:

```
Created June 2003 (JKD)
```

1.0.100 getevalfv (Source File: getevalfv.f90)

INTERFACE:

```
Subroutine getevalfv (vpl, evalfv)
```

USES:

```
Use modmain
Use modinput
Use modmpi
```

DESCRIPTION:

The file where the (first-variational) eigenvalues are stored is `EVALFV.OUT`. It is a direct-access binary file, the record length of which can be determined with the help of the array sizes and data type information. One record of this file has the following structure

k_{lat}	N_{stfv}	N_{spfv}	E

The following table explains the parts of the record in more detail

name	type	shape	description
k_{lat}	real(8)	3	k-point in lattice coordinates
N_{stfv}	integer	1	number of (first-variational) states (without core states)
N_{spfv}	integer	1	first-variational spins (always equals 1)
E	real(8)	$N_{\text{stfv}} \times N_{\text{spfv}}$	(first-variational) eigenvalue array

REVISION HISTORY:

```
Documentation added, Dec 2009 (S. Sagmeister)
```

1.0.101 rfmtinp (Source File: rfmtinp.f90)

INTERFACE:

```
Real (8) Function rfmtinp (lrstp, lmax, nr, r, ld, rfmt1, rfmt2)
```

INPUT/OUTPUT PARAMETERS:

```
lrstp : radial step length (in,integer)
lmax  : maximum angular momentum (in,integer)
nr    : number of radial mesh points (in,integer)
r     : radial mesh (in,real(nr))
ld    : the leading dimension (in,integer)
rfmt1 : first real function inside muffin-tin (in,real(ld,nr))
rfmt2 : second real function inside muffin-tin (in,real(ld,nr))
```

DESCRIPTION:

Calculates the inner product of two real fuctions in the muffin-tin. So given two real functions of the form

$$f(\mathbf{r}) = \sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} f_{lm}(r) R_{lm}(\hat{\mathbf{r}})$$

where R_{lm} are the real spherical harmonics, the function returns

$$I = \int \sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} f^1_{lm}(r) f^2_{lm}(r) r^2 dr \ .$$

The radial integral is performed using a high accuracy cubic spline method. See routines `genrlm` and `fderiv`.

REVISION HISTORY:

```
Created November 2003 (Sharma)
```

1.0.102 symrf (Source File: symrf.f90)

INTERFACE:

```
Subroutine symrf (lrstp, rfmt, rfir)
```

USES:

```
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
lrstp : radial step length (in,integer)
rfmt  : real muffin-tin function (inout,real(lmmaxvr,nrmtmax,natmtot))
rfir  : real intersitial function (inout,real(ngrtot))
```

DESCRIPTION:

Symmetrises a real scalar function defined over the entire unit cell using the full set of crystal symmetries. In the muffin-tin of a particular atom the spherical harmonic coefficients of every equivlent atom are rotated and averaged. The interstitial part of the function is first Fourier transformed to G-space, and then averaged over each symmetry by rotating the Fourier coefficients and multiplying them by a phase factor corresponding to the symmetry translation. See routines `symrfmt` and `symrfir`.

REVISION HISTORY:

```
Created May 2007 (JKD)
```

1.0.103 updatpos (Source File: updatpos.f90)

INTERFACE:

```
Subroutine updatpos
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Updates the current atomic positions according to the force on each atom. If $\mathbf{r}_{ij}^m$ is the position and $\mathbf{F}_{ij}^m$ is the force acting on it for atom j of species i and after time step m, then the new position is calculated by

$$\mathbf{r}_{ij}^{m+1}=\mathbf{r}_{ij}^m+\tau_{ij}^m\left(\mathbf{F}_{ij}^m+\mathbf{F}_{ij}^{m-1}\right),$$

where τ_{ij}^m is a parameter governing the size of the displacement. If $\mathbf{F}_{ij}^m\cdot\mathbf{F}_{ij}^{m-1}>0$ then τ_{ij}^m is increased, otherwise it is decreased.

REVISION HISTORY:

```
Created June 2003 (JKD)
```

1.0.104 getevecsv (Source File: getevecsv.f90)

INTERFACE:

```
Subroutine getevecsv (vpl, evecsv)
```

USES:

```
Use modmain
Use modinput
Use modmpi
```

DESCRIPTION:

The file where the (second-variational) eigenvectors are stored is `EVECSV.OUT`. It is a direct-access binary file, the record length of which can be determined with the help of the array sizes and data type information. One record of this file has the following structure

$k_{\rm lat}$	$N_{\rm stsv}$	Φ

The following table explains the parts of the record in more detail

name	type	shape	description
$k_{\rm lat}$	real(8)	3	k-point in lattice coordinates
$N_{\rm stsv}$	integer	1	number of (second-variational) states (without core states)
Φ	complex(8)	$N_{\rm stsv} \times N_{\rm stsv}$	(second-variational) eigenvector array

REVISION HISTORY:

```
Documentation added, Dec 2009 (S. Sagmeister)
```

1.0.105 autoradmt (Source File: autoradmt.f90)

INTERFACE:

```
Subroutine autoradmt
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Automatically determines the muffin-tin radii from the formula

$$R_i \propto 1 + \zeta |Z_i|^{1/3},$$

where Z_i is the atomic number of the ith species, ζ is a user-supplied constant (~ 0.625). The parameter ζ is stored in `rmtapm(1)` and the value which governs the distance between the muffin-tins is stored in `rmtapm(2)`. When `rmtapm(2)` $= 1$, the closest muffin-tins will touch.

REVISION HISTORY:

```
Created March 2005 (JKD)
Changed the formula, September 2006 (JKD)
```

1.0.106 ggair (Source File: ggair.f90)

INTERFACE:

```
Subroutine ggair (grhoir, gupir, gdnir, g2upir, g2dnir, g3rhoir, &
& g3upir, g3dnir)
```

INPUT/OUTPUT PARAMETERS:

```
  Use modinput
grhoir  : |grad rho| (out,real(ngrtot))
gupir   : |grad rhoup| (out,real(ngrtot))
gdnir   : |grad rhodn| (out,real(ngrtot))
g2upir  : grad^2 rhoup (out,real(ngrtot))
g2dnir  : grad^2 rhodn (out,real(ngrtot))
g3rhoir : (grad rho).(grad |grad rho|) (out,real(ngrtot))
g3upir  : (grad rhoup).(grad |grad rhoup|) (out,real(ngrtot))
g3dnir  : (grad rhodn).(grad |grad rhodn|) (out,real(ngrtot))
```

DESCRIPTION:

Computes $|\nabla\rho|$, $|\nabla\rho^{\uparrow}|$, $|\nabla\rho^{\downarrow}|$, $\nabla^2\rho^{\uparrow}$, $\nabla^2\rho^{\downarrow}$, $\nabla\rho\cdot(\nabla|\nabla\rho|)$, $\nabla\rho^{\uparrow}\cdot(\nabla|\nabla\rho^{\uparrow}|)$ and $\nabla\rho^{\downarrow}\cdot(\nabla|\nabla\rho^{\downarrow}|)$ for the interstitial charge density, as required by the generalised gradient approximation functional for spin-polarised densities. In the case of spin unpolarised calculations, $|\nabla\rho|$, $\nabla^2\rho$ and $\nabla\rho\cdot(\nabla|\nabla\rho|)$ are returned in the arrays `gupir`, `g2upir` and `g3upir`, respectively, while `grhoir`, `gdnir`, `g2dnir`, `g3rhoir` and `g3dnir` are not referenced. See routines `potxc` and `modxcifc`.

REVISION HISTORY:

```
Created October 2004 (JKD)
```

1.0.107 ggamt (Source File: ggamt.f90)

INTERFACE:

```
Subroutine ggamt (is, ia, grhomt, gupmt, gdnmt, g2upmt, g2dnmt, &
& g3rhomt, g3upmt, g3dnmt)
```

USES:

```
  Use modinput
  Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
is      : species number (in,integer)
ia      : atom number (in,integer)
grhomt  : |grad rho| (out,real(lmmaxvr,nrmtmax))
gupmt   : |grad rhoup| (out,real(lmmaxvr,nrmtmax))
```

```
gdnmt   : |grad rhodn| (out,real(lmmaxvr,nrmtmax))
g2upmt  : grad^2 rhoup (out,real(lmmaxvr,nrmtmax))
g2dnmt  : grad^2 rhodn (out,real(lmmaxvr,nrmtmax))
g3rhomt : (grad rho).(grad |grad rho|) (out,real(lmmaxvr,nrmtmax))
g3upmt  : (grad rhoup).(grad |grad rhoup|) (out,real(lmmaxvr,nrmtmax))
g3dnmt  : (grad rhodn).(grad |grad rhodn|) (out,real(lmmaxvr,nrmtmax))
```

DESCRIPTION:

Computes $|\nabla\rho|$, $|\nabla\rho^{\uparrow}|$, $|\nabla\rho^{\downarrow}|$, $\nabla^2\rho^{\uparrow}$, $\nabla^2\rho^{\downarrow}$, $\nabla\rho\cdot(\nabla|\nabla\rho|)$, $\nabla\rho^{\uparrow}\cdot(\nabla|\nabla\rho^{\uparrow}|)$ and $\nabla\rho^{\downarrow}\cdot(\nabla|\nabla\rho^{\downarrow}|)$ for a muffin-tin charge density, as required by the generalised gradient approximation functional for spin-polarised densities. In the case of spin unpolarised calculations, $|\nabla\rho|$, $\nabla^2\rho$ and $\nabla\rho\cdot(\nabla|\nabla\rho|)$ are returned in the arrays `gupmt`, `g2upmt` and `g3upmt`, respectively, while `grhomt`, `gdnmt`, `g2dnmt`, `g3rhomt` and `g3dnmt` are not referenced. The input densities are in terms of real spherical harmonic expansions but the returned functions are in spherical coordinates. See routines `potxc`, `modxcifc`, `gradrfmt`, `genrlm` and `genshtmat`.

REVISION HISTORY:

```
Created April 2004 (JKD)
```

1.0.108 writekpts (Source File: writekpts.f90)

INTERFACE:

```
Subroutine writekpts
```

USES:

```
Use modmain
```

DESCRIPTION:

Writes the k-points in lattice coordinates, weights and number of $\mathbf{G}+\mathbf{k}$-vectors to the file `KPOINTS.OUT`.

REVISION HISTORY:

```
Created June 2003 (JKD)
```

1.0.109 genylmgq (Source File: genylmgq.F90)

INTERFACE:

```
Subroutine genylmgq (iq, lmax)
```

USES:

```
Use modmain
Use modxs
```

DESCRIPTION:

Generates a set of spherical harmonics, $Y_{lm}(\widehat{\mathbf{G}+\mathbf{q}})$, with angular momenta up to `lmax` for the set of $\mathbf{G}+\mathbf{q}$-vectors. Based upon the routine genylmg.

REVISION HISTORY:

```
Created October 2006 (Sagmeister)
```

1.0.110 pade (Source File: pade.F90)

INTERFACE:

```
Subroutine pade (m, z, n, iw, ih, h)
```

INPUT/OUTPUT PARAMETERS:

```
m     : number of values in array below (in,integer)
z     : values for which analytic continuation is to be performed
        (in,complex(m))
n     : number of values in arrays below (in,integer)
iw    : values for which function is evaluated
        (in,complex(n))
ih    : function evaluated at values "iw" (in,complex(n))
h     : function evaluated at values "z" (out,complex(n))
```

USES:

```
Use m_ctdfrac
```

DESCRIPTION:

Implementation of a Padé approximant using Thiele's reciprocal-difference method. This routine takes a complex function ($f_n = f(z_n)$) evaluated at an initial set of arguments, (z_n) approximates the function with the help of Padé approximants, and evaluates (extrapolates/rotates) this approximation at a given set of arguments (z). The N-point Padé approximant then reads

$$P_N(z) = \cfrac{a_1}{1 + \cfrac{a_2(z-z_1)}{\cdots + \cfrac{a_n(z-z_{N-1})}{1+(z-z_N)g_{N+1}(z)}}}$$

where

$$g_n(z) = \frac{g_{n-1}(z_{n-1}) - g_{n-1}(z)}{(z-z_{n-1})g_{n-1}(z)},\ n \geq 2$$

and

$$a_n = g_n(z_n),\ g_1(z_n) = f_n,\ n = 1, \ldots, N.$$

For simplicity, the expression $g_{N+1}(z)$ is set to zero in the continued fraction expression of the approximant. Expressions are taken from K. Lee and K. Chang, Phys. Rev. B 54, R8285 (1996). See also H. Vidberg and J. Serene, J. Low Temp. Phys., 29, 179 (1977).

REVISION HISTORY:

```
Created December 2006 (S. Sagmeister)
Documentation added, July 2011 (S. Sagmeister)
```

1.0.111 transijst (Source File: transijst.F90)

INTERFACE:

```
logical function transijst(ik,ist,jst)
```

USES:

```
use modinput
```

DESCRIPTION:

This function returns "true" if the transition specified by the input k-point initial and final state matches with the ones specified in the `transitions` element. Wether a state is included or excluded depends on the sequence of the transitions specified (using the "include" and "exclude" attribute).

REVISION HISTORY:

```
Created July 2010 (Sagmeister)
```

1.0.112 kernxc_bse (Source File: kernxc_bse.F90)

INTERFACE:

```
Subroutine kernxc_bse
```

USES:

```
Use modinput
Use modmain
Use modtetra
Use modxs
Use m_xsgauntgen
Use m_findgntn0
Use m_writegqpts
Use m_genwgrid
Use m_xszoutpr3
Use m_getpemat
Use m_getunit
Use m_genfilname
```

INPUT/OUTPUT PARAMETERS:

```
oct   : optical diagonal tensor component (in,integer)
```

DESCRIPTION:

BSE-kernel of A. Marini, Phys. Rev. Lett. 91, 256402 (2003)
REVISION HISTORY:

```
Created March 2008 (Sagmeister)
```

1.0.113 ctdfrac (Source File: ctdfrac.F90)

INTERFACE:

```
Subroutine ctdfrac (n, a, b, f)
```

INPUT/OUTPUT PARAMETERS:

```
n     : depth of continued fraction (in,integer)
a     : a-coefficients (in,complex(n))
b     : b-coefficients (in,complex(0:n))
f     : continued fraction result
```

DESCRIPTION:

Straight forward evaluation of a continued fraction with depth n

$$b_0 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{\cdots + \cfrac{a_n}{b_n}}}}$$

without any checking of convergence.
REVISION HISTORY:

```
Created March 2006 (Sagmeister)
```

1.0.114 fxc_bse_ma03 (Source File: fxc_bse_ma03.F90)

INTERFACE:

```
Subroutine fxc_bse_ma03 (msiz, oct, sw, iw, fxc)
```

USES:

```
Use modinput
Use mod_constants, Only: zzero
Use modmpi, Only:
Use modxs, Only: unitout, bzsampl
```

```
Use invert
Use m_xsgauntgen
Use m_findgntn0
Use m_writegqpts
Use m_genfilname
Use m_getunit
```

INPUT/OUTPUT PARAMETERS:

```
msiz  : matrix size of local field effects (in,integer)
sw    : true for inclusion of local field effects (in,logical)
alpha : real constant (in,real)
fxc   : xc-kernel Fourier coefficients (out,complex(:,:))
```

DESCRIPTION:

BSE-kernel of A. Marini, Phys. Rev. Lett. 91, 256402 (2003). Interface function.

REVISION HISTORY:

```
Created March 2008 (Sagmeister)
```

1.0.115 wavefmt_lo (Source File: wavefmt_lo.F90)

INTERFACE:

```
Subroutine wavefmt_lo (lrstp, lmax, is, ia, ngp, evecfv, ld, wfmt)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
lrstp  : radial step length (in,integer)
lmax   : maximum angular momentum required (in,integer)
is     : species number (in,integer)
ia     : atom number (in,integer)
ngp    : number of G+p-vectors (in,integer)
evecfv : first-variational eigenvector (in,complex(nmatmax))
ld     : leading dimension (in,integer)
wfmt   : muffin-tin wavefunction (out,complex(ld,*))
```

DESCRIPTION:

Muffin-tin wavefunction built up by local orbital contribution only. Based upon the routine `wavefmt`.

REVISION HISTORY:

```
Created May 2008 (Sagmeister)
```

1.0.116 df (Source File: df.F90)

INTERFACE:

```
Subroutine df
```

USES:

```
Use modinput
Use modmain
Use modxs
Use modmpi
Use m_writegqpts
Use m_xsgauntgen
Use m_findgntn0
Use m_genfilname
```

DESCRIPTION:

Control routine for setting up the Kohn-Sham response function or the microscopic dielectric function/matrix for all specified **q**-points. Can be run with MPI parallelization for **q**-points.

REVISION HISTORY:

```
Created March 2006 (Sagmeister)
```

1.0.117 gentetlinkp (Source File: gentetlinkp.F90)

INTERFACE:

```
Subroutine gentetlinkp (vpl, tqw)
```

USES:

```
Use modinput
Use modmain
Use modtetra
```

DESCRIPTION:

Interface to the `gentetlink` routine.

REVISION HISTORY:

```
Created July 2008 (Sagmeister)
```

1.0.118 connecta (Source File: connecta.F90)

INTERFACE:

```
Subroutine connecta (cvec, nv, np, vvl, vpl, dv, dp)
```

INPUT/OUTPUT PARAMETERS:

```
cvec : matrix of (reciprocal) lattice vectors stored column-wise
       (in,real(3,3))
nv   : number of vertices (in,integer)
np   : number of connecting points (in,integer)
vvl  : vertex vectors in lattice coordinates (in,real(3,nv))
vpl  : connecting point vectors in lattice coordinates (out,real(3,np))
dv   : cummulative distance to each vertex (out,real(nv))
dp   : cummulative distance to each connecting point (out,real(np))
```

DESCRIPTION:

Generates a set of points which interpolate between a given set of vertices. Vertex points are supplied in lattice coordinates in the array `vvl` and converted to Cartesian coordinates with the matrix `cvec`. Interpolating points are stored in the array `vpl`. The cummulative distances to the vertices and points along the path are stored in arrays `dv` and `dp`, respectively. The given vertex points are contained in the set of output vectors. Based upon the routine `connect`.

REVISION HISTORY:

```
Created June 2007 (Sagmeister)
```

1.0.119 bse (Source File: bse.F90)

INTERFACE:

```
Subroutine bse
```

USES:

```
Use modinput
Use modmain
use modmpi
Use modxs
Use m_genwgrid
Use m_getpmat
Use m_genfilname
Use m_getunit
Use m_genloss
Use m_gensigma
Use m_gensumrls
Use m_writeeps
```

```
Use m_writeloss
Use m_writesigma
Use m_writesumrls
```

DESCRIPTION:

Solves the Bethe-Salpeter equation (BSE). The BSE is treated as equivalent effective eigenvalue problem (thanks to the spectral theorem that can be applied to the original BSE in the case of a statically screened Coulomb interaction). The effective BSE-Hamiltonian consists of three parts originating from different sources. It reads

$$H^{\mathrm{eff}} = H^{\mathrm{diag}} + 2H^{\mathrm{x}} + H^{\mathrm{c}},$$

where H^{diag} is the diagonal part stemming from the independent particle transitions, H^{x} denotes the exchange-term caused by the unscreened (bare) Coulomb interaction, whereas H^{c} accounts for the particle-hole correlations and is originating from the screened Coulomb interaction. For the purpose of describing independent particle transitions with the BSE only the diagonal term is referred to:

$$H^{\mathrm{eff}} = H^{\mathrm{diag}}.$$

By neglecting the correlation part in the effective Hamiltonian we arrive at the *random phase approximation* (RPA)

$$H^{\mathrm{eff}} = H^{\mathrm{diag}} + 2H^{\mathrm{x}}.$$

Investigations on the spin-structure of the BSE-Hamiltonian show that there are tow channels, namely the *singlet*-channel as solution to the Hamiltonian

$$H^{\mathrm{eff}} = H^{\mathrm{diag}} + 2H^{\mathrm{x}} + H^{\mathrm{c}}$$

and a *triplet* channel with the exchange-part being absent.

$$H^{\mathrm{eff}} = H^{\mathrm{diag}} + H^{\mathrm{c}}.$$

The equation of the eigenvalue problem is given by

$$\sum_{v'c'\mathbf{k}'} H^{\mathrm{eff}}_{vc\mathbf{k},v'c'\mathbf{k}'} A^{\lambda}_{v'c'\mathbf{k}'} = \varepsilon_{\lambda} A^{\lambda}_{vc\mathbf{k}}.$$

For the diagonalization of the Hamiltonian, a LAPACK-routine (`zheevx`) is invoked to obtain the eigenvalues ε_{λ} and eigenvectors $A^{\lambda}_{vc\mathbf{k}}$ (alternatively, a time-evolution method shall be implemented to obtain the macroscopic dielectric function directly). Consequently, the transition amplitudes t_{λ} are calculated according to

$$t^{i}_{\lambda} = \left| \sum_{vc\mathbf{k}} A^{\lambda}_{vc\mathbf{k}} \frac{p^{i}_{vc\mathbf{k}}}{\varepsilon_{c\mathbf{k}} - \varepsilon_{v\mathbf{k}}} \right|^2.$$

Here, the index i labels the polarization and the matrix elements $p^{i}_{vc\mathbf{k}}$ are the ones for the i-th component of the momentum operator in Cartesian coordinates. The macroscopic dielectric function (MDF) is obtained by the realation

$$\mathrm{Im}\ \epsilon^{i}_{\mathrm{M}}(\omega) = \frac{8\pi^2}{V} \sum_{\lambda} t^{i}_{\lambda} \delta(\omega - \varepsilon_{\lambda} + \Delta),$$

where $\epsilon_{\rm M}^i$ is the MDF for the i-th polarization, V denotes the crystal volume and Δ is a constant shift of the conduction bands (scissors shift). The delta-function in the latter expression is convoluted with a (symmetrized) Lorentzian

$$\pi\delta(\omega-\omega_0)=\lim_{\eta\to 0}\left[\frac{\eta}{(\omega-\omega_0)^2+\eta^2}+\frac{\eta}{(-\omega-\omega_0)^2-\eta^2}\right]=\pi\delta(\omega-\omega_0)+\pi\delta(\omega+\omega_0)$$

which is true for $\omega\geq 0$ if $\omega_0>0$. In doing so, the analytic property ${\rm Im}\epsilon_{\rm M}(0)=0$ is fulfilled. The broadening η in the latter expression is adjusted by the `broad` parameter. (All parts of the documentation written by S. Sagmeister are part of the author's PhD-thesis.)

REVISION HISTORY:

```
Created June 2008 (S. Sagmeister)
Addition of explicit energy ranges for states below and above the Fermi
   level for the treatment of core excitations (using local orbitals).
   October 2010 (Weine Olovsson)
```

1.0.120 zoutpr (Source File: zoutpr.F90)

INTERFACE:

```
Subroutine zoutpr (n1, n2, alpha, x, y, a)
```

INPUT/OUTPUT PARAMETERS:

```
n1,n2 : size of vectors and matrix, respectively (in,integer)
alpha : complex constant (in,complex)
x     : first input vector (in,complex(n1))
y     : second input vector (in,complex(n2))
a     : output matrix (out,complex(n1,n2))
```

DESCRIPTION:

Performs the rank-2 operation

$$A_{ij}\to\alpha\mathbf{x}_i^*\mathbf{y}_j+A_{ij}.$$

REVISION HISTORY:

```
Created April 2008 (Sagmeister)
```

1.0.121 xcd_pwca (Source File: xcd_pwca.F90)

INTERFACE:

```
Subroutine xcd_pwca (n, rho, dvx, dvc)
```

INPUT/OUTPUT PARAMETERS:

```
n     : number of density points (in,integer)
rho   : charge density (in,real(n))
dvx   : exchange potential derivative (out,real(n))
dvc   : correlation potential derivative (out,real(n))
```

DESCRIPTION:

Spin-unpolarised exchange-correlation potential derivative of the Perdew-Wang parameterisation of the Ceperley-Alder electron gas, Phys. Rev. B 45, 13244 (1992) and Phys. Rev. Lett. 45, 566 (1980). Based upon the routine `xc_pwca`.

REVISION HISTORY:

```
Created February 2007 (Sagmeister)
```

1.0.122 wavefmt_apw (Source File: wavefmt_apw.F90)

INTERFACE:

```
Subroutine wavefmt_apw (lrstp, lmax, is, ia, ngp, apwalm, evecfv, ld, &
& wfmt)
```

USES:

```
  Use modinput
  Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
lrstp  : radial step length (in,integer)
lmax   : maximum angular momentum required (in,integer)
is     : species number (in,integer)
ia     : atom number (in,integer)
ngp    : number of G+p-vectors (in,integer)
apwalm : APW matching coefficients
         (in,complex(ngkmax,apwordmax,lmmaxapw,natmtot))
evecfv : first-variational eigenvector (in,complex(nmatmax))
ld     : leading dimension (in,integer)
wfmt   : muffin-tin wavefunction (out,complex(ld,*))
```

DESCRIPTION:

Muffin-tin wavefunction built up by APW contribution only. Based upon the routine `wavefmt`.

REVISION HISTORY:

```
Created May 2008 (Sagmeister)
```

1.0.123 findgroupq (Source File: findgroupq.F90)

INTERFACE:

```
Subroutine findgroupq (tfbz, vql, epslat, bvec, symlat, nsymcrys, &
& lsplsymc, nsymcrysq, scqmap, ivscwrapq)
```

DESCRIPTION:

Find the (little) group of **q** (which includes finding the small group of **q**). All symmetries, where the rotational part transforms **q** into an equivalent vector are collected for the small group of **q**. Inclusion of non-primitive translations yields the little group of **q**.

REVISION HISTORY:

```
Created March 2006 (Sagmeister)
```

1.0.124 writeangmom (Source File: writeangmom.F90)

INTERFACE:

```
Subroutine writeangmom (un)
```

USES:

```
Use modinput
Use modmain
Use modxs
```

INPUT/OUTPUT PARAMETERS:

DESCRIPTION:

Outputs information about the angular momentum cutoffs to the file `INFOXS.OUT`.

REVISION HISTORY:

```
Created October 2006 (Sagmeister)
```

1.0.125 gradzfmtr (Source File: gradzfmtr.F90)

INTERFACE:

```
Subroutine gradzfmtr (lmax, nr, r, l1, m1, ld1, ld2, fmt, gfmt)
```

INPUT/OUTPUT PARAMETERS:

```
lmax  : maximum angular momentum (in,integer)
nr    : number of radial mesh points (in,integer)
r     : radial mesh (in,real(nr))
ld1   : leading dimension 1 (in,integer)
ld2   : leading dimension 2 (in,integer)
fmt  : real muffin-tin function (in,real(nr))
gfmt : gradient of zfmt (out,real(ld1,ld2,3))
```

DESCRIPTION:

Calculates the gradient of a muffin-tin function with real spherical harmonics expansion coefficients, $f(r)$, corresponding to a specific lm-combination. The gradient is given in a spherical harmonics representation. The y-component is divided by i to be expressed as a real number. See routine `gradzfmt`.

REVISION HISTORY:

```
Created April 2008 (Sagmeister)
```

1.0.126 xszoutpr3 (Source File: xszoutpr3.F90)

INTERFACE:

```
Subroutine xszoutpr3 (n1, n2, alpha, x, y, a)
```

INPUT/OUTPUT PARAMETERS:

```
n1,n2 : size of vectors and matrix, respectively (in,integer)
alpha : complex constant (in,complex)
x     : first input vector (in,complex(n1))
y     : second input vector (in,complex(n2))
a     : output matrix (out,complex(n1,n2))
```

DESCRIPTION:

Performs the rank-2 operation

$$A_{ij} \rightarrow \alpha \mathbf{x}_i \mathbf{y}_j + A_{ij}.$$

REVISION HISTORY:

```
Created March 2008 (Sagmeister)
```

1.0.127 kernxc_bse3 (Source File: kernxc_bse3.F90)

INTERFACE:

```
Subroutine kernxc_bse3
```

USES:

```
Use modinput
Use modmain
Use modxs
Use m_getevalsvr
Use m_getpemat
Use m_xsgauntgen
Use m_findgntn0
Use m_genwgrid
Use m_xszoutpr3
Use m_genfilname
Use m_getunit
Use m_xszoutpr3
```

INPUT/OUTPUT PARAMETERS:

DESCRIPTION:

BSE-kernel of A. Marini, Phys. Rev. Lett. 91, 256402 (2003)
REVISION HISTORY:

```
Created March 2009 (Sagmeister)
```

1.0.128 fxc_alda_check (Source File: fxc_alda_check.F90)

INTERFACE:

```
Subroutine fxc_alda_check
```

USES:

```
Use modinput
Use modmain, Only: lmmaxvr, ngrtot, rhoir, rhomt, vxcir, vxcmt,xctype
Use modxcifc, Only: xcifc
Use modfxcifc
```

DESCRIPTION:

Checks the validity of the analytical expressions for the ALDA exchange-correlation kernel.
REVISION HISTORY:

```
Created April 2007 (Sagmeister)
```

1.0.129 dfq (Source File: dfq.F90)

INTERFACE:

```
Subroutine dfq (iq)
```

USES:

```
use ioarray
      Use modinput
      Use mod_constants
      Use mod_eigenvalue_occupancy
      Use mod_misc
      Use mod_Gkvector
      Use mod_APW_LO
      Use mod_Gvector
      Use mod_kpoint
      Use mod_qpoint
      Use mod_lattice
      Use mod_DOS_optics_response
      Use modxs
      Use modtetra
      Use modmpi
      Use m_genwgrid
      Use m_getpemat
      Use m_dftim
      Use m_gettetcw
      Use m_putx0
      Use m_getunit
      Use m_writevars
      Use m_filedel
      Use m_genfilname
```

DESCRIPTION:

Calculates the symmetrized Kohn-Sham response function $\chi^0_{\mathbf{G}\mathbf{G}'}(\mathbf{q},\omega)$ for one $\mathbf{q}$-point according to

$$\chi^0_{\mathbf{G}\mathbf{G}'}(\mathbf{q},\omega) = \sum_{ou\mathbf{k}} \left[M^{\mathbf{G}}_{ou\mathbf{k}}(\mathbf{q}) M^{\mathbf{G}'}_{ou\mathbf{k}}(\mathbf{q})^* w_{ou\mathbf{k}}(\mathbf{q},\omega) + M^{\mathbf{G}}_{uo\mathbf{k}}(\mathbf{q}) M^{\mathbf{G}'}_{uo\mathbf{k}}(\mathbf{q})^* w_{uo\mathbf{k}}(\mathbf{q},\omega) \right]$$

It is related to the Kohn-Sham response function $\chi^0_{\mathbf{G}\mathbf{G}'}(\mathbf{q},\omega)$ by

$$\chi^0_{\mathbf{G}\mathbf{G}'}(\mathbf{q},\omega) = v^{-\frac{1}{2}}_{\mathbf{G}}(\mathbf{q}) \bar{\chi}^0_{\mathbf{G}\mathbf{G}'}(\mathbf{q},\omega) v^{-\frac{1}{2}}_{\mathbf{G}'}(\mathbf{q})$$

and is well defined in the limit as $\mathbf{q}$ goes to zero. The symmetrized matrix elements are defined as

$$M^{\mathbf{G}}_{ou\mathbf{k}}(\mathbf{q}) = v^{-\frac{1}{2}}_{\mathbf{G}}(\mathbf{q}) \bar{M}^{\mathbf{G}}_{ou\mathbf{k}}(\mathbf{q}),$$

where

$$\bar{M}^{\mathbf{G}}_{ou\mathbf{k}}(\mathbf{q}) = \langle o\mathbf{k} | e^{-i(\mathbf{G}+\mathbf{q})\mathbf{r}} | u\mathbf{k}+\mathbf{q} \rangle.$$

For $\mathbf{G} = 0$ we have to consider three vectors stemming from the limits as $\mathbf{q} \to 0$ along the Cartezian basis vectors $\mathbf{e_i}$, i.e., we can think of $\mathbf{0}_1, \mathbf{0}_2, \mathbf{0}_3$ in place of $\mathbf{0}$. The weights

$w_{\mathrm{ou}\mathbf{k}}(\mathbf{q},\omega)$ are defined as

$$w_{nm\mathbf{k}}(\mathbf{q},\omega)=\lambda_{\mathbf{k}}\frac{f_{n\mathbf{k}}-f_{m\mathbf{k}+\mathbf{q}}}{\varepsilon_{n\mathbf{k}}-\varepsilon_{m\mathbf{k}+\mathbf{q}}+\Delta_{n\mathbf{k}}-\Delta_{m\mathbf{k}+\mathbf{q}}+\omega+i\eta}$$

in the case where we use a Lorentzian broadening η. In the above expression $\lambda_{\mathbf{k}}$ is the weight of the $\mathbf{k}$-point, $\varepsilon_{n\mathbf{k}}$ and $\varepsilon_{m\mathbf{k}+\mathbf{q}}$ are the DFT Kohn-Sham energies, $\Delta_{n\mathbf{k}}$ and $\Delta_{m\mathbf{k}+\mathbf{q}}$ are the scissors corrections that are non-zero in the case where $m\mathbf{k}+\mathbf{q}$ corresponds to a conduction state. The indices o and u denote *at least partially occupied* and *at least partially unoccupied* states, respectively. The symmetrized Kohn-Sham response function can also be calculated for imaginary frequencies $i\omega$ without broadening η. In this case the replacement

$$\omega+i\eta\mapsto i\omega$$

is applied to the expressions for the weights. Optionally, the weights can be calculated with the help of the linear tetrahedron method (including Bloechl's correction). This routine can be run with MPI parallelization for energies.

REVISION HISTORY:

```
Created March 2005 (Sagmeister)
Added band and k-point analysis, 2007-2008 (Sagmeister)
```

1.0.130 writepmatxs (Source File: writepmatxs.F90)

INTERFACE:

```
Subroutine writepmatxs
```

USES:

```
 Use modinput
 Use modmain, Only: nkpt, ngkmax, apwordmax, lmmaxapw, natmtot, &
& nmatmax, nstfv, nstsv, nlotot, nlomax, lolmax, task, vkl, vgkl, &
& ngk, gkc, tpgkc, sfacgk, igkig, vgkc, filext
 Use modmpi
 Use modxs
 Use m_putpmat
 Use m_genfilname
```

DESCRIPTION:

Calculates the momentum matrix elements using routine `genpmat` and writes them to direct access file `PMAT.OUT`, `PMAT_XS.OUT` or `PMAT_SCR.OUT` depending on the context of the execution.

REVISION HISTORY:

```
Created 2006 (S. Sagmeister)
Modifications, August 2010 (S. Sagmeister)
```

1.0.131 gengqvec (Source File: gengqvec.F90)

INTERFACE:

```
Subroutine gengqvec (iq, vpl, vpc, ngp, igpig, vgpl, vgpc, gpc, tpgpc)
```

USES:

```
Use modinput
Use modmain
Use modxs
```

INPUT/OUTPUT PARAMETERS:

```
vpl   : p-point vector in lattice coordinates (in,real(3))
vpc   : p-point vector in Cartesian coordinates (in,real(3))
ngp   : number of G+p-vectors returned (out,integer)
igpig : index from G+p-vectors to G-vectors (out,integer(ngkmax))
vgpl  : G+p-vectors in lattice coordinates (out,real(3,ngkmax))
vgpc  : G+p-vectors in Cartesian coordinates (out,real(3,ngkmax))
gpc   : length of G+p-vectors (out,real(ngkmax))
tpgpc : (theta, phi) coordinates of G+p-vectors (out,real(2,ngkmax))
```

DESCRIPTION:

Generates a set of $\mathbf{G}+\mathbf{p}$-vectors for the input $\mathbf{p}$-point with length less than `gkmax`. These are used as the plane waves in the APW functions. Also computes the spherical coordinates of each vector. Based on `gengpvec`.

REVISION HISTORY:

```
Created October 2006 (Sagmeister)
```

1.0.132 xszoutpr (Source File: xszoutpr.F90)

INTERFACE:

```
Subroutine xszoutpr (n1, n2, alpha, x, y, a)
```

INPUT/OUTPUT PARAMETERS:

```
n1,n2 : size of vectors and matrix, respectively (in,integer)
alpha : complex constant (in,complex)
x     : first input vector (in,complex(n1))
y     : second input vector (in,complex(n2))
a     : output matrix (out,complex(n1,n2))
```

DESCRIPTION:

Performs the rank-2 operation

$$A_{ij} \rightarrow \alpha \mathbf{x}_i^* \mathbf{y}_j + A_{ij}.$$

REVISION HISTORY:

Created March 2008 (Sagmeister)

1.0.133 findsymi (Source File: findsymi.F90)

INTERFACE:

```
Subroutine findsymi (epslat, maxsymcrys, nsymcrys, symlat, lsplsymc, &
& vtlsymc, isymlat, scimap)
```

DESCRIPTION:

Throughout the code the symmetries are understood to be applied in a way

$$(\alpha_S|\alpha_R|\mathbf{t})\mathbf{x} = \alpha_S\alpha_R(\mathbf{x}+\mathbf{t})$$

which is different from the commonly used definition $\{\alpha|\tau\}x = \alpha x + \tau$ – see routine `findsymcrys`. This difference affects the inverse of the fractional translation but has no effect on the inverse of the rotational part, so the inverse spacegroup symmetry operations are the same for both definitions.

REVISION HISTORY:

Created April 2007 (Sagmeister)

1.0.134 writeqpts (Source File: writeqpts.F90)

INTERFACE:

```
Subroutine writeqpts
```

USES:

```
Use modinput
Use modmain
Use modxs
Use m_getunit
Use m_genfilname
```

DESCRIPTION:

Writes the $\mathbf{q}$-points in lattice coordinates, weights and number of $\mathbf{G}+\mathbf{q}$-vectors to the file `QPOINTS.OUT`. Based on the routine `writekpts`.

REVISION HISTORY:

Created October 2006 (Sagmeister)

1.0.135 writepwmat (Source File: writepwmat.F90)

INTERFACE:

```
Subroutine writepwmat
```

USES:

```
Use modmain
Use modxs
Use m_genfilname
```

DESCRIPTION:

Calculates the matrix elements of the plane wave $e^{-i(\mathbf{G}+\mathbf{q})\mathbf{r}}$ using routine `genpwmat` and writes them to direct access file `PWMAT.OUT`.

REVISION HISTORY:

```
Created November 2007 (Sagmeister)
```

1.0.136 writesymi (Source File: writesymi.F90)

INTERFACE:

```
Subroutine writesymi
```

USES:

```
Use modinput
Use modmain
Use modxs
```

DESCRIPTION:

Outputs the crystal and symmetries including their inverse elements to file `SYMINV.OUT`

REVISION HISTORY:

```
Created December 2007 (Sagmeister)
```

1.0.137 scrcoulint (Source File: scrcoulint.F90)

INTERFACE:

```
Subroutine scrcoulint
```

USES:

```
Use modmain
Use modinput
Use modmpi
Use modxs
Use summations
Use m_xsgauntgen
Use m_findgntn0
Use m_writevars
Use m_genfilname
Use m_getunit
```

DESCRIPTION:

Calculates the direct term of the Bethe-Salpeter Hamiltonian.

REVISION HISTORY:

```
Created June 2008 (S. Sagmeister)
Addition of explicit energy ranges for states below and above the Fermi
   level for the treatment of core excitations (using local orbitals).
   October 2010 (Weine Olovsson)
```

1.0.138 dyson (Source File: dyson.F90)

INTERFACE:

```
Subroutine dyson (n, s0, k, s)
```

USES:

```
Use invert
```

INPUT/OUTPUT PARAMETERS:

```
n     : matrix size of local field effects (in,integer)
s0    : S0 matrix (in,complex(:,:))
k     : kernel matrix (in,complex(:,:))
s     : S (solution) matrix (in,complex(:,:))
```

DESCRIPTION:

Solve Dyson's equation

$$S = S_0 + S_0 K S$$

for S by inversion;

$$S = [1 + S_0 K]^{-1} S_0.$$

The inversion is carried out using the LAPACK routines `zgetrf` and `zgetri`.

REVISION HISTORY:

```
Created March 2005 (Sagmeister)
```

1.0.139 writegqpts (Source File: writegqpts.F90)

INTERFACE:

```
Subroutine writegqpts (iq, filex)
```

USES:

```
Use modmain
Use modxs
Use m_getunit
```

DESCRIPTION:

Writes the $\mathbf{G}+\mathbf{q}$-points in lattice coordinates, Cartesian coordinates, and lengths of $\mathbf{G}+\mathbf{q}$-vectors to the file `GQPOINTS.OUT`.

REVISION HISTORY:

```
Created October 2006 (Sagmeister)
```

1.0.140 xssave0 (Source File: xssave0.F90)

INTERFACE:

```
Subroutine xssave0
```

USES:

```
Use modmain
Use modxs
```

DESCRIPTION:

This routine should be called after init0, init1 and init2 in order to save variables related to the k-point set for $\mathbf{q}=0$.

REVISION HISTORY:

```
Created March 2005 (Sagmeister)
```

1.0.141 writesymt2 (Source File: writesymt2.F90)

INTERFACE:

```
Subroutine writesymt2
```

USES:

```
Use modmain
Use modxs
```

DESCRIPTION:

Outputs the symmetrization matrices for the tensor components of a rank-2 tensor. The tensor(-field) t_{ij} in reciprocal space must be invariant under coordinate transforms of the system wrt. the rotational part of the crystal symmetries, so we can average:

$$t_{ij}^{\mathrm{sym}} = \frac{1}{N_\alpha} \sum_\alpha \sum_{k,l} \alpha_{ik} \alpha_{jl} t_{kl}.$$

The symmetrized tensor t_{ij}^{sym} can then be written as

$$t_{ij}^{\mathrm{sym}} = \sum_{k,l} T_{ij,kl} t_{kl},$$

with the symmetrization tensor

$$T_{ij,kl} = \frac{1}{N_\alpha} \sum_\alpha \alpha_{ik} \alpha_{jl}$$

where N_α is the number of symmetry operations in the space group. For each component ij the symmetrization tensor $T_{ij,kl}$ is written as a matrix in the components kl to the file `SYMT2.OUT`.

REVISION HISTORY:

`Created October 2008 (Sagmeister)`

1.0.142 fxcifc (Source File: modfxcifc.F90)

INTERFACE:

```
 Subroutine fxcifc (fxctype, w, iw, iq, ng, oct, alrc, alrcd, &
& blrcd, fxcg)
    Use m_fxc_lrc, Only: fxc_lrc
    Use m_fxc_lrcd, Only: fxc_lrcd
    Use m_fxc_alda, Only: fxc_alda
    Use m_fxc_bse_ma03, Only: fxc_bse_ma03
```

INPUT/OUTPUT PARAMETERS:

`fxctype : type of exchange-correlation functional (in,integer)`

DESCRIPTION:

Interface to the exchange-correlation kernel routines. This makes it relatively simple to add new functionals which do not necessarily depend only on all input parameters. Based upon the routine `modxcifc`.

REVISION HISTORY:

`Created October 2007 (Sagmeister)`

1.0.143 getfxcdata (Source File: modfxcifc.F90)

INTERFACE:

```
Subroutine getfxcdata (fxctype, fxcdescr, fxcspin)
```

INPUT/OUTPUT PARAMETERS:

```
    Use modinput
fxctype  : type of exchange-correlation functional (in,integer)
fxcdescr : description of functional (out,character(256))
fxcspin  : spin treatment (out,integer)
fxcgrad  : gradient treatment (out,integer)
```

DESCRIPTION:

Returns data on the exchange-correlation functional labelled by `fxctype`. The character array `fxctype` contains a short description of the functional including journal references. The variable `fxcspin` is set to 1 or 0 for spin-polarised or -unpolarised functionals, respectively.

REVISION HISTORY:

```
Created October 2007 (Sagmeister)
```

1.0.144 zfinp2 (Source File: zfinp2.F90)

INTERFACE:

```
Complex (8) Function zfinp2 (ngp1, ngp2, igpig, zfmt1, zfmt2, zfir1, &
& zfir2)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
zfmt1 : first complex function in spherical harmonics for all muffin-tins
        (in,complex(lmmaxvr,nrcmtmax,natmtot))
zfmt2 : second complex function in spherical harmonics for all muffin-tins
        (in,complex(lmmaxvr,nrcmtmax,natmtot))
zfir1 : first complex interstitial function in real-space
        (in,complex(ngrtot))
zfir2 : second complex interstitial function in real-space
        (in,complex(ngrtot))
```

DESCRIPTION:

Calculates the inner product of two complex fuctions over the entire unit cell. The muffin-tin functions should be stored on the coarse radial grid and have angular momentum cut-off

`lmaxvr`. In the intersitial region, the integrand is multiplied with the smooth characteristic function, $\tilde{\Theta}(\mathbf{r})$, to remove the contribution from the muffin-tin. See routines `zfmtinp` and `gencfun`. Based upon the routine `zfinp`.

REVISION HISTORY:

```
Created January 2007 (Sagmeister)
```

1.0.145 kernxc (Source File: kernxc.F90)

INTERFACE:

```
Subroutine kernxc
```

DESCRIPTION:

Computes the ALDA exchange-correlation kernel. In the muffin-tin, the density is transformed from spherical harmonic coefficients ρ_{lm} to spherical coordinates (θ, ϕ) with a backward spherical harmonic transformation (SHT). Once calculated, the exchange-correlation potential and energy density are transformed with a forward SHT. This routine is based upon the routine `potxc`.

REVISION HISTORY:

```
Created March 2007 (Sagmeister)
```

1.0.146 dysonsym (Source File: dysonsym.F90)

INTERFACE:

```
Subroutine dysonsym (n, s0, k, s)
```

USES:

```
Use invert
Use modxs
```

INPUT/OUTPUT PARAMETERS:

```
n     : matrix size of local field effects (in,integer)
s0    : S0 matrix (in,complex(:,:))
k     : kernel matrix multiplied by S0 from both sides (in,complex(:,:))
s     : S (solution) matrix (in,complex(:,:))
```

DESCRIPTION:

Solve symmetric form of Dyson's equation

$$S = S_0 + S_0(1 + S_0^{-1} K S_0^{-1})S$$

for S by inversion;

$$S = S_0 \left[S_0(1 - S0) - T \right]^{-1} S_0.$$

The inversion is carried out using the LAPACK routines `zgetrf` and `zgetri`.
REVISION HISTORY:

```
Created October 2008 (Sagmeister)
```

1.0.147 copyfilesq0 (Source File: copyfilesq0.F90)

INTERFACE:

```
Subroutine copyfilesq0
      Use modinput
```

USES:

```
      Use modmain
      Use modxs
      Use m_getapwcmt
      Use m_getlocmt
```

DESCRIPTION:

If a finite momentum transfer Q-point is the Gamma-point, eigenvalues eigenvectors, occupancies and muffin-tin expansion coefficients are identical to those corresponding to the unshifted mesh. Files are copied using the associated generic routines for ISO compatibility, or links are generated if ISO compatibility is dropped.
REVISION HISTORY:

```
Created January 2008 (Sagmeister)
```

2 Introduction

Welcome to the XS/EXCITING code developers' manual. This manual is supposed to collect the routines and modules belonging exclusively to the excited states (TDDFT and BSE) part into one document. The content of this manual is partially taken from the author's PhD thesis.
& S. Sagmeister& Leoben, August 2008

2.0.148 fxc_lrcd (Source File: fxc_lrcd.F90)

INTERFACE:

```
Subroutine fxc_lrcd (msiz, sw, alpha, beta, w, fxc)
```

USES:

```
Use mod_constants, Only: fourpi
Use modxs, Only: unitout
```

INPUT/OUTPUT PARAMETERS:

```
msiz  : matrix size of local field effects (in,integer)
sw    : true for inclusion of local field effects (in,logical)
alpha : real constant (in,real)
w     : frequency grid (in,complex(:))
fxc   : xc-kernel Fourier coefficients (out,complex(:,:))
```

DESCRIPTION:

Dynamical long range xc-kernel; S. Botti, PRB 72, 125203 (2005). Calculates the symmetrized xc-kernel for the static long range model. According to the switch `sw` either

$$f_{\rm xc}(\mathbf{G},\mathbf{G}') = -\frac{1}{4\pi}(\alpha+\beta\omega^2)\delta(\mathbf{G},\mathbf{G}'),$$

if the switch is true, or

$$f_{\rm xc}(\mathbf{G},\mathbf{G}') = -\frac{1}{4\pi}(\alpha+\beta\omega^2)\delta(\mathbf{G},\mathbf{G}')\delta(\mathbf{G},\mathbf{0}),$$

otherwise.

REVISION HISTORY:

```
Created March 2006 (Sagmeister)
```

2.0.149 transik (Source File: transik.F90)

INTERFACE:

```
logical function transik(ik)
```

USES:

```
use modinput
```

DESCRIPTION:

This function returns "true" if the transition specified by the input k-point initial and final state matches with the ones specified in the `transitions` element. Wether a state is included or excluded depends on the sequence of the transitions specified (using the "include" and "exclude" attribute).

REVISION HISTORY:

```
Created July 2010 (Sagmeister)
```

2.0.150 xszoutpr2 (Source File: xszoutpr2.F90)

INTERFACE:

```
Subroutine xszoutpr2 (n1, n2, alpha, x, y, a)
```

INPUT/OUTPUT PARAMETERS:

```
n1,n2 : size of vectors and matrix, respectively (in,integer)
alpha : complex constant (in,complex)
x     : first input vector (in,complex(n1))
y     : second input vector (in,complex(n2))
a     : output matrix (out,complex(n1,n2))
```

DESCRIPTION:

Performs the rank-2 operation

$$A_{ij} \rightarrow \alpha \mathbf{x}_i \mathbf{y}_j^* + A_{ij}.$$

REVISION HISTORY:

```
Created March 2008 (Sagmeister)
```

2.0.151 genpmatxs (Source File: genpmatxs.F90)

INTERFACE:

```
Subroutine genpmatxs (ngp, igpig, vgpc, evecfv, evecsv, pmat)
```

USES:

```
Use modinput
Use modmain
Use modxs, Only: apwcmt, locmt, ripaa, ripalo, riploa, riplolo
```

INPUT/OUTPUT PARAMETERS:

```
ngp    : number of G+p-vectors (in,integer)
igpig  : index from G+p-vectors to G-vectors (in,integer(ngkmax))
vgpc   : G+p-vectors in Cartesian coordinates (in,real(3,ngkmax))
evecfv : first-variational eigenvector (in,complex(nmatmax,nstfv))
evecsv : second-variational eigenvectors (in,complex(nstsv,nstsv))
pmat   : momentum matrix elements (out,complex(3,nstsv,nstsv))
```

DESCRIPTION:

Calculates the momentum matrix elements

$$p_{ij} = \langle \Psi_{i,\mathbf{k}} | -i\nabla | \Psi_{j,\mathbf{k}} \rangle.$$

The gradient is applied explicitly only to the radial functions and corresponding spherical harmonics for the muffin-tin part. In the interstitial region the gradient is evaluated analytically. Parts taken from the routine `genpmat`.

REVISION HISTORY:

```
Created April 2008 (Sagmeister)
```

2.0.152 symtr2 (Source File: symtr2.F90)

INTERFACE:

```
Subroutine symtr2 (t2)
```

USES:

```
Use modmain
Use modxs
```

DESCRIPTION:

Symmetrizes a rank-2 tensor with respect to the rotational part of the crystal symmetries:

$$t_{ij}^{\mathrm{sym}} = \frac{1}{N_\alpha} \sum_{\alpha} \sum_{k,l=1}^{3} \alpha_{ik} \alpha_{jl} t_{kl}.$$

Here, t_{ij} are the components of the rank-2 tensor, α_{ij} denotes the rotational part of the crystal symmetries and N_α stands for the total number of crystal symmetries.

REVISION HISTORY:

```
Created October 2008 (Sagmeister)
```

2.0.153 genwiqggp (Source File: genwiqggp.F90)

INTERFACE:

```
Subroutine genwiqggp (flag, iq, igq1, igq2, clwt)
```

USES:

```
Use modmain
Use modxs
Use m_genfilname
Use m_getunit
```

DESCRIPTION:

Effective integrals of Coulomb interaction. See routine `genwiq2`.

REVISION HISTORY:

```
Created February 2008 (Sagmeister)
```

2.0.154 genfilname (Source File: genfilname.F90)

INTERFACE:

```
 Subroutine genfilname (nodotpar, basename, etype, asc, bzsampl, &
& acont, nar, tord, nlf, fxctype, fxctypestr, scrtype, bsetype, markfxcbse, &
& tq0, oc1, oc2, iq, iqmt, procs, rank, dotext, setfilext, &
& revertfilext, appfilext, filnam, fileext)
```

USES:

```
Use modmain, Only: filext
Use modxs, Only: filextrevert
```

DESCRIPTION:

Generates file name and extension according to optional input parameters (see routine). Interpret bzsampl variable as default (Lorentzian) for 0, as tetrahedron method for 1. Trilinear method to be followed.

REVISION HISTORY:

```
Created October 2007 (Sagmeister)
```

2.0.155 genpwmat (Source File: genpwmat.F90)

INTERFACE:

```
Subroutine genpwmat (vpl, ngpmax, ngp, vgpc, gpc, igpig, ylmgp, sfacgp, &
& vklk, ngkk, igkigk, apwalmk, evecfvk, evecsvk, vklkp, ngkkp, igkigkp, &
& apwalmkp, evecfvkp, evecsvkp, pwmat)
```

USES:

```
Use modmain
Use modxs
Use modinput
```

INPUT/OUTPUT PARAMETERS:

DESCRIPTION:

Calculates the matrix elements of the plane wave

$$M_{ij\mathbf{k}} = \langle \Psi_{i,\mathbf{k}} | e^{-i(\mathbf{G}+\mathbf{q})\mathbf{r}} | \Psi_{j,\mathbf{k}} \rangle.$$

Straightforward implementation for checking.
REVISION HISTORY:

```
Created November 2007 (Sagmeister)
```

2.0.156 getngqmax (Source File: getngqmax.F90)

INTERFACE:

```
Subroutine getngqmax
```

USES:

```
Use modinput
Use modmain
Use modxs
```

DESCRIPTION:

Determines the largest number of $\mathbf{G}+\mathbf{q}$-vectors with length less than `gqmax` over all the $\mathbf{q}$-points and stores it in the global variable `ngqmax`. This variable is used for allocating arrays. Based upon the routine `getngkmax`.
REVISION HISTORY:

```
Created October 2006 (Sagmeister)
```

2.0.157 kramkron (Source File: kramkron.F90)

INTERFACE:

```
Subroutine kramkron (i1, i2, eps, n, w, im, re)
```

INPUT/OUTPUT PARAMETERS:

```
i1,i2 : tensor components of dielectric function tensor (in,integer)
eps   : zero frequency tolerance
n     : number of frequency points (in,integer)
w     : frequency grid (in,real(n))
im    : imaginary part of dielectric function tensor component (in,real(n))
re    : real part of dielectric function tensor component (out,real(n))
```

DESCRIPTION:

Performs a Kramers-Kronig transformation from the imaginary part to the real part of the dielectric function. Algorithm taken from routine `linopt`.

REVISION HISTORY:

```
Created November 2007 (Sagmeister)
```

2.0.158 bandgap (Source File: bandgap.F90)

INTERFACE:

```
Subroutine bandgap (n, e, ef, egf, ego, ikgf, ikgo, istho)
```

USES:

```
Use modmain
```

DESCRIPTION:

Determines the fundamental and optical band gap if present.

REVISION HISTORY:

```
Created July 2007 (Sagmeister)
```

2.0.159 fxc_lrc (Source File: fxc_lrc.F90)

INTERFACE:

```
Subroutine fxc_lrc (msiz, sw, alpha, fxc)
```

USES:

```
Use mod_constants, Only: fourpi
Use modxs, Only: unitout
```

INPUT/OUTPUT PARAMETERS:

```
msiz  : matrix size of local field effects (in,integer)
sw    : true for inclusion of local field effects (in,logical)
alpha : real constant (in,real)
fxc   : xc-kernel Fourier coefficients (out,complex(:,:))
```

DESCRIPTION:

Static long range xc-kernel; S. Botti, PRB 70, 045301 (2004). Calculates the symmetrized xc-kernel for the static long range model. According to the switch `sw` either

$$f_{\rm xc}(\mathbf{G},\mathbf{G}') = -\frac{\alpha}{4\pi}\delta(\mathbf{G},\mathbf{G}'),$$

if the switch is true, or

$$f_{\rm xc}(\mathbf{G},\mathbf{G}') = -\frac{\alpha}{4\pi}\delta(\mathbf{G},\mathbf{G}')\delta(\mathbf{G},\mathbf{0}),$$

otherwise.

REVISION HISTORY:

```
Created March 2006 (Sagmeister)
```

2.0.160 gensumrls (Source File: gensumrls.F90)

INTERFACE:

```
Subroutine gensumrls (w, eps, sumrls)
```

USES:

```
Use modxs
```

INPUT/OUTPUT PARAMETERS:

```
w      : frequency grid (in,real(:))
eps    : dielectric function tensor component (in,complex(:))
sumrls : values of three different sumrules (out,real(3))
```

DESCRIPTION:

Expressions for the sumrules taken from C. Ambrosch-Draxl, CPC 175 (2006) 1-14, p5, Eq. 26.

REVISION HISTORY:

```
Created March 2006 (Sagmeister)
```

2.0.161 exccoulint (Source File: exccoulint.F90)

INTERFACE:

```
Subroutine exccoulint
```

USES:

```
Use modmain
Use modinput
Use modmpi
Use modxs
Use ioarray
Use m_xsgauntgen
Use m_findgntn0
Use m_writegqpts
Use m_genfilname
Use m_getunit
```

DESCRIPTION:

Calculates the exchange term of the Bethe-Salpeter Hamiltonian.
REVISION HISTORY:

```
Created June 2008 (S. Sagmeister)
Addition of explicit energy ranges for states below and above the Fermi
   level for the treatment of core excitations (using local orbitals).
   October 2010 (Weine Olovsson)
```

2.0.162 fermisurf_dx (Source File: fermisurf_dx.f90)

INTERFACE:

```
Subroutine fermisurf_dx
```

USES:

```
Use modmain
```

DESCRIPTION:

Add-on to the `fermisurf` routine. Generates input files for the visualization of the Fermi surface in the first Brillouin zone with OpenDX. Prerequisite is a previous run of `fermisurf` for a $2 \times 2 \times 2$ supercell of reciprocal asymmetric units. It generates a `FERMI.dx` and a `FERMI_BZ.dx` file where the data for the visualization is stored together with the clipping planes for the construction of the first Brillouin zone. The OpenDX input is complete together with the template `FERMI.net` and the macro `clipvmacro.net`.
REVISION HISTORY:

```
Created Jan 2005 (Sagmeister)
Revised, May 2008 (Sagmeister)
```

2.0.163 seceqn (Source File: residualvector.F90)

Subroutine residualvectors (n, iunconverged, h, s, evalfv, r, rnorms) Use modmain, Only: nmatmax, zone, zzero
INPUT/OUTPUT PARAMETERS:

DESCRIPTION:

the residual is

$$|\mathbf{R}\left(|\mathbf{A}^{ap}\rangle, E^{ap}\right)\rangle = (\mathbf{H} - E^{ap}\mathbf{S})|\mathbf{A}^{ap}\rangle$$

REVISION HISTORY:

```
Created March 2004 (JKD)
```

2.0.164 mixpulay (Source File: mixpulay.f90)

INTERFACE:

```
Subroutine mixpulay (iscl, n, maxsd, nu, mu, f, d)
```

INPUT/OUTPUT PARAMETERS:

```
iscl  : self-consistent loop number (in,integer)
n     : vector length (in,integer)
maxsd : maximum subspace dimension (in,integer)
nu    : current output vector as well as the next input vector of the
        self-consistent loop (inout,real(n))
mu    : used internally (inout,real(n,maxsd))
f     : used internally (inout,real(n,maxsd))
d     : RMS difference between old and new output vectors (out,real)
```

DESCRIPTION:

Pulay's mixing scheme which uses direct inversion in the iterative subspace (DIIS). See *Chem. Phys. Lett.* **73**, 393 (1980).

REVISION HISTORY:

```
Created October 2008 (S. Suehara, NIMS)
Modified, October 2008 (JKD)
```

2.0.165 mixmsec (Source File: mixmsec.f90)

INTERFACE:

```
Subroutine mixmsec (iscl, potential, residualnorm, n)
```

INPUT/OUTPUT PARAMETERS:

```
      Use modinput
  iscl     : self-consistent loop number (in,integer)
  potential : potential coefitients packed in one array
  residualnorm: measure for convergence
  n : length of mixing vector
 config params:
      Use mod_potential_and_density, Only:
      Use mod_charge_and_moment, Only: chgir, chgmttot, chgtot
  persistent arrays and create/desdruct functions
      Use modmixermsec, Only: residual, last_outputp, work2, work3, &
     & initmixermsec, freearraysmixermsec, noldstepsmax, &
     & noldstepsin_file, noldsteps, qmx, dmix, dmixout, TCharge, &
     & SCharge, splane, tplane, qmx_input, qtot
```

2.0.166 mixadapt (Source File: mixadapt.f90)

INTERFACE:

```
Subroutine mixadapt (iscl, beta0, betainc, betadec, n, nu, mu, beta, f, &
& d)
```

INPUT/OUTPUT PARAMETERS:

```
iscl    : self-consistent loop number (in,integer)
beta0   : initial value for mixing parameter (in,real)
betainc : mixing parameter increase (in,real)
betadec : mixing parameter decrease (in,real)
n       : vector length (in,integer)
nu      : current output vector as well as the next input vector of the
          self-consistent loop (inout,real(n))
mu      : used internally (inout,real(n))
beta    : used internally (inout,real(n))
f       : used internally (inout,real(n))
d       : RMS difference between old and new output vectors (out,real)
```

DESCRIPTION:

Given the input vector μ^i and output vector ν^i of the ith self-consistent loop, this routine generates the next input vector to the loop using an adaptive mixing scheme. The jth component of the output vector is mixed with a fraction of the same component of the input vector:

$$\mu_j^{i+1} = \beta_j^i \nu_j^i + (1 - \beta_j^i)\mu_j^i,$$

where β_j^i is set to β_0 at initialisation and increased by scaling with $\beta_{\rm inc}$ (> 1) if $f_j^i \equiv \nu_j^i - \mu_j^i$ does not change sign between loops. If f_j^i does change sign, then β_j^i is scaled by $\beta_{\rm dec}$ (< 1). Note that the array `nu` serves for both input and output, and the arrays `mu`, `beta` and `f` are used internally and should not be changed between calls. The routine is initialised at the first iteration and is thread-safe so long as each thread has its own independent work array. Complex arrays may be passed as real arrays with n doubled.

REVISION HISTORY:

```
Created March 2003 (JKD)
Modified, September 2008 (JKD)
```

2.0.167 sumrule (Source File: sumrule.f90)

INTERFACE:

```
Subroutine sumrule (dynq)
```

INPUT/OUTPUT PARAMETERS:

```
  Use modinput
dynq : dynamical matrices on q-point set (in,real(3*natmtot,3*natmtot,nqpt))
```

DESCRIPTION:

Applies the same correction to all the dynamical matrices such that the matrix for $\mathbf{q} = 0$ satisfies the acoustic sum rule. In other words, the matrices are updated with

$$D^{\mathbf{q}}_{ij} \rightarrow D^{\mathbf{q}}_{ij} - \sum_{k=1}^{3} \omega^0_k v^0_{k;i} v^0_{k;j}$$

for all $\mathbf{q}$, where ω^0_k is the kth eigenvalue of the $\mathbf{q} = 0$ dynamical matrix and $v^0_{k;i}$ the ith component of its eigenvector. The eigenvalues are assumed to be arranged in ascending order. This ensures that the $\mathbf{q} = 0$ dynamical matrix has 3 zero eigenvalues, which the uncorrected matrix may not have due to the finite exchange-correlation grid.

REVISION HISTORY:

```
Created May 2005 (JKD)
```

2.0.168 reformatdynamicalmatrices (Source File: reformatdynamicalmatrices.F90)

INTERFACE:

```
subroutine reformatdynamicalmatrices
```

USES:

```
use modinput
Use mod_atoms
use mod_qpoint
use mod_constants
```

DESCRIPTION:

Collecting pieces of the dynamical matrices and assembling them in a nicer way, such that 3×3 matrices are displayed for each combination of atoms and each $\mathbf{q}$ point.

REVISION HISTORY:

```
Created February 2010 (Sagmeister)
```

2.0.169 writedynamicalmatrices (Source File: writedynamicalmatrices.F90)

INTERFACE:

```
subroutine writedynamicalmatrices(fname,dynmq)
```

USES:

```
use modinput
Use mod_atoms
use mod_qpoint
use mod_constants
```

DESCRIPTION:

Write out the dynamical matrices to file `DYNMAT.OUT`.

REVISION HISTORY:

```
Created February 2010 (Sagmeister)
```

2.0.170 genpmat (Source File: genpmat.f90)

INTERFACE:

```
Subroutine genpmat (ngp, igpig, vgpc, apwalm, evecfv, evecsv, pmat)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
ngp    : number of G+p-vectors (in,integer)
igpig  : index from G+p-vectors to G-vectors (in,integer(ngkmax))
vgpc   : G+p-vectors in Cartesian coordinates (in,real(3,ngkmax))
apwalm : APW matching coefficients
         (in,complex(ngkmax,apwordmax,lmmaxapw,natmtot))
evecfv : first-variational eigenvector (in,complex(nmatmax,nstfv))
evecsv : second-variational eigenvectors (in,complex(nstsv,nstsv))
pmat   : momentum matrix elements (out,complex(3,nstsv,nstsv))
```

DESCRIPTION:

Calculates the momentum matrix elements

$$p_{ij} = \langle \Psi_{i,\mathbf{k}} | -i\nabla | \Psi_{j,\mathbf{k}} \rangle .$$

REVISION HISTORY:

```
Created November 2003 (Sharma)
Fixed bug found by Juergen Spitaler, September 2006 (JKD)
```

2.0.171 stheta_sq (Source File: stheta_sq.f90)

INTERFACE:

```
Real (8) Function stheta_sq (x)
```

INPUT/OUTPUT PARAMETERS:

```
x : real argument (in,real)
```

DESCRIPTION:

Returns the Heaviside step function corresponding to the square-wave pulse approximation to the Dirac delta function

$$\tilde{\Theta}(x) = \begin{cases} 0 & x \leq -1/2 \\ x + 1/2 & -1/2 < x < 1/2 \\ 1 & x \geq 1 \end{cases}$$

REVISION HISTORY:

```
Created July 2008 (JKD)
```

2.0.172 hermite (Source File: hermite.f90)

INTERFACE:

```
Real (8) Function hermite (n, x)
```

INPUT/OUTPUT PARAMETERS:

```
n : order of Hermite polynomial (in,integer)
x : real argument (in,real)
```

DESCRIPTION:

Returns the nth Hermite polynomial. The recurrence relation

$$H_i(x) = 2xH_{i-1}(x) - 2nH_{i-2}(x),$$

with $H_0 = 1$ and $H_1 = 2x$, is used. This procedure is numerically stable and accurate to near machine precision for $n \leq 20$.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.173 spline (Source File: spline.f90)

INTERFACE:

```
Subroutine spline (n, x, ld, f, cf)
```

INPUT/OUTPUT PARAMETERS:

```
n  : number of points (in,integer)
x  : abscissa array (in,real(n))
ld : leading dimension (in,integer)
f  : input data array (in,real(ld,n))
cf : cubic spline coefficients (out,real(3,n))
```

DESCRIPTION:

Calculates the coefficients of a cubic spline fitted to input data. In other words, given a set of data points f_i defined at x_i, where $i = 1 \ldots n$, the coefficients c_j^i are determined such that

$$y_i(x) = f_i + c_1^i(x - x_i) + c_2^i(x - x_i)^2 + c_3^i(x - x_i)^3,$$

is the interpolating function for $x \in [x_i, x_{i+1})$. This is done by determining the end-point coefficients c_2^1 and c_2^n from the first and last three points, and then solving the tridiagonal system

$$d_{i-1}c_2^{i-1} + 2(d_{i-1} + d_i)c_2^i + d_i c_2^{i+1} = 3\left(\frac{f_{i+1} - f_i}{d_i} - \frac{f_i - f_{i-1}}{d_{i-1}}\right),$$

where $d_i = x_{i+1} - x_i$, for the intermediate coefficients.

REVISION HISTORY:

```
Created October 2004 (JKD)
Improved speed and accuracy, April 2006 (JKD)
Optimisations and improved end-point coefficients, February 2008 (JKD)
```

2.0.174 stheta_mp (Source File: stheta_mp.f90)

INTERFACE:

```
Real (8) Function stheta_mp (n, x)
```

INPUT/OUTPUT PARAMETERS:

```
n : order (in,integer)
x : real argument (in,real)
```

DESCRIPTION:

Returns the smooth approximation to the Heaviside step function of order N given by Methfessel and Paxton, *Phys. Rev. B* **40**, 3616 (1989),

$$\tilde{\Theta}(x) = 1 - S_N(x)$$

where

$$S_N(x) = S_0(x) + \sum_{i=1}^{N} \frac{(-1)^i}{i!4^n\sqrt{\pi}} H_{2i-1}(x)e^{-x^2},$$
$$S_0(x) = \frac{1}{2}(1 - \mathrm{erf}(x))$$

and H_j is the jth-order Hermite polynomial. This procedure is numerically stable and accurate to near machine precision for $N \leq 10$.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.175 stheta_fd (Source File: stheta_fd.f90)

INTERFACE:

```
Real (8) Function stheta_fd (x)
```

INPUT/OUTPUT PARAMETERS:

```
x : real argument (in,real)
```

DESCRIPTION:

Returns the Fermi-Dirac approximation to the Heaviside step function

$$\tilde{\Theta}(x) = \frac{1}{1 + e^{-x}}.$$

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.176 rdirac (Source File: rdirac.f90)

INTERFACE:

```
Subroutine rdirac (n, l, k, np, nr, r, vr, eval, g0, f0)
```

INPUT/OUTPUT PARAMETERS:

```
n    : principal quantum number (in,integer)
l    : quantum number l (in,integer)
k    : quantum number k (l or l+1) (in,integer)
np   : order of predictor-corrector polynomial (in,integer)
nr   : number of radial mesh points (in,integer)
r    : radial mesh (in,real(nr))
vr   : potential on radial mesh (in,real(nr))
eval : eigenvalue without rest-mass energy (inout,real)
g0   : major component of the radial wavefunction (out,real(nr))
f0   : minor component of the radial wavefunction (out,real(nr))
```

DESCRIPTION:

Finds the solution to the radial Dirac equation for a given potential $v(r)$ and quantum numbers n, k and l. The method involves integrating the equation using the predictor-corrector method and adjusting E until the number of nodes in the wavefunction equals $n-l-1$. The calling routine must provide an initial estimate for the eigenvalue. Note that the arrays `g0` and `f0` represent the radial functions multiplied by r.

REVISION HISTORY:

```
Created September 2002 (JKD)
```

2.0.177 r3frac (Source File: r3frac.f90)

INTERFACE:

```
Subroutine r3frac (eps, v, iv)
```

INPUT/OUTPUT PARAMETERS:

```
eps : zero component tolerance (in,real)
v   : input vector (inout,real(3))
iv  : integer parts of v (out,integer(3))
```

DESCRIPTION:

Finds the fractional part of each component of a real 3-vector using the function $\mathrm{frac}\,(x) = x - \lfloor x \rfloor$. A component is taken to be zero if it lies within the intervals $[0, \epsilon)$ or $(1-\epsilon, 1]$. The integer components of `v` are returned in the variable `iv`.

REVISION HISTORY:

```
Created January 2003 (JKD)
```

2.0.178 zmatinp (Source File: zmatinp.f90)

INTERFACE:

```
Subroutine zmatinp (tapp, n, alpha, x, y, v, a)
```

INPUT/OUTPUT PARAMETERS:

```
tapp  : .true. if the matrix is to be applied to the input vector v,
        .false. if the full matrix is to be calculated (in,logical)
n     : length of vectors (in,integer)
alpha : complex constant (in,complex)
x     : first input vector (in,complex(n))
y     : second input vector (in,complex(n))
v     : input vector to which matrix is applied if tapp is .true., otherwise
```

```
          not referenced (in,complex(n))
     a    : matrix applied to v if tapp is .true., otherwise the full matrix in
          packed form (inout,complex(n+(n-1)*n/2))
```

DESCRIPTION:

Performs the rank-2 operation

$$A_{ij} \rightarrow \alpha \mathbf{x}_i^* \mathbf{y}_j + \alpha^* \mathbf{y}_i^* \mathbf{x}_j + A_{ij},$$

where A is stored in packed form. This is similar to the `BLAS` routine `zhpr2`, except that here a matrix of inner products is formed instead of an outer product of vectors. If `tapp` is `.true.` then the matrix is applied to an input vector, rather than calculated explicitly.

REVISION HISTORY:

```
Created June 2003 (JKD)
```

2.0.179 r3mtm (Source File: r3mtm.f90)

INTERFACE:

```
Subroutine r3mtm (a, b, c)
```

INPUT/OUTPUT PARAMETERS:

```
a : input matrix 1 (in,real(3,3))
b : input matrix 2 (in,real(3,3))
c : output matrix (out,real(3,3))
```

DESCRIPTION:

Multiplies the transpose of one real 3×3 matrix with another.

REVISION HISTORY:

```
Created January 2003 (JKD)
```

2.0.180 i3minv (Source File: i3minv.f90)

INTERFACE:

```
Subroutine i3minv (a, b)
```

INPUT/OUTPUT PARAMETERS:

```
a : input matrix (in,integer(3,3))
b : output matrix (in,integer(3,3))
```

DESCRIPTION:

Computes the inverse of a integer 3×3 matrix: $B = A^{-1}$.

REVISION HISTORY:

```
Created November 2003 (JKD)
```

2.0.181 sdelta_fd (Source File: sdelta_fd.f90)

INTERFACE:

```
Real (8) Function sdelta_fd (x)
```

INPUT/OUTPUT PARAMETERS:

```
x : real argument (in,real)
```

DESCRIPTION:

Returns the Fermi-Dirac approximation to the Dirac delta function

$$\tilde{\delta}(x) = \frac{e^{-x}}{(1+e^{-x})^2}.$$

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.182 flushifc (Source File: flushifc.f90)

INTERFACE:

```
Subroutine flushifc (fnum)
```

INPUT/OUTPUT PARAMETERS:

```
  Use modinput
fnum : unit specifier for file (in,integer)
```

DESCRIPTION:

Interface to the Fortran `flush` statement. Some compilers do not support the `flush` command, which is very useful for keeping small formatted files up-to-date on the disk. The routine implimented below is a machine-independent emulation of `flush`, but may be replaced with the intrinsic command if preferred.

REVISION HISTORY:

```
Created September 2002 (JKD)
```

2.0.183 clebgor (Source File: clebgor.f90)

INTERFACE:

```
Real (8) Function clebgor (j1, j2, j3, m1, m2, m3)
```

INPUT/OUTPUT PARAMETERS:

```
j1, j2, j3 : angular momentum quantum numbers (in,integer)
m1, m2, m3 : magnetic quantum numbers (in,integer)
```

DESCRIPTION:

Returns the Clebsch-Gordan coefficients using the Wigner $3j$-symbols

$$C(J_1J_2J_3|m_1m_2m_3) = (-1)^{J_1-J_2+m_3}\sqrt{2J_3+1}\begin{pmatrix} J_1 & J_2 & J_3 \\ m_1 & m_2 & -m_3 \end{pmatrix}.$$

Suitable for $J_i \leq 50$.

REVISION HISTORY:

```
Created September 2003 (JKD)
```

2.0.184 gauntyry (Source File: gauntyry.f90)

INTERFACE:

```
Complex (8) Function gauntyry (l1, l2, l3, m1, m2, m3)
```

INPUT/OUTPUT PARAMETERS:

```
l1, l2, l3 : angular momentum quantum numbers (in,integer)
m1, m2, m3 : magnetic quantum numbers (in,integer)
```

DESCRIPTION:

Returns the complex Gaunt-like coefficient given by $\langle Y^{l_1}_{m_1}|R^{l_2}_{m_2}|Y^{l_3}_{m_3}\rangle$, where Y_{lm} and R_{lm} are the complex and real spherical harmonics, respectively. Suitable for l_i less than 50. See routine `genrlm`.

REVISION HISTORY:

```
Created November 2002 (JKD)
```

2.0.185 r3mtv (Source File: r3mtv.f90)

INTERFACE:

```
Subroutine r3mtv (a, x, y)
```

INPUT/OUTPUT PARAMETERS:

```
a : input matrix (in,real(3,3))
x : input vector (in,real(3))
y : output vector (out,real(3))
```

DESCRIPTION:

Multiplies the transpose of a real 3×3 matrix with a vector.
REVISION HISTORY:

```
Created January 2003 (JKD)
```

2.0.186 rtozflm (Source File: rtozflm.f90)

INTERFACE:

```
Subroutine rtozflm (lmax, rflm, zflm)
```

INPUT/OUTPUT PARAMETERS:

```
lmax : maximum angular momentum (in,integer)
rflm : coefficients of real spherical harmonic expansion
       (in,real((lmax+1)**2)))
zflm : coefficients of complex spherical harmonic expansion
       (out,complex((lmax+1)**2)))
```

DESCRIPTION:

Converts a real function, r_{lm}, expanded in terms of real spherical harmonics into a complex spherical harmonic expansion, z_{lm}:

$$z_{lm} = \begin{cases} \frac{1}{\sqrt{2}}(r_{lm} + i(-1)^m r_{l-m}) & m > 0 \\ \frac{1}{\sqrt{2}}((-1)^m r_{l-m} - i r_{lm}) & m < 0 \\ r_{lm} & m = 0 \end{cases} .$$

See routine `genrlm`.
REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.187 gradzfmt (Source File: gradzfmt.f90)

INTERFACE:

```
Subroutine gradzfmt (lmax, nr, r, ld1, ld2, zfmt, gzfmt)
```

INPUT/OUTPUT PARAMETERS:

```
lmax  : maximum angular momentum (in,integer)
nr    : number of radial mesh points (in,integer)
r     : radial mesh (in,real(nr))
ld1   : leading dimension 1 (in,integer)
ld2   : leading dimension 2 (in,integer)
zfmt  : complex muffin-tin function (in,complex(ld1,nr))
gzfmt : gradient of zfmt (out,complex(ld1,ld2,3))
```

DESCRIPTION:

Calculates the gradient of a complex muffin-tin function. In other words, given the spherical harmonic expansion coefficients, $f_{lm}(r)$, of a function $f(\mathbf{r})$, the routine returns $\mathbf{F}_{lm}$ where

$$\sum_{lm} \mathbf{F}_{lm}(r) Y_{lm}(\hat{\mathbf{r}}) = \nabla f(\mathbf{r}).$$

This is done using the identity

$$\begin{aligned}\nabla \left[f_{lm}(r) Y_{lm}(\hat{\mathbf{r}}) \right] = &- \left[\frac{l+1}{2l+1} \right]^{1/2} \left[\frac{d}{dr} - \frac{l}{r} \right] f_{lm}(r) \mathbf{Y}_{ll+1m}(\hat{\mathbf{r}}) \\ &+ \left[\frac{l}{2l+1} \right]^{1/2} \left[\frac{d}{dr} + \frac{l+1}{r} \right] f_{lm}(r) \mathbf{Y}_{ll-1m}(\hat{\mathbf{r}}),\end{aligned}$$

where the vector spherical harmonics are given by

$$\mathbf{Y}_{ll'm}(\hat{\mathbf{r}}) = \sum_{m'=-l'}^{l'} \sum_{m''=-1}^{1} C(l'1l|m'm''m)\, Y_{lm}(\hat{\mathbf{r}})\, \hat{\mathbf{e}}_{m''},$$

C is a Clebsch-Gordan coefficient and

$$\hat{\mathbf{e}}_{+1} = -\frac{\hat{\mathbf{x}} + i\hat{\mathbf{y}}}{\sqrt{2}}, \qquad \hat{\mathbf{e}}_0 = \hat{\mathbf{z}}, \qquad \hat{\mathbf{e}}_{-1} = \frac{\hat{\mathbf{x}} - i\hat{\mathbf{y}}}{\sqrt{2}}$$

are unit vectors. Note that the gradient returned is in terms of the global $(\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}})$ coordinate system.

REVISION HISTORY:

```
Created August 2003 (JKD)
```

2.0.188 rdiracdme (Source File: rdiracdme.f90)

INTERFACE:

```
Subroutine rdiracdme (m, kpa, e, np, nr, r, vr, nn, g0, g1, f0, f1)
```

INPUT/OUTPUT PARAMETERS:

```
m   : order of energy derivative (in,integer)
kpa : quantum number kappa (in,integer)
e   : energy (in,real)
np  : order of predictor-corrector polynomial (in,integer)
nr  : number of radial mesh points (in,integer)
r   : radial mesh (in,real(nr))
vr  : potential on radial mesh (in,real(nr))
nn  : number of nodes (out,integer)
g0  : m th energy derivative of the major component multiplied by r
      (out,real(nr))
g1  : radial derivative of g0 (out,real(nr))
f0  : m th energy derivative of the minor component multiplied by r
      (out,real(nr))
f1  : radial derivative of f0 (out,real(nr))
```

DESCRIPTION:

Finds the solution to the mth energy derivative of the radial Dirac equation using the routine `rdiracint`.

REVISION HISTORY:

```
Created March 2003 (JKD)
```

2.0.189 factnm (Source File: factnm.f90)

INTERFACE:

```
Real (8) Function factnm (n, m)
```

INPUT/OUTPUT PARAMETERS:

```
n : input (in,integer)
m : order of multifactorial (in,integer)
```

DESCRIPTION:

Returns the multifactorial

$$n \underbrace{!!\ldots!}_{m\,\text{times}} = \prod_{i\geq 0,\, n-im>0} n-im$$

for n, $m \geq 0$. n should be less than 150.

REVISION HISTORY:

```
Created January 2003 (JKD)
```

2.0.190 euler (Source File: euler.f90)

INTERFACE:

```
Subroutine euler (rot, ang)
```

INPUT/OUTPUT PARAMETERS:

```
rot : rotation matrix (in,real(3,3))
ang : euler angles (alpha, beta, gamma) (out,real(3))
```

DESCRIPTION:

Given a rotation matrix

$$R(\alpha,\beta,\gamma) = \begin{pmatrix} \cos\gamma\cos\beta\cos\alpha - \sin\gamma\sin\alpha & \cos\gamma\cos\beta\sin\alpha + \sin\gamma\cos\alpha & -\cos\gamma\sin\beta \\ -\sin\gamma\cos\beta\cos\alpha - \cos\gamma\sin\alpha & -\sin\gamma\cos\beta\sin\alpha + \cos\gamma\cos\alpha & \sin\gamma\sin\beta \\ \sin\beta\cos\alpha & \sin\beta\sin\alpha & \cos\beta \end{pmatrix},$$

this routine determines the Euler angles, (α,β,γ). This corresponds to the so-called "y-convention", which involves the following successive rotations of the coordinate system:

1. The x_1'-, x_2'-, x_3'-axes are rotated anticlockwise through an angle α about the x_3 axis
2. The x_1''-, x_2''-, x_3''-axes are rotated anticlockwise through an angle β about the x_2' axis
3. The x_1'''-, x_2'''-, x_3'''-axes are rotated anticlockwise through an angle γ about the x_3'' axis

Note that the Euler angles are not necessarily unique for a given rotation matrix.

REVISION HISTORY:

```
Created May 2003 (JKD)
```

2.0.191 factr (Source File: factr.f90)

INTERFACE:

```
Real (8) Function factr (n, d)
```

INPUT/OUTPUT PARAMETERS:

```
n : numerator (in,integer)
d : denominator (in,integer)
```

DESCRIPTION:

Returns the ratio $n!/d!$ for $n,d\geq 0$. Performs no under- or overflow checking.

REVISION HISTORY:

```
Created October 2002 (JKD)
```

2.0.192 fderiv (Source File: fderiv.f90)

INTERFACE:

```
Subroutine fderiv (m, n, x, f, g, cf)
```

INPUT/OUTPUT PARAMETERS:

```
m  : order of derivative (in,integer)
n  : number of points (in,integer)
x  : abscissa array (in,real(n))
f  : function array (in,real(n))
g  : (anti-)derivative of f (out,real(n))
cf : spline coefficients (out,real(3,n))
```

DESCRIPTION:

Given function f defined on a set of points x_i then if $m \geq 0$ this routine computes the mth derivative of f at each point. If $m < 0$ the anti-derivative of f given by

$$g(x_i) = \int_{x_1}^{x_i} f(x)\,dx$$

is calculated by fitting the function to a clamped cubic spline. See routine `spline`.

REVISION HISTORY:

```
Created May 2002 (JKD)
```

2.0.193 brzint (Source File: brzint.f90)

INTERFACE:

```
Subroutine brzint (nsm, ngridk, nsk, ikmap, nw, wint, n, ld, e, f, g)
```

INPUT/OUTPUT PARAMETERS:

```
nsm    : level of smoothing for output function (in,integer)
ngridk : k-point grid size (in,integer(3))
nsk    : k-point subdivision grid size (in,integer(3))
ikmap  : map from grid to k-point set
         (in,integer(0:ngridk(1)-1,0:ngridk(2)-1,0:ngridk(3)-1))
nw     : number of energy divisions (in,integer)
wint   : energy interval (in,real(2))
n      : number of functions to integrate (in,integer)
ld     : leading dimension (in,integer)
e      : array of energies as a function of k-points (in,real(ld,*))
f      : array of weights as a function of k-points (in,real(ld,*))
g      : output function (out,real(nw))
```

DESCRIPTION:

Given energy and weight functions, e and f, on the Brillouin zone and a set of equidistant energies ω_i, this routine computes the integrals

$$g(\omega_i) = \frac{\Omega}{(2\pi)^3} \int_{\mathrm{BZ}} f(\mathbf{k})\delta(\omega_i - e(\mathbf{k}))d\mathbf{k},$$

where Ω is the unit cell volume. This is done by first interpolating e and f on a finer k-point grid using the trilinear method. Then for each $e(\mathbf{k})$ on the finer grid the nearest ω_i is found and $f(\mathbf{k})$ is accumulated in $g(\omega_i)$. If the output function is noisy then either `nsk` should be increased or `nw` decreased. Alternatively, the output function can be artificially smoothed up to a level given by `nsm`. See routine `fsmooth`.

REVISION HISTORY:

```
Created October 2003 (JKD)
Improved efficiency May 2007 (Sebastian Lebegue)
```

2.0.194 zflmconj (Source File: zflmconj.f90)

INTERFACE:

```
Subroutine zflmconj (lmax, zflm1, zflm2)
```

INPUT/OUTPUT PARAMETERS:

```
lmax  : maximum angular momentum (in,integer)
zflm1 : coefficients of input complex spherical harmonic expansion
        (in,complex((lmax+1)**2)))
zflm2 : coefficients of output complex spherical harmonic expansion
        (out,complex((lmax+1)**2)))
```

DESCRIPTION:

Returns the complex conjugate of a function expanded in spherical harmonics. In other words, given the input function coefficients z_{lm}, the routine returns $z'_{lm} = (-1)^m z^*_{l-m}$ so that

$$\sum_{lm} z'_{lm} Y_{lm}(\theta,\phi) = \left(\sum_{lm} z_{lm} Y_{lm}(\theta,\phi) \right)^*$$

for all (θ,ϕ). Note that `zflm1` and `zflm2` can refer to the same array.

REVISION HISTORY:

```
Created April 2004 (JKD)
```

2.0.195 rschrodint (Source File: rschrodint.f90)

INTERFACE:

```
Subroutine rschrodint (m, l, e, np, nr, r, vr, nn, p0p, p0, p1, q0, q1)
```

INPUT/OUTPUT PARAMETERS:

```
m   : order of energy derivative (in,integer)
l   : angular momentum quantum number (in,integer)
e   : energy (in,real)
np  : order of predictor-corrector polynomial (in,integer)
nr  : number of radial mesh points (in,integer)
r   : radial mesh (in,real(nr))
vr  : potential on radial mesh (in,real(nr))
nn  : number of nodes (out,integer)
p0p : m-1 th energy derivative of P (in,real(nr))
p0  : m th energy derivative of P (out,real(nr))
p1  : radial derivative of p0 (out,real(nr))
q0  : m th energy derivative of Q (out,real(nr))
q1  : radial derivative of q0 (out,real(nr))
```

DESCRIPTION:

Integrates the mth energy derivative of the scalar relativistic radial Schrödinger equation from $r = 0$ outwards. This involves using the predictor-corrector method to solve the coupled first-order equations (in atomic units)

$$\frac{d}{dr}P_l^{(m)} = 2MQ_l^{(m)} + \frac{1}{r}P_l^{(m)}$$
$$\frac{d}{dr}Q_l^{(m)} = -\frac{1}{r}Q_l^{(m)} + \left[\frac{l(l+1)}{2Mr^2} + (V-E)\right]P_l^{(m)} - mP_l^{(m-1)},$$

where V is the external potential, E is the eigenenergy and $M = 1 - V/2c^2$. Following the convention of Koelling and Harmon, *J. Phys. C: Solid State Phys.* **10**, 3107 (1977), the functions P_l and Q_l are defined by

$$P_l = rg_l$$
$$Q_l = \frac{r}{2M}\frac{dg_l}{dr},$$

where g_l is the major component of the Dirac equation (see the routine `rdiracint`). Note that in order to make the equations linear in energy, the full definition $M = 1+(E-V)/2c^2$ is not used. If $m = 0$ then the array `p0p` is not referenced.

REVISION HISTORY:

```
Created October 2003 (JKD)
```

2.0.196 sphcrd (Source File: sphcrd.f90)

INTERFACE:

```
Subroutine sphcrd (v, r, tp)
```

INPUT/OUTPUT PARAMETERS:

```
v  : input vector (in,real(3))
r  : length of v (out,real)
tp : (theta, phi) coordinates (out,real(2))
```

DESCRIPTION:

Returns the spherical coordinates (r, θ, ϕ) of a vector

$$\mathbf{v} = (r\sin(\theta)\cos(\phi), r\sin(\theta)\sin(\phi), r\cos(\theta)).$$

REVISION HISTORY:

```
Created October 2002 (JKD)
```

2.0.197 i3mdet (Source File: i3mdet.f90)

INTERFACE:

```
Integer Function i3mdet (a)
```

INPUT/OUTPUT PARAMETERS:

```
a : input matrix (in,integer(3,3))
```

DESCRIPTION:

Returns the determinant of an integer 3×3 matrix A.

REVISION HISTORY:

```
Created October 2004 (JKD)
```

2.0.198 i3mtv (Source File: i3mtv.f90)

INTERFACE:

```
Subroutine i3mtv (a, x, y)
```

INPUT/OUTPUT PARAMETERS:

```
a : input matrix (in,integer(3,3))
x : input vector (in,integer(3))
y : output vector (out,integer(3))
```

DESCRIPTION:

Multiplies the transpose of an integer 3×3 matrix with a vector.
REVISION HISTORY:

```
Created April 2007 (JKD)
```

2.0.199 genrlm (Source File: genrlm.f90)

INTERFACE:

```
Subroutine genrlm (lmax, tp, rlm)
```

INPUT/OUTPUT PARAMETERS:

```
lmax : maximum angular momentum (in,integer)
tp   : (theta, phi) coordinates (in,real(2))
rlm  : array of real spherical harmonics (out,real((lmax+1)**2))
```

DESCRIPTION:

Generates a sequence of real spherical harmonics evaluated at angles (θ, ϕ) for $0 < l < l_{\max}$. The values are returned in a packed array `rlm` indexed with $j = l(l+1) + m + 1$. Real spherical harmonics are defined by

$$R_{lm}(\theta,\phi) = \begin{cases} \sqrt{2}\,\Re\{Y_{lm}(\theta,\phi)\} & m > 0 \\ \sqrt{2}\,\Im\{Y_{lm}(\theta,\phi)\} & m < 0\,, \\ \Re\{Y_{lm}(\theta,\phi)\} & m = 0 \end{cases}$$

where Y_{lm} are the complex spherical harmonics. These functions are orthonormal and complete and may be used for expanding real-valued functions on the sphere. This routine is numerically stable and accurate to near machine precision for $l \leq 50$. See routine `genylm`.
REVISION HISTORY:

```
Created March 2004 (JKD)
```

2.0.200 findband (Source File: findband.f90)

INTERFACE:

```
Subroutine findband (findlinentype, l, k, np, nr, r, vr, de0, etol, e, tfnd)
```

INPUT/OUTPUT PARAMETERS:

```
l   : angular momentum quantum number (in,integer)
k   : quantum number k, zero if Dirac eqn. is not to be used (in,integer)
np  : order of predictor-corrector polynomial (in,integer)
nr  : number of radial mesh points (in,integer)
```

```
r   : radial mesh (in,real(nr))
vr  : potential on radial mesh (in,real(nr))
de0 : default energy step size (in,real)
e   : input energy and returned band energy (inout,real)
```

DESCRIPTION:

Finds the band energies for a given radial potential and angular momentum. This is done by first searching upwards in energy until the radial wavefunction at the muffin-tin radius is zero. This is the energy at the top of the band, denoted $E_{\rm t}$. A downward search is now performed from $E_{\rm t}$ until the slope of the radial wavefunction at the muffin-tin radius is zero. This energy, $E_{\rm b}$, is at the bottom of the band. The band energy is taken as $(E_{\rm t}+E_{\rm b})/2$. If either $E_{\rm t}$ or $E_{\rm b}$ cannot be found then the band energy is set to the input value.

REVISION HISTORY:

```
Created September 2004 (JKD)
```

2.0.201 rdiracint (Source File: rdiracint.f90)

INTERFACE:

```
Subroutine rdiracint (m, kpa, e, np, nr, r, vr, nn, g0p, f0p, g0, g1, &
& f0, f1)
```

INPUT/OUTPUT PARAMETERS:

```
m   : order of energy derivative (in,integer)
kpa : quantum number kappa (in,integer)
e   : energy (in,real)
np  : order of predictor-corrector polynomial (in,integer)
nr  : number of radial mesh points (in,integer)
r   : radial mesh (in,real(nr))
vr  : potential on radial mesh (in,real(nr))
nn  : number of nodes (out,integer)
g0p : m-1 th energy derivative of the major component multiplied by r
      (in,real(nr))
f0p : m-1 th energy derivative of the minor component multiplied by r
      (in,real(nr))
g0  : m th energy derivative of the major component multiplied by r
      (out,real(nr))
g1  : radial derivative of g0 (out,real(nr))
f0  : m th energy derivative of the minor component multiplied by r
      (out,real(nr))
f1  : radial derivative of f0 (out,real(nr))
```

DESCRIPTION:

Integrates the mth energy derivative of the radial Dirac equation from $r = 0$ outwards. This involves using the predictor-corrector method to solve the coupled first-order equations (in atomic units)

$$\left(\frac{d}{dr}+\frac{\kappa}{r}\right)G_\kappa^{(m)} = \frac{1}{c}\{2E_0+E-V\}F_\kappa^{(m)}+\frac{m}{c}F_\kappa^{(m-1)}$$
$$\left(\frac{d}{dr}-\frac{\kappa}{r}\right)F_\kappa^{(m)} = -\frac{1}{c}\{E-V\}G_\kappa^{(m)}-\frac{m}{c}G_\kappa^{(m-1)},$$

where $G_\kappa^{(m)} = rg_\kappa^{(m)}$ and $F_\kappa^{(m)} = rf_\kappa^{(m)}$ are the mth energy derivatives of the major and minor components multiplied by r, respectively; V is the external potential; E_0 is the electron rest energy; E is the eigen energy (excluding E_0); and $\kappa = l$ for $j = l - \frac{1}{2}$ or $\kappa = -(l+1)$ for $j = l + \frac{1}{2}$. If $m = 0$ then the arrays `g0p` and `f0p` are not referenced.

REVISION HISTORY:

```
Created September 2002 (JKD)
```

2.0.202 fsmooth (Source File: fsmooth.f90)

Subroutine fsmooth (m, n, ld, f) **INPUT/OUTPUT PARAMETERS:**

```
m  : number of 3-point running averages to perform (in,integer)
n  : number of point (in,integer)
ld : leading dimension (in,integer)
f  : function array (inout,real(ld,n))
```

DESCRIPTION:

Removes numerical noise from a function by performing m successive 3-point running averages on the data. The endpoints are kept fixed.

REVISION HISTORY:

```
Created December 2005 (JKD)
```

2.0.203 sphcover (Source File: sphcover.f90)

INTERFACE:

```
Subroutine sphcover (n, tp)
```

INPUT/OUTPUT PARAMETERS:

```
n  : number of required points (in,integer)
tp : (theta, phi) coordinates (out,real(2,n))
```

DESCRIPTION:

Produces a set of N points which cover the unit sphere nearly optimally. The points in (θ, ϕ) coordinates are generated using the explicit formula

$$\theta_k = \arccos(h_k), \qquad h_k = \frac{2(k-1)}{N-1} - 1, \qquad 1 \le k \le N$$
$$\phi_k = \left(\phi_{k-1} + C/\sqrt{N(1-h_k^2)}\right) \pmod{2\pi}, \qquad 2 \le k \le N-1, \qquad \phi_1 = \phi_N = 0,$$

where $C = (8\pi/\sqrt{3})^{1/2}$. See E. B. Saff and A. B. J. Kuijlaars, *Math. Intell.* **19**, 5 (1997).
REVISION HISTORY:

```
Created April 2008 (JKD)
```

2.0.204 sbesseldm (Source File: sbesseldm.f90)

INTERFACE:

```
Subroutine sbesseldm (m, lmax, x, djl)
```

INPUT/OUTPUT PARAMETERS:

```
m    : order of derivatve (in,integer)
lmax : maximum order of Bessel function (in,integer)
x    : real argument (in,real)
djl  : array of returned values (out,real(0:lmax))
```

DESCRIPTION:

Computes the mth derivative of the spherical Bessel function of the first kind, $j_l(x)$, for argument x and $l = 0, 1, \ldots, l_{\max}$. For $x \ge 1$ this is done by repeatedly using the relations

$$\frac{d}{dx} j_l(x) = \frac{l}{x} j_l(x) - j_{l+1}(x)$$
$$j_{l+1}(x) = \frac{2l+1}{x} j_l(x) - j_{l-1}(x).$$

While for $x < 1$ the series expansion of the Bessel function is used

$$\frac{d^m}{dx^m} j_l(x) = \sum_{i=0}^{\infty} \frac{(2i+l)!}{(-2)^i i!(2i+l-m)!(2i+2l+1)!!} x^{2i+l-m}.$$

This procedure is numerically stable and accurate to near machine precision for $l \le 30$ and $m \le 6$.
REVISION HISTORY:

```
Created March 2003 (JKD)
Modified to return an array of values, October 2004 (JKD)
```

2.0.205 vecfbz (Source File: vecfbz.f90)

INTERFACE:

```
Subroutine vecfbz (eps, bvec, vpl, iv)
```

INPUT/OUTPUT PARAMETERS:

```
eps  : zero component tolerance (in,real)
bvec : reciprocal lattice vectors (in,real(3,3))
vpl  : input vector in lattice coordinates (inout,real(3))
iv   : integer parts of vpl (out,integer(3))
```

DESCRIPTION:

Maps a vector in lattice coordinates to the first Brillouin zone. This is done by first removing its integer components and then adding primitive reciprocal lattice vectors until the shortest vector is found.

REVISION HISTORY:

```
Created September 2008 (JKD)
```

2.0.206 genylm (Source File: genylm.f90)

INTERFACE:

```
Subroutine genylm (lmax, tp, ylm)
```

INPUT/OUTPUT PARAMETERS:

```
lmax : maximum angular momentum (in,integer)
tp   : (theta, phi) coordinates (in,real(2))
ylm  : array of spherical harmonics (out,complex((lmax+1)**2))
```

DESCRIPTION:

Generates a sequence of spherical harmonics, including the Condon-Shortley phase, evaluated at angles (θ, ϕ) for $0 < l < l_{\max}$. The values are returned in a packed array `ylm` indexed with $j = l(l+1) + m + 1$. The algorithm of Masters and Richards-Dinger is used, *Geophys. J. Int.* **135**, 307 (1998). This routine is numerically stable and accurate to near machine precision for $l \leq 50$.

REVISION HISTORY:

```
Created March 2004 (JKD)
Improved stability, December 2005 (JKD)
```

2.0.207 rfinterp (Source File: rfinterp.f90)

INTERFACE:

```
Subroutine rfinterp (ni, xi, ldi, fi, no, xo, ldo, fo)
```

INPUT/OUTPUT PARAMETERS:

```
ni  : number of input points (in,integer)
xi  : input abscissa array (in,real(ni))
ldi : leading dimension (in,integer)
fi  : input data array (in,real(ldi,ni)
no  : number of output points (in,integer)
xo  : output abscissa array (in,real(ni))
ldo : leading dimension (in,integer)
fo  : output interpolated function (out,real(ldo,no))
```

DESCRIPTION:

Given a function defined on a set of input points, this routine uses a clamped cubic spline to interpolate the function on a different set of points. See routine `spline`.

REVISION HISTORY:

```
Created January 2005 (JKD)
```

2.0.208 gcd (Source File: gcd.f90)

INTERFACE:

```
Integer Function gcd (x, y)
```

INPUT/OUTPUT PARAMETERS:

```
x : first integer (in,integer)
y : second integer (in,integer)
```

DESCRIPTION:

Computes the greatest common divisor (GCD) of two integers using Euclid's algorithm.

REVISION HISTORY:

```
Created September 2004 (JKD)
```

2.0.209 sbessel (Source File: sbessel.f90)

INTERFACE:

```
Subroutine sbessel (lmax, x, jl)
```

INPUT/OUTPUT PARAMETERS:

```
lmax : maximum order of Bessel function (in,integer)
x    : real argument (in,real)
jl   : array of returned values (out,real(0:lmax))
```

DESCRIPTION:

Computes the spherical Bessel functions of the first kind, $j_l(x)$, for argument x and $l = 0, 1, \ldots, l_{\max}$. The recursion relation

$$j_{l+1}(x) = \frac{2l+1}{x} j_l(x) - j_{l-1}(x)$$

is used either downwards for $x < l$ or upwards for $x \geq l$. For $x \ll 1$ the asymtotic form is used

$$j_l(x) \approx \frac{x^l}{(2l+1)!!}.$$

This procedure is numerically stable and accurate to near machine precision for $l \leq 50$.

REVISION HISTORY:

```
Created January 2003 (JKD)
Modified to return an array of values, October 2004 (JKD)
Improved stability, August 2006 (JKD)
```

2.0.210 sdelta_mp (Source File: sdelta_mp.f90)

INTERFACE:

```
Real (8) Function sdelta_mp (n, x)
```

INPUT/OUTPUT PARAMETERS:

```
n : order (in,integer)
x : real argument (in,real)
```

DESCRIPTION:

Returns the smooth approximation to the Dirac delta function of order N given by Methfessel and Paxton, *Phys. Rev. B* **40**, 3616 (1989),

$$\tilde{\delta}(x) = \sum_{i=0}^{N} \frac{(-1)^i}{i! 4^n \sqrt{\pi}} H_{2i}(x) e^{-x^2},$$

where H_j is the jth-order Hermite polynomial. This function has the property

$$\int_{-\infty}^{\infty} \tilde{\delta}(x) P(x) = P(0),$$

where $P(x)$ is any polynomial of degree $2N+1$ or less. The case $N = 0$ corresponds to Gaussian smearing. This procedure is numerically stable and accurate to near machine precision for $N \leq 10$.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.211 r3dot (Source File: r3dot.f90)

INTERFACE:

```
Real (8) Function r3dot (x, y)
```

INPUT/OUTPUT PARAMETERS:

```
x : input vector 1 (in,real(3))
y : input vector 2 (in,real(3))
```

DESCRIPTION:

Returns the dot-product of two real 3-vectors.

REVISION HISTORY:

```
Created January 2003 (JKD)
```

2.0.212 gradrfmt (Source File: gradrfmt.f90)

INTERFACE:

```
Subroutine gradrfmt (lmax, nr, r, ld1, ld2, rfmt, grfmt)
```

INPUT/OUTPUT PARAMETERS:

```
lmax  : maximum angular momentum (in,integer)
nr    : number of radial mesh points (in,integer)
r     : radial mesh (in,real(nr))
ld1   : leading dimension 1 (in,integer)
ld2   : leading dimension 2 (in,integer)
rfmt  : real muffin-tin function (in,real(ld1,nr))
grfmt : gradient of rfmt (out,real(ld1,ld2,3))
```

DESCRIPTION:

Calculates the gradient of a real muffin-tin function. In other words, given the real spherical harmonic expansion coefficients, $f_{lm}(r)$, of a function $f(\mathbf{r})$, the routine returns $\mathbf{F}_{lm}$ where

$$\sum_{lm} \mathbf{F}_{lm}(r) R_{lm}(\hat{\mathbf{r}}) = \nabla f(\mathbf{r}),$$

and R_{lm} is a real spherical harmonic function. This is done by first converting the function to a complex spherical harmonic expansion and then using the routine `gradzfmt`. See routine `genrlm`.

REVISION HISTORY:

```
Created August 2003 (JKD)
```

2.0.213 sortidx (Source File: sortidx.f90)

INTERFACE:

```
Subroutine sortidx (n, a, idx)
```

INPUT/OUTPUT PARAMETERS:

```
n   : number of elements in array (in,integer)
idx : permutation index (out,integer(n))
a   : real array (in,real(n))
```

DESCRIPTION:

Finds the permutation index `idx` which sorts the real array `a` into ascending order. No sorting of the array `a` itself is performed. Uses the heapsort algorthim.

REVISION HISTORY:

```
Created October 2002 (JKD)
Included tolerance eps, April 2006 (JKD)
```

2.0.214 z2mm (Source File: z2mm.f90)

INTERFACE:

```
Subroutine z2mm (a, b, c)
```

INPUT/OUTPUT PARAMETERS:

```
a   : input matrix 1 (in,complex(2,2))
b   : input matrix 2 (in,complex(2,2))
c   : output matrix (out,complex(2,2))
```

DESCRIPTION:

Multiplies two complex 2×2 matrices. Note that the output matrix cannot be one of the input matrices.

REVISION HISTORY:

```
Created October 2007 (JKD)
```

2.0.215 gaunt (Source File: gaunt.f90)

INTERFACE:

```
Real (8) Function gaunt (l1, l2, l3, m1, m2, m3)
```

INPUT/OUTPUT PARAMETERS:

```
l1, l2, l3 : angular momentum quantum numbers (in,integer)
m1, m2, m3 : magnetic quantum numbers (in,integer)
```

DESCRIPTION:

Returns the Gaunt coefficient given by

$$\langle Y^{l_1}_{m_1}|Y^{l_2}_{m_2}|Y^{l_3}_{m_3}\rangle = (-1)^{m_1}\left[\frac{(2l_1+1)(2l_2+1)(2l_3+1)}{4\pi}\right]^{\frac{1}{2}}\begin{pmatrix} l_1 & l_2 & l_3 \\ 0 & 0 & 0\end{pmatrix}\begin{pmatrix} l_1 & l_2 & l_3 \\ -m_1 & m_2 & m_3\end{pmatrix}.$$

Suitable for l_i less than 50.

REVISION HISTORY:

```
Created November 2002 (JKD)
```

2.0.216 r3cross (Source File: r3cross.f90)

INTERFACE:

```
Subroutine r3cross (x, y, z)
```

INPUT/OUTPUT PARAMETERS:

```
x : input vector 1 (in,real(3))
y : input vector 2 (in,real(3))
z : output cross-product (out,real(3))
```

DESCRIPTION:

Returns the cross product of two real 3-vectors.

REVISION HISTORY:

```
Created September 2002 (JKD)
```

2.0.217 r3mmt (Source File: r3mmt.f90)

INTERFACE:

```
Subroutine r3mmt (a, b, c)
```

INPUT/OUTPUT PARAMETERS:

```
a : input matrix 1 (in,real(3,3))
b : input matrix 2 (in,real(3,3))
c : output matrix (out,real(3,3))
```

DESCRIPTION:

Multiplies a real matrix with the transpose of another.

REVISION HISTORY:

```
Created January 2003 (JKD)
```

2.0.218 r3dist (Source File: r3dist.f90)

INTERFACE:

```
Real (8) Function r3dist (x, y)
```

INPUT/OUTPUT PARAMETERS:

```
x : input vector 1 (in,real(3))
y : input vector 2 (in,real(3))
```

DESCRIPTION:

Returns the distance between two real 3-vectors: $d = |\mathbf{x} - \mathbf{y}|$.

REVISION HISTORY:

```
Created January 2003 (JKD)
```

2.0.219 rschroddme (Source File: rschroddme.f90)

INTERFACE:

```
Subroutine rschroddme (m, l, k, e, np, nr, r, vr, nn, p0, p1, q0, q1)
```

INPUT/OUTPUT PARAMETERS:

```
m   : order of energy derivative (in,integer)
l   : angular momentum quantum number (in,integer)
k   : quantum number k, zero if Dirac eqn. is not to be used (in,integer)
e   : energy (in,real)
np  : order of predictor-corrector polynomial (in,integer)
nr  : number of radial mesh points (in,integer)
r   : radial mesh (in,real(nr))
vr  : potential on radial mesh (in,real(nr))
nn  : number of nodes (out,integer)
p0  : m th energy derivative of P (out,real(nr))
p1  : radial derivative of p0 (out,real(nr))
q0  : m th energy derivative of Q (out,real(nr))
q1  : radial derivative of q0 (out,real(nr))
```

DESCRIPTION:

Finds the solution to the mth energy derivative of the scalar relativistic radial Schrödinger equation using the routine `rschrodint`.

REVISION HISTORY:

```
Created June 2003 (JKD)
```

2.0.220 rotaxang (Source File: rotaxang.f90)

INTERFACE:

```
Subroutine rotaxang (eps, rot, det, v, th)
```

INPUT/OUTPUT PARAMETERS:

```
eps : zero vector tolerance (in,real)
rot : rotation matrix (in,real(3,3))
det : matrix determinant (out,real)
v   : normalised axis vector (out,real(3))
th  : rotation angle (out,real)
```

DESCRIPTION:

Given a rotation matrix

$$R(\hat{\mathbf{v}},\theta)=\begin{pmatrix}\cos\theta+x^2(1-\cos\theta) & xy(1-\cos\theta)+z\sin\theta & xz(1-\cos\theta)-y\sin\theta\\ xy(1-\cos\theta)-z\sin\theta & \cos\theta+y^2(1-\cos\theta) & yz(1-\cos\theta)+x\sin\theta\\ xz(1-\cos\theta)+y\sin\theta & yz(1-\cos\theta)-x\sin\theta & \cos\theta+z^2(1-\cos\theta)\end{pmatrix},$$

this routine determines the axis of rotation $\hat{\mathbf{v}}$ and the angle of rotation θ. If R corresponds to an improper rotation then only the proper part is used and `det` is set to -1.

REVISION HISTORY:

```
Created Decmeber 2006 (JKD)
Changed "intent(inout)" to "intent(in)" for argument "rot", 2009 (Sagmeister)
```

2.0.221 axangsu2 (Source File: axangsu2.f90)

Subroutine axangsu2 (v, th, su2) **INPUT/OUTPUT PARAMETERS:**

```
v   : rotation axis vector (in,real(3))
th  : rotation angle (in,real)
su2 : SU(2) representation of rotation (out,complex(2,2))
```

DESCRIPTION:

Finds the complex SU(2) representation of a rotation defined by an axis vector $\hat{\mathbf{v}}$ and angle θ. The spinor rotation matrix is given explicitly by

$$R^{1/2}(\hat{\mathbf{v}},\theta)=I\cos\frac{\theta}{2}+i(\hat{\mathbf{v}}\cdot\vec{\sigma})\sin\frac{\theta}{2}.$$

REVISION HISTORY:

```
Created August 2007 (JKD)
Reversed rotation direction, February 2008 (L. Nordstrom)
```

2.0.222 stheta (Source File: stheta.f90)

INTERFACE:

```
Real (8) Function stheta (stype, x)
```

INPUT/OUTPUT PARAMETERS:

```
stype : smearing type (in,integer)
x     : real argument (in,real)
```

DESCRIPTION:

Returns the Heaviside step function corresponding to the smooth approximation to the Dirac delta function:

$$\tilde{\Theta}(x) = \int_{-\infty}^{x} dt\, \tilde{\delta}(t).$$

See function `sdelta` for details.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.223 z2mctm (Source File: z2mctm.f90)

INTERFACE:

```
Subroutine z2mctm (a, b, c)
```

INPUT/OUTPUT PARAMETERS:

```
a   : input matrix 1 (in,complex(2,2))
b   : input matrix 2 (in,complex(2,2))
c   : output matrix (out,complex(2,2))
```

DESCRIPTION:

Multiplies the conjugate transpose of one complex 2×2 matrix with another. Note that the output matrix cannot be one of the input matrices.

REVISION HISTORY:

```
Created October 2007 (JKD)
```

2.0.224 erf (Source File: erf.f90)

INTERFACE:

```
Real (8) Function erf (x)
```

INPUT/OUTPUT PARAMETERS:

```
x : real argument (in,real)
```

DESCRIPTION:

Returns the error function $\mathrm{erf}(x)$ using a rational function approximation. This procedure is numerically stable and accurate to near machine precision.

REVISION HISTORY:

```
Modified version of a NSWC routine, April 2003 (JKD)
```

2.0.225 z2mmct (Source File: z2mmct.f90)

INTERFACE:

```
Subroutine z2mmct (a, b, c)
```

INPUT/OUTPUT PARAMETERS:

```
a   : input matrix 1 (in,complex(2,2))
b   : input matrix 2 (in,complex(2,2))
c   : output matrix (out,complex(2,2))
```

DESCRIPTION:

Multiplies a 2×2 matrix with the conjugate transpose of another. Note that the output matrix cannot be one of the input matrices.

REVISION HISTORY:

```
Created October 2007 (JKD)
```

2.0.226 r3mdet (Source File: r3mdet.f90)

INTERFACE:

```
Real (8) Function r3mdet (a)
```

INPUT/OUTPUT PARAMETERS:

```
a : input matrix (in,real(3,3))
```

DESCRIPTION:

Returns the determinant of a real 3×3 matrix A.

REVISION HISTORY:

```
Created May 2003 (JKD)
```

2.0.227 r3minv (Source File: r3minv.f90)

INTERFACE:

```
Subroutine r3minv (a, b)
```

INPUT/OUTPUT PARAMETERS:

```
a : input matrix (in,real(3,3))
b : output matrix (in,real(3,3))
```

DESCRIPTION:

Computes the inverse of a real 3×3 matrix.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.228 sdelta_sq (Source File: sdelta_sq.f90)

INTERFACE:

```
Real (8) Function sdelta_sq (x)
```

INPUT/OUTPUT PARAMETERS:

```
x : real argument (in,real)
```

DESCRIPTION:

Returns the square-wave pulse approximation to the Dirac delta function

$$\tilde{\delta}(x) = \begin{cases} 1 & |x| \leq 1/2 \\ 0 & |x| > 1/2 \end{cases}$$

REVISION HISTORY:

```
Created July 2008 (JKD)
```

2.0.229 polynom (Source File: polynom.f90)

INTERFACE:

```
Real (8) Function polynom (m, np, xa, ya, c, x)
```

INPUT/OUTPUT PARAMETERS:

```
m  : order of derivative (in,integer)
np : number of points to fit (in,integer)
xa : abscissa array (in,real(np))
ya : ordinate array (in,real(np))
c  : work array (out,real(np))
x  : evaluation abscissa (in,real)
```

DESCRIPTION:

Fits a polynomial of order $n_p - 1$ to a set of n_p points. If $m \geq 0$ the function returns the mth derviative of the polynomial at x, while for $m < 0$ the integral of the polynomial from the first point in the array to x is returned.

REVISION HISTORY:

```
Created October 2002 (JKD)
```

2.0.230 rschrodapp (Source File: rschrodapp.f90)

INTERFACE:

```
Subroutine rschrodapp (l, nr, r, vr, p0, q0, q1, hp0)
```

INPUT/OUTPUT PARAMETERS:

```
l   : angular momentum quantum number (in,integer)
nr  : number of radial mesh points (in,integer)
r   : radial mesh (in,real(nr))
vr  : potential on radial mesh (in,real(nr))
p0  : m th energy derivative of P (in,real(nr))
q0  : m th energy derivative of Q (in,real(nr))
q1  : radial derivative of q0 (in,real(nr))
hp0 : H applied to P (out,real(nr))
```

DESCRIPTION:

Applies the scalar relativistic radial Hamiltonian, H, to a radial wavefunction, P_l. This is an approximation since we assume P_l is a scalar wavefunction, normalisable to unity. A Hamiltonian which satisfies $HP_l = EP_l$ is given implicitly by

$$HP_l = \left[\frac{l(l+1)}{2Mr^2} + V\right] P_l - \frac{1}{r} Q_l - \frac{d}{dr} Q_l,$$

where V is the external potential, $M = 1 - V/2c^2$ and Q_l is obtained from integrating the coupled scalar relativistic equations. See the routine `rschrodint` for further details.

REVISION HISTORY:

```
Created October 2003 (JKD)
```

2.0.231 wigner3j (Source File: wigner3j.f90)

INTERFACE:

```
Real (8) Function wigner3j (j1, j2, j3, m1, m2, m3)
```

INPUT/OUTPUT PARAMETERS:

```
j1, j2, j3 : angular momentum quantum numbers (in,integer)
m1, m2, m3 : magnetic quantum numbers (in,integer)
```

DESCRIPTION:

Returns the Wigner $3j$-symbol. There are many equivalent definitions for the $3j$-symbols, the following provides high accuracy for $j \leq 50$

$$\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} = (-1)^{j1+j2+m3}$$
$$\times \sqrt{\frac{(j_1+m_1)!(j_2+m_2)!(j_3+m_3)!(j_3-m_3)!(j_1-m_1)!(j_2-m_2)!}{(j_2-j_1+j_3)!(j_1-j_2+j_3)!(j_1+j_2-j_3)!(1+j_1+j_2+j_3)!}} \times \sum_{\max(0,j_2-j_3-m_1,j_1-j_3+m_2)}^{\min(j_1+j_2-j_3,j_1-m_1,j_2+m_2)}$$
$$(-1)^k \frac{(j_2-j_1+j_3)!(j_1-j_2+j_3)!(j_1+j_2-j_3)!}{(j_3-j_1-m_2+k)!(j_3-j_2+m_1+k)!(j_1+j_2-j_3-k)!k!(j_1-m_1-k)!(j_2+m_2-k)}.$$

REVISION HISTORY:

```
Created November 2002 (JKD)
```

2.0.232 r3mv (Source File: r3mv.f90)

INTERFACE:

```
Subroutine r3mv (a, x, y)
```

INPUT/OUTPUT PARAMETERS:

```
a : input matrix (in,real(3,3))
x : input vector (in,real(3))
y : output vector (out,real(3))
```

DESCRIPTION:

Multiplies a real 3×3 matrix with a vector.

REVISION HISTORY:

```
Created January 2003 (JKD)
```

2.0.233 r3mm (Source File: r3mm.f90)

INTERFACE:

```
Subroutine r3mm (a, b, c)
```

INPUT/OUTPUT PARAMETERS:

```
a : input matrix 1 (in,real(3,3))
b : input matrix 2 (in,real(3,3))
c : output matrix (out,real(3,3))
```

DESCRIPTION:

Multiplies two real 3×3 matrices.
REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.234 sdelta (Source File: sdelta.f90)

INTERFACE:

```
Function sdelta (stype, x)
```

INPUT/OUTPUT PARAMETERS:

```
stype : smearing type (in,integer)
x     : real argument (in,real)
```

DESCRIPTION:

Returns a normalised smooth approximation to the Dirac delta function. These functions are defined such that

$$\int \tilde{\delta}(x)dx = 1.$$

The effective width, w, of the delta function may be varied by using the normalising transformation

$$\tilde{\delta}_w(x) \equiv \frac{\tilde{\delta}(x/w)}{w}.$$

Currently implimented are:

0. Gaussian
1. Methfessel-Paxton order 1
2. Methfessel-Paxton order 2
3. Fermi-Dirac
4. Square-wave impulse

See routines `stheta`, `sdelta_mp`, `sdelta_fd` and `sdelta_sq`.
REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.235 getsdata (Source File: sdelta.f90)

INTERFACE:

```
Subroutine getsdata (stype, sdescr)
```

INPUT/OUTPUT PARAMETERS:

```
stype  : smearing type (in,integer)
sdescr : smearing scheme description (out,character(256))
```

DESCRIPTION:

Returns a description of the smearing scheme as string `sdescr` up to 256 characters long.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.236 r3taxi (Source File: r3taxi.f90)

INTERFACE:

```
Real (8) Function r3taxi (x, y)
```

INPUT/OUTPUT PARAMETERS:

```
x : input vector 1 (in,real(3))
y : input vector 2 (in,real(3))
```

DESCRIPTION:

Returns the taxi-cab distance between two real 3-vectors: $d = |x_1 - y_1| + |x_2 - y_2| + |x_3 - y_3|$.

REVISION HISTORY:

```
Created March 2006 (JKD)
```

2.0.237 connect (Source File: connect.f90)

INTERFACE:

```
Subroutine connect (cvec, plotdef, nv, np, vpl, dv, dp)
```

INPUT/OUTPUT PARAMETERS:

```
cvec : matrix of (reciprocal) lattice vectors stored column-wise
       (in,real(3,3))
nv   : number of vertices (in,integer)
np   : number of connecting points (in,integer)
vvl  : vertex vectors in lattice coordinates (in,real(3,nv))
vpl  : connecting point vectors in lattice coordinates (out,real(3,np))
dv   : cummulative distance to each vertex (out,real(nv))
dp   : cummulative distance to each connecting point (out,real(np))
```

DESCRIPTION:

Generates a set of points which interpolate between a given set of vertices. Vertex points are supplied in lattice coordinates in the array `vvl` and converted to Cartesian coordinates with the matrix `cvec`. Interpolating points are stored in the array `vpl`. The cummulative distances to the vertices and points along the path are stored in arrays `dv` and `dp`, respectively.

REVISION HISTORY:

```
Created June 2003 (JKD)
Improved September 2007 (JKD)
```

2.0.238 lopzflm (Source File: lopzflm.f90)

INTERFACE:

```
Subroutine lopzflm (lmax, zflm, ld, zlflm)
```

INPUT/OUTPUT PARAMETERS:

```
lmax  : maximum angular momentum (in,integer)
zflm  : coefficients of input spherical harmonic expansion
        (in,complex((lmax+1)**2))
ld    : leading dimension (in,integer)
zlflm : coefficients of output spherical harmonic expansion
        (out,complex(ld,3))
```

DESCRIPTION:

Applies the angular momentum operator $\mathbf{L}$ to a function expanded in terms of complex spherical harmonics. This makes use of the identities

$$(L_x + iL_y)Y_{lm}(\theta,\phi) = \sqrt{(l-m)(l+m+1)}Y_{lm+1}(\theta,\phi)$$
$$(L_x - iL_y)Y_{lm}(\theta,\phi) = \sqrt{(l+m)(l-m+1)}Y_{lm-1}(\theta,\phi)$$
$$L_zY_{lm}(\theta,\phi) = mY_{lm}(\theta,\phi).$$

REVISION HISTORY:

```
Created March 2004 (JKD)
```

2.0.239 ztorflm (Source File: ztorflm.f90)

INTERFACE:

```
Subroutine ztorflm (lmax, zflm, rflm)
```

INPUT/OUTPUT PARAMETERS:

```
lmax : maximum angular momentum (in,integer)
zflm : coefficients of complex spherical harmonic expansion
       (in,complex((lmax+1)**2)))
rflm : coefficients of real spherical harmonic expansion
       (out,real((lmax+1)**2)))
```

DESCRIPTION:

Converts a real function, z_{lm}, expanded in terms of complex spherical harmonics into a real spherical harmonic expansion, r_{lm}:

$$r_{lm}=\begin{cases}\frac{1}{\sqrt{2}}\Re(z_{lm}+(-1)^m z_{l-m}) & m>0\\ \frac{1}{\sqrt{2}}\Im(-z_{lm}+(-1)^m z_{l-m}) & m<0\\ \Re(z_{lm}) & m=0\end{cases}.$$

See routine `genrlm`.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.240 genwiq2 (Source File: genwiq2.f90)

INTERFACE:

```
Subroutine genwiq2
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

The Fock matrix elements

$$V^{\rm NL}_{ij\mathbf{k}}\equiv\sum_{l\mathbf{k}'}\int\frac{\Psi^{\dagger}_{i\mathbf{k}}(\mathbf{r})\cdot\Psi_{l\mathbf{k}'}(\mathbf{r})\Psi^{\dagger}_{l\mathbf{k}'}(\mathbf{r}')\cdot\Psi_{j\mathbf{k}}(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\,d\mathbf{r}\,d\mathbf{r}'$$

contain a divergent term in the sum over $\mathbf{k}'$ which behaves as $1/q^2$, where $\mathbf{q}\equiv\mathbf{k}-\mathbf{k}'$ is in the first Brillouin zone. The resulting convergence with respect to the number of discrete q-points is very slow. This routine computes the weights

$$w_{\mathbf{q}_i}\equiv\int_{V_i}\frac{1}{q^2}\,d\mathbf{q}\,, \tag{1}$$

where the integral is over the small parallelepiped centered on $\mathbf{q}_i$, so that integrals over the first Brillouin zone of the form

$$I=\int_{\rm BZ}\frac{f(\mathbf{q})}{q^2}\,d\mathbf{q}\,,$$

can be approximated by the sum

$$I \approx \sum_i w_{\mathbf{q}_i} f(\mathbf{q}_i)$$

which converges rapidly with respect to the number of q-points for smooth functions f. The integral in (1) is determined by evaluating it numerically on increasingly finer grids and extrapolating to the continuum. Agreement with Mathematica to at least 10 significant figures.

REVISION HISTORY:

Created August 2004 (JKD,SS)

2.0.241 vnlrhomt (Source File: vnlrhomt.f90)

INTERFACE:

```
Subroutine vnlrhomt (tsh, is, wfmt1, wfmt2, zrhomt)
```

USES:

```
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
tsh    : .true. if the density is to be in spherical harmonics (in,logical)
is     : species number (in,integer)
wfmt1  : muffin-tin part of wavefunction 1 in spherical coordinates
         (in,complex(lmmaxvr,nrcmtmax,natmtot,nspinor))
wfmt2  : muffin-tin part of wavefunction 2 in spherical coordinates
         (in,complex(lmmaxvr,nrcmtmax,natmtot,nspinor))
zrhomt : muffin-tin charge density in spherical harmonics/coordinates
         (out,complex(lmmaxvr,nrcmtmax))
```

DESCRIPTION:

Calculates the complex overlap density in a single muffin-tin from two input wavefunctions expressed in spherical coordinates. If `tsh` is `.true.` then the output density is converted to a spherical harmonic expansion. See routine `vnlrho`.

REVISION HISTORY:

Created November 2004 (Sharma)

2.0.242 vnlrho (Source File: vnlrho.f90)

INTERFACE:

```
Subroutine vnlrho (tsh, wfmt1, wfmt2, wfir1, wfir2, zrhomt, zrhoir)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
tsh    : .true. if the muffin-tin density is to be in spherical harmonics
         (in,logical)
wfmt1  : muffin-tin part of wavefunction 1 in spherical coordinates
         (in,complex(lmmaxvr,nrcmtmax,natmtot,nspinor))
wfmt2  : muffin-tin part of wavefunction 2 in spherical coordinates
         (in,complex(lmmaxvr,nrcmtmax,natmtot,nspinor))
wfir1  : interstitial wavefunction 1 (in,complex(ngrtot))
wfir2  : interstitial wavefunction 2 (in,complex(ngrtot))
zrhomt : muffin-tin charge density in spherical harmonics/coordinates
         (out,complex(lmmaxvr,nrcmtmax,natmtot))
zrhoir : interstitial charge density (out,complex(ngrtot))
```

DESCRIPTION:

Calculates the complex overlap charge density from two input wavefunctions:

$$\rho(\mathbf{r}) \equiv \Psi_1^{\dagger}(\mathbf{r}) \cdot \Psi_2(\mathbf{r}).$$

Note that the muffin-tin wavefunctions are provided in spherical coordinates and the returned density is either in terms of spherical harmonic coefficients or spherical coordinates when `tsh` is `.true.` or `.false.`, respectively. See also the routine `vnlrhomt`.

REVISION HISTORY:

```
Created November 2004 (Sharma)
```

2.0.243 olprad (Source File: olprad.f90)

INTERFACE:

```
Subroutine olprad
```

USES:

```
Use modmain
```

DESCRIPTION:

Calculates the radial overlap integrals of the APW and local-orbital basis functions. In other words, for spin σ and atom j of species i, it computes integrals of the form

$$o_{qp}^{\sigma;ij} = \int_0^{R_i} u_{q;l_p}^{\sigma;ij}(r) v_p^{\sigma;ij}(r) r^2 dr$$

and

$$o_{pp'}^{\sigma;ij} = \int_0^{R_i} v_p^{\sigma;ij}(r) v_{p'}^{\sigma;ij}(r) r^2 dr, \quad l_p = l_{p'}$$

where $u_{q;l}^{\sigma;ij}$ is the qth APW radial function for angular momentum l; and $v_p^{\sigma;ij}$ is the pth local-orbital radial function and has angular momentum l_p.

REVISION HISTORY:

```
Created November 2003 (JKD)
```

2.0.244 hmlrad (Source File: hmlrad.f90)

INTERFACE:

```
Subroutine hmlrad
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Calculates the radial Hamiltonian integrals of the APW and local-orbital basis functions. In other words, for spin σ and atom j of species i, it computes integrals of the form

$$h_{qq';ll'l''m''}^{\sigma;ij} = \begin{cases} \int_0^{R_i} u_{q;l}^{\sigma;ij}(r) H u_{q';l'}^{\sigma;ij}(r) r^2 dr & l''=0 \\ \int_0^{R_i} u_{q;l}^{\sigma;ij}(r) V_{l''m''}^{\sigma;ij}(r) u_{q';l'}^{\sigma;ij}(r) r^2 dr & l''>0 \end{cases},$$

where $u_{q;l}^{\sigma;ij}$ is the qth APW radial function for angular momentum l; H is the Hamiltonian of the radial Schrödinger equation; and $V_{l''m''}^{\sigma;ij}$ is the effective muffin-tin potential. Similar integrals are calculated for APW-local-orbital and local-orbital-local-orbital contributions.

REVISION HISTORY:

```
Created December 2003 (JKD)
```

2.0.245 hmlistl (Source File: hmlistl.f90)

INTERFACE:

```
Subroutine hmlistl (tapp, ngp, igpig, vgpc, v, h)
```

USES:

```
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
tapp  : .true. if the Hamiltonian is to be applied to the input vector,
        .false. if the full matrix is to be calculated (in,logical)
ngp   : number of G+p-vectors (in,integer)
igpig : index from G+p-vectors to G-vectors (in,integer(ngkmax))
vgpc  : G+p-vectors in Cartesian coordinates (in,real(3,ngkmax))
v     : input vector to which H is applied if tapp is .true., otherwise
        not referenced (in,complex(nmatmax))
h     : H applied to v if tapp is .true., otherwise it is the Hamiltonian
        matrix in packed form (inout,complex(*))
```

DESCRIPTION:

Computes the interstitial contribution to the Hamiltonian matrix for the APW basis functions. The Hamiltonian is given by

$$H^{\rm I}(\mathbf{G}+\mathbf{k},\mathbf{G}'+\mathbf{k}) = \frac{1}{2}(\mathbf{G}+\mathbf{k})\cdot(\mathbf{G}'+\mathbf{k})\tilde{\Theta}(\mathbf{G}-\mathbf{G}') + V^{\sigma}(\mathbf{G}-\mathbf{G}'),$$

where V^{σ} is the effective interstitial potential for spin σ and $\tilde{\Theta}$ is the characteristic function. See routine `gencfun`.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.246 hmlaa (Source File: hmlaa.f90)

INTERFACE:

```
Subroutine hmlaa (tapp, is, ia, ngp, apwalm, v, h)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
tapp   : .true. if the Hamiltonian is to be applied to the input vector,
         .false. if the full matrix is to be calculated (in,logical)
is     : species number (in,integer)
ia     : atom number (in,integer)
ngp    : number of G+p-vectors (in,integer)
apwalm : APW matching coefficients
         (in,complex(ngkmax,apwordmax,lmmaxapw,natmtot))
v      : input vector to which H is applied if tapp is .true., otherwise
         not referenced (in,complex(nmatmax))
h      : H applied to v if tapp is .true., otherwise it is the Hamiltonian
         matrix in packed form (inout,complex(*))
```

DESCRIPTION:

Calculates the APW-APW contribution to the Hamiltonian matrix.
REVISION HISTORY:

```
Created October 2002 (JKD)
```

2.0.247 hmlaa (Source File: hmlaan.f90)

INTERFACE:

```
Subroutine hmlaan (hamilton, is, ia, ngp, apwalm)
```

USES:

```
Use modinput
Use modmain
Use modfvsystem
```

INPUT/OUTPUT PARAMETERS:

```
tapp   : .true. if the Hamiltonian is to be applied to the input vector,
         .false. if the full matrix is to be calculated (in,logical)
is     : species number (in,integer)
ia     : atom number (in,integer)
ngp    : number of G+p-vectors (in,integer)
apwalm : APW matching coefficients
         (in,complex(ngkmax,apwordmax,lmmaxapw,natmtot))
v      : input vector to which H is applied if tapp is .true., otherwise
         not referenced (in,complex(nmatmax))
h      : H applied to v if tapp is .true., otherwise it is the Hamiltonian
         matrix in packed form (inout,complex(npmatmax))
```

DESCRIPTION:

Calculates the APW-APW contribution to the Hamiltonian matrix.
REVISION HISTORY:

```
Created October 2002 (JKD)
```

2.0.248 olpistl (Source File: olpistl.f90)

INTERFACE:

```
Subroutine olpistl (tapp, ngp, igpig, v, o)
```

USES:

```
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
ngp   : number of G+p-vectors (in,integer)
igpig : index from G+p-vectors to G-vectors (in,integer(ngkmax))
v     : input vector to which O is applied if tapp is .true., otherwise
        not referenced (in,complex(nmatmax))
o     : O applied to v if tapp is .true., otherwise it is the overlap
        matrix in packed form (inout,complex(*))
```

DESCRIPTION:

Computes the interstitial contribution to the overlap matrix for the APW basis functions. The overlap is given by

$$O^{\rm I}(\mathbf{G}+\mathbf{k},\mathbf{G}'+\mathbf{k})=\tilde{\Theta}(\mathbf{G}-\mathbf{G}'),$$

where $\tilde{\Theta}$ is the characteristic function. See routine `gencfun`.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.249 olpistl (Source File: olpistln.f90)

INTERFACE:

```
Subroutine olpistln (overlap, ngp, igpig)
```

USES:

```
Use modmain
Use modfvsystem
```

INPUT/OUTPUT PARAMETERS:

```
ngp   : number of G+p-vectors (in,integer)
igpig : index from G+p-vectors to G-vectors (in,integer(ngkmax))
v     : input vector to which O is applied if tapp is .true., otherwise
        not referenced (in,complex(nmatmax))
o     : O applied to v if tapp is .true., otherwise it is the overlap
      : matrix in packed form (inout,complex(npmatmax))
```

DESCRIPTION:

Computes the interstitial contribution to the overlap matrix for the APW basis functions. The overlap is given by

$$O^{\rm I}(\mathbf{G}+\mathbf{k},\mathbf{G}'+\mathbf{k})=\tilde{\Theta}(\mathbf{G}-\mathbf{G}'),$$

where $\tilde{\Theta}$ is the characteristic function. See routine `gencfun`.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.250 hmlistl (Source File: hmlistln.f90)

INTERFACE:

```
Subroutine hmlistln (hamilton, ngp, igpig, vgpc)
```

USES:

```
Use modmain
Use modfvsystem
```

INPUT/OUTPUT PARAMETERS:

```
tapp  : .true. if the Hamiltonian is to be applied to the input vector,
        .false. if the full matrix is to be calculated (in,logical)
ngp   : number of G+p-vectors (in,integer)
igpig : index from G+p-vectors to G-vectors (in,integer(ngkmax))
vgpc  : G+p-vectors in Cartesian coordinates (in,real(3,ngkmax))
v     : input vector to which H is applied if tapp is .true., otherwise
        not referenced (in,complex(nmatmax))
h     : H applied to v if tapp is .true., otherwise it is the Hamiltonian
        matrix in packed form (inout,complex(npmatmax))
```

DESCRIPTION:

Computes the interstitial contribution to the Hamiltonian matrix for the APW basis functions. The Hamiltonian is given by

$$H^{\rm I}(\mathbf{G}+\mathbf{k},\mathbf{G}'+\mathbf{k})=\frac{1}{2}(\mathbf{G}+\mathbf{k})\cdot(\mathbf{G}'+\mathbf{k})\tilde{\Theta}(\mathbf{G}-\mathbf{G}')+V^{\sigma}(\mathbf{G}-\mathbf{G}'),$$

where V^{σ} is the effective interstitial potential for spin σ and $\tilde{\Theta}$ is the characteristic function. See routine `gencfun`.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.251 mpiresumeevec (Source File: mpiresumeevecfiles.F90.orig)

INTERFACE:

```
Subroutine mpiresumeevecfiles
      Use modmain
#ifdef MPI
      Use modmpi
```

DESCRIPTION:

Routine reads the lokal EIGVECk1-k2.OUT files that were created to avoid file system inconsistencies. And writes them into the standard EIGVEC.OUT File

REVISION HISTORY:

```
Created October SEPT 2006 (MULEOBEN)
by Cristian Meisenbichler
```

2.0.252 mpisumrhoandmag (Source File: mpisumrhoandmag.F90)

INTERFACE:

```
Subroutine mpisumrhoandmag
#ifdef MPI
      Use modinput
```

USES:

```
      Use modmpi
      Use modmain
```

DESCRIPTION:

This subroutine adds up the partial sums for the charge density, the magnetisation, and the partial charges for all processors and broadcasts it to all processors.

REVISION HISTORY:

```
Created October September 2006 (Christian Meisenbichler)
Modifications, August 2010 (Stephan Sagmeister)
```

2.0.253 mpiresumeevec (Source File: mpiresumeevecfiles.F90)

INTERFACE:

```
Subroutine mpiresumeevecfiles
      Use modmain
#ifdef MPI
      Use modmpi
```

DESCRIPTION:

Routine reads the lokal EIGVECk1-k2.OUT files that were created to avoid file system inconsistencies. And writes them into the standard EIGVEC.OUT File

REVISION HISTORY:

```
Created October SEPT 2006 (MULEOBEN)
by Cristian Meisenbichler
```

2.0.254 ggair_2a (Source File: ggair_2a.f90)

INTERFACE:

```
subroutine ggair_2a(g2rho, gvrho, grho2)
```

USES:

```
use modinput
use mod_Gvector
use mod_potential_and_density
```

DESCRIPTION:

Spin-unpolarised version of ggair_sp_2a.

REVISION HISTORY:

```
Created November 2009 (JKD and TMcQ)
```

2.0.255 xc_pbe (Source File: xc_pbe.f90)

INTERFACE:

```
subroutine xc_pbe(n,kappa,mu,beta,rhoup,rhodn,grho,gup,gdn,g2up,g2dn,g3rho, &
 g3up,g3dn,ex,ec,vxup,vxdn,vcup,vcdn)
```

INPUT/OUTPUT PARAMETERS:

```
n     : number of density points (in,integer)
kappa : parameter for large-gradient limit (in,real)
mu    : gradient expansion coefficient (in,real)
beta  : gradient expansion coefficient (in,real)
rhoup : spin-up charge density (in,real(n))
rhodn : spin-down charge density (in,real(n))
grho  : |grad rho| (in,real(n))
gup   : |grad rhoup| (in,real(n))
gdn   : |grad rhodn| (in,real(n))
g2up  : grad^2 rhoup (in,real(n))
g2dn  : grad^2 rhodn (in,real(n))
g3rho : (grad rho).(grad |grad rho|) (in,real(n))
g3up  : (grad rhoup).(grad |grad rhoup|) (in,real(n))
g3dn  : (grad rhodn).(grad |grad rhodn|) (in,real(n))
ex    : exchange energy density (out,real(n))
ec    : correlation energy density (out,real(n))
vxup  : spin-up exchange potential (out,real(n))
vxdn  : spin-down exchange potential (out,real(n))
vcup  : spin-up correlation potential (out,real(n))
vcdn  : spin-down correlation potential (out,real(n))
```

DESCRIPTION:

Spin-polarised exchange-correlation potential and energy of the generalised gradient approximation functional of J. P. Perdew, K. Burke and M. Ernzerhof *Phys. Rev. Lett.* **77**, 3865 (1996) and **78**, 1396(E) (1997). The parameter κ, which controls the large-gradient limit, can be set to 0.804 or 1.245 corresponding to the value in the original article or the revised version of Y. Zhang and W. Yang, *Phys. Rev. Lett.* **80**, 890 (1998).
REVISION HISTORY:

```
Modified routines written by K. Burke, October 2004 (JKD)
```

2.0.256 xc_pzca (Source File: xc_pzca.f90)

INTERFACE:

```
subroutine xc_pzca(n,rho,ex,ec,vx,vc)
```

INPUT/OUTPUT PARAMETERS:

```
n   : number of density points (in,integer)
rho : charge density (in,real(n))
ex  : exchange energy density (out,real(n))
ec  : correlation energy density (out,real(n))
vx  : exchange potential (out,real(n))
vc  : correlation potential (out,real(n))
```

DESCRIPTION:

Spin-unpolarised exchange-correlation potential and energy of the Perdew-Zunger parameterisation of Ceperley-Alder electron gas: *Phys. Rev. B* **23**, 5048 (1981) and *Phys. Rev. Lett.* **45**, 566 (1980).
REVISION HISTORY:

```
Created October 2002 (JKD)
```

2.0.257 ggamt_1 (Source File: ggamt_1.f90)

INTERFACE:

```
subroutine ggamt_1(is, ia, grho, g2rho, g3rho)
```

USES:

```
use modinput
use mod_Gvector
use mod_muffin_tin
use mod_SHT
use mod_atoms
use mod_potential_and_density
```

DESCRIPTION:

Spin-unpolarised version of `ggamt_sp_1`.

REVISION HISTORY:

```
Created November 2009 (JKD)
```

2.0.258 ggair_sp_1 (Source File: ggair_sp_1.f90)

INTERFACE:

```
subroutine ggair_sp_1(rhoup, rhodn, grho, gup, gdn, g2up, g2dn, g3rho, g3up, g3dn)
```

INPUT/OUTPUT PARAMETERS:

```
use mod_Gvector
    rhoup : spin-up density (in,real(ngrtot))
    rhodn : spin-down density (in,real(ngrtot))
    grho  : |grad rho| (out,real(ngrtot))
    gup   : |grad rhoup| (out,real(ngrtot))
    gdn   : |grad rhodn| (out,real(ngrtot))
    g2up  : grad^2 rhoup (out,real(ngrtot))
    g2dn  : grad^2 rhodn (out,real(ngrtot))
    g3rho : (grad rho).(grad |grad rho|) (out,real(ngrtot))
    g3up  : (grad rhoup).(grad |grad rhoup|) (out,real(ngrtot))
    g3dn  : (grad rhodn).(grad |grad rhodn|) (out,real(ngrtot))
```

DESCRIPTION:

Computes $|\nabla\rho|$, $|\nabla\rho^{\uparrow}|$, $|\nabla\rho^{\downarrow}|$, $\nabla^2\rho^{\uparrow}$, $\nabla^2\rho^{\downarrow}$, $\nabla\rho\cdot(\nabla|\nabla\rho|)$, $\nabla\rho^{\uparrow}\cdot(\nabla|\nabla\rho^{\uparrow}|)$ and $\nabla\rho^{\downarrow}\cdot(\nabla|\nabla\rho^{\downarrow}|)$ for the interstitial charge density, as required by the generalised gradient approximation functionals of type 1 for spin-polarised densities. See routines `ggair_1` `potxc` and `modxcifc`.

REVISION HISTORY:

```
Created October 2004 (JKD)
Simplified and improved, October 2009 (JKD)
Modified third order gradients: improved numerical stability, April 2011
(S. Sagmeister)
```

2.0.259 ggair_2b (Source File: ggair_2b.f90)

INTERFACE:

```
subroutine ggair_2b(g2rho, gvrho, vx, vc, dxdg2, dcdg2)
```

USES:

```
use modinput
use mod_Gvector
use mod_potential_and_density
```

DESCRIPTION:

Spin-unpolarised version of `ggair_sp_2b`.

REVISION HISTORY:

```
Created November 2009 (JKD and TMcQ)
```

2.0.260 xc_xalpha (Source File: xc_xalpha.f90)

INTERFACE:

```
subroutine xc_xalpha(n,rho,exc,vxc)
```

INPUT/OUTPUT PARAMETERS:

```
n   : number of density points (in,integer)
rho : charge density (in,real(n))
exc : exchange-correlation energy density (out,real(n))
vxc : exchange-correlation potential (out,real(n))
```

DESCRIPTION:

X_α approximation to the exchange-correlation potential and energy density. See J. C. Slater, *Phys. Rev.* **81**, 385 (1951).

REVISION HISTORY:

```
Modified an ABINIT routine, September 2006 (JKD)
```

2.0.261 ggair_1 (Source File: ggair_1.f90)

INTERFACE:

```
subroutine ggair_1(grho, g2rho, g3rho)
```

USES:

```
use mod_Gvector
use mod_potential_and_density
```

DESCRIPTION:

Spin-unpolarised version of `ggair_sp_1`. The gradient $(\nabla\rho)\cdot\nabla(|\nabla\rho|)$ is evaluated using the relation

$$(\nabla\rho)\cdot\nabla(|\nabla\rho|) = \frac{(\nabla\rho)\cdot(\nabla\otimes\nabla\rho)\cdot(\nabla\rho)}{|\nabla\rho|}.$$

REVISION HISTORY:

```
Created November 2009 (JKD)
Modified third order gradients: improved numerical stability, April 2011
(S. Sagmeister)
```

2.0.262 ggamt_sp_2b (Source File: ggamt_sp_2b.f90)

INTERFACE:

```
subroutine ggamt_sp_2b(is, g2up, g2dn, gvup, gvdn, vxup, vxdn, vcup, vcdn, dxdgu2, &
 dxdgd2, dxdgud, dcdgu2, dcdgd2, dcdgud)
```

USES:

```
use modinput
use mod_Gvector
use mod_muffin_tin
use mod_SHT
use mod_atoms
```

DESCRIPTION:

Post processing step of muffin-tin gradients for GGA type 2. See routine `ggamt_sp_2a` for full details.

REVISION HISTORY:

```
Created November 2009 (JKD and TMcQ)
```

2.0.263 xc_vbh (Source File: xc_vbh.f90)

INTERFACE:

```
subroutine xc_vbh(n,rhoup,rhodn,ex,ec,vxup,vxdn,vcup,vcdn)
```

INPUT/OUTPUT PARAMETERS:

```
n     : number of density points (in,integer)
rhoup : spin-up charge density (in,real(n))
rhodn : spin-down charge density (in,real(n))
ex    : exchange energy density (out,real(n))
ec    : correlation energy density (out,real(n))
vxup  : spin-up exchange potential (out,real(n))
vxdn  : spin-down exchange potential (out,real(n))
vcup  : spin-up correlation potential (out,real(n))
vcdn  : spin-down correlation potential (out,real(n))
```

DESCRIPTION:

Spin-polarised exchange-correlation potential and energy functional of von Barth and Hedin: *J. Phys. C* **5**, 1629 (1972). Note that the implementation is in Rydbergs in order to follow the paper step by step, at the end the potential and energy are converted to Hartree.

REVISION HISTORY:

Created September 2007 (F. Cricchio)

2.0.264 ggamt_sp_2a (Source File: ggamt_sp_2a.f90)

INTERFACE:

```
subroutine ggamt_sp_2a(is, rhoup, rhodn, g2up, g2dn, gvup, gvdn, gup2, gdn2, gupdn)
```

USES:

```
use modinput
use mod_Gvector
use mod_muffin_tin
use mod_SHT
use mod_atoms
use mod_potential_and_density
```

DESCRIPTION:

Computes the muffin-tin gradients $\nabla^2\rho^\uparrow$, $\nabla^2\rho^\downarrow$, $\nabla\rho^\uparrow$, $\nabla\rho^\downarrow$, $(\nabla\rho^\uparrow)^2$, $(\nabla\rho^\downarrow)^2$ and $\nabla\rho^\uparrow\cdot\nabla\rho^\downarrow$, which are passed in to GGA functional subroutines of type 2. The exchange-correlation energy in these routines has the functional form

$$E_{xc}[\rho^\uparrow,\rho^\downarrow]=\int d^3r\,\hat{\epsilon}_{xc}\big(\rho^\uparrow(\mathbf{r}),\rho^\downarrow(\mathbf{r}),(\nabla\rho^\uparrow(\mathbf{r}))^2,(\nabla\rho^\downarrow(\mathbf{r}))^2,\nabla\rho^\uparrow(\mathbf{r})\cdot\nabla\rho^\downarrow(\mathbf{r})\big),$$

where $\hat{\epsilon}_{xc}(\mathbf{r})=\epsilon_{xc}(\mathbf{r})\rho(\mathbf{r})$ is the xc energy per unit volume, with ϵ_{xc} being the xc energy per electron, and $\rho=\rho^\uparrow+\rho^\downarrow$. From the gradients above, type 2 GGA routines return ϵ_{xc}, but not directly the xc potentials. Instead they generate the derivatives $\partial\hat{\epsilon}_{xc}/\partial\rho^\uparrow(\mathbf{r})$, $\partial\hat{\epsilon}_{xc}/\partial(\nabla\rho^\uparrow(\mathbf{r}))^2$, and the same for down spin, as well as $\partial\hat{\epsilon}_{xc}/\partial(\nabla\rho^\uparrow(\mathbf{r})\cdot\nabla\rho^\downarrow(\mathbf{r}))$. In a post-processing step invoked by `ggamt_sp_2b`, integration by parts is used to obtain the xc potential explicitly with

$$V_{xc}^\uparrow(\mathbf{r})=\frac{\partial\hat{\epsilon}_{xc}}{\partial\rho^\uparrow(\mathbf{r})}-2\left(\nabla\frac{\partial\hat{\epsilon}_{xc}}{\partial(\nabla\rho^\uparrow)^2}\right)\cdot\nabla\rho^\uparrow-2\frac{\hat{\epsilon}_{xc}}{\partial(\nabla\rho^\uparrow)^2}\nabla^2\rho^\uparrow\&\quad-\left(\nabla\frac{\hat{\epsilon}_{xc}}{\partial(\nabla\rho^\uparrow\cdot\nabla\rho^\downarrow)}\right)\cdot\nabla\rho^\downarrow-\frac{\partial\hat{\epsilon}_{xc}}{\partial(\nabla\rho^\uparrow\cdot\nabla\rho^\downarrow)}\nabla^2\rho^\downarrow,$$

and similarly for $V_{xc}^\downarrow$.

REVISION HISTORY:

Created November 2009 (JKD and TMcQ)

2.0.265 xc_pwca (Source File: xc_pwca.f90)

INTERFACE:

```
subroutine xc_pwca(n,rhoup,rhodn,ex,ec,vxup,vxdn,vcup,vcdn)
```

INPUT/OUTPUT PARAMETERS:

```
n     : number of density points (in,integer)
rhoup : spin-up charge density (in,real(n))
rhodn : spin-down charge density (in,real(n))
ex    : exchange energy density (out,real(n))
ec    : correlation energy density (out,real(n))
vxup  : spin-up exchange potential (out,real(n))
vxdn  : spin-down exchange potential (out,real(n))
vcup  : spin-up correlation potential (out,real(n))
vcdn  : spin-down correlation potential (out,real(n))
```

DESCRIPTION:

Spin-polarised exchange-correlation potential and energy of the Perdew-Wang parameterisation of the Ceperley-Alder electron gas: *Phys. Rev. B* **45**, 13244 (1992) and *Phys. Rev. Lett.* **45**, 566 (1980).

REVISION HISTORY:

```
Created January 2004 (JKD)
```

2.0.266 ggair_sp_2a (Source File: ggair_sp_2a.f90)

INTERFACE:

```
subroutine ggair_sp_2a(rhoup, rhodn, g2up, g2dn, gvup, gvdn, gup2, gdn2, gupdn)
```

USES:

```
use modinput
use mod_Gvector
```

DESCRIPTION:

Computes the interstitial gradients $\nabla^2\rho^\uparrow$, $\nabla^2\rho^\downarrow$, $\nabla\rho^\uparrow$, $\nabla\rho^\downarrow$, $(\nabla\rho^\uparrow)^2$, $(\nabla\rho^\downarrow)^2$ and $\nabla\rho^\uparrow\cdot\nabla\rho^\downarrow$. These are used for GGA functionals of type 2. See `ggamt_sp_2a` for full details.

REVISION HISTORY:

```
Created November 2009 (JKD and TMcQ)
```

2.0.267 xcifc (Source File: modxcifc.f90)

INTERFACE:

```
subroutine xcifc(xctype,n,rho,rhoup,rhodn,grho,gup,gdn,g2rho,g2up,g2dn,g3rho, &
 g3up,g3dn,grho2,gup2,gdn2,gupdn,ex,ec,vx,vc,vxup,vxdn,vcup,vcdn,dxdg2,dxdgu2, &
 dxdgd2,dxdgud,dcdg2,dcdgu2,dcdgd2,dcdgud)
```

INPUT/OUTPUT PARAMETERS:

```
xctype : type of exchange-correlation functional (in,integer(3))
n      : number of density points (in,integer)
rho    : spin-unpolarised charge density (in,real(n),optional)
rhoup  : spin-up charge density (in,real(n),optional)
rhodn  : spin-down charge density (in,real(n),optional)
grho   : |grad rho| (in,real(n),optional)
gup    : |grad rhoup| (in,real(n),optional)
gdn    : |grad rhodn| (in,real(n),optional)
g2rho  : grad^2 rho (in,real(n),optional)
g2up   : grad^2 rhoup (in,real(n),optional)
g2dn   : grad^2 rhodn (in,real(n),optional)
g3rho  : (grad rho).(grad |grad rho|) (in,real(n),optional)
g3up   : (grad rhoup).(grad |grad rhoup|) (in,real(n),optional)
g3dn   : (grad rhodn).(grad |grad rhodn|) (in,real(n),optional)
grho2  : |grad rho|^2 (in,real(n),optional)
gup2   : |grad rhoup|^2 (in,real(n),optional)
gdn2   : |grad rhodn|^2 (in,real(n),optional)
gupdn  : (grad rhoup).(grad rhodn) (in,real(n),optional)
ex     : exchange energy density (out,real(n),optional)
ec     : correlation energy density (out,real(n),optional)
vx     : spin-unpolarised exchange potential (out,real(n),optional)
vc     : spin-unpolarised correlation potential (out,real(n),optional)
vxup   : spin-up exchange potential (out,real(n),optional)
vxdn   : spin-down exchange potential (out,real(n),optional)
vcup   : spin-up correlation potential (out,real(n),optional)
vcdn   : spin-down correlation potential (out,real(n),optional)
dxdg2  : de_x/d(|grad rho|^2) (out,real(n),optional)
dxdgu2 : de_x/d(|grad rhoup|^2) (out,real(n),optional)
dxdgd2 : de_x/d(|grad rhodn|^2) (out,real(n),optional)
dxdgud : de_x/d((grad rhoup).(grad rhodn)) (out,real(n),optional)
dcdg2  : de_c/d(|grad rho|^2) (out,real(n),optional)
dcdgu2 : de_c/d(|grad rhoup|^2) (out,real(n),optional)
dcdgd2 : de_c/d(|grad rhodn|^2) (out,real(n),optional)
dcdgud : de_c/d((grad rhoup).(grad rhodn)) (out,real(n),optional)
```

DESCRIPTION:

Interface to the exchange-correlation routines.

REVISION HISTORY:

```
Created October 2002 (JKD)
```

2.0.268 getxcdata (Source File: modxcifc.f90)

INTERFACE:

```
subroutine getxcdata(xctype,xcdescr,xcspin,xcgrad)
```

INPUT/OUTPUT PARAMETERS:

```
xctype  : type of exchange-correlation functional (in,integer(3))
xcdescr : description of functional (out,character(256))
xcspin  : spin treatment (out,integer)
xcgrad  : gradient treatment (out,integer)
```

DESCRIPTION:

Returns data on the exchange-correlation functional labeled by `xctype`. The character array `xcdescr` contains a short description of the functional including journal references. The variable `xcspin` is set to 1 or 0 for spin-polarised or -unpolarised functionals, respectively. For functionals which require the gradients of the density `xcgrad` is set to 1, otherwise it is set to 0.

REVISION HISTORY:

```
Created October 2002 (JKD)
```

2.0.269 ggamt_sp_1 (Source File: ggamt_sp_1.f90)

INTERFACE:

```
subroutine ggamt_sp_1(is, rhoup, rhodn, grho, gup, gdn, g2up, g2dn, g3rho, g3up, g3dn)
```

USES:

```
use modinput
use mod_Gvector
use mod_muffin_tin
use mod_SHT
use mod_atoms
use mod_potential_and_density
```

INPUT/OUTPUT PARAMETERS:

```
is    : species number (in,integer)
rhoup : spin-up density in spherical coordinates (in,real(lmmaxvr,nrmtmax))
rhodn : spin-down density (in,real(lmmaxvr,nrmtmax))
grho  : |grad rho| (out,real(lmmaxvr,nrmtmax))
gup   : |grad rhoup| (out,real(lmmaxvr,nrmtmax))
gdn   : |grad rhodn| (out,real(lmmaxvr,nrmtmax))
g2up  : grad^2 rhoup (out,real(lmmaxvr,nrmtmax))
```

```
g2dn  : grad^2 rhodn (out,real(lmmaxvr,nrmtmax))
g3rho : (grad rho).(grad |grad rho|) (out,real(lmmaxvr,nrmtmax))
g3up  : (grad rhoup).(grad |grad rhoup|) (out,real(lmmaxvr,nrmtmax))
g3dn  : (grad rhodn).(grad |grad rhodn|) (out,real(lmmaxvr,nrmtmax))
```

DESCRIPTION:

Computes $|\nabla\rho|$, $|\nabla\rho^{\uparrow}|$, $|\nabla\rho^{\downarrow}|$, $\nabla^2\rho^{\uparrow}$, $\nabla^2\rho^{\downarrow}$, $\nabla\rho\cdot(\nabla|\nabla\rho|)$, $\nabla\rho^{\uparrow}\cdot(\nabla|\nabla\rho^{\uparrow}|)$ and $\nabla\rho^{\downarrow}\cdot(\nabla|\nabla\rho^{\downarrow}|)$ for a muffin-tin charge density, as required by the generalised gradient approximation functionals of type 1 for spin-polarised densities. The input densities and output gradients are in terms of spherical coordinates. See routines `potxc` and `modxcifc`.

REVISION HISTORY:

```
Created April 2004 (JKD)
Simplified and improved, October 2009 (JKD)
```

2.0.270 ggamt_2b (Source File: ggamt_2b.f90)

INTERFACE:

```
subroutine ggamt_2b(is, g2rho, gvrho, vx, vc, dxdg2, dcdg2)
```

USES:

```
use modinput
use mod_Gvector
use mod_muffin_tin
use mod_SHT
use mod_atoms
```

DESCRIPTION:

Spin-unpolarised version of `ggamt_sp_2b`.

REVISION HISTORY:

```
Created November 2009 (JKD and TMcQ)
```

2.0.271 ggamt_2a (Source File: ggamt_2a.f90)

INTERFACE:

```
subroutine ggamt_2a(is, ia, g2rho, gvrho, grho2)
```

USES:

```
use modinput
use mod_Gvector
use mod_muffin_tin
use mod_SHT
use mod_atoms
use mod_potential_and_density
```

DESCRIPTION:

Spin-unpolarised version of `ggamt_sp_2a`.

REVISION HISTORY:

```
Created November 2009 (JKD and TMcQ)
```

2.0.272 spline (Source File: splline4.f90)

INTERFACE:

```
subroutine spline4(n,x,ld,f,cf)
```

INPUT/OUTPUT PARAMETERS:

```
n  : number of points (in,integer)
x  : abscissa array (in,real(n))
ld : leading dimension (in,integer)
f  : input data array (in,real(ld,n))
cf : cubic spline coefficients (1,2,3) and work space (4) (out,real(4,n))
```

DESCRIPTION:

Calculates the coefficients of a cubic spline fitted to input data. In other words, given a set of data points f_i defined at x_i, where $i = 1 \ldots n$, the coefficients c_j^i are determined such that

$$y_i(x) = f_i + c_1^i(x - x_i) + c_2^i(x - x_i)^2 + c_3^i(x - x_i)^3,$$

is the interpolating function for $x \in [x_i, x_{i+1})$. This is done by determining the end-point coefficients c_2^1 and c_2^n from the first and last three points, and then solving the tridiagonal system

$$d_{i-1}c_2^{i-1} + 2(d_{i-1} + d_i)c_2^i + d_i c_2^{i+1} = 3\left(\frac{f_{i+1} - f_i}{d_i} - \frac{f_i - f_{i-1}}{d_{i-1}}\right),$$

where $d_i = x_{i+1} - x_i$, for the intermediate coefficients.

REVISION HISTORY:

```
Created October 2004 (JKD)
Improved speed and accuracy, April 2006 (JKD)
Optimisations and improved end-point coefficients, February 2008 (JKD)
```

2.0.273 xcifc_libxc (Source File: libxcifc.f90)

INTERFACE:

```
subroutine xcifc_libxc(xctype,n,rho,rhoup,rhodn,grho2,gup2,gdn2,gupdn,ex,ec, &
 vx,vc,vxup,vxdn,vcup,vcdn,dxdg2,dxdgu2,dxdgd2,dxdgud,dcdg2,dcdgu2,dcdgd2, &
 dcdgud)
```

INPUT/OUTPUT PARAMETERS:

```
xctype : type of exchange-correlation functional (in,integer(3))
n      : number of density points (in,integer)
rho    : spin-unpolarised charge density (in,real(n),optional)
rhoup  : spin-up charge density (in,real(n),optional)
rhodn  : spin-down charge density (in,real(n),optional)
grho2  : |grad rho|^2 (in,real(n),optional)
gup2   : |grad rhoup|^2 (in,real(n),optional)
gdn2   : |grad rhodn|^2 (in,real(n),optional)
gupdn  : (grad rhoup).(grad rhodn) (in,real(n),optional)
ex     : exchange energy density (out,real(n),optional)
ec     : correlation energy density (out,real(n),optional)
vx     : spin-unpolarised exchange potential (out,real(n),optional)
vc     : spin-unpolarised correlation potential (out,real(n),optional)
vxup   : spin-up exchange potential (out,real(n),optional)
vxdn   : spin-down exchange potential (out,real(n),optional)
vcup   : spin-up correlation potential (out,real(n),optional)
vcdn   : spin-down correlation potential (out,real(n),optional)
dxdg2  : de_x/d(|grad rho|^2) (out,real(n),optional)
dxdgu2 : de_x/d(|grad rhoup|^2) (out,real(n),optional)
dxdgd2 : de_x/d(|grad rhodn|^2) (out,real(n),optional)
dxdgud : de_x/d((grad rhoup).(grad rhodn)) (out,real(n),optional)
dcdg2  : de_c/d(|grad rho|^2) (out,real(n),optional)
dcdgu2 : de_c/d(|grad rhoup|^2) (out,real(n),optional)
dcdgd2 : de_c/d(|grad rhodn|^2) (out,real(n),optional)
dcdgud : de_c/d((grad rhoup).(grad rhodn)) (out,real(n),optional)
```

DESCRIPTION:

Interface to the `libxc` exchange-correlation functional library: `http://www.tddft.org/programs/octopus/wiki/index.php/Libxc`. The second and third integers in `xctype` define the exchange and correlation functionals in `libxc`, respectively.

REVISION HISTORY:

```
Created April 2009 (Tyrel McQueen)
Modified September 2009 (JKD and TMQ)
```

2.0.274 ggair_sp_2b (Source File: ggair_sp_2b.f90)

INTERFACE:

```
subroutine ggair_sp_2b(g2up, g2dn, gvup, gvdn, vxup, vxdn, vcup, vcdn, dxdgu2, dxdgd2, &
 dxdgud, dcdgu2, dcdgd2, dcdgud)
use modinput
```

USES:

```
use mod_Gvector
```

DESCRIPTION:

Post processing step of interstitial gradients for GGA type 2. See routine `ggamt_sp_2a` for full details.

REVISION HISTORY:

```
Created November 2009 (JKD and TMcQ)
```

2.0.275 xc_am05 (Source File: xc_am05.f90)

INTERFACE:

```
subroutine xc_am05(n,rho,grho,g2rho,g3rho,ex,ec,vx,vc)
```

INPUT/OUTPUT PARAMETERS:

```
n     : number of density points (in,integer)
rho   : charge density (in,real(n))
grho  : |grad rho| (in,real(n))
g2rho : grad^2 rho (in,real(n))
g3rho : (grad rho).(grad |grad rho|) (in,real(n))
ex    : exchange energy density (out,real(n))
ec    : correlation energy density (out,real(n))
vx    : spin-unpolarised exchange potential (out,real(n))
vc    : spin-unpolarised correlation potential (out,real(n))
```

DESCRIPTION:

Spin-unpolarised exchange-correlation potential and energy functional of R. Armiento and A. E. Mattsson, *Phys. Rev. B* **72**, 085108 (2005).

REVISION HISTORY:

```
Created April 2005 (RAR); based on xc_pbe
```

2.0.276 xc_am05_point (Source File: xc_am05.f90)

INTERFACE:

```
subroutine xc_am05_point(rho,s,u,v,ex,ec,vx,vc,pot)
```

INPUT/OUTPUT PARAMETERS:

```
rho : electron density (in,real)
s   : gradient of n / (2 kF n)
u   : grad n * grad | grad n | / (n**2 (2 kF)**3)
v   : laplacian of density / (n**2 (2.d0*kf)**3)
ex  : exchange energy density (out,real)
ec  : correlation energy density (out,real)
vx  : spin-unpolarised exchange potential (out,real)
vc  : spin-unpolarised correlation potential (out,real)
```

DESCRIPTION:

Calculate the spin-unpolarised exchange-correlation potential and energy for the Armiento-Mattsson 05 functional for a single point.

REVISION HISTORY:

```
Created April 2005 (RAR)
```

2.0.277 xc_am05_ldax (Source File: xc_am05.f90)

INTERFACE:

```
subroutine xc_am05_ldax(n,ex,vx)
```

INPUT/OUTPUT PARAMETERS:

```
n  : electron density (in,real)
ex : exchange energy per electron (out,real)
vx : exchange potential (out,real)
```

DESCRIPTION:

Local density approximation exchange.

REVISION HISTORY:

```
Created April 2005 (RAR)
```

2.0.278 xc_am05_ldapwc (Source File: xc_am05.f90)

INTERFACE:

```
subroutine xc_am05_ldapwc(n,ec,vc)
```

INPUT/OUTPUT PARAMETERS:

```
n  : electron density (in,real)
ec : correlation energy per electron (out,real)
vc : correlation potential (out,real)
```

DESCRIPTION:

Correlation energy and potential of the Perdew-Wang parameterisation of the Ceperley-Alder electron gas *Phys. Rev. B* **45**, 13244 (1992) and *Phys. Rev. Lett.* **45**, 566 (1980). This is a clean-room implementation from paper.

REVISION HISTORY:

```
Created April 2005 (RAR)
```

2.0.279 xc_am05_labertw (Source File: xc_am05.f90)

INTERFACE:

```
subroutine xc_am05_labertw(z,val)
```

INPUT/OUTPUT PARAMETERS:

```
z   : function argument (in,real)
val : value of lambert W function of z (out,real)
```

DESCRIPTION:

Lambert W-function using the method of Corless, Gonnet, Hare, Jeffrey and Knuth, *Adv. Comp. Math.* **5**, 329 (1996). The approach is based loosely on that in GNU Octave by N. N. Schraudolph, but this implementation is for real values and the principal branch only.

REVISION HISTORY:

```
Created April 2005 (RAR)
```

2.0.280 loadinputDOM (Source File: modinputdom.f90)

INTERFACE:

```
Subroutine loadinputDOM (deffilename)
```

INPUT/OUTPUT PARAMETERS:

```
deffilename: default filename if no file name argument is in  argv()
    implicit none

    character(*),intent(in)::deffilename
```

DESCRIPTION:

Loads the contents of the inputfile into the DOM and initializes some data that is used by getstructinput(inputnp)

2.0.281 handleunknownnodes (Source File: modinputdom.f90)

INTERFACE:

```
Subroutine handleunknownnodes (np)
```

INPUT/OUTPUT PARAMETERS:

```
np: node pointer type containing node with all
    known nodes removed, thus containing only unknown that need to be reported.
```

DESCRIPTION:

This writes the error message when the getstruct... function sees an unknown (illegal) entry

---

INTERFACE:

```
Subroutine gensymcmut (eps, maxsymcrys, nsymcrys, symlat, lsplsymc, &
& vtlsymc, scmut, tabel, tspainvsym)
```

DESCRIPTION:

Sets up the group multiplication table. The table is checked for consistency in a way that it is required that every elements occurs once and only once in each row and column of the table. The first row and colmuns must consist of the indentity since the first symmetry element is the identity by convention.

REVISION HISTORY:

```
Created July 2008 (Sagmeister)
```

2.0.282 getevecfv (Source File: getevecfv.f90)

INTERFACE:

```
Subroutine getevecfv (vpl, vgpl, evecfv)
```

USES:

```
Use modmain
Use modinput
Use modmpi
```

DESCRIPTION:

The file where the (first-variational) eigenvectors are stored is `EVECFV.OUT`. It is a direct-access binary file, the record length of which can be determined with the help of the array sizes and data type information. One record of this file corresponds to one k-point in the irreducible Brillouin zone and has the following structure

k_{lat}	N_{mat}	N_{stfv}	N_{spfv}	Φ

The following table explains the parts of the record in more detail

name	type	shape	description
k_{lat}	real(8)	3	k-point in lattice coordinates
N_{mat}	integer	1	(L)APW basis size including local orbitals (maximum over k-points)
N_{stfv}	integer	1	number of (first-variational) states (without core states)
N_{spfv}	integer	1	first-variational spins (2 for spin-spirals, 1 otherwise)
Φ	complex(8)	$N_{\mathrm{mat}} \times N_{\mathrm{stfv}} \times N_{\mathrm{spfv}}$	(first-variational) eigenvector array

REVISION HISTORY:

```
Created Feburary 2007 (JKD)
Fixed transformation error, October 2007 (JKD, Anton Kozhevnikov)
Documentation added, Dec 2009 (S. Sagmeister)
Fixed l.o. rotation, June 2010 (A. Kozhevnikov)
```

2.0.283 getngkmax (Source File: getngkmax.f90)

INTERFACE:

```
Subroutine getngkmax
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Determines the largest number of $\mathbf{G}+\mathbf{k}$-vectors with length less than `gkmax` over all the k-points and stores it in the global variable `ngkmax`. This variable is used for allocating arrays.

REVISION HISTORY:

```
Created October 2004 (JKD)
```

2.0.284 init1 (Source File: init1.f90)

INTERFACE:

```
Subroutine init1
```

USES:

```
      Use modinput
      Use modmain
#ifdef TETRA
      Use modtetra
#endif
#ifdef XS
      Use modxs
#endif
```

DESCRIPTION:

Generates the k-point set and then allocates and initialises global variables which depend on the k-point set.

REVISION HISTORY:

```
Created January 2004 (JKD)
```

2.0.285 zpotcoul (Source File: zpotcoul.f90)

INTERFACE:

```
Subroutine zpotcoul (nr, nrmax, ld, r, igp0, gpc, jlgpr, ylmgp, sfacgp, &
& zn, zrhomt, zrhoir, zvclmt, zvclir, zrho0)
      Use modinput
```

USES:

```
      Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
nr     : number of radial points for each species (in,integer(nspecies))
nrmax  : maximum nr over all species (in,integer)
ld     : leading dimension of r (in,integer)
r      : radial mesh for each species (in,real(ld,nspecies))
igp0   : index of the shortest G+p-vector (in,integer)
gpc    : G+p-vector lengths (in,real(ngvec))
jlgpr  : spherical Bessel functions for evergy G+p-vector and muffin-tin
         radius (in,real(0:lmaxvr+npsden+1,ngvec,nspecies))
ylmgp  : spherical harmonics of the G+p-vectors (in,complex(lmmaxvr,ngvec))
sfacgp : structure factors of the G+p-vectors (in,complex(ngvec,natmtot))
zn     : nuclear charges at the atomic centers (in,real(nspecies))
zrhomt : muffin-tin charge density (in,complex(lmmaxvr,nrmax,natmtot))
zrhoir : interstitial charge density (in,complex(ngrtot))
zvclmt : muffin-tin Coulomb potential (out,complex(lmmaxvr,nrmax,natmtot))
zvclir : interstitial Coulomb potential (out,complex(ngrtot))
zrho0  : G+p=0 term of the pseudocharge density (out,complex)
```

DESCRIPTION:

Calculates the Coulomb potential of a complex charge density by solving Poisson's equation. First, the multipole moments of the muffin-tin charge are determined for the jth atom of the ith species by

$$q^{\rm MT}_{ij;lm} = \int_0^{R_i} r^{l+2}\rho_{ij;lm}(r)dr + z_{ij}Y_{00}\,\delta_{l,0}\ ,$$

where R_i is the muffin-tin radius and z_{ij} is a point charge located at the atom center (usually the nuclear charge, which should be taken as **negative**). Next, the multipole moments of the continuation of the interstitial density, $\rho^{\rm I}$, into the muffin-tin are found with

$$q^{\rm I}_{ij;lm} = 4\pi i^l R_i^{l+3}\sum_{\mathbf{G}}\frac{j_{l+1}(GR_i)}{GR_i}\rho^{\rm I}(\mathbf{G})\exp(i\mathbf{G}\cdot\mathbf{r}_{ij})Y^*_{lm}(\hat{\mathbf{G}}),$$

remembering that

$$\lim_{x\to 0}\frac{j_{l+n}(x)}{x^n} = \frac{1}{(2n+1)!!}\delta_{l,0}$$

should be used for the case $\mathbf{G}=0$. A pseudocharge is now constructed which is equal to the real density in the interstitial region and whose multipoles are the difference between the real and interstitial muffin-tin multipoles. This pseudocharge density is smooth in the sense that it can be expanded in terms of the finite set of $\mathbf{G}$-vectors. In each muffin-tin the pseudocharge has the form

$$\rho^{\rm P}_{ij}(\mathbf{r}) = \rho^{\rm I}(\mathbf{r}-\mathbf{r}_{ij}) + \sum_{lm}\rho^{\rm P}_{ij;lm}\frac{1}{R_i^{l+3}}\left(\frac{r}{R_i}\right)^l\left(1-\frac{r^2}{R_i^2}\right)^{N_i}Y_{lm}(\hat{\mathbf{r}})$$

where

$$\rho^{\rm P}_{ij;lm} = \frac{(2l+2N_i+3)!!}{2_i^N N_i!(2l+1)!!}\left(q^{\rm MT}_{ij;lm}-q^{\rm I}_{ij;lm}\right)$$

and $N_i \approx \frac{1}{2}R_iG_{\rm max}$ is generally a good choice. The pseudocharge in reciprocal-space is given by

$$\rho^{\rm P}(\mathbf{G}) = \rho^{\rm I}(\mathbf{G}) + \sum_{ij;lm}2^{N_i}N_i!\frac{4\pi(-i)^l}{\Omega R_i^l}\frac{j_{l+N_i+1}(GR_i)}{(GR_i)^{N_i+1}}\rho^{\rm P}_{ij;lm}\exp(-i\mathbf{G}\cdot\mathbf{r}_{ij})Y_{lm}(\hat{\mathbf{G}})$$

which may be used for solving Poisson's equation directly

$$V^{\rm P}(\mathbf{G}) = \begin{cases} 4\pi\frac{\rho^{\rm P}(\mathbf{G})}{G^2} & G>0 \\ 0 & G=0 \end{cases}.$$

The usual Green's function approach is then employed to determine the potential in the muffin-tin sphere due to charge in the sphere. In other words

$$V^{\rm MT}_{ij;lm}(r) = \frac{4\pi}{2l+1}\left(\frac{1}{r^{l+1}}\int_0^r\rho^{\rm MT}_{ij;lm}(r')r'^{l+2}dr' + r^l\int_r^{R_i}\frac{\rho^{\rm MT}_{ij;lm}(r')}{r'^{l-1}}dr'\right) + \frac{1}{Y_{00}}\frac{z_{ij}}{r}\delta_{l,0}$$

where the last term is the monopole arising from the point charge. All that remains is to add the homogenous solution of Poisson's equation,

$$V^{\rm H}_{ij}(\mathbf{r}) = \sum_{lm}V^{\rm H}_{ij;lm}\left(\frac{r}{R_i}\right)^l Y_{lm}(\hat{\mathbf{r}}),$$

to the muffin-tin potential so that it is continuous at the muffin-tin boundary. Therefore the coefficients, $\rho^{\rm H}_{ij;lm}$, are given by

$$V^{\rm H}_{ij;lm} = 4\pi i^l \sum_{\mathbf{G}} j_l(Gr) V^{\rm P}(\mathbf{G}) \exp(i\mathbf{G}\cdot\mathbf{r}_{ij}) Y^*_{lm}(\hat{\mathbf{G}}) - V^{\rm MT}_{ij;lm}(R_i).$$

Finally note that the **G**-vectors passed to the routine can represent vectors with a non-zero offset, $\mathbf{G}+\mathbf{p}$ say, which is required for calculating Coulomb matrix elements.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.286 getevalsv (Source File: getevalsv.f90)

INTERFACE:

```
Subroutine getevalsv (vpl, evalsvp)
```

USES:

```
Use modmain
Use modinput
Use modmpi
```

DESCRIPTION:

The file where the (second-variational) eigenvalues are stored is `EVALSV.OUT`. It is a direct-access binary file, the record length of which can be determined with the help of the array sizes and data type information. One record of this file has the following structure

$k_{\rm lat}$	$N_{\rm stsv}$	E

The following table explains the parts of the record in more detail

name	type	shape	description
$k_{\rm lat}$	real(8)	3	k-point in lattice coordinates
$N_{\rm stsv}$	integer	1	number of (second-variational) states (without core states)
E	real(8)	$N_{\rm stsv}$	(second-variational) eigenvalue array

REVISION HISTORY:

```
Documentation added, Dec 2009 (S. Sagmeister)
```

2.0.287 findsymlat (Source File: findsymlat.f90)

Subroutine findsymlat **USES:**

```
Use modinput
Use modmain
```

DESCRIPTION:

Finds the point group symmetries which leave the Bravais lattice invariant. Let A be the matrix consisting of the lattice vectors in columns, then

$$g = A^{\mathrm{T}} A$$

is the metric tensor. Any 3×3 matrix S with elements -1, 0 or 1 is a point group symmetry of the lattice if $\det(S)$ is -1 or 1, and

$$S^{\mathrm{T}} g S = g.$$

The first matrix in the set returned is the identity.
REVISION HISTORY:

```
Created January 2003 (JKD)
Removed arguments and simplified, April 2007 (JKD)
```

2.0.288 gencfun (Source File: gencfun.f90)

INTERFACE:

```
Subroutine gencfun
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Generates the smooth characteristic function. This is the function which is 0 within the muffin-tins and 1 in the intersitial region and is constructed from radial step function form-factors with $G < G_{\max}$. The form factors are given by

$$\tilde{\Theta}_i(G) = \begin{cases} \frac{4\pi R_i^3}{3\Omega} & G = 0 \\ \frac{4\pi R_i^3}{\Omega} \frac{j_1(GR_i)}{GR_i} & 0 < G \le G_{\max} \\ 0 & G > G_{\max} \end{cases},$$

where R_i is the muffin-tin radius of the ith species and Ω is the unit cell volume. Therefore the characteristic function in G-space is

$$\tilde{\Theta}(\mathbf{G}) = \delta_{G,0} - \sum_{ij} \exp(-i\mathbf{G} \cdot \mathbf{r}_{ij}) \tilde{\Theta}_i(G),$$

where $\mathbf{r}_{ij}$ is the position of the jth atom of the ith species.
REVISION HISTORY:

```
Created January 2003 (JKD)
```

2.0.289 create_plotlablels (Source File: mod_plotlabels.f90)

INTERFACE:

```
function create_plotlablels(title,filename,dimension) result (thisplotlabels)
```

DESCRIPTION:

This function creates a plotlabels type and initializes the axis descriptions
INPUT/OUTPUT PARAMETERS:

```
title     :  string for plot title (in, character(512))
dimension :  number of dimensions. e.g 1d plot has one dimension but 2 axes (in, integer)
create_plotlablels : pointer to plotlabels type  (out, type(plotlabels))
```

REVISION HISTORY:

```
Created April 2007 (JKD)
```

2.0.290 create_plotlablels (Source File: mod_plotlabels.f90)

INTERFACE:

```
subroutine set_plotlabel_axis(plotlabels_,axis,label,latexunit,graceunit)
```

DESCRIPTION:

This subroutine sets the axis labels for dinmension axis
INPUT/OUTPUT PARAMETERS:

```
plotlabels_  :  plotlabels type of which an axis schould be labeled (inout, type(plotlabel
axis :  number of the axis to set (in,integer)
label: (in,character*)
unit : (in,character*)
```

REVISION HISTORY:

```
Created April 2007 (JKD)
```

2.0.291 writeiad (Source File: writeiad.f90)

INTERFACE:

```
Subroutine writeiad (topt)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

topt : if .true. then the filename will be {\tt IADIST_OPT.OUT}, otherwise {\tt IADIST.OUT} (in,logical)

DESCRIPTION:

Outputs the interatomic distances to file.

REVISION HISTORY:

Created May 2005 (JKD)

2.0.292 writeeval (Source File: writeeval.f90)

INTERFACE:

Subroutine writeeval

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Outputs the second-variational eigenvalues and occupation numbers to the file EIGVAL.OUT.

REVISION HISTORY:

Created June 2003 (JKD)

2.0.293 rhoplot (Source File: rhoplot.f90)

INTERFACE:

Subroutine rhoplot

USES:

```
Use modinput
Use modmain
use modplotlabels
```

DESCRIPTION:

Outputs the charge density and the charge density gradients (modulus) read in from STATE.OUT, for 1D, 2D or 3D plotting.

REVISION HISTORY:

Created June 2003 (JKD)
Density gradients are added, March 2011 (DIN)

2.0.294 writelinen (Source File: writelinen.f90)

INTERFACE:

```
Subroutine writelinen
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Writes the linearisation energies for all APW and local-orbital functions to the file `LINENGY.OUT`.

REVISION HISTORY:

```
Created February 2004 (JKD)
```

2.0.295 fsmfield (Source File: fsmfield.f90)

INTERFACE:

```
Subroutine fsmfield
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Updates the effective magnetic field, $\mathbf{B}_{\rm FSM}$, required for fixing the spin moment to a given value, $\boldsymbol{\mu}_{\rm FSM}$. This is done by adding a vector to the field which is proportional to the difference between the moment calculated in the ith self-consistent loop and the required moment:

$$\mathbf{B}_{\rm FSM}^{i+1}=\mathbf{B}_{\rm FSM}^{i}+\lambda\left(\boldsymbol{\mu}^{i}-\boldsymbol{\mu}_{\rm FSM}\right),$$

where λ is a scaling factor.

REVISION HISTORY:

```
Created March 2005 (JKD)
```

2.0.296 plot2d (Source File: plot2d.f90)

INTERFACE:

```
Subroutine plot2d (labels, nf, lmax, ld, rfmt, rfir, plotdef)
```

USES:

```
      Use modinput
 use mod_muffin_tin
  use mod_atoms
  use mod_Gvector
      Use FoX_wxml
      use modmpi
  use modplotlabels
  use mod_plotting
```

INPUT/OUTPUT PARAMETERS:

```
fname : plot file name character(len=*)
nf    : number of functions (in,integer)
lmax : maximum angular momentum (in,integer)
ld    : leading dimension (in,integer)
rfmt : real muffin-tin function (in,real(ld,nrmtmax,natmtot,nf))
rfir : real intersitial function (in,real(ngrtot,nf))
plotdef:type(plot2d) defines plot region
```

DESCRIPTION:

Produces a 2D plot of the real functions contained in arrays `rfmt` and `rfir` on the parallelogram defined by the corner vertices in the global array `vclp2d`. See routine `rfarray`.

REVISION HISTORY:

```
Created June 2003 (JKD)
```

2.0.297 rhonorm (Source File: rhonorm.f90)

INTERFACE:

```
Subroutine rhonorm
```

USES:

```
      Use modmain
```

DESCRIPTION:

Loss of precision of the calculated total charge can result because the muffin-tin density is computed on a set of (θ, ϕ) points and then transformed to a spherical harmonic representation. This routine adds a constant to the density so that the total charge is correct. If the error in total charge exceeds a certain tolerance then a warning is issued.

REVISION HISTORY:

```
Created April 2003 (JKD)
Changed from rescaling to adding, September 2006 (JKD)
```

2.0.298 writegeom (Source File: writegeom.f90)

INTERFACE:

```
Subroutine writegeom (topt)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
topt : if .true. then the filename will be {\tt GEOMETRY_OPT.OUT}, otherwise
       {\tt GEOMETRY.OUT} (in,logical)
```

DESCRIPTION:

Outputs the lattice vectors and atomic positions to file, in a format which may be then used directly in `exciting.in`.

REVISION HISTORY:

```
Created January 2004 (JKD)
```

2.0.299 packeff (Source File: packeff.f90)

INTERFACE:

```
Subroutine packeff (tpack, n, nu)
```

USES:

```
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
tpack : .true. for packing, .false. for unpacking (in,logical)
n     : total number of real values stored (out,integer)
nu    : packed potential (inout,real(*))
```

DESCRIPTION:

Packs/unpacks the muffin-tin and interstitial parts of the effective potential and magnetic field into/from the single array `nu`. This array can then be passed directly to the mixing routine. See routine `rfpack`.

REVISION HISTORY:

```
Created June 2003 (JKD)
```

2.0.300 getoccsv (Source File: getoccsv.f90)

INTERFACE:

```
Subroutine getoccsv (vpl, occsvp)
```

USES:

```
Use modmain
Use modinput
Use modmpi
```

DESCRIPTION:

The file where the (second-variational) occupation numbers are stored is `OCCSV.OUT`. The maximum occupancies for spin-unpolarized systems is 2, whereas for spin-polarized systems it is 1. It is a direct-access binary file, the record length of which can be determined with the help of the array sizes and data type information. One record of this file has the following structure

k_{lat}	N_{stsv}	o

The following table explains the parts of the record in more detail

name	type	shape	description
k_{lat}	real(8)	3	k-point in lattice coordinates
N_{stsv}	integer	1	number of (second-variational) states (without core states)
o	real(8)	N_{stsv}	(second-variational) occupation number array

REVISION HISTORY:

```
Documentation added, Dec 2009 (S. Sagmeister)
```

2.0.301 writefermi (Source File: writefermi.f90)

INTERFACE:

```
Subroutine writefermi
```

USES:

```
Use modmain
```

DESCRIPTION:

Writes the Fermi energy to the file `EFERMI.OUT`.

REVISION HISTORY:

```
Created March 2005 (JKD)
```

2.0.302 potplot (Source File: potplot.f90)

INTERFACE:

```
Subroutine potplot
```

USES:

```
Use modinput
Use modmain
use modplotlabels
```

DESCRIPTION:

Outputs the exchange, correlation and Coulomb potentials, read in from `STATE.OUT`, for 1D, 2D or 3D plotting.

REVISION HISTORY:

```
Created June 2003 (JKD)
```

2.0.303 genapwfr (Source File: genapwfr.f90)

INTERFACE:

```
Subroutine genapwfr
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Generates the APW radial functions. This is done by integrating the scalar relativistic Schrödinger equation (or its energy deriatives) at the current linearisation energies using the spherical part of the effective potential. The number of radial functions at each l-value is given by the variable `apword` (at the muffin-tin boundary, the APW functions have continuous derivatives up to order `apword` $-$ 1). Within each l, these functions are orthonormalised with the Gram-Schmidt method. The radial Hamiltonian is applied to the orthonormalised functions and the results are stored in the global array `apwfr`.

REVISION HISTORY:

```
Created March 2003 (JKD)
```

2.0.304 writeefg (Source File: writeefg.f90)

INTERFACE:

```
Subroutine writeefg
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Computes the electric field gradient (EFG) tensor for each atom, α, and writes it to the file `EFG.OUT` along with its eigenvalues. The EFG is defined by

$$V_{ij}^{\alpha} \equiv \left.\frac{\partial^2 V_{\rm C}'(\mathbf{r})}{\partial \mathbf{r}_i \partial \mathbf{r}_j}\right|_{\mathbf{r}=\mathbf{r}_{\alpha}},$$

where $V_{\rm C}'$ is the Coulomb potential with the $l=m=0$ component removed in each muffin-tin. The derivatives are computed explicitly using the routine `gradrfmt`.

REVISION HISTORY:

```
Created May 2004 (JKD)
Fixed serious problem, November 2006 (JKD)
```

2.0.305 energy (Source File: energy.f90)

INTERFACE:

```
Subroutine energy
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Computes the total energy and its individual contributions. The kinetic energy is given by

$$T_{\rm s} = \sum_i n_i \epsilon_i - \int \rho(\mathbf{r})[v_{\rm C}(\mathbf{r}) + v_{\rm xc}(\mathbf{r})]d\mathbf{r} - \int \mathbf{m}(\mathbf{r}) \cdot (\mathbf{B}_{\rm xc}(\mathbf{r}) + \mathbf{B}_{\rm ext}(\mathbf{r}))d\mathbf{r},$$

where n_i are the occupancies and ϵ_i are the eigenvalues of both the core and valence states; ρ is the density; $\mathbf{m}$ is the magnetisation density; $v_{\rm C}$ is the Coulomb potential; $v_{\rm xc}$ and $\mathbf{B}_{\rm xc}$ are the exchange-correlation potential and effective magnetic field, respectively; and $\mathbf{B}_{\rm ext}$ is

the external magnetic field. The Hartree, electron-nuclear and nuclear-nuclear electrostatic energies are combined into the Coulomb energy:

$$\begin{aligned} E_{\rm C} &= E_{\rm H} + E_{\rm en} + E_{\rm nn} \\ &= \frac{1}{2}V_{\rm C} + E_{\rm Mad}, \end{aligned}$$

where

$$V_{\rm C} = \int \rho(\mathbf{r}) v_{\rm C}(\mathbf{r}) d\mathbf{r}$$

is the Coulomb potential energy. The Madelung energy is given by

$$E_{\rm Mad} = \frac{1}{2}\sum_{\alpha} z_{\alpha} R_{\alpha},$$

where

$$R_{\alpha} = \lim_{r\to 0}\left(v^{\rm C}_{\alpha;00}(r)Y_{00} + \frac{z_{\alpha}}{r}\right)$$

for atom α, with $v^{\rm C}_{\alpha;00}$ being the $l = 0$ component of the spherical harmonic expansion of $v_{\rm C}$ in the muffin-tin, and z_{α} is the nuclear charge. Using the nuclear-nuclear energy determined at the start of the calculation, the electron-nuclear and Hartree energies can be isolated with

$$E_{\rm en} = 2\left(E_{\rm Mad} - E_{\rm nn}\right)$$

and

$$E_{\rm H} = \frac{1}{2}(E_{\rm C} - E_{\rm en}).$$

Finally, the total energy is

$$E = T_{\rm s} + E_{\rm C} + E_{\rm xc},$$

where $E_{\rm xc}$ is obtained either by integrating the exchange-correlation energy density, or in the case of exact exchange, the explicit calculation of the Fock exchange integral. The energy from the external magnetic fields in the muffin-tins, `bfcmt`, is always removed from the total since these fields are non-physical: their field lines do not close. The energy of the physical external field, `bfieldc`, is also not included in the total because this field, like those in the muffin-tins, is used for breaking spin symmetry and taken to be infintesimal. If this field is intended to be finite, then the associated energy, `engybext`, should be added to the total by hand. See `potxc`, `exxengy` and related subroutines.

REVISION HISTORY:

```
Created May 2003 (JKD)
```

2.0.306 readfermi (Source File: readfermi.f90)

INTERFACE:

```
Subroutine readfermi
```

USES:

```
Use modmain
```

DESCRIPTION:

Reads the Fermi energy from the file `EFERMI.OUT`.

REVISION HISTORY:

```
Created March 2005 (JKD)
```

2.0.307 rtorat (Source File: modtetra.F90)

INTERFACE:

```
Subroutine rtorat (eps, n, x, k, div)
```

DESCRIPTION:

This subroutine factorizes the real coordinates of a vector $\mathbf{x}$. The output is an integer vector $\mathbf{k}$, such that

$$|x(i) - k(i)/\mathrm{div}| < \mathrm{eps}$$

for all $i = 1, \ldots, n$.

REVISION HISTORY:

```
Created July 2008 by Sagmeister
```

2.0.308 gentetlink (Source File: modtetra.F90)

INTERFACE:

```
Subroutine gentetlink (vpl, tqw, eps, bvec, ngridk, vkloff, nkpt, &
& nkptnr, vklnr, ikmapnr)
```

DESCRIPTION:

Generates an array connecting the tetrahedra of the $\mathbf{k}$-point with the ones of the $\mathbf{k}+\mathbf{q}$-point. Interface routine referencing the `libbzint` library of Ricardo Gomez-Abal and Xinzheng Li.

REVISION HISTORY:

```
Created January 2008 (Sagmeister)
```

2.0.309 rfmtctof (Source File: rfmtctof.f90)

INTERFACE:

```
Subroutine rfmtctof (rfmt)
```

INPUT/OUTPUT PARAMETERS:

```
  Use modinput
rfmt : real muffin-tin function (in,real(lmmaxvr,nrmtmax,natmtot))
```

DESCRIPTION:

Converts a real muffin-tin function from a coarse to a fine radial mesh by using cubic spline interpolation. See routines `rfinterp` and `spline`.

REVISION HISTORY:

```
Created October 2003 (JKD)
```

2.0.310 force (Source File: force.f90)

INTERFACE:

```
Subroutine force
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Computes the various contributions to the atomic forces. In principle, the force acting on a nucleus is simply the gradient at that site of the classical electrostatic potential from the other nuclei and the electronic density. This is a result of the Hellmann-Feynman theorem. However because the basis set is dependent on the nuclear coordinates and is not complete, the Hellman-Feynman force is inacurate and corrections to it are required. The first is the core correction which arises because the core wavefunctions were determined by neglecting the non-spherical parts of the effective potential v_s. Explicitly this is given by

$$\mathbf{F}^{\alpha}_{\rm core}=\int_{{\rm MT}_{\alpha}} v_{\rm s}(\mathbf{r})\nabla\rho^{\alpha}_{\rm core}(\mathbf{r})\,d\mathbf{r}$$

for atom α. The second, which is the incomplete basis set (IBS) correction, is due to the position dependence of the APW functions, and is derived by considering the change in total energy if the eigenvector coefficients were fixed and the APW functions themselves were changed. This would result in changes to the first-variational Hamiltonian and overlap matrices given by

$$\delta H^{\alpha}_{\mathbf{G},\mathbf{G}'}=i(\mathbf{G}-\mathbf{G}')\left(H^{\alpha}_{\mathbf{G}+\mathbf{k},\mathbf{G}'+\mathbf{k}}-\frac{1}{2}(\mathbf{G}+\mathbf{k})\cdot(\mathbf{G}'+\mathbf{k})\tilde{\Theta}_{\alpha}(\mathbf{G}-\mathbf{G}')e^{-i(\mathbf{G}-\mathbf{G}')\cdot\mathbf{r}_{\alpha}}\right)$$

$$\delta O^{\alpha}_{\mathbf{G},\mathbf{G}'}=i(\mathbf{G}-\mathbf{G}')\left(O^{\alpha}_{\mathbf{G}+\mathbf{k},\mathbf{G}'+\mathbf{k}}-\tilde{\Theta}_{\alpha}(\mathbf{G}-\mathbf{G}')e^{-i(\mathbf{G}-\mathbf{G}')\cdot\mathbf{r}_{\alpha}}\right)$$

where both $\mathbf{G}$ and $\mathbf{G}'$ run over the APW indices; $\tilde{\Theta}_\alpha$ is the form factor of the smooth step function for muffin-tin α; and H^α and O^α are the muffin-tin Hamiltonian and overlap matrices, respectively. The APW-local-orbital part is given by

$$\delta H^\alpha_{\mathbf{G},\mathbf{G}'} = i(\mathbf{G}+\mathbf{k})H^\alpha_{\mathbf{G}+\mathbf{k},\mathbf{G}'+\mathbf{k}}$$
$$\delta O^\alpha_{\mathbf{G},\mathbf{G}'} = i(\mathbf{G}+\mathbf{k})O^\alpha_{\mathbf{G}+\mathbf{k},\mathbf{G}'+\mathbf{k}}$$

where $\mathbf{G}$ runs over the APW indices and $\mathbf{G}'$ runs over the local-orbital indices. There is no contribution from the local-orbital-local-orbital part of the matrices. We can now write the IBS correction in terms of the basis of first-variational states as

$$\mathbf{F}^{\alpha\mathbf{k}}_{ij} = \sum_{\mathbf{G},\mathbf{G}'} b^{i\mathbf{k}*}_{\mathbf{G}} b^{j\mathbf{k}}_{\mathbf{G}'} \left(\delta H^\alpha_{\mathbf{G},\mathbf{G}'} - \epsilon_j \delta O^\alpha_{\mathbf{G},\mathbf{G}'}\right),$$

where $b^{i\mathbf{k}}$ is the first-variational eigenvector. Finally, the $\mathbf{F}^{\alpha\mathbf{k}}_{ij}$ matrix elements can be multiplied by the second-variational coefficients, and contracted over all indices to obtain the IBS force:

$$\mathbf{F}^\alpha_{\rm IBS} = \sum_{\mathbf{k}} w_{\mathbf{k}} \sum_{l\sigma} n_{l\mathbf{k}} \sum_{ij} c^{l\mathbf{k}*}_{\sigma i} c^{l\mathbf{k}}_{\sigma j} \mathbf{F}^{\alpha\mathbf{k}}_{ij} + \int_{{\rm MT}_\alpha} v_{\rm s}(\mathbf{r}) \nabla \left[\rho(\mathbf{r}) - \rho^\alpha_{\rm core}(\mathbf{r})\right] d\mathbf{r},$$

where $c^{l\mathbf{k}}$ are the second-variational coefficients, $w_{\mathbf{k}}$ are the k-point weights, $n_{l\mathbf{k}}$ are the occupancies, and $v_{\rm s}$ is the Kohn-Sham effective potential. See routines `hmlaa`, `olpaa`, `hmlalo`, `olpalo`, `energy`, `seceqn` and `gencfun`.

REVISION HISTORY:

```
Created January 2004 (JKD)
Fixed problem with second-variational forces, May 2008 (JKD)
```

2.0.311 forcek (Source File: forcek.f90)

INTERFACE:

```
Subroutine forcek (ik, ffacg)
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Computes the $\mathbf{k}$-dependent contribution to the incomplete basis set (IBS) force. See the calling routine `force` for a full description.

REVISION HISTORY:

```
Created June 2006 (JKD)
Updated for spin-spiral case, May 2007 (Francesco Cricchio and JKD)
```

2.0.312 rhoinit (Source File: rhoinit.f90)

INTERFACE:

```
Subroutine rhoinit
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Initialises the crystal charge density. Inside the muffin-tins it is set to the spherical atomic density. In the interstitial region it is taken to be constant such that the total charge is correct. Requires that the atomic densities have already been calculated.

REVISION HISTORY:

```
Created January 2003 (JKD)
```

2.0.313 genrmesh (Source File: genrmesh.f90)

INTERFACE:

```
Subroutine genrmesh
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Generates the coarse and fine radial meshes for each atomic species in the crystal. Also determines which points are in the inner part of the muffin-tin using the value of `fracinr`. For species i the radial mesh starts from the value $R_0 = r(1)$, hits the muffin-tin surface at $R_{\rm MT} = r(N)$, and ends at the effective infinity value $R_\infty = r(N_\infty)$. The number of points up to the effective infinity are determined by the number of points N up to the muffin-tin radius as well as by the smallest and largest mesh point and the muffin-tin radius, and is given by

$$N_\infty = \text{round}\left[\frac{(N-1)\ln(R_\infty/R_0)}{\ln(R_{\rm MT}/R_0)}\right] + 1.$$

The radial mesh points are finally defined by

$$r(j) = R_0\left(\frac{R_{\rm MT}}{R_0}\right)^{\frac{j-1}{N-1}},$$

for $j = 1, \ldots, N_\infty$.
Note: The number of mesh points initially defined in species file is adapted to be commensurate with the coarse mesh of step size `lradstep`

$$N = N^* - \mod (N^*, \texttt{lradstep}),$$

if N^* was the number of points defined in the species file. The number of mesh points $\tilde{N}$ of the coarse mesh $\tilde{r}(j)$ reads

$$\tilde{N} = \left\lfloor \frac{N-1}{\texttt{lradstep}} \right\rfloor + 1.$$

It is given by

$$\tilde{r}(j) = r([j-1] * \texttt{lradstep} + 1),$$

for $j = 1, \ldots, \tilde{N}$ and has the properties $\tilde{r}(1) = r(1) = R_0$ and $\tilde{r}(\tilde{N}) = r(N) = R_{\rm MT}$.
REVISION HISTORY:

```
Created September 2002 (JKD)
Revised and updated documentation, April 2011 (S. Sagmeister)
```

2.0.314 genpchgs (Source File: genpchgs.F90)

INTERFACE:

```
subroutine genpchgs(ik,evecfv,evecsv)
```

USES:

```
use modinput
use modmain
```

DESCRIPTION:

Generate partial charges for each state j, atom α and for each (lm) combination.
REVISION HISTORY:

```
Created 2010 (Sagmeister)
```

2.0.315 gensdmat (Source File: gensdmat.f90)

INTERFACE:

```
Subroutine gensdmat (evecsv, sdmat)
```

USES:

```
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
evecsv : second-variational eigenvectors (in,complex(nstsv,nstsv))
sdmat  : spin density matrices (out,complex(nspinor,nspinor,nstsv))
```

DESCRIPTION:

Computes the spin density matrices for a set of second-variational states.
REVISION HISTORY:

```
Created September 2008 (JKD)
```

2.0.316 mossbauer (Source File: mossbauer.f90)

INTERFACE:

```
Subroutine mossbauer
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Computes the contact charge density and contact magnetic hyperfine field for each atom and outputs the data to the file `MOSSBAUER.OUT`. The nuclear radius used for the contact quantities is approximated by the empirical formula $R_{\rm N} = 1.25Z^{1/3}$ fm, where Z is the atomic number.
REVISION HISTORY:

```
Created May 2004 (JKD)
```

2.0.317 seceqn (Source File: seceqn.f90)

Subroutine seceqn (ik, evalfv, evecfv, evecsv) **USES:**

```
Use modinput
Use modmain
Use modmpi
Use sclcontroll
Use diisinterfaces
```

INPUT/OUTPUT PARAMETERS:

```
ik     : k-point number (in,integer)
evalfv : first-variational eigenvalues (out,real(nstfv))
evecfv : first-variational eigenvectors (out,complex(nmatmax,nstfv))
evecsv : second-variational eigenvectors (out,complex(nstsv,nstsv))
```

DESCRIPTION:

Solves the first- and second-variational secular equations. See routines `match`, `seceqnfv`, `seceqnss` and `seceqnsv`.

REVISION HISTORY:

```
Created March 2004 (JKD)
```

2.0.318 symrfmt (Source File: symrfmt.f90)

INTERFACE:

```
Subroutine symrfmt (lrstp, is, rot, rfmt, srfmt)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
lrstp : radial step length (in,integer)
is    : species number (in,integer)
rot   : rotation matrix (in,real(3,3))
rfmt  : input muffin-tin function (in,real(lmmaxvr,nrmtmax))
srfmt : output muffin-tin function (out,real(lmmaxvr,nrmtmax))
```

DESCRIPTION:

Applies a symmetry operation (in the form of a rotation matrix) to a real muffin-tin function. The input function can also be the output function. See the routines `rtozflm` and `rotzflm`.

REVISION HISTORY:

```
Created May 2003 (JKD)
```

2.0.319 gencore (Source File: gencore.f90)

INTERFACE:

```
Subroutine gencore
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Computes the core radial wavefunctions, eigenvalues and densities. The radial Dirac equation is solved in the spherical part of the effective potential to which the atomic potential has been appended for $r > R^{\mathrm{MT}}$. In the case of spin-polarised calculations, the effective magnetic field is ignored.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.320 writeinfo (Source File: writeinfo.f90)

INTERFACE:

```
Subroutine writeinfo (fnum)
```

USES:

```
      Use modinput
      Use modmain
      use modmpi, only: procs
#ifdef TETRA
      Use modtetra
#endif
```

INPUT/OUTPUT PARAMETERS:

```
fnum : unit specifier for INFO.OUT file (in,integer)
```

DESCRIPTION:

Outputs basic information about the run to the file `INFO.OUT`. Does not close the file afterwards.

REVISION HISTORY:

```
Created January 2003 (JKD)
```

2.0.321 gndstate (Source File: gndstate.f90)

INTERFACE:

```
Subroutine gndstate
```

USES:

```
      Use modinput
      Use modmain
      Use modmpi
      Use scl_xml_out_Module
```

DESCRIPTION:

Computes the self-consistent Kohn-Sham ground-state. General information is written to the file `INFO.OUT`. First- and second-variational eigenvalues, eigenvectors and occupancies are written to the unformatted files `EVALFV.OUT`, `EVALSV.OUT`, `EVECFV.OUT`, `EVECSV.OUT` and `OCCSV.OUT`.

REVISION HISTORY:

```
Created October 2002 (JKD)
```

2.0.322 elfplot (Source File: elfplot.f90)

INTERFACE:

```
Subroutine elfplot
```

USES:

```
Use modinput
Use modmain
 use modplotlabels
```

DESCRIPTION:

Outputs the electron localisation function (ELF) for 1D, 2D or 3D plotting. The spin-averaged ELF is given by

$$f_{\rm ELF}(\mathbf{r}) = \frac{1}{1+[D(\mathbf{r})/D^0(\mathbf{r})]^2},$$

where

$$D(\mathbf{r}) = \frac{1}{2}\left(\tau(\mathbf{r}) - \frac{1}{4}\frac{[\nabla n(\mathbf{r})]^2}{n(\mathbf{r})}\right)$$

and

$$\tau(\mathbf{r}) = \sum_{i=1}^{N} |\nabla \Psi_i(\mathbf{r})|^2$$

is the spin-averaged kinetic energy density from the spinor wavefunctions. The function D^0 is the kinetic energy density for the homogeneous electron gas evaluated for $n(\mathbf{r})$:

$$D^0(\mathbf{r}) = \frac{3}{5}(6\pi^2)^{2/3}\left(\frac{n(\mathbf{r})}{2}\right)^{5/3}.$$

The ELF is useful for the topological classification of bonding. See for example T. Burnus, M. A. L. Marques and E. K. U. Gross [Phys. Rev. A 71, 10501 (2005)].

REVISION HISTORY:

```
Created September 2003 (JKD)
Fixed bug found by F. Wagner (JKD)
```

2.0.323 findsymcrys (Source File: findsymcrys.f90)

INTERFACE:

```
Subroutine findsymcrys
```

USES:

```
      Use modinput
      Use modmain
#ifdef XS
      Use modxs
#endif
```

DESCRIPTION:

Finds the complete set of symmetries which leave the crystal structure (including the magnetic fields) invariant. A crystal symmetry is of the form $\{\alpha_S|\alpha_R|\mathbf{t}\}$, where $\mathbf{t}$ is a translation vector, α_R is a spatial rotation operation and α_S is a global spin rotation. Note that the order of operations is important and defined to be from right to left, i.e. translation followed by spatial rotation followed by spin rotation. In the case of spin-orbit coupling $\alpha_S = \alpha_R$. In order to determine the translation vectors, the entire atomic basis is shifted so that the first atom in the smallest set of atoms of the same species is at the origin. Then all displacement vectors between atoms in this set are checked as possible symmetry translations. If the global variable `tshift` is set to `.false.` then the shift is not performed. See L. M. Sandratskii and P. G. Guletskii, *J. Phys. F: Met. Phys.* **16**, L43 (1986) and the routine `findsym`.

REVISION HISTORY:

```
Created April 2007 (JKD)
```

2.0.324 poteff (Source File: poteff.f90)

INTERFACE:

```
Subroutine poteff
```

USES:

```
      Use modmain
```

DESCRIPTION:

Computes the effective potential by adding together the Coulomb and exchange-correlation potentials. See routines `potcoul` and `potxc`.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.325 genshtmat (Source File: genshtmat.f90)

INTERFACE:

```
Subroutine genshtmat
```

USES:

```
      Use modinput
      Use modmain
#ifdef XS
      Use modxs
#endif
```

DESCRIPTION:

Generates the forward and backward spherical harmonic transformation (SHT) matrices using the spherical covering set produced by the routine `sphcover`. These matrices are used to transform a function between its (l,m)-expansion coefficients and its values at the (θ,ϕ) points on the sphere.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.326 writewiq2 (Source File: writewiq2.f90)

INTERFACE:

```
Subroutine writewiq2
```

USES:

```
      Use modmain
```

DESCRIPTION:

Outputs the integrals of $1/q^2$ in the small parallelepiped around each q-point to the file `WIQ2.OUT`. Note that the integrals are calculated after the q-point has been mapped to the first Brillouin zone. See routine genwiq2.

REVISION HISTORY:

```
Created June 2005 (JKD)
```

2.0.327 checkmt (Source File: checkmt.f90)

INTERFACE:

```
Subroutine checkmt
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Checks for overlapping muffin-tins. If any muffin-tins are found to intersect the program is terminated with error.

REVISION HISTORY:

```
Created May 2003 (JKD)
```

2.0.328 readstate (Source File: readstate.f90)

INTERFACE:

```
Subroutine readstate
```

USES:

```
      Use modinput
      Use modmain
#ifdef XS
      Use modxs, Only: isreadstate0
#endif
```

DESCRIPTION:

Reads in the charge density and other relevant variables from the file `STATE.OUT`. Checks for version and parameter compatibility.

REVISION HISTORY:

```
Created May 2003 (JKD)
```

2.0.329 allatoms (Source File: allatoms.f90)

INTERFACE:

```
Subroutine allatoms
```

USES:

```
      Use modinput
      Use modmain
```

DESCRIPTION:

Solves the Kohn-Sham-Dirac equations for each atom type in the solid and finds the self-consistent radial wavefunctions, eigenvalues, charge densities and potentials. The atomic densities can then be used to initialise the crystal densities, and the atomic self-consistent potentials can be appended to the muffin-tin potentials to solve for the core states. Note that, irrespective of the value of `xctype`, exchange-correlation functional type 3 is used. See also `atoms`, `rhoinit`, `gencore` and `modxcifc`.

REVISION HISTORY:

```
Created September 2002 (JKD)
Modified for GGA, June 2007 (JKD)
```

2.0.330 writegeometryxml (Source File: writegeometryxml.f90)

INTERFACE:

```
Subroutine writegeometryxml (topt)
```

USES:

```
Use modmain
Use modinput
Use modsp
Use modspdb
Use FoX_wxml
```

INPUT/OUTPUT PARAMETERS:

```
topt : if .true. then the filename will be {\ttgeometry_opt.xml}, otherwise
       {\tt geometry.xml} (in,logical)
```

DESCRIPTION:

Outputs the lattice vectors and atomic positions to file, in a format which may be then used directly in `exciting.in`.

REVISION HISTORY:

```
Created January 2004 (JKD)
```

2.0.331 nfftifc (Source File: nfftifc.f90)

INTERFACE:

```
Subroutine nfftifc (n)
```

INPUT/OUTPUT PARAMETERS:

```
n : required/avalable grid size (in,integer)
```

DESCRIPTION:

Interface to the grid requirements of the fast Fourier transform routine. Most routines restrict n to specific prime factorisations. This routine returns the next largest grid size allowed by the FFT routine.

REVISION HISTORY:

```
Created October 2002 (JKD)
```

2.0.332 genidxlo (Source File: genidxlo.f90)

INTERFACE:

```
Subroutine genidxlo
```

USES:

```
Use modmain
```

DESCRIPTION:

Generates an index array which maps the local-orbitals in each atom to their locations in the overlap or Hamiltonian matrices. Also finds the total number of local-orbitals.

REVISION HISTORY:

```
Created June 2003 (JKD)
```

2.0.333 zfmtinp (Source File: zfmtinp.f90)

INTERFACE:

```
Complex (8) Function zfmtinp (tsh, lmax, nr, r, ld, zfmt1, zfmt2)
```

INPUT/OUTPUT PARAMETERS:

```
tsh   : .true. if the functions are in spherical harmonics (in,logical)
lmax  : maximum angular momentum
nr    : number of radial mesh points (in,integer)
r     : radial mesh (in,real(nr))
ld    : leading dimension (in,integer)
zfmt1 : first complex muffin-tin function in spherical harmonics/
        coordinates (in,complex(ld,nr))
zfmt2 : second complex muffin-tin function in spherical harmonics/
        coordinates (in,complex(ld,nr))
```

DESCRIPTION:

Calculates the inner product of two complex fuctions in the muffin-tin. In other words, given two complex functions of the form

$$f(\mathbf{r}) = \sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} f_{lm}(r) Y_{lm}(\hat{\mathbf{r}}),$$

the function returns

$$I = \sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} \int f_{lm}^{1*}(r) f_{lm}^{2}(r) r^2 \, dr \; .$$

Note that if `tsh` is `.false.` the functions are in spherical coordinates rather than spherical harmonics. In this case I is multiplied by $4\pi/(l_{\max}+1)^2$.

REVISION HISTORY:

```
Created November 2003 (Sharma)
```

2.0.334 zfftifc (Source File: zfftifc.f90)

INTERFACE:

```
Subroutine zfftifc (nd, n, sgn, z)
```

INPUT/OUTPUT PARAMETERS:

```
nd   : number of dimensions (in,integer)
n    : grid sizes (in,integer(nd))
sgn  : FFT direction, -1: forward; 1: backward (in,integer)
z    : array to transform (inout,complex(n(1)*n(2)*...*n(nd)))
```

DESCRIPTION:

Interface to the double-precision complex fast Fourier transform routine. This is to allow machine-optimised routines to be used without affecting the rest of the code. See routine `nfftifc`.

REVISION HISTORY:

```
Created October 2002 (JKD)
```

2.0.335 potcoul (Source File: potcoul.f90)

INTERFACE:

```
Subroutine potcoul
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Calculates the Coulomb potential of the real charge density stored in the global variables `rhomt` and `rhoir` by solving Poisson's equation. These variables are converted to complex representations and passed to the routine `zpotcoul`.

REVISION HISTORY:

`Created April 2003 (JKD)`

2.0.336 init0 (Source File: init0.f90)

INTERFACE:

```
Subroutine init0
```

USES:

```
      Use modinput
      Use modmain
      Use modxcifc
#ifdef XS
      Use modxs
#endif
```

DESCRIPTION:

Performs basic consistency checks as well as allocating and initialising global variables not dependent on the k-point set.

REVISION HISTORY:

`Created January 2004 (JKD)`

2.0.337 portstate (Source File: portstate.F90)

INTERFACE:

```
Subroutine portstate (act)
```

USES:

```
      Use ioarray
      use mod_misc, only: refversion_gitstate
```

DESCRIPTION:

Toggle file format of `STATE.OUT`. If tb2a is true an ASCII file with the name `STATE.xml` is generated and the data from `STATE.OUT` is transferred. If tb2a is false the conversion goes in the other direction. Based upon the routines `readstate` and `writestate`.

REVISION HISTORY:

`Created 2007 (Sagmeister)`

2.0.338 genveffig (Source File: genveffig.f90)

INTERFACE:

```
Subroutine genveffig
```

USES:

```
Use modmain
```

DESCRIPTION:

Generates the Fourier transform of the effective potential in the intersitial region. The potential is first multiplied by the characteristic function which zeros it in the muffin-tins. See routine `gencfun`.

REVISION HISTORY:

```
Created January 2004 (JKD)
```

2.0.339 charge (Source File: charge.f90)

INTERFACE:

```
Subroutine charge
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Computes the muffin-tin, interstitial and total charges by integrating the density.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.340 genppts (Source File: genppts.f90)

INTERFACE:

```
Subroutine genppts (reducep, tfbz, ngridp, boxl, nppt, ipmap, ivp, vpl, &
& vpc, wppt)
```

USES:

```
      Use modinput
      Use modmain
#ifdef XS
      Use modxs
#endif
```

INPUT/OUTPUT PARAMETERS:

```
reducep : .true. if p-point set is to be reduced (in,logical)
tfbz    : .true. if vpl and vpc should be mapped to the first Brillouin
          zone (in,logical)
ngridp  : p-point grid size (in,integer(3))
boxl    : corners of box containing p-points in lattice coordinates, the
          first vector is the origin (in,real(3,4))
nppt    : total number of p-points (out,integer)
ipmap   : map from integer grid to p-point index
          (out,integer(0:ngridp(1)-1,0:ngridp(2)-1,0:ngridp(3)-1))
ivp     : integer coordinates of the p-points
          (out,integer(3,ngridp(1)*ngridp(2)*ngridp(3)))
vpl     : lattice coordinates of each p-point
          (out,real(3,ngridp(1)*ngridp(2)*ngridp(3)))
vpc     : Cartesian coordinates of each p-point
          (out,real(3,ngridp(1)*ngridp(2)*ngridp(3)))
wppt    : weights of each p-point (out,real(ngridp(1)*ngridp(2)*ngridp(3)))
```

DESCRIPTION:

This routine is used for generating k-point or q-point sets. Since these are stored in global arrays, the points passed to this and other routines are referred to as p-points. If `reducep` is `.true.` the set is reduced with the spatial part of the crystal symmetries. In lattice coordinates, the $\mathbf{p}$ vectors are given by

$$\mathbf{p}=\begin{pmatrix}\mathbf{B}_2-\mathbf{B}_1 & \mathbf{B}_3-\mathbf{B}_1 & \mathbf{B}_4-\mathbf{B}_1\end{pmatrix}\begin{pmatrix}i_1/n_1\\ i_2/n_2\\ i_3/n_3\end{pmatrix}+\mathbf{B}_1$$

where i_j runs from 0 to n_j-1, and the $\mathbf{B}$ vectors define the corners of a box with $\mathbf{B}_1$ as the origin. If `tfbz` is `.true.` then the vectors `vpl` (and `vpc`) are mapped to the first Brillouin zone. If `tfbz` is `.false.` and `reducep` is `.true.` then the coordinates of `vpl` are mapped to the $[0,1)$ interval. The p-point weights are stored in `wppt` and the array `ipmap` contains the map from the integer coordinates to the reduced index.

REVISION HISTORY:

```
Created August 2002 (JKD)
Updated April 2007 (JKD)
Modifications for excited states, November 2007 (Sagmeister)
Added mapping to the first Brillouin zone, September 2008 (JKD)
```

2.0.341 readinput (Source File: setdefault.f90)

INTERFACE:

```
Subroutine setdefault
```

USES:

```
      Use modinput
      Use modmain
#ifdef TETRA
      Use modtetra
#endif
#ifdef XS
      Use modmpi, Only: rank
      Use modxs
#endif
      Use sclcontroll
```

DESCRIPTION:

Sets default values for the input parameters.

REVISION HISTORY:

```
Created September 2002 (JKD)
Additional parmeters for excited states and tetrahedron method
  2004-2008 (Sagmeister)
```

2.0.342 seceqnfv (Source File: seceqnfv.f90)

INTERFACE:

```
Subroutine seceqnfv (nmatp, ngp, igpig, vgpc, apwalm, evalfv, evecfv)
```

USES:

```
      Use modinput
      Use modmain
      Use modfvsystem
```

INPUT/OUTPUT PARAMETERS:

```
nmatp  : order of overlap and Hamiltonian matrices (in,integer)
ngp    : number of G+k-vectors for augmented plane waves (in,integer)
igpig  : index from G+k-vectors to G-vectors (in,integer(ngkmax))
vgpc   : G+k-vectors in Cartesian coordinates (in,real(3,ngkmax))
apwalm : APW matching coefficients
         (in,complex(ngkmax,apwordmax,lmmaxapw,natmtot))
evalfv : first-variational eigenvalues (out,real(nstfv))
evecfv : first-variational eigenvectors (out,complex(nmatmax,nstfv))
```

DESCRIPTION:

Solves the secular equation,

$$(H - \epsilon O)b = 0,$$

for the all the first-variational states of the input k-point.

REVISION HISTORY:

Created March 2004 (JKD)

2.0.343 potxc (Source File: potxc.f90)

INTERFACE:

```
subroutine potxc
```

USES:

```
use modmain
use modxcifc
```

DESCRIPTION:

Computes the exchange-correlation potential and energy density. In the muffin-tin, the density is transformed from spherical harmonic coefficients ρ_{lm} to spherical coordinates (θ, ϕ) with a backward spherical harmonic transformation (SHT). Once calculated, the exchange-correlation potential and energy density are transformed with a forward SHT.

REVISION HISTORY:

Created April 2003 (JKD)

2.0.344 symrfir (Source File: symrfir.f90)

Subroutine symrfir (ngv, rfir) **USES:**

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
ngv  : number of G-vectors to be used for the Fourier space rotation
       (in,integer)
rfir : real intersitial function (inout,real(ngrtot))
```

DESCRIPTION:

Symmetrises a real scalar interstitial function. The function is first Fourier transformed to G-space, and then averaged over each symmetry by rotating the Fourier coefficients and multiplying them by a phase factor corresponding to the symmetry translation.

REVISION HISTORY:

Created July 2007 (JKD)

2.0.345 zfinp (Source File: zfinp.f90)

INTERFACE:

```
Complex (8) Function zfinp (tsh, zfmt1, zfmt2, zfir1, zfir2)
```

USES:

```
Use modmain
Use modinput
```

INPUT/OUTPUT PARAMETERS:

```
tsh   : .true. if the muffin-tin functions are in spherical harmonics
        (in,logical)
zfmt1 : first complex function in spherical harmonics/coordinates for all
        muffin-tins (in,complex(lmmaxvr,nrcmtmax,natmtot))
zfmt2 : second complex function in spherical harmonics/coordinates for all
        muffin-tins (in,complex(lmmaxvr,nrcmtmax,natmtot))
zfir1 : first complex interstitial function in real-space
        (in,complex(ngrtot))
zfir2 : second complex interstitial function in real-space
        (in,complex(ngrtot))
```

DESCRIPTION:

Calculates the inner product of two complex fuctions over the entire unit cell. The muffin-tin functions should be stored on the coarse radial grid and have angular momentum cut-off `lmaxvr`. In the intersitial region, the integrand is multiplied with the characteristic function, to remove the contribution from the muffin-tin. See routines `zfmtinp` and `gencfun`.

REVISION HISTORY:

```
Created July 2004 (Sharma)
```

2.0.346 gensfacgp (Source File: gensfacgp.f90)

INTERFACE:

```
Subroutine gensfacgp (ngp, vgpc, ld, sfacgp)
```

USES:

```
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
ngp    : number of G+p-vectors (in,integer)
vgpc   : G+p-vectors in Cartesian coordinates (in,real(3,*))
ld     : leading dimension (in,integer)
sfacgp : structure factors of G+p-vectors (out,complex(ld,natmtot))
```

DESCRIPTION:

Generates the atomic structure factors for a set of $\mathbf{G}+\mathbf{p}$-vectors:

$$S_\alpha(\mathbf{G}+\mathbf{p}) = \exp(i(\mathbf{G}+\mathbf{p})\cdot\mathbf{r}_\alpha),$$

where $\mathbf{r}_\alpha$ is the position of atom α.

REVISION HISTORY:

```
Created January 2003 (JKD)
```

2.0.347 zpotclmt (Source File: zpotclmt.f90)

INTERFACE:

```
Subroutine zpotclmt (ptnucl, lmax, nr, r, zn, ld, zrhomt, zvclmt)
```

INPUT/OUTPUT PARAMETERS:

```
ptnucl : .true. if the nucleus is a point particle (in,logical)
lmax   : maximum angular momentum (in,integer)
nr     : number of radial mesh points (in,integer)
r      : radial mesh (in,real(nr))
zn     : nuclear charge at the atomic center (in,real)
ld     : leading dimension (in,integer)
zrhomt : muffin-tin charge density (in,complex(ld,nr))
zvclmt : muffin-tin Coulomb potential (out,complex(ld,nr))
```

DESCRIPTION:

Solves the Poisson equation for the charge density contained in an isolated muffin-tin using the Green's function approach. In other words, the spherical harmonic expansion of the Coulomb potential, V_{lm}, is obtained from the density expansion, ρ_{lm}, by

$$V_{lm}(r) = \frac{4\pi}{2l+1}\left(\frac{1}{r^{l+1}}\int_0^r \rho_{lm}(r')r'^{l+2}dr' + r^l\int_r^R \frac{\rho_{lm}(r')}{r'^{l-1}}dr'\right) + \frac{1}{Y_{00}}\frac{z}{r}\delta_{l,0}$$

where the last term is the monopole arising from the point charge z, and R is the muffin-tin radius.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.348 wavefmt (Source File: wavefmt.f90)

INTERFACE:

```
Subroutine wavefmt (lrstp, lmax, is, ia, ngp, apwalm, evecfv, ld, wfmt)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
lrstp  : radial step length (in,integer)
lmax   : maximum angular momentum required (in,integer)
is     : species number (in,integer)
ia     : atom number (in,integer)
ngp    : number of G+p-vectors (in,integer)
apwalm : APW matching coefficients
         (in,complex(ngkmax,apwordmax,lmmaxapw,natmtot))
evecfv : first-variational eigenvector (in,complex(nmatmax))
ld     : leading dimension (in,integer)
wfmt   : muffin-tin wavefunction (out,complex(ld,*))
```

DESCRIPTION:

Calculates the first-variational wavefunction in the muffin-tin in terms of a spherical harmonic expansion. For atom α and a particular k-point $\mathbf{p}$, the r-dependent (l,m)-coefficients of the wavefunction for the ith state are given by

$$\Phi^{i\mathbf{p}}_{\alpha lm}(r)=\sum_{\mathbf{G}}b^{i\mathbf{p}}_{\mathbf{G}}\sum_{j=1}^{M^{\alpha}_{l}}A^{\alpha}_{jlm}(\mathbf{G}+\mathbf{p})u^{\alpha}_{jl}(r)+\sum_{j=1}^{N^{\alpha}}b^{i\mathbf{p}}_{(\alpha,j,m)}v^{\alpha}_{j}(r)\delta_{l,l_j},$$

where $b^{i\mathbf{p}}$ is the ith eigenvector returned from routine `seceqn`; $A^{\alpha}_{jlm}(\mathbf{G}+\mathbf{p})$ is the matching coefficient; M^{α}_{l} is the order of the APW; u^{α}_{jl} is the APW radial function; N^{α} is the number of local-orbitals; v^{α}_{j} is the jth local-orbital radial function; and (α,j,m) is a compound index for the location of the local-orbital in the eigenvector. See routines `genapwfr`, `genlofr`, `match` and `seceqn`.

REVISION HISTORY:

```
Created April 2003 (JKD)
Fixed description, October 2004 (C. Brouder)
Removed argument ist, November 2006 (JKD)
```

2.0.349 wavefmt_add (Source File: wavefmt.f90)

INTERFACE:

```
Subroutine wavefmt_add (nr, ld, wfmt, a, b, lrstp, fr)
```

INPUT/OUTPUT PARAMETERS:

```
nr     : number of radial mesh points (in,integer)
ld     : leading dimension (in,integer)
```

```
wfmt   : complex muffin-tin wavefunction passed in as a real array
         (inout,real(2*ld,*))
a      : real part of complex constant (in,real)
b      : imaginary part of complex constant (in,real)
lrstp  : radial step length (in,integer)
fr     : real radial function (in,real(lrstp,*))
```

DESCRIPTION:

Adds a real function times a complex constant to a complex muffin-tin wavefunction as efficiently as possible. See routine `wavefmt`.

REVISION HISTORY:

```
Created December 2006 (JKD)
```

2.0.350 linengy (Source File: linengy.f90)

INTERFACE:

```
Subroutine linengy
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Calculates the new linearisation energies for both the APW and local-orbital radial functions. See the routine `findband`.

REVISION HISTORY:

```
Created May 2003 (JKD)
```

2.0.351 plot1d (Source File: plot1d.f90)

INTERFACE:

```
Subroutine plot1d (labels, nf, lmax, ld, rfmt, rfir, plotdef)
```

USES:

```
Use modinput
use modmpi
Use FoX_wxml
use mod_muffin_tin
use mod_atoms
use mod_Gvector
use modplotlabels
use mod_plotting
```

INPUT/OUTPUT PARAMETERS:

```
lables : plot labels (character*)
nf     : number of functions (in,integer)
lmax   : maximum angular momentum (in,integer)
ld     : leading dimension (in,integer)
rfmt   : real muffin-tin function (in,real(ld,nrmtmax,natmtot,nf))
rfir   : real intersitial function (in,real(ngrtot,nf))
plotdef: type(plot1d) defining plotregion
```

DESCRIPTION:

Produces a 1D plot of the real functions contained in arrays `rfmt` and `rfir` along the lines connecting the vertices in the global array `vvlp1d`. See routine `rfarray`.

REVISION HISTORY:

```
Created June 2003 (JKD)
```

2.0.352 findprim (Source File: findprim.f90)

INTERFACE:

```
Subroutine findprim
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

This routine finds the smallest primitive cell which produces the same crystal structure as the conventional cell. This is done by searching through all the vectors which connect atomic positions and finding those which leave the crystal structure invariant. Of these, the three shortest which produce a non-zero unit cell volume are chosen.

REVISION HISTORY:

```
Created April 2007 (JKD)
```

2.0.353 gengpvec (Source File: gengpvec.f90)

INTERFACE:

```
Subroutine gengpvec (vpl, vpc, ngp, igpig, vgpl, vgpc, gpc, tpgpc)
```

USES:

```
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
vpl   : p-point vector in lattice coordinates (in,real(3))
vpc   : p-point vector in Cartesian coordinates (in,real(3))
ngp   : number of G+p-vectors returned (out,integer)
igpig : index from G+p-vectors to G-vectors (out,integer(ngkmax))
vgpl  : G+p-vectors in lattice coordinates (out,real(3,ngkmax))
vgpc  : G+p-vectors in Cartesian coordinates (out,real(3,ngkmax))
gpc   : length of G+p-vectors (out,real(ngkmax))
tpgpc : (theta, phi) coordinates of G+p-vectors (out,real(2,ngkmax))
```

DESCRIPTION:

Generates a set of $\mathbf{G}+\mathbf{p}$-vectors for the input p-point with length less than `gkmax`. These are used as the plane waves in the APW functions. Also computes the spherical coordinates of each vector.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.354 match (Source File: match.f90)

INTERFACE:

```
Subroutine match (ngp, gpc, tpgpc, sfacgp, apwalm)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
ngp    : number of G+p-vectors (in,integer)
gpc    : length of G+p-vectors (in,real(ngkmax))
tpgpc  : (theta, phi) coordinates of G+p-vectors (in,real(2,ngkmax))
sfacgp : structure factors of G+p-vectors (in,complex(ngkmax,natmtot))
apwalm : APW matching coefficients
         (out,complex(ngkmax,apwordmax,lmmaxapw,natmtot))
```

DESCRIPTION:

Computes the $(\mathbf{G}+\mathbf{p})$-dependent matching coefficients for the APW basis functions. Inside muffin-tin α, the APW functions are given by

$$\phi^{\alpha}_{\mathbf{G}+\mathbf{p}}(\mathbf{r}) = \sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} \sum_{j=1}^{M^{\alpha}_{l}} A^{\alpha}_{jlm}(\mathbf{G}+\mathbf{p}) u^{\alpha}_{jl}(r) Y_{lm}(\hat{\mathbf{r}}),$$

where $A^{\alpha}_{jlm}(\mathbf{G}+\mathbf{p})$ is the matching coefficient, M^{α}_{l} is the order of the APW and u^{α}_{jl} is the radial function. In the interstitial region, an APW function is a plane wave, $\exp(i(\mathbf{G}+\mathbf{p})\cdot$

$\mathbf{r})/\sqrt{\Omega}$, where Ω is the unit cell volume. Ensuring continuity up to the $(M_l^\alpha - 1)$th derivative across the muffin-tin boundary therefore requires that the matching coefficients satisfy

$$\sum_{j=1}^{M_l^\alpha} D_{ij} A_{jlm}^\alpha (\mathbf{G}+\mathbf{p}) = b_i \ ,$$

where

$$D_{ij} = \left. \frac{d^{i-1} u_{jl}^\alpha(r)}{dr^{i-1}} \right|_{r=R_\alpha}$$

and

$$b_i = \frac{4\pi i^l}{\sqrt{\Omega}} |\mathbf{G}+\mathbf{p}|^{i-1} j_l^{(i-1)}(|\mathbf{G}+\mathbf{p}|R_\alpha) \exp(i(\mathbf{G}+\mathbf{p})\cdot\mathbf{r}_\alpha) Y_{lm}^*(\widehat{\mathbf{G}+\mathbf{p}}),$$

with $\mathbf{r}_\alpha$ the atomic position and R_α the muffin-tin radius. See routine `wavefmt`.

REVISION HISTORY:

```
Created April 2003 (JKD)
Fixed documentation, June 2006 (JKD)
```

2.0.355 chgdist (Source File: chgdist.F90)

INTERFACE:

```
Subroutine chgdist
```

USES:

```
use modmain
```

DESCRIPTION:

Calculated the charge distance between two charge densities of the current and of the last iteration according to the expression

$$\Delta Q = \int_\Omega \mathrm{d}^3 r [\rho^{(n)}(\mathbf{r}) - \rho^{(n-1)}(\mathbf{r})].$$

Based on the routine `charge`.

REVISION HISTORY:

```
Created 2010 (Sagmeister)
```

2.0.356 rhovalk (Source File: rhovalk.f90)

INTERFACE:

```
Subroutine rhovalk (ik, evecfv, evecsv)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
ik     : k-point number (in,integer)
evecfv : first-variational eigenvectors (in,complex(nmatmax,nstfv,nspnfv))
evecsv : second-variational eigenvectors (in,complex(nstsv,nstsv))
```

DESCRIPTION:

Generates the partial valence charge density from the eigenvectors at k-point `ik`. In the muffin-tin region, the wavefunction is obtained in terms of its (l,m)-components from both the APW and local-orbital functions. Using a backward spherical harmonic transform (SHT), the wavefunction is converted to real-space and the density obtained from its modulus squared. This density is then transformed with a forward SHT and accumulated in the global variable `rhomt`. A similar proccess is used for the intersitial density in which the wavefunction in real-space is obtained from a Fourier transform of the sum of APW functions. The interstitial density is added to the global array `rhoir`. See routines `wavefmt`, `genshtmat` and `seceqn`.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.357 gridsize (Source File: gridsize.f90)

INTERFACE:

```
Subroutine gridsize
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Finds the **G**-vector grid which completely contains the vectors with $G < G_{\max}$ and is compatible with the FFT routine. The optimal sizes are given by

$$n_i = \frac{G_{\max}|\mathbf{a}_i|}{\pi} + 1,$$

where $\mathbf{a}_i$ is the ith lattice vector.

REVISION HISTORY:

```
Created July 2003 (JKD)
```

2.0.358 gengvec (Source File: gengvec.f90)

INTERFACE:

```
Subroutine gengvec
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Generates a set of **G**-vectors used for the Fourier transform of the charge density and potential and sorts them according to length. Integers corresponding to the vectors in lattice coordinates are stored, as well as the map from these integer coordinates to the **G**-vector index. A map from the **G**-vector set to the standard FFT array structure is also generated. Finally, the number of **G**-vectors with magnitude less than `gmaxvr` is determined.

REVISION HISTORY:

```
Created October 2002 (JKD)
Increased number of G-vectors to ngrtot, July 2007 (JKD)
```

2.0.359 atom (Source File: atom.f90)

INTERFACE:

```
Subroutine atom (ptnucl, zn, nst, n, l, k, occ, xctype, xcgrad, np, nr, &
& r, eval, rho, vr, rwf)
```

USES:

```
Use modxcifc
```

INPUT/OUTPUT PARAMETERS:

```
ptnucl : .true. if the nucleus is a point particle (in,logical)
zn     : nuclear charge (in,real)
nst    : number of states to solve for (in,integer)
n      : priciple quantum number of each state (in,integer(nst))
l      : quantum number l of each state (in,integer(nst))
k      : quantum number k (l or l+1) of each state (in,integer(nst))
occ    : occupancy of each state (inout,real(nst))
xctype : exchange-correlation type (in,integer)
xcgrad : 1 for GGA functional, 0 otherwise (in,integer)
np     : order of predictor-corrector polynomial (in,integer)
nr     : number of radial mesh points (in,integer)
r      : radial mesh (in,real(nr))
eval   : eigenvalue without rest-mass energy for each state (out,real(nst))
```

```
rho    : charge density (out,real(nr))
vr     : self-constistent potential (out,real(nr))
rwf    : major and minor components of radial wavefunctions for each state
         (out,real(nr,2,nst))
```

DESCRIPTION:

Solves the Dirac-Kohn-Sham equations for an atom using the exchange-correlation functional `xctype` and returns the self-consistent radial wavefunctions, eigenvalues, charge densities and potentials. The variable `np` defines the order of polynomial used for performing numerical integration. Requires the exchange-correlation interface routine `xcifc`.

REVISION HISTORY:

```
Created September 2002 (JKD)
Fixed s.c. convergence problem, October 2003 (JKD)
Added support for GGA functionals, June 2006 (JKD)
```

2.0.360 addrhocr (Source File: addrhocr.f90)

INTERFACE:

```
Subroutine addrhocr
```

USES:

```
Use modmain
```

DESCRIPTION:

Adds the core density to the muffin-tin and interstitial densities. A uniform background density is added in the interstitial region to take into account leakage of core charge from the muffin-tin spheres.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.361 plot3d (Source File: plot3d.f90)

INTERFACE:

```
Subroutine plot3d (plotlabels3d, nf, lmax, ld, rfmt, rfir, plotdef)
```

USES:

```
use modplotlabels
   Use modinput
   use mod_muffin_tin
   use mod_atoms
   use mod_Gvector
   Use FoX_wxml
   use modmpi
```

INPUT/OUTPUT PARAMETERS:

```
plotlabels : plot file number (in,integer)
nf   : number of functions (in,integer)
lmax : maximum angular momentum (in,integer)
ld   : leading dimension (in,integer)
rfmt : real muffin-tin function (in,real(ld,nrmtmax,natmtot,nf))
rfir : real intersitial function (in,real(ngrtot,nf))
```

DESCRIPTION:

Produces a 3D plot of the real functions contained in arrays `rfmt` and `rfir` in the parallelepiped defined by the corner vertices in the global array `vclp3d`. See routine `rfarray`.

REVISION HISTORY:

```
Created June 2003 (JKD)
Modified, October 2008 (F. Bultmark, F. Cricchio, L. Nordstrom)
Modified, February 2011 (D. Nabok)
```

2.0.362 symrvfir (Source File: symrvfir.f90)

Subroutine symrvfir (ngv, rvfir) **USES:**

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
ngv   : number of G-vectors to be used for the Fourier space rotation
        (in,integer)
rvfir : real interstitial vector function (inout,real(ngrtot,ndmag))
```

DESCRIPTION:

Symmetrises a real interstitial vector function. See routines `symrvf` and `symrfir` for details.

REVISION HISTORY:

```
Created July 2007 (JKD)
```

2.0.363 moment (Source File: moment.f90)

INTERFACE:

```
Subroutine moment
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Computes the muffin-tin, interstitial and total moments by integrating the magnetisation.

REVISION HISTORY:

```
Created January 2005 (JKD)
```

2.0.364 rfinp (Source File: rfinp.f90)

INTERFACE:

```
Function rfinp (lrstp, rfmt1, rfmt2, rfir1, rfir2)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
lrstp : radial step length (in,integer)
rfmt1 : first function in real spherical harmonics for all muffin-tins
        (in,real(lmmaxvr,nrmtmax,natmtot))
rfmt2 : second function in real spherical harmonics for all muffin-tins
        (in,real(lmmaxvr,nrmtmax,natmtot))
rfir1 : first real interstitial function in real-space (in,real(ngrtot))
rfir2 : second real interstitial function in real-space (in,real(ngrtot))
```

DESCRIPTION:

Calculates the inner product of two real fuctions over the entire unit cell. The input muffin-tin functions should have angular momentum cut-off `lmaxvr`. In the intersitial region, the integrand is multiplied with the characteristic function, $\tilde{\Theta}(\mathbf{r})$, to remove the contribution from the muffin-tin. See routines `rfmtinp` and `gencfun`.

REVISION HISTORY:

```
Created July 2004 (JKD)
```

2.0.365 reciplat (Source File: reciplat.f90)

INTERFACE:

```
Subroutine reciplat
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Generates the reciprocal lattice vectors from the real-space lattice vectors

$$\mathbf{b}_1 = \frac{2\pi}{s}(\mathbf{a}_2 \times \mathbf{a}_3)$$
$$\mathbf{b}_2 = \frac{2\pi}{s}(\mathbf{a}_3 \times \mathbf{a}_1)$$
$$\mathbf{b}_3 = \frac{2\pi}{s}(\mathbf{a}_1 \times \mathbf{a}_2)$$

and finds the unit cell volume $\Omega = |s|$, where $s = \mathbf{a}_1 \cdot (\mathbf{a}_2 \times \mathbf{a}_3)$.

REVISION HISTORY:

```
Created September 2002 (JKD)
```

2.0.366 findsym (Source File: findsym.f90)

INTERFACE:

```
Subroutine findsym (apl1, apl2, nsym, lspl, lspn, iea)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
apl1 : first set of atomic positions in lattice coordinates
       (in,real(3,maxatoms,maxspecies))
apl2 : second set of atomic positions in lattice coordinates
       (in,real(3,maxatoms,maxspecies))
nsym : number of symmetries (out,integer)
lspl : spatial rotation element in lattice point group for each symmetry
       (out,integer(48))
lspn : spin rotation element in lattice point group for each symmetry
       (out,integer(48))
iea  : equivalent atom index for each symmetry
       (out,integer(iea(natmmax,nspecies,48))
```

DESCRIPTION:

Finds the symmetries which rotate one set of atomic positions into another. Both sets of positions differ only by a translation vector and have the same muffin-tin magnetic fields (stored in the global array `bfcmt`). Any symmetry element consists of a spatial rotation of the atomic position vectors followed by a global magnetic rotation: $\{\alpha_S|\alpha_R\}$. In the case of spin-orbit coupling $\alpha_S = \alpha_R$. The symmetries are returned as indices of elements in the Bravais lattice point group. An index to equivalent atoms is stored in the array `iea`.
REVISION HISTORY:

```
Created April 2007 (JKD)
Fixed use of proper rotations for spin, February 2008 (L. Nordstrom)
```

2.0.367 symvect (Source File: symvect.f90)

INTERFACE:

```
Subroutine symvect (tspin, vc)
```

USES:

```
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
tspin : .true. if the global spin rotations should be used instead of the
        spatial rotations (in,logical)
vc    : vectors in Cartesian coordinates for all atoms (in,real(3,natmtot))
```

DESCRIPTION:

Symmetrises a 3-vector at each atomic site by rotating and averaging over equivalent atoms. Depending on `tspin`, either the spatial or spin part of the crystal symmetries are used.
REVISION HISTORY:

```
Created June 2004 (JKD)
Modified for spin rotations, May 2008 (JKD)
```

2.0.368 symrvf (Source File: symrvf.f90)

INTERFACE:

```
Subroutine symrvf (lrstp, rvfmt, rvfir)
```

USES:

```
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
lrstp : radial step length (in,integer)
rvfmt : real muffin-tin vector field
        (in,real(lmmaxvr,nrmtmax,natmtot,ndmag))
rvfir : real interstitial vector field
        (in,real(ngrtot,ndmag))
```

DESCRIPTION:

Symmetrises a vector field defined over the entire unit cell using the full set of crystal symmetries. If a particular symmetry involves rotating atom 1 into atom 2, then the spatial and spin rotations of that symmetry are applied to the vector field in atom 2 (expressed in spherical harmonic coefficients), which is then added to the field in atom 1. This is repeated for all symmetry operations. The fully symmetrised field in atom 1 is then rotated and copied to atom 2. Symmetrisation of the interstitial part of the field is performed by `symrvfir`. See also `symrfmt` and `findsym`.

REVISION HISTORY:

```
Created May 2007 (JKD)
Fixed problem with improper rotations, February 2008 (L. Nordstrom,
 F. Bultmark and F. Cricchio)
```

2.0.369 dos (Source File: dos.f90)

INTERFACE:

```
Subroutine dos
```

USES:

```
Use modinput
Use modmain
Use FoX_wxml
```

DESCRIPTION:

Produces a total and partial density of states (DOS) for plotting. The total DOS is written to the file `TDOS.OUT` while the partial DOS is written to the file `PDOS_Sss_Aaaaa.OUT` for atom `aaaa` of species `ss`. In the case of the partial DOS, each symmetrised (l,m)-projection is written consecutively and separated by blank lines. If the global variable `lmirep` is `.true.`, then the density matrix from which the (l,m)-projections are obtained is first rotated into a irreducible representation basis, i.e. one that block diagonalises all the site symmetry matrices in the Y_{lm} basis. Eigenvalues of a quasi-random matrix in the Y_{lm} basis which has been symmetrised with the site symmetries are written to `ELMIREP.OUT`. This allows for identification of the irreducible representations of the site symmetries, for example e_g or t_{2g}, by the degeneracies of the eigenvalues. In the plot, spin-up is made positive and spin-down negative. See the routines `gendmat` and `brzint`.

REVISION HISTORY:

```
Created January 2004 (JKD)
```

2.0.370 genlofr (Source File: genlofr.f90)

INTERFACE:

```
Subroutine genlofr
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Generates the local-orbital radial functions. This is done by integrating the scalar relativistic Schrödinger equation (or its energy deriatives) at the current linearisation energies using the spherical part of the effective potential. For each local-orbital, a linear combination of `lorbord` radial functions is constructed such that its radial derivatives up to order `lorbord`-1 are zero at the muffin-tin radius. This function is normalised and the radial Hamiltonian applied to it. The results are stored in the global array `lofr`.

REVISION HISTORY:

```
Created March 2003 (JKD)
```

2.0.371 vecplot (Source File: vecplot.f90)

INTERFACE:

```
Subroutine vecplot
```

DESCRIPTION:

Use modinput use modplotlabels Outputs a 2D or 3D vector field for plotting. The vector field can be the magnetisation vector field, $\mathbf{m}$; the exchange-correlation magnetic field, $\mathbf{B}_{\rm xc}$; or the electric field $\mathbf{E}\equiv-\nabla V_{\rm C}$. The magnetisation is obtained from the spin density matrix, $\rho_{\alpha\beta}$, by solving

$$\rho_{\alpha\beta}(\mathbf{r})=\frac{1}{2}\left(n(\mathbf{r})\delta_{\alpha\beta}+\sigma\cdot\mathbf{m}(\mathbf{r})\right),$$

where $n\equiv\mathrm{tr}\,\rho_{\alpha\beta}$ is the total density. In the case of 2D plots, the magnetisation vectors are still 3D, but are in the coordinate system of the plane.

REVISION HISTORY:

```
Created August 2004 (JKD)
Included electric field plots, August 2006 (JKD)
```

2.0.372 genwfsv (Source File: genwfsv.f90)

INTERFACE:

```
Subroutine genwfsv (tocc, ngp, igpig, evalsvp, apwalm, evecfv, evecsv, &
& wfmt, wfir)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
tocc    : .true. if only occupied wavefunctions are required (in,logical)
ngp     : number of G+p-vectors (in,integer)
igpig   : index from G+p-vectors to G-vectors (in,integer(ngkmax))
evalsvp : second-variational eigenvalue for every state (in,real(nstsv))
apwalm  : APW matching coefficients
          (in,complex(ngkmax,apwordmax,lmmaxapw,natmtot))
evecfv  : first-variational eigenvectors (in,complex(nmatmax,nstfv))
evecsv  : second-variational eigenvectors (in,complex(nstsv,nstsv))
wfmt    : muffin-tin part of the wavefunctions for every state in spherical
          coordinates (out,complex(lmmaxvr,nrcmtmax,natmtot,nspinor,nstsv))
wfir    : interstitial part of the wavefunctions for every state
          (out,complex(ngrtot,nspinor,nstsv))
```

DESCRIPTION:

Calculates the second-variational spinor wavefunctions in both the muffin-tin and interstitial regions for every state of a particular k-point. The wavefunctions in both regions are stored on a real-space grid. A coarse radial mesh is assumed in the muffin-tins with with angular momentum cut-off of `lmaxvr`. If `tocc` is `.true.`, then only the occupied states (those below the Fermi energy) are calculated.

REVISION HISTORY:

```
Created November 2004 (Sharma)
```

2.0.373 bandstr (Source File: bandstr.f90)

INTERFACE:

```
Subroutine bandstr
```

USES:

```
Use modinput
Use modmain
Use FoX_wxml
```

DESCRIPTION:

Produces a band structure along the path in reciprocal-space which connects the vertices in the array `vvlp1d`. The band structure is obtained from the second-variational eigenvalues and is written to the file `BAND.OUT` with the Fermi energy set to zero. If required, band structures are plotted to files `BAND_Sss_Aaaaa.OUT` for atom `aaaa` of species `ss`, which include the band characters for each l component of that atom in columns 4 onwards. Column 3 contains the sum over l of the characters. Vertex location lines are written to `BANDLINES.OUT`.

REVISION HISTORY:

```
Created June 2003 (JKD)
```

2.0.374 writesym (Source File: writesym.f90)

INTERFACE:

```
Subroutine writesym
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Outputs the Bravais, crystal and site symmetry matrices to files `SYMLAT.OUT`, `SYMCRYS.OUT` and `SYMSITE.OUT`, respectively. Also writes out equivalent atoms and related crystal symmetries to `EQATOMS.OUT`.

REVISION HISTORY:

```
Created October 2002 (JKD)
```

2.0.375 rvfcross (Source File: rvfcross.f90)

INTERFACE:

```
Subroutine rvfcross (rvfmt1, rvfmt2, rvfir1, rvfir2, rvfmt3, rvfir3)
```

USES:

```
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
rvfmt1 : first input muffin-tin field (in,real(lmmaxvr,nrmtmax,natmtot,3))
rvfmt2 : second input muffin-tin field (in,real(lmmaxvr,nrmtmax,natmtot,3))
rvfir1 : first input interstitial field (in,real(ngrtot,3))
rvfir2 : second input interstitial field (in,real(ngrtot,3))
rvfmt3 : output muffin-tin field (out,real(lmmaxvr,nrmtmax,natmtot,3))
rvfir3 : output interstitial field (out,real(ngrtot,3))
```

DESCRIPTION:

Given two real vector fields, $\mathbf{f}_1$ and $\mathbf{f}_2$, defined over the entire unit cell, this routine computes the local cross product

$$\mathbf{f}_3(\mathbf{r}) \equiv \mathbf{f}_1(\mathbf{r}) \times \mathbf{f}_2(\mathbf{r}).$$

REVISION HISTORY:

```
Created February 2007 (JKD)
```

2.0.376 writestate (Source File: writestate.f90)

INTERFACE:

```
Subroutine writestate
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Writes the charge density, potentials and other relevant variables to the file `STATE.OUT`. Note to developers: changes to the way the variables are written should be mirrored in `readstate`.

REVISION HISTORY:

```
Created May 2003 (JKD)
```

2.0.377 writepchgs (Source File: writepchgs.F90)

INTERFACE:

```
subroutine writepchgs(fnum,lmax)
```

USES:

```
use modinput
use modmain
```

DESCRIPTION:

Write partial charges to file.

REVISION HISTORY:

```
Created 2010 (Sagmeister)
```

2.0.378 rotzflm (Source File: rotzflm.f90)

INTERFACE:

```
Subroutine rotzflm (rot, lmax, n, ld, zflm1, zflm2)
```

INPUT/OUTPUT PARAMETERS:

```
rot   : rotation matrix (in,real(3,3))
lmax  : maximum angular momentum (in,integer)
n     : number of functions to rotate (in,integer)
ld    : leading dimension (in,integer)
zflm1 : coefficients of complex spherical harmonic expansion for each
        function (in,complex(ld,n))
zflm2 : coefficients of rotated functions (out,complex(ld,n))
```

DESCRIPTION:

Rotates a set of functions

$$f_i(\mathbf{r}) = \sum_{lm} f_{lm}^i Y_{lm}(\hat{\mathbf{r}})$$

for all i, given the coefficients f_{lm}^i and a rotation matrix R. This is done by first the computing the Euler angles (α, β, γ) of R^{-1} (see routine `euler`) and then generating the rotation matrix for spherical harmonics, $D_{mm'}^l(\alpha, \beta, \gamma)$, with which

$$Y_{lm}(\theta', \phi') = \sum_{m'} D_{mm'}^l(\alpha, \beta, \gamma) Y_{lm'}(\theta, \phi),$$

where (θ', ϕ') are the angles (θ, ϕ) rotated by R. The matrix D is given explicitly by

$$D_{mm'}^l(\alpha, \beta, \gamma) = \sum_i \frac{(-1)^i \sqrt{(l+m)!(l-m)!(l+m')!(l-m')!}}{(l-m'-i)!(l+m-i)!i!(i+m'-m)!} \times \left(\cos\frac{\beta}{2}\right)^{2l+m-m'-2i} \left(\sin\frac{\beta}{2}\right)^{2i+m'-m} e^{-i(m\alpha+m'\gamma)},$$

where the sum runs over all i which make the factorial arguments non-negative. For improper rotations, i.e. those which are a combination of a rotation and inversion, the rotation is first made proper with $R \to -R$ and D is modified with $D_{mm'}^l \to (-1)^l D_{mm'}^l$.

REVISION HISTORY:

```
Created April 2003 (JKD)
```

2.0.379 occupy (Source File: occupy.f90)

INTERFACE:

```
Subroutine occupy
```

USES:

```
      Use modinput
      Use modmain
#ifdef TETRAOCC_DOESNTWORK
      Use modtetra
#endif
```

DESCRIPTION:

Finds the Fermi energy and sets the occupation numbers for the second-variational states using the routine `fermi`.

REVISION HISTORY:

```
Created February 2004 (JKD)
Modifiactions for tetrahedron method, November 2007 (RGA alias
  Ricardo Gomez-Abal)
Modifications for tetrahedron method, 2007-2010 (Sagmeister)
```

2.0.380 genylmg (Source File: genylmg.f90)

INTERFACE:

```
Subroutine genylmg
```

USES:

```
      Use modinput
      Use modmain
```

DESCRIPTION:

Generates a set of spherical harmonics, $Y_{lm}(\hat{\mathbf{G}})$, with angular momenta up to `lmaxvr` for the set of $\mathbf{G}$-vectors.

REVISION HISTORY:

```
Created June 2003 (JKD)
```

2.0.381 getevalfv (Source File: getevalfv.f90)

INTERFACE:

```
Subroutine getevalfv (vpl, evalfv)
```

USES:

```
      Use modmain
      Use modinput
      Use modmpi
```

DESCRIPTION:

The file where the (first-variational) eigenvalues are stored is `EVALFV.OUT`. It is a direct-access binary file, the record length of which can be determined with the help of the array sizes and data type information. One record of this file has the following structure

k_{lat}	N_{stfv}	N_{spfv}	E

The following table explains the parts of the record in more detail

name	type	shape	description
k_{lat}	real(8)	3	k-point in lattice coordinates
N_{stfv}	integer	1	number of (first-variational) states (without core states)
N_{spfv}	integer	1	first-variational spins (always equals 1)
E	real(8)	$N_{\text{stfv}} \times N_{\text{spfv}}$	(first-variational) eigenvalue array

REVISION HISTORY:

```
Documentation added, Dec 2009 (S. Sagmeister)
```

2.0.382 rfmtinp (Source File: rfmtinp.f90)

INTERFACE:

```
Real (8) Function rfmtinp (lrstp, lmax, nr, r, ld, rfmt1, rfmt2)
```

INPUT/OUTPUT PARAMETERS:

```
lrstp : radial step length (in,integer)
lmax  : maximum angular momentum (in,integer)
nr    : number of radial mesh points (in,integer)
r     : radial mesh (in,real(nr))
ld    : the leading dimension (in,integer)
rfmt1 : first real function inside muffin-tin (in,real(ld,nr))
rfmt2 : second real function inside muffin-tin (in,real(ld,nr))
```

DESCRIPTION:

Calculates the inner product of two real fuctions in the muffin-tin. So given two real functions of the form

$$f(\mathbf{r}) = \sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} f_{lm}(r) R_{lm}(\hat{\mathbf{r}})$$

where R_{lm} are the real spherical harmonics, the function returns

$$I = \int \sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} f^1_{lm}(r) f^2_{lm}(r) r^2 dr \ .$$

The radial integral is performed using a high accuracy cubic spline method. See routines `genrlm` and `fderiv`.

REVISION HISTORY:

```
Created November 2003 (Sharma)
```

2.0.383 symrf (Source File: symrf.f90)

INTERFACE:

```
Subroutine symrf (lrstp, rfmt, rfir)
```

USES:

```
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
lrstp : radial step length (in,integer)
rfmt  : real muffin-tin function (inout,real(lmmaxvr,nrmtmax,natmtot))
rfir  : real intersitial function (inout,real(ngrtot))
```

DESCRIPTION:

Symmetrises a real scalar function defined over the entire unit cell using the full set of crystal symmetries. In the muffin-tin of a particular atom the spherical harmonic coefficients of every equivlent atom are rotated and averaged. The interstitial part of the function is first Fourier transformed to G-space, and then averaged over each symmetry by rotating the Fourier coefficients and multiplying them by a phase factor corresponding to the symmetry translation. See routines `symrfmt` and `symrfir`.

REVISION HISTORY:

```
Created May 2007 (JKD)
```

2.0.384 updatpos (Source File: updatpos.f90)

INTERFACE:

```
Subroutine updatpos
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Updates the current atomic positions according to the force on each atom. If $\mathbf{r}_{ij}^{m}$ is the position and $\mathbf{F}_{ij}^{m}$ is the force acting on it for atom j of species i and after time step m, then the new position is calculated by

$$\mathbf{r}_{ij}^{m+1}=\mathbf{r}_{ij}^{m}+\tau_{ij}^{m}\left(\mathbf{F}_{ij}^{m}+\mathbf{F}_{ij}^{m-1}\right),$$

where τ_{ij}^m is a parameter governing the size of the displacement. If $\mathbf{F}_{ij}^m \cdot \mathbf{F}_{ij}^{m-1} > 0$ then τ_{ij}^m is increased, otherwise it is decreased.

REVISION HISTORY:

```
Created June 2003 (JKD)
```

2.0.385 getevecsv (Source File: getevecsv.f90)

INTERFACE:

```
Subroutine getevecsv (vpl, evecsv)
```

USES:

```
Use modmain
Use modinput
Use modmpi
```

DESCRIPTION:

The file where the (second-variational) eigenvectors are stored is `EVECSV.OUT`. It is a direct-access binary file, the record length of which can be determined with the help of the array sizes and data type information. One record of this file has the following structure

k_{lat}	N_{stsv}	Φ

The following table explains the parts of the record in more detail

name	type	shape	description
k_{lat}	real(8)	3	k-point in lattice coordinates
N_{stsv}	integer	1	number of (second-variational) states (without core states)
Φ	complex(8)	$N_{\text{stsv}} \times N_{\text{stsv}}$	(second-variational) eigenvector array

REVISION HISTORY:

```
Documentation added, Dec 2009 (S. Sagmeister)
```

2.0.386 autoradmt (Source File: autoradmt.f90)

INTERFACE:

```
Subroutine autoradmt
```

USES:

```
Use modinput
Use modmain
```

DESCRIPTION:

Automatically determines the muffin-tin radii from the formula

$$R_i \propto 1 + \zeta |Z_i|^{1/3},$$

where Z_i is the atomic number of the ith species, ζ is a user-supplied constant (~ 0.625). The parameter ζ is stored in `rmtapm(1)` and the value which governs the distance between the muffin-tins is stored in `rmtapm(2)`. When `rmtapm(2)` $= 1$, the closest muffin-tins will touch.

REVISION HISTORY:

```
Created March 2005 (JKD)
Changed the formula, September 2006 (JKD)
```

2.0.387 ggair (Source File: ggair.f90)

INTERFACE:

```
Subroutine ggair (grhoir, gupir, gdnir, g2upir, g2dnir, g3rhoir, &
& g3upir, g3dnir)
```

INPUT/OUTPUT PARAMETERS:

```
  Use modinput
grhoir  : |grad rho| (out,real(ngrtot))
gupir   : |grad rhoup| (out,real(ngrtot))
gdnir   : |grad rhodn| (out,real(ngrtot))
g2upir  : grad^2 rhoup (out,real(ngrtot))
g2dnir  : grad^2 rhodn (out,real(ngrtot))
g3rhoir : (grad rho).(grad |grad rho|) (out,real(ngrtot))
g3upir  : (grad rhoup).(grad |grad rhoup|) (out,real(ngrtot))
g3dnir  : (grad rhodn).(grad |grad rhodn|) (out,real(ngrtot))
```

DESCRIPTION:

Computes $|\nabla\rho|$, $|\nabla\rho^{\uparrow}|$, $|\nabla\rho^{\downarrow}|$, $\nabla^2\rho^{\uparrow}$, $\nabla^2\rho^{\downarrow}$, $\nabla\rho\cdot(\nabla|\nabla\rho|)$, $\nabla\rho^{\uparrow}\cdot(\nabla|\nabla\rho^{\uparrow}|)$ and $\nabla\rho^{\downarrow}\cdot(\nabla|\nabla\rho^{\downarrow}|)$ for the interstitial charge density, as required by the generalised gradient approximation functional for spin-polarised densities. In the case of spin unpolarised calculations, $|\nabla\rho|$, $\nabla^2\rho$ and $\nabla\rho\cdot(\nabla|\nabla\rho|)$ are returned in the arrays `gupir`, `g2upir` and `g3upir`, respectively, while `grhoir`, `gdnir`, `g2dnir`, `g3rhoir` and `g3dnir` are not referenced. See routines `potxc` and `modxcifc`.

REVISION HISTORY:

```
Created October 2004 (JKD)
```

2.0.388 ggamt (Source File: ggamt.f90)

INTERFACE:

```
Subroutine ggamt (is, ia, grhomt, gupmt, gdnmt, g2upmt, g2dnmt, &
& g3rhomt, g3upmt, g3dnmt)
```

USES:

```
Use modinput
Use modmain
```

INPUT/OUTPUT PARAMETERS:

```
is      : species number (in,integer)
ia      : atom number (in,integer)
grhomt  : |grad rho| (out,real(lmmaxvr,nrmtmax))
gupmt   : |grad rhoup| (out,real(lmmaxvr,nrmtmax))
gdnmt   : |grad rhodn| (out,real(lmmaxvr,nrmtmax))
g2upmt  : grad^2 rhoup (out,real(lmmaxvr,nrmtmax))
g2dnmt  : grad^2 rhodn (out,real(lmmaxvr,nrmtmax))
g3rhomt : (grad rho).(grad |grad rho|) (out,real(lmmaxvr,nrmtmax))
g3upmt  : (grad rhoup).(grad |grad rhoup|) (out,real(lmmaxvr,nrmtmax))
g3dnmt  : (grad rhodn).(grad |grad rhodn|) (out,real(lmmaxvr,nrmtmax))
```

DESCRIPTION:

Computes $|\nabla\rho|$, $|\nabla\rho^{\uparrow}|$, $|\nabla\rho^{\downarrow}|$, $\nabla^2\rho^{\uparrow}$, $\nabla^2\rho^{\downarrow}$, $\nabla\rho\cdot(\nabla|\nabla\rho|)$, $\nabla\rho^{\uparrow}\cdot(\nabla|\nabla\rho^{\uparrow}|)$ and $\nabla\rho^{\downarrow}\cdot(\nabla|\nabla\rho^{\downarrow}|)$ for a muffin-tin charge density, as required by the generalised gradient approximation functional for spin-polarised densities. In the case of spin unpolarised calculations, $|\nabla\rho|$, $\nabla^2\rho$ and $\nabla\rho\cdot(\nabla|\nabla\rho|)$ are returned in the arrays `gupmt`, `g2upmt` and `g3upmt`, respectively, while `grhomt`, `gdnmt`, `g2dnmt`, `g3rhomt` and `g3dnmt` are not referenced. The input densities are in terms of real spherical harmonic expansions but the returned functions are in spherical coordinates. See routines `potxc`, `modxcifc`, `gradrfmt`, `genrlm` and `genshtmat`.

REVISION HISTORY:

```
Created April 2004 (JKD)
```

2.0.389 writekpts (Source File: writekpts.f90)

INTERFACE:

```
Subroutine writekpts
```

USES:

```
Use modmain
```

DESCRIPTION:

Writes the k-points in lattice coordinates, weights and number of $\mathbf{G}+\mathbf{k}$-vectors to the file `KPOINTS.OUT`.

REVISION HISTORY:

```
Created June 2003 (JKD)
```