The Elk Code Manual

Version 7.2.42

J. K. Dewhurst, S. Sharma
L. Nordström, F. Cricchio, O. Gråsäs
E. K. U. Gross

Contents

1 Introduction

Welcome to the Elk Code! Elk is an all-electron full-potential linearised augmented-plane-wave (FP-LAPW) code for determining the properties of crystalline solids. It was developed originally at the Karl-Franzens-Universität Graz as part of the EXCITING EU Research and Training Network project[1]. The guiding philosophy during the implementation of the code was to keep it as simple as possible for both users and developers without compromising on its capabilities. All the routines are released under either the GNU General Public License (GPL) or the GNU Lesser General Public License (LGPL) in the hope that they may inspire other scientists to implement new developments in the field of density functional theory and beyond.

2 Acknowledgments

Lots of people contributed to the Elk code with ideas, checking and testing, writing code or documentation and general encouragement. They include Claudia Ambrosch-Draxl, Clas Persson, Fredrik Bultmark, Christian Brouder, Rickard Armiento, Andrew Chizmeshya, Per Anderson, Igor Nekrasov, Sushil Auluck, Frank Wagner, Fateh Kalarasse, Jürgen Spitaler, Stefano Pittalis, Nektarios Lathiotakis, Tobias Burnus, Stephan Sagmeister, Christian Meisenbichler, Sébastien Lebègue, Yigang Zhang, Fritz Körmann, Alexey Baranov, Anton Kozhevnikov, Shigeru Suehara, Frank Essenberger, Antonio Sanna, Tyrel McQueen, Tim Baldsiefen, Marty Blaber, Anton Filanovich, Torbjörn Björkman, Martin Stankovski, Jerzy Goraus, Markus Meinert, Daniel Rohr, Vladimir Nazarov, Kevin Krieger, Pink Floyd, Arkardy Davydov, Florian Eich, Aldo Romero Castro, Koichi Kitahara, James Glasbrenner, Konrad Bussmann, Igor Mazin, Matthieu Verstraete, David Ernsting, Stephen Dugdale, Peter Elliott, Marcin Dulak, José A. Flores Livas, Stefaan Cottenier, Yasushi Shinohara, Michael Fechner, Yaroslav Kvashnin, Tristan Müller, Arsenii Gerasimov, Manh Duc Le, Jon Lafuente Bartolomé, René Wirnata, Jagdish Kumar, Andrew Shyichuk, Nisha Singh, Pietro Bonfa, Ronald Cohen and Alyn James. Special mention of David Singh's very useful book on the LAPW method[2] must also be made. Finally we would like to acknowledge the generous support of Karl-Franzens-Universität Graz, the EU Marie-Curie Research Training Networks initiative, the Max Born Institute and the Max Planck Society.

Kay Dewhurst
Sangeeta Sharma
Lars Nordström
Francesco Cricchio
Oscar Grånäs
Hardy Gross

Berlin, Halle, Jerusalem and Uppsala, June 2021

[1] EXCITING code developed under the Research and Training Network EXCITING funded by the EU, contract No. HPRN-CT-2002-00317

[2] D. J. Singh, *Planewaves, Pseudopotentials and the LAPW Method* (Kluwer Academic Publishers, Boston, 1994).

3 Units

Unless explicitly stated otherwise, Elk uses atomic units. In this system $\hbar = 1$, the electron mass $m = 1$, the Bohr radius $a_0 = 1$ and the electron charge $e = 1$ (note that the electron charge is positive, so that the atomic numbers Z are negative). Thus the atomic unit of length is 0.529177210903(80) Å, and the atomic unit of energy is the Hartree which equals 27.211386245988(53) eV. The unit of the external magnetic fields is defined such that one unit of magnetic field in `elk.in` equals 1715.255541 Tesla.

4 Compiling and running Elk

4.1 Compiling the code

Unpack the code from the archive file. Run the command

```
setup
```

in the `elk` directory and select the appropriate system and compiler. We highly recommend that you edit the file `make.inc` and tune the compiler options for your computer system. In particular, use of machine-optimised BLAS/LAPACK libraries can result in significant increase in performance, but make sure they are of version 3.x. Following this, run

```
make
```

This will hopefully compile the entire code and all the libraries into one executable, `elk`, located in the `elk/src` directory. It will also compile two useful auxiliary programs, namely `spacegroup` for producing crystal geometries from spacegroup data and `eos` for fitting equations of state to energy-volume data. If you want to compile everything all over again, then run `make clean` from the `elk` directory, followed by `make`.

4.1.1 Parallelism in Elk

Three forms of parallelism are implemented in Elk, and all can be used in combination with each other, with efficiency depending on the particular task, crystal structure and computer system. You may need to contact your system administrator for assistance with running Elk in parallel.

1. OpenMP works for symmetric multiprocessors, i.e. computers that have many cores with the same unified memory accessible to each. It is enabled by setting the appropriate command-line options (e.g. `-qopenmp` for the Intel compiler) before compiling, and also at runtime by the environment variable

   ```
   export OMP_NUM_THREADS=n
   ```

   where n is the number of cores available on a particular node. The same can be accomplished in `elk.in` with

```
maxthd
 n
```

In addition, some vendor-supplied BLAS/LAPACK libraries use OpenMP internally. The maximum number of threads used for LAPACK operations by Intel's MKL can be set with

```
maxthdmkl
 n
```

2. The message passing interface (MPI) is particularly suitable for running Elk across multiple nodes of a cluster, with scaling to hundreds of processors possible. To enable MPI, comment out the lines indicated in `elk/make.inc`. Then run `make clean` followed by `make`. If y is the number of nodes and x is the number of cores per node, then at runtime envoke

```
mpirun -np z ./elk
```

where $z = xy$ is the total number of cores available on the machine. Highest efficiency is obtained by using hybrid parallelism with OpenMP on each node and MPI across nodes. This can be done by compiling the code using the MPI Fortran compiler in combination with the OpenMP command-line option. At runtime set `export OMP_NUM_THREADS=x` and start the MPI run with *one process per node* as follows

```
mpirun -pernode -np y ./elk
```

The number of MPI processes is reported in the file `INFO.OUT` which serves as a check that MPI is running correctly. Note that version 2 of the MPI libraries is required to run Elk.

3. Phonon calculations use a simple form of parallelism by just examining the run directory for dynamical matrix files. These files are of the form

```
DYN_Qqqqq_qqqq_qqqq_Sss_Aaa_Pp.OUT
```

and contain a single row of a particular dynamical matrix. Elk simply finds which `DYN` files do not exist, chooses one and runs it. This way many independent runs of Elk can be started in the same directory on a networked file system (NFS), and will run until all the dynamical matrices files are completed. Should a particular run crash, then delete the associated empty `DYN` file and rerun Elk.

4.2 Memory requirements

Elk is a memory-bound code and runs best on processors with large caches and a large number of memory channels per core. Some tasks in Elk require a considerable amount of memory which can exceed the physical memory of the computer. In such cases, the number of threads at the first nesting level can be reduced with (for example)

```
maxthd1
 -4
```

which restricts the number of threads at the first nesting level to maxthd/4. Deeper nesting levels, which generally require less memory, will still utilise the full compliment of available threads.

4.2.1 Stack space

The latest versions of Elk use stack space aggressively. This is because accessing variables is faster on the stack than on the heap. This can, however, result in the code crashing as threads run out of their stack space. To avoid this, increase the stack size for each OpenMP thread with (for example)

```
export OMP_STACKSIZE=64M
```

before running the code.

4.3 Linking with the Libxc functional library

Libxc is the ETSF library of exchange-correlation functionals. Elk can use the complete set of LDA and GGA functionals available in Libxc as well as the potential-only metaGGA's. In order to enable this, first download and compile Libxc version 5. This should have produced the files `libxc.a` and `libxcf90.a`. Copy these files to the `elk/src` directory and then uncomment the lines indicated for Libxc in the file `elk/make.inc`. Once this is done, run `make clean` followed by `make`. To select a particular functional of Libxc, use the block

```
xctype
 100 nx nc
```

where `nx` and `nc` are, respectively, the numbers of the exchange and correlation functionals in the Libxc library. See the file `elk/src/libxcf90.f90` for a list of the functionals and their associated numbers.

4.4 Running the code

As a rule, all input files for the code are in lower case and end with the extension `.in`. All output files are uppercase and have the extension `.OUT`. For most cases, the user will only need to modify the file `elk.in`. In this file input parameters are arranged in blocks. Each block consists of a block name on one line and the block variables on subsequent lines. Almost all blocks are optional: the code uses reasonable default values in cases where they are absent. Blocks can appear in any order, if a block is repeated then the second instance is used. Comment lines can be included in the input file and begin with the `!` character.

4.4.1 Species files

The only other input files are those describing the atomic species which go into the crystal. These files are found in the `species` directory and are named with the element symbol and the extension `.in`, for example `Sb.in`. They contain parameters like the atomic charge, mass, muffin-tin radius, occupied atomic states and the type of linearisation required. Here as an example is the copper species file `Cu.in`:

```
 'Cu'                                          : spsymb
 'copper'                                      : spname
-29.0000                                       : spzn
 115837.2716                                   : spmass
 0.371391E-06    2.0000   34.8965   500       : rminsp, rmt, rmaxsp, nrmt
 10                                            : nstsp
 1   0   1   2.00000    T                     : nsp, lsp, ksp, occsp, spcore
 2   0   1   2.00000    T
 2   1   1   2.00000    T
 2   1   2   4.00000    T
 3   0   1   2.00000    T
 3   1   1   2.00000    F
 3   1   2   4.00000    F
 3   2   2   4.00000    F
 3   2   3   6.00000    F
 4   0   1   1.00000    F
 1                                             : apword
 0.1500   0  F                                 : apwe0, apwdm, apwve
 1                                             : nlx
 2   2                                         : lx, apword
 0.1500   0  T                                 : apwe0, apwdm, apwve
 0.1500   1  T
 4                                             : nlorb
 0   2                                         : lorbl, lorbord
 0.1500   0  F                                 : lorbe0, lorbdm, lorbve
 0.1500   1  F
 1   2
 0.1500   0  F
 0.1500   1  F
 2   2
 0.1500   0  F
 0.1500   1  F
 1   3
 0.1500   0  F
 0.1500   1  F
-2.8652   0  T
```

The input parameters are defined as follows:

`spsymb`
The symbol of the element.

`spname`
The name of the element.

`spzn`
Nuclear charge: should be negative since the electron charge is taken to be postive in the code; it can also be fractional for purposes of doping.

`spmass`
Nuclear mass in atomic units.

`rminsp`, `rmt`, `rmaxsp`, `nrmt`
Respectively, the minimum radius on logarithmic radial mesh; muffin-tin radius; effective infinity for atomic radial mesh; and number of radial mesh points to muffin-tin radius.

`nstsp`
Number of atomic states.

`nsp`, `lsp`, `ksp`, `occsp`, `spcore`
Respectively, the principal quantum number of the radial Dirac equation; quantum number l; quantum number k (l or $l+1$); occupancy of atomic state (can be fractional); `.T.` if state is in the core and therefore treated with the Dirac equation in the spherical part of the muffin-tin Kohn-Sham potential.

`apword`
Default APW function order, i.e. the number of radial functions and therefore the order of the radial derivative matching at the muffin-tin surface.

`apwe0`, `apwdm`, `apwve`
Respectively, the default APW linearisation energy; the order of the energy derivative of the APW radial function $\partial^m u(r)/\partial E^m$; and `.T.` if the linearisation energy is allowed to vary.

`nlx`
The number of exceptions to the default APW configuration. These should be listed on subsequent lines for particular angular momenta. In this example, the fixed energy APW with angular momentum d (`lx` $= 2$) is replaced with a LAPW, which has variable linearisation energy.

`nlorb`
Number of local-orbitals.

`lorbl`, `lorbord`
Respectively, the angular momentum l of the local-orbital; and the order of the radial derivative which goes to zero at the muffin-tin surface.

`lorbe0`, `lorbdm`, `lorbve`
Respectively, the default local-orbital linearisation energy; the order of the energy derivative of the local-orbital radial function; and `.T.` if the linearisation energy is allowed to vary.

4.4.2 Examples

The best way to learn to use Elk is to run the examples included with the package. These can be found in the `examples` directory and use many of the code's capabilities. The following section which describes all the input parameters will be of invaluable assistance.

5 Input blocks

This section lists all the input blocks available. It is arranged with the name of the block followed by a table which lists each parameter name, what the parameter does, its type and default value. A horizontal line in the table indicates a new line in `elk.in`. Below the table is a brief overview of the block's function.

5.1 atoms

`nspecies`	number of species	integer	0
`spfname(i)`	species filename for species i	string	-
`natoms(i)`	number of atoms for species i	integer	-
`atposl(j,i)`	atomic position in lattice coordinates for atom j	real(3)	-
`bfcmt(j,i)`	muffin-tin external magnetic field in Cartesian coordinates for atom j	real(3)	-

Defines the atomic species as well as their positions in the unit cell and the external magnetic field applied throughout the muffin-tin. These fields are used to break spin symmetry and should be considered infinitesimal as they do not contribute directly to the total energy. Collinear calculations are more efficient if the field is applied in the z-direction. One could, for example, set up an antiferromagnetic crystal by pointing the field on one atom in the positive z-direction and in the opposite direction on another atom. If `molecule` is `.true.` then the atomic positions are assumed to be in Cartesian coordinates. See also `sppath`, `bfieldc` and `molecule`.

5.2 autokpt

`autokpt`	`.true.` if the k-point set is to be determined automatically	logical	`.false.`

See `radkpt` for details.

5.3 autolinengy

`autolinengy`	`.true.` if the fixed linearisation energies are to be determined automatically	logical	`.false.`

See `dlefe` for details.

5.4 autoswidth

`autoswidth`	`.true.` if the smearing parameter `swidth` should be determined automatically	logical	`.false.`

Calculates the smearing width from the k-point density, V_{BZ}/n_k; the valence band width, W; and an effective mass parameter, m^*; according to

$$\sigma = \frac{\sqrt{2W}}{m^*}\left(\frac{3}{4\pi}\frac{V_{\mathrm{BZ}}}{n_k}\right)^{1/3}.$$

The variable `mstar` then replaces `swidth` as the control parameter of the smearing width. A large value of m^* gives a narrower smearing function. Since `swidth` is adjusted according to the fineness of the **k**-mesh, the smearing parameter can then be eliminated. It is

not recommended that `autoswidth` be used in conjunction with the Fermi-Dirac smearing function, since the electronic temperature will then be a function of the k-point mesh. See T. Björkman and O. Gränäs, *Int. J. Quant. Chem.* DOI: 10.1002/qua.22476 (2010) for details. See also `stype` and `swidth`.

5.5 avec

`avec(1)`	first lattice vector	real(3)	$(1.0, 0.0, 0.0)$
`avec(2)`	second lattice vector	real(3)	$(0.0, 1.0, 0.0)$
`avec(3)`	third lattice vector	real(3)	$(0.0, 0.0, 1.0)$

Lattice vectors of the crystal in atomic units (Bohr).

5.6 beta0

`beta0`	adaptive mixing parameter	real	0.05

This determines how much of the potential from the previous self-consistent loop is mixed with the potential from the current loop. It should be made smaller if the calculation is unstable. See `betamax` and also the routine `mixadapt`.

5.7 betamax

`betamax`	maximum adaptive mixing parameter	real	0.5

Maximum allowed mixing parameter used in routine `mixadapt`.

5.8 bfieldc

`bfieldc`	global external magnetic field in Cartesian coordinates	real(3)	$(0.0, 0.0, 0.0)$

This is a constant magnetic field applied throughout the entire unit cell and enters the second-variational Hamiltonian as

$$\frac{g_e}{4c}\,\vec{\sigma}\cdot\mathbf{B}_{\text{ext}},$$

where g_e is the electron g-factor. This field is normally used to break spin symmetry for spin-polarised calculations and considered to be infinitesimal with no direct contribution to the total energy. In cases where the magnetic field is finite (for example when computing magnetic response) the external $\mathbf{B}$-field energy reported in `INFO.OUT` should be added to the total by hand. This field is applied throughout the entire unit cell. To apply magnetic fields in particular muffin-tins use the `bfcmt` vectors in the `atoms` block. Collinear calculations are more efficient if the field is applied in the z-direction.

5.9 broydpm

`broydpm`	Broyden mixing parameters α and w_0	real	$(0.4, 0.15)$

See `mixtype` and `mixsdb`.

5.10 c_tb09

`c_tb09`	Tran-Blaha constant c	real	-

Sets the constant c in the Tran-Blaha '09 functional. Normally this is calculated from the density, but there may be situations where this needs to be adjusted by hand. See *Phys. Rev. Lett.* **102**, 226401 (2009).

5.11 chgexs

`chgexs`	excess electronic charge	real	0.0

This controls the amount of charge in the unit cell beyond that required to maintain neutrality. It can be set positive or negative depending on whether electron or hole doping is required.

5.12 cmagz

`cmagz`	.true. if z-axis collinear magnetism is to be enforced	logical	`.false.`

This variable can be set to .true. in cases where the magnetism is predominantly collinear in the z-direction, for example a ferromagnet with spin-orbit coupling. This will make the calculation considerably faster at the slight expense of precision.

5.13 deltaem

`deltaem`	the size of the **k**-vector displacement used when calculating numerical derivatives for the effective mass tensor	real	0.025

See `ndspem` and `vklem`.

5.14 deltaph

`deltaph`	size of the atomic displacement used for calculating dynamical matrices	real	0.01

Phonon calculations are performed by constructing a supercell corresponding to a particular **q**-vector and making a small periodic displacement of the atoms. The magnitude of this displacement is given by `deltaph`. This should not be made too large, as anharmonic terms could then become significant, neither should it be too small as this can introduce numerical error.

5.15 deltast

`deltast`	size of the change in lattice vectors used for calculating the stress tensor	real	0.005

The stress tensor is computed by changing the lattice vector matrix A by

$$A \to (1 + \delta t\, e_i)A,$$

where dt is an infinitesimal equal in practice to `deltast` and e_i is the i^{th} strain tensor. Numerical finite differences are used to compute the stress tensor as the derivative of the total energy dE_i/dt.

5.16 dft+u

`dftu`	type of DFT+U calculation	integer	0
`inpdftu`	type of input for DFT+U calculation	integer	1
`is`	species number	integer	-
`l`	angular momentum value	integer	-1
`u`	the desired U value	real	0.0
`j`	the desired J value	real	0.0

This block contains the parameters required for an DFT+U calculation, with the list of parameters for each species terminated with a blank line. The type of double counting required is set with the parameter `dftu`. Currently implemented are:

0 No DFT+U calculation
1 Fully localised limit (FLL)
2 Around mean field (AFM)
3 An interpolation between FLL and AFM

The type of input parameters is set with the parameter `inpdftu`. The current possibilities are:

1 U and J
2 Slater parameters
3 Racah parameters
4 Yukawa screening length
5 U and determination of corresponding Yukawa screening length

See (amongst others) *Phys. Rev. B* **67**, 153106 (2003), *Phys. Rev. B* **52**, R5467 (1995), *Phys. Rev. B* **60**, 10763 (1999), and *Phys. Rev. B* **80**, 035121 (2009).

5.17 dlefe

`dlefe`	difference between the fixed linearisation energy and the Fermi energy	real	-0.1

When `autolinengy` is `.true.` then the fixed linearisation energies are set to the Fermi energy plus `dlefe`.

5.18 dncgga

`dncgga`	small constant used to stabilise non-collinear GGA	real	1×10^{-8}

This small constant, d, is required in order to remove the infinite gradients obtained when using 'Kubler's trick' in conjunction with GGA and non-collinear magnetism. It is applied by calculating the up and down densities as

$$\rho^{\uparrow}(\mathbf{r}) = \rho(\mathbf{r}) + \widetilde{m}(\mathbf{r}) \qquad \rho^{\downarrow}(\mathbf{r}) = \rho(\mathbf{r}) - \widetilde{m}(\mathbf{r}),$$

where $\widetilde{m}(\mathbf{r}) = \sqrt{\mathbf{m}^2(\mathbf{r}) + d}$, and should be taken as the smallest value for which the exchange-correlation magnetic field $\mathbf{B}_{\rm xc}$ is smooth.

5.19 dosmsum

`dosmsum`	`.true.` if the partial DOS is to be summed over m	logical	`.false.`

By default, the partial density of states is resolved over (l, m) quantum numbers. If `dosmsum` is set to `.true.` then the partial DOS is summed over m, and thus depends only on l.

5.20 dosssum

`dosssum`	`.true.` if the partial DOS is to be summed over spin	logical	`.false.`

By default, the partial density of states for spin-polarised systems is spin resolved.

5.21 dtimes

`dtimes`	time step used in time evolution run	real	0.1

See also `tstime`.

5.22 epsband

`epsband`	convergence tolerance for determining band energies	real	1×10^{-12}

APW and local-orbital linearisation energies are determined from the band energies. 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_{t}. A downward search is now performed from E_{t} until the slope of the radial wavefunction at the muffin-tin radius is zero. This energy, E_{b}, is at the bottom of the band. The band energy is taken as $(E_{\mathrm{t}} + E_{\mathrm{b}})/2$. If either E_{t} or E_{b} is not found, then the band energy is set to the default value.

5.23 epschg

`epschg`	maximum allowed error in the calculated total charge beyond which a warning message will be issued	real	1×10^{-3}

5.24 epsengy

`epsengy`	convergence criterion for the total energy	real	1×10^{-4}

See `epspot`.

5.25 epsforce

`epsforce`	convergence tolerance for the forces during a geometry optimisation run	real	2×10^{-3}

If the mean absolute value of the atomic forces is less than `epsforce` then the geometry optimisation run is ended. See also `tasks` and `latvopt`.

5.26 epslat

`epslat`	vectors with lengths less than this are considered zero	real	10^{-6}

Sets the tolerance for determining if a vector or its components are zero. This is to account for any numerical error in real or reciprocal space vectors.

5.27 epsocc

epsocc	smallest occupancy for which a state will contribute to the density	real	1×10^{-8}

5.28 epspot

epspot	convergence criterion for the Kohn-Sham potential and field	real	1×10^{-6}

If the RMS change in the Kohn-Sham potential and magnetic field is smaller than epspot and the absolute change in the total energy is less than epsengy, then the self-consistent loop is considered converged and exited. For geometry optimisation runs this results in the forces being calculated, the atomic positions updated and the loop restarted. See also epsengy and maxscl.

5.29 epsstress

epsstress	convergence tolerance for the stress tensor during a geometry optimisation run with lattice vector relaxation	real	5×10^{-4}

See also epsforce and latvopt.

5.30 emaxelnes

emaxelnes	maximum allowed initial-state eigenvalue for ELNES calculations	real	-1.2

5.31 emaxrf

emaxrf	energy cut-off used when calculating Kohn-Sham response functions	real	10^6

A typical Kohn-Sham response function is of the form

$$\chi_s(\mathbf{r},\mathbf{r}',\omega) \equiv \frac{\delta\rho(\mathbf{r},\omega)}{\delta v_s(\mathbf{r}',\omega)} = \frac{1}{N_k}\sum_{i\mathbf{k},j\mathbf{k}'}(f_{i\mathbf{k}} - f_{j\mathbf{k}'})\frac{\langle i\mathbf{k}|\hat{\rho}(\mathbf{r})|j\mathbf{k}'\rangle\langle j\mathbf{k}'|\hat{\rho}(\mathbf{r}')|i\mathbf{k}\rangle}{w + (\varepsilon_{i\mathbf{k}} - \varepsilon_{j\mathbf{k}'}) + i\eta},$$

where $\hat{\rho}$ is the density operator; N_k is the number of k-points; $\varepsilon_{i\mathbf{k}}$ and $f_{i\mathbf{k}}$ are the eigenvalues and occupation numbers, respectively. The variable emaxrf is an energy window which limits the summation over states in the formula above so that $|\varepsilon_{i\mathbf{k}} - \varepsilon_{\text{Fermi}}| <$ emaxrf. Reducing this can result in a faster calculation at the expense of accuracy.

5.32 fracinr

fracinr	fraction of the muffin-tin radius up to which lmaxi is used as the angular momentum cut-off	real	0.01

If fracinr is negative then the fraction is determined from $f = \sqrt{(\texttt{lmaxi}+1)^2/(\texttt{lmaxo}+1)^2}$ in order to maintain a minimum density of points throughout the muffin-tin. See lmaxi and lmaxo.

5.33 fsmtype

`fsmtype`	0 for no fixed spin moment (FSM), 1 for total FSM, 2 for local muffin-tin FSM, and 3 for both total and local FSM	integer	0

Set to 1, 2 or 3 for fixed spin moment calculations. To fix only the direction and not the magnitude set to -1, -2 or -3. See also `momfix`, `mommtfix`, `taufsm` and `spinpol`.

5.34 ftmtype

`ftmtype`	1 to enable a fixed tensor moment (FTM) calculation, 0 otherwise	integer	0

If `ftmtype` is -1 then the symmetry corresponding to the tensor moment is broken but no FTM calculation is performed. See *Phys. Rev. Lett.* **103**, 107202 (2009) and also `tmomfix`.

5.35 fxclrc

`fxclrc`	parameters for the dynamical long-range contribution (LRC) to the TDDFT exchange-correlation kernel	real(2)	$(0.0, 0.0)$

These are the parameters α and β for the kernel proposed in *Phys. Rev. B* **72**, 125203 (2005), namely

$$f_{xc}(\mathbf{G}, \mathbf{G}', \mathbf{q}, \omega) = -\frac{\alpha + \beta\omega^2}{q^2}\delta_{\mathbf{G},\mathbf{G}'}\delta_{\mathbf{G},\mathbf{0}}.$$

5.36 fxctype

`fxctype`	integer defining the type of exchange-correlation kernel $f_{\rm xc}$	integer	-1

The acceptable values are:

- -1 $f_{\rm xc}$ defined by `xctype`
- 0,1 RPA ($f_{\rm xc} = 0$)
- 200 Long-range contribution (LRC) kernel, S. Botti *et al.*, *Phys. Rev. B* **72**, 125203 (2005); see `fxclrc`
- 210 'Bootstrap' kernel, S. Sharma, J. K. Dewhurst, A. Sanna and E. K. U. Gross, *Phys. Rev. Lett.* **107**, 186401 (2011)
- 211 Single iteration bootstrap

5.37 gmaxrf

`gmaxrf`	maximum length of $\|\mathbf{G}\|$ for computing response functions	real	3.0

5.38 gmaxvr

`gmaxvr`	maximum length of $\|\mathbf{G}\|$ for expanding the interstitial density and potential	real	12.0

This variable has a lower bound which is enforced by the code as follows:

$$\text{gmaxvr} \to \max\left(\text{gmaxvr}, 2 \times \text{gkmax} + \text{epslat}\right)$$

See `rgkmax`.

5.39 hdbse

`hdbse`	`.true.` if the direct term is to be included in the BSE Hamiltonian	logical	`.true.`

5.40 highq

`highq`	`.true.` if a high quality parameter set should be used	logical	`.false.`

Setting this to `.true.` results in some default parameters being changed to ensure good convergence in most situations. These changes can be overruled by subsequent blocks in the input file. See also `vhighq`.

5.41 hmaxvr

`hmaxvr`	maximum length of **H**-vectors	real	6.0

The **H**-vectors are used for calculating X-ray and magnetic structure factors. They are also used in linear response phonon calculations for expanding the density and potential in plane waves. See also `gmaxvr`, `vhmat`, `reduceh`, `wsfac` and `hkmax`.

5.42 hxbse

`hxbse`	`.true.` if the exchange term is to be included in the BSE Hamiltonian	`.true.`

5.43 hybrid

`hybrid`	`.true` if a hybrid functional is to be used when running a Hartree-Fock calculation	logical	`.false`

See also `hybridc` and `xctype`.

5.44 hybridc

`hybridc`	hybrid functional mixing coefficient	real	1.0

5.45 intraband

`intraband`	`.true.` if the intraband (Drude-like) contribution is to be added to the dieletric tensor	logical	`.false.`

5.46 isgkmax

`isgkmax`	species for which the muffin-tin radius will be used for calculating `gkmax`	integer	-1

The APW cut-off is determined from `gkmax` $=$ `rgkmax`$/R$. The variable `isgkmax` determines which muffin-tin radius is to be used for R. These are the options:

-4	Use the largest radius
-3	Use the smallest radius
-2	Use the fixed value $R = 2.0$
-1	Use the average of the muffin-tin radii
$n \geq 1$	Use the radius of species n

5.47 kstlist

`kstlist(i)`	ith k-point and state pair	integer(2)	$(1, 1)$

This is a user-defined list of k-point and state index pairs which are those used for plotting wavefunctions and writing **L**, **S** and **J** expectation values. Only the first pair is used by the aforementioned tasks. The list should be terminated by a blank line.

5.48 latvopt

`latvopt`	type of lattice vector optimisation to be performed during structural relaxation	integer	0

Optimisation of the lattice vectors will be performed with `task` $= 2, 3$ when `latvopt` $\neq 0$. When `latvopt` $= 1$ the lattice vector optimisation will be constrained only by symmetry. Optimisation over all symmetry-preserving strains except isotropic scaling is performed when `latvopt` $= 2$. If `latvopt` < 0 then the optimisation will be over strain number |`latvopt`|. The list of symmetric strain tensors can be produced with `task` $= 430$. By default (`latvopt` $= 0$) no lattice vector optimisation is performed during structural relaxation. See also `tau0latv` and `atpopt`.

5.49 lmaxapw

`lmaxapw`	angular momentum cut-off for the APW functions	integer	8

5.50 lmaxdos

`lmaxdos`	angular momentum cut-off for the partial DOS plot	integer	3

5.51 lmaxi

`lmaxi`	angular momentum cut-off for the muffin-tin density and potential on the inner part of the muffin-tin	integer	2

Close to the nucleus, the density and potential is almost spherical and therefore the spherical harmonic expansion can be truncated a low angular momentum. See also `fracinr`.

5.52 lmaxo

`lmaxo`	angular momentum cut-off for the muffin-tin density and potential	integer	6

5.53 lmirep

`lmirep`	`.true.` if the Y_{lm} basis is to be transformed into the basis of irreducible representations of the site symmetries for DOS plotting	logical	`.true.`

When lmirep is set to .true., the spherical harmonic basis is transformed into one in which the site symmetries are block diagonal. Band characters determined from the density matrix expressed in this basis correspond to irreducible representations, and allow the partial DOS to be resolved into physically relevant contributions, for example e_g and t_{2g}.

5.54 lorbcnd

`lorbcnd`	`.true.` if conduction state local-orbitals are to be automatically added to the basis	logical	`.false.`

Adding these higher energy local-orbitals can improve calculations which rely on accurate unoccupied states, such as the response function. See also `lorbordc`.

5.55 lorbordc

`lorbordc`	the order of the conduction state local-orbitals	integer	2

See `lorbcnd`.

5.56 lradstp

`lradstp`	radial step length for determining coarse radial mesh	integer	4

Some muffin-tin functions (such as the density) are calculated on a coarse radial mesh and then interpolated onto a fine mesh. This is done for the sake of efficiency. `lradstp` defines the step size in going from the fine to the coarse radial mesh. If it is too large, loss of precision may occur.

5.57 maxitoep

`maxitoep`	maximum number of iterations when solving the exact exchange integral equations	integer	300

See `tau0oep`.

5.58 maxscl

`maxscl`	maximum number of self-consistent loops allowed	integer	200

This determines after how many loops the self-consistent cycle will terminate if the convergence criterion is not met. If `maxscl` is 1 then the density and potential file, `STATE.OUT`, will **not** be written to disk at the end of the loop. See `epspot`.

5.59 mixtype

`mixtype`	type of mixing required for the potential	integer	1

Currently implemented are:

0 Linear mixing
1 Adaptive linear mixing
3 Broyden mixing, *J. Phys. A: Math. Gen.* **17**, L317 (1984)

5.60 mixsdb

`mixsdb`	subspace dimension for Broyden mixing	integer	5

This is the number of mixing vectors which define the subspace in which the Hessian matrix is calculated. See `mixtype` and `broydpm`.

5.61 molecule

`molecule`	`.true.` if the system is an isolated molecule	logical	`.false.`

If `molecule` is `.true.`, then the atomic positions, **a**, given in the `atoms` block are assumed to be in Cartesian coordinates.

5.62 momfix

`momfix`	the desired total moment for a FSM calculation	real(3)	$(0.0, 0.0, 0.0)$

Note that all three components must be specified (even for collinear calculations). See `fsmtype`, `taufsm` and `spinpol`.

5.63 mommtfix

`is`	species number	integer	0
`ia`	atom number	integer	0
`mommtfix`	the desired muffin-tin moment for a FSM calculation	real(3)	$(0.0, 0.0, 0.0)$

The local muffin-tin moments are specified for a subset of atoms, with the list terminated with a blank line. Note that all three components must be specified (even for collinear calculations). See `fsmtype`, `taufsm` and `spinpol`.

5.64 msmooth

`msmooth`	amount of smoothing to be applied to the exchange-correlation potentials and magnetic field	integer	0

Smoothing operations can be applied to the exchange-correlation potentials v_{xc}, w_{xc} and the magnetic field $\mathbf{B}_{xc}$ in order to improve convergence. In the muffin-tin, this smoothing takes the form of m successive three-point running averages applied to the radial component. In the interstitial region, the potential is first Fourier transformed to G-space, then a low-pass filter of the form $\exp[-2m(G/G_{\max})^8]$ is applied and the function is transformed back to real-space.

5.65 mstar

`mstar`	value of the effective mass parameter used for adaptive determination of `swidth`	real	10.0

See `autoswidth`.

5.66 mustar

`mustar`	Coulomb pseudopotential, μ^*, used in the McMillan-Allen-Dynes equation	real	0.15

This is used when calculating the superconducting critical temperature with the formula *Phys. Rev. B 12, 905 (1975)*

$$T_c = \frac{\omega_{\log}}{1.2k_B} \exp\left[\frac{-1.04(1+\lambda)}{\lambda - \mu^*(1+0.62\lambda)}\right],$$

where $\omega_{\log}$ is the logarithmic average frequency and λ is the electron-phonon coupling constant.

5.67 ncbse

`ncbse`	number of conduction states to be used for BSE calculations	integer	3

See also `nvbse`.

5.68 ndspem

`ndspem`	the number of **k**-vector displacements in each direction around `vklem` when computing the numerical derivatives for the effective mass tensor	integer	1

See `deltaem` and `vklem`.

5.69 nempty

`nempty`	the number of empty states per atom and spin	real	4.0

Defines the number of eigenstates beyond that required for charge neutrality. When running metals it is not known *a priori* how many states will be below the Fermi energy for each k-point. Setting `nempty` greater than zero allows the additional states to act as a buffer in such cases. Furthermore, magnetic calculations use the first-variational eigenstates as a basis for setting up the second-variational Hamiltonian, and thus `nempty` will determine the size of this basis set. Convergence with respect to this quantity should be checked.

5.70 ngridk

`ngridk`	the k-point mesh sizes	integer(3)	(1, 1, 1)

The **k**-vectors are generated using

$$\mathbf{k} = (\frac{i_1 + v_1}{n_1}, \frac{i_2 + v_2}{n_2}, \frac{i_3 + v_3}{n_3}),$$

where i_j runs from 0 to $n_j - 1$ and $0 \leq v_j < 1$ for $j = 1, 2, 3$. The vector **v** is given by the variable `vkloff`. See also `reducek`.

5.71 ngridq

`ngridq`	the phonon q-point mesh sizes	integer(3)	$(1, 1, 1)$

Same as `ngridk`, except that this mesh is for the phonon q-points and other tasks. See also `reduceq`.

5.72 nosource

`nosource`	when set to `.true.`, source fields are projected out of the exchange-correlation magnetic field	logical	`.false.`

Experimental feature.

5.73 notes

`notes(i)`	the ith line of the notes	string	-

This block allows users to add their own notes to the file `INFO.OUT`. The block should be terminated with a blank line, and no line should exceed 80 characters.

5.74 npmae

`npmae`	number or distribution of directions for MAE calculations	integer	-1

Automatic determination of the magnetic anisotropy energy (MAE) requires that the total energy is determined for a set of directions of the total magnetic moment. This variable controls the number or distribution of these directions. The convention is:

$-4, -3, -2, -1$	Cardinal directions given by the primitive translation vectors $n_1\mathbf{A}_1 + n_2\mathbf{A}_2 + n_3\mathbf{A}_3$, where $1 \leq n_i \leq \lvert$`npmae`$\rvert$
2	Cartesian x and z directions
3	Cartesian x, y and z directions
$4, 5, \ldots$	Even distribution of `npmae` directions

5.75 ntemp

`ntemp`	number of temperature steps	integer	40

This is the number of temperature steps to be used in the Eliashberg gap and thermodynamic properties calculations.

5.76 nvbse

`nvbse`	number of valence states to be used for BSE calculations	integer	2

See also `ncbse`.

5.77 nwrite

`nwrite`	number of self-consistent loops after which `STATE.OUT` is to be written	integer	0

Normally, the density and potentials are written to the file `STATE.OUT` only after completion of the self-consistent loop. By setting `nwrite` to a positive integer the file will instead be written every `nwrite` loops.

5.78 nxoapwlo

`nxoapwlo`	extra order of radial functions to be added to the existing APW and local-orbital set	integer	0

Setting this variable will result in the APWs and local-orbitals for all species becoming higher order with corresponding increase in derivative matching at the muffin-tin surface. For example, setting `nxoapwlo`=1 turns all APWs into LAPWs.

5.79 optcomp

`optcomp`	the components of the first- or second-order optical tensor to be calculated	integer(3)	$(1, 1, 1)$

This selects which components of the optical tensor you would like to plot. Only the first two are used for the first-order tensor. Several components can be listed one after the other with a blank line terminating the list.

5.80 phwrite

`nphwrt`	number of q-points for which phonon modes are to be found	integer	1
`vqlwrt(i)`	the ith q-point in lattice coordinates	real(3)	$(0.0, 0.0, 0.0)$

This is used in conjunction with `task`=230. The code will write the phonon frequencies and eigenvectors to the file `PHONON.OUT` for all the q-points in the list. The q-points can be anywhere in the Brillouin zone and do not have to lie on the mesh defined by `ngridq`. Obviously, all the dynamical matrices have to be computed first using `task`=200.

5.81 plot1d

`nvp1d`	number of vertices	integer	2
`npp1d`	number of plotting points	integer	200
`vvlp1d(i)`	lattice coordinates for vertex i	real(3)	$(0.0, 0.0, 0.0) \rightarrow (1.0, 1.0, 1.0)$

Defines the path in either real or reciprocal space along which the 1D plot is to be produced. The user should provide `nvp1d` vertices in lattice coordinates.

5.82 plot2d

`vclp2d(0)`	zeroth corner (origin)	real(3)	$(0.0, 0.0, 0.0)$
`vclp2d(1)`	first corner	real(3)	$(1.0, 0.0, 0.0)$
`vclp2d(2)`	second corner	real(3)	$(0.0, 1.0, 0.0)$
`np2d`	number of plotting points in both directions	integer(2)	$(40, 40)$

Defines the corners of a parallelogram and the grid size used for producing 2D plots.

5.83 plot3d

vclp3d(0)	zeroth corner (origin)	real(3)	$(0.0, 0.0, 0.0)$
vclp3d(1)	first corner	real(3)	$(1.0, 0.0, 0.0)$
vclp3d(2)	second corner	real(3)	$(0.0, 1.0, 0.0)$
vclp3d(3)	third corner	real(3)	$(0.0, 0.0, 1.0)$
np3d	number of plotting points each direction	integer(3)	$(20, 20, 20)$

Defines the corners of a box and the grid size used for producing 3D plots.

5.84 primcell

primcell	.true. if the primitive unit cell should be found	logical	.false.

Allows the primitive unit cell to be determined automatically from the conventional cell. This is done by searching for lattice vectors among all those which connect atomic sites, and using the three shortest which produce a unit cell with non-zero volume.

5.85 pulse

n	number of pulses	integer	-
a0(i)	polarisation vector (including amplitude)	real(3)	-
w(i)	frequency	real	-
phi(i)	phase in degrees	real	-
rc(i)	chirp rate	real	-
t0(i)	peak time	real	-
d(i)	full-width at half-maximum	real	-

Parameters used to generate a time-dependent vector potential $\mathbf{A}(t)$ representing a laser pulse. The total vector potential is the sum of individual pulses and is given by the formula

$$\mathbf{A}(t) = \sum_{i=1}^{n} \mathbf{A}_0^i \exp\left[-(t-t_0^i)^2/2\sigma_i^2\right] \sin\left[w_i(t-t_0^i) + \phi_i + r_c^i t^2/2\right],$$

where $\sigma = d/2\sqrt{2\ln 2}$. See also ramp.

5.86 radkpt

radkpt	radius of sphere used to determine k-point density	real	40.0

Used for the automatic determination of the k-point mesh. If autokpt is set to .true. then the mesh sizes will be determined by $n_i = R_k|\mathbf{B}_i| + 1$, where $\mathbf{B}_i$ are the primitive reciprocal lattice vectors.

5.87 ramp

n	number of ramps	integer	-
a0(i)	polarisation vector (including amplitude)	real(3)	-
t0(i)	ramp start time	real	-
c1(i)	linear coefficient of $\mathbf{A}(t)$	real	-
c2(i)	quadratic coefficient	real	-

Parameters used to generate a time-dependent vector potential $\mathbf{A}(t)$ representing a constant or linearly increasing electric field $\mathbf{E}(t) = -\partial\mathbf{A}(t)/\partial t$. The vector potential is given by

$$\mathbf{A}(t) = \sum_{i=1}^{n} \mathbf{A}_0^i \left[c_1(t-t_0) + c_2(t-t_0)^2\right] \Theta(t-t_0).$$

5.88 readadu

`readadu`	set to `.true.` if the interpolation constant for DFT+U should be read from file rather than calculated	logical	`.false.`

When `dftu`=3, the DFT+U energy and potential are interpolated between FLL and AFM. The interpolation constant, α, is normally calculated from the density matrix, but can also be read in from the file `ALPHADU.OUT`. This allows the user to fix α, but is also necessary when calculating forces, since the contribution of the potential of the variation of α with respect to the density matrix is not computed. See `dft+u`.

5.89 reducebf

`reducebf`	reduction factor for the external magnetic fields	real	1.0

After each self-consistent loop, the external magnetic fields are multiplied with `reducebf`. This allows for a large external magnetic field at the start of the self-consistent loop to break spin symmetry, while at the end of the loop the field will be effectively zero, i.e. infinitesimal. See `bfieldc` and `atoms`.

5.90 reduceh

`reduceh`	set to `.true.` if the reciprocal **H**-vectors should be reduced by the symmorphic crystal symmetries	logical	.true.

See `hmaxvr` and `vmat`.

5.91 reducek

`reducek`	type of reduction of the k-point set	integer	1

Types of reduction are defined by the symmetry group used:

0 no reduction
1 reduce with full crystal symmetry group (including non-symmorphic symmetries)
2 reduce with symmorphic symmetries only

See also `ngridk` and `vkloff`.

5.92 reduceq

`reduceq`	type of reduction of the q-point set	integer	1

See `reducek` and `ngridq`.

5.93 rgkmax

rgkmax	$R_{\rm min}^{\rm MT} \times \max\{\|\mathbf{G}+\mathbf{k}\|\}$	real	7.0

This sets the maximum length for the $\mathbf{G}+\mathbf{k}$ vectors, defined as `rgkmax` divided by the average muffin-tin radius. See `isgkmax`.

5.94 rotavec

axang	axis-angle representation of lattice vector rotation	real(4)	$(0.0, 0.0, 0.0, 0.0)$

This determines the rotation matrix which is applied to the lattice vectors prior to any calculation. The first three components specify the axis and the last component is the angle in degrees. The 'right-hand rule' convention is followed.

5.95 scale

scale	lattice vector scaling factor	real	1.0

Scaling factor for all three lattice vectors. Applied in conjunction with `scale1`, `scale2` and `scale3`.

5.96 scale1/2/3

scale1/2/3	separate scaling factors for each lattice vector	real	1.0

5.97 scissor

scissor	the scissor correction	real	0.0

This is the scissor shift applied to states above the Fermi energy *Phys. Rev. B* **43**, 4187 (1991). Affects optics calculations only.

5.98 scrpath

scrpath	scratch space path	string	null

This is the scratch space path where the eigenvector files `EVALFV.OUT` and `EVALSV.OUT` will be written. If the run directory is accessed via a network then `scrpath` can be set to a directory on the local disk, for example `/tmp/`. Note that the forward slash `/` at the end of the path must be included.

5.99 socscf

socscf	scaling factor for the spin-orbit coupling term in the Hamiltonian	real	1.0

This can be used to enhance the effect of spin-orbit coupling in order to accurately determine the magnetic anisotropy energy (MAE).

5.100 spincore

spincore	set to `.true.` if the core should be spin-polarised	logical	`.false.`

5.101 spinorb

spinorb	set to .true. if a spin-orbit coupling is required	logical	.false.

If spinorb is .true., then a $\boldsymbol{\sigma}\cdot\mathbf{L}$ term is added to the second-variational Hamiltonian. See spinpol.

5.102 spinpol

spinpol	set to .true. if a spin-polarised calculation is required	logical	.false.

If spinpol is .true., then the spin-polarised Hamiltonian is solved as a second-variational step using two-component spinors in the Kohn-Sham magnetic field. The first variational scalar wavefunctions are used as a basis for setting this Hamiltonian.

5.103 spinsprl

spinsprl	set to .true. if a spin-spiral calculation is required	logical	.false.

Experimental feature for the calculation of spin-spiral states. See vqlss for details.

5.104 sppath

sppath	path where the species files can be found	string	null

Note that the forward slash / at the end of the path must be included.

5.105 ssdph

ssdph	set to .true. if a complex de-phasing factor is to be used in spin-spiral calculations	logical	.true.

If this is .true. then spin-spiral wavefunctions in each muffin-tin at position $\mathbf{r}_\alpha$ are de-phased by the matrix

$$\begin{pmatrix} e^{-i\mathbf{q}\cdot\mathbf{r}_\alpha/2} & 0 \\ 0 & e^{i\mathbf{q}\cdot\mathbf{r}_\alpha/2} \end{pmatrix}.$$

In simple situations, this has the advantage of producing magnon dynamical matrices which are already in diagonal form. This option should be used with care, and a full understanding of the spin-spiral configuration is required. See spinsprl.

5.106 stype

stype	integer defining the type of smearing to be used	integer	3

A smooth approximation to the Dirac delta function is needed to compute the occupancies of the Kohn-Sham states. The variable swidth determines the width of the approximate delta function. Currently implemented are

0 Gaussian
1 Methfessel-Paxton order 1, Phys. Rev. B **40**, 3616 (1989)
2 Methfessel-Paxton order 2
3 Fermi-Dirac

See also autoswidth, swidth and tempk.

5.107 swidth

swidth	width of the smooth approximation to the Dirac delta function	real	0.001

See stype for details and the variable tempk.

5.108 tasks

task(i)	the ith task	integer	-1

A list of tasks for the code to perform sequentially. The list should be terminated with a blank line. Each task has an associated integer as follows:

-1	Write out the version number of the code.
0	Ground state run starting from the atomic densities.
1	Resumption of ground-state run using density in `STATE.OUT`.
2	Geometry optimisation run starting from the atomic densities, with atomic positions written to `GEOMETRY.OUT`.
3	Resumption of geometry optimisation run using density in `STATE.OUT` but with positions from `elk.in`.
5	Ground state Hartree-Fock run.
10	Total, partial and interstitial density of states (DOS).
14	Plots the smooth Dirac delta and Heaviside step functions used by the code to calculate occupancies.
15	Output **L**, **S** and **J** total expectation values.
16	Output **L**, **S** and **J** expectation values for each k-point and state in `kstlist`.
20	Band structure plot.
21	Band structure plot which includes total and angular momentum characters for every atom.
22	Band structure plot which includes (l, m) character for every atom.
23	Band structure plot which includes spin character for every atom.
25	Compute the effective mass tensor at the k-point given by `vklem`.
31, 32, 33	1/2/3D charge density plot.
41, 42, 43	1/2/3D exchange-correlation and Coulomb potential plots.
51, 52, 53	1/2/3D electron localisation function (ELF) plot.
61, 62, 63	1/2/3D wavefunction plot: $\lvert\Psi_{i\mathbf{k}}(\mathbf{r})\rvert^2$.
65	Write the core wavefunctions to file for plotting.
71, 72, 73	1/2/3D plot of magnetisation vector field, $\mathbf{m}(\mathbf{r})$.
81, 82, 83	1/2/3D plot of exchange-correlation magnetic vector field, $\mathbf{B}_{\rm xc}(\mathbf{r})$.
91, 92, 93	1/2/3D plot of $\nabla \cdot \mathbf{B}_{\rm xc}(\mathbf{r})$.
100	3D Fermi surface plot using the scalar product $p(\mathbf{k}) = \Pi_i(\epsilon_{i\mathbf{k}} - \epsilon_{\rm F})$.
101	3D Fermi surface plot using separate bands (minus the Fermi energy).
102	3D Fermi surface which can be plotted with XCrysDen.
105	3D nesting function plot.
110	Calculation of Mössbauer contact charge densities and magnetic fields at the nuclear sites.
115	Calculation of the electric field gradient (EFG) at the nuclear sites.
120	Output of the momentum matrix elements $\langle\Psi_{i\mathbf{k}}\rvert - i\nabla\lvert\Psi_{j\mathbf{k}}\rangle$.
121	Linear optical dielectric response tensor calculated within the random phase approximation (RPA) and in the $q \rightarrow 0$ limit, with no microscopic contributions.
122	Magneto optical Kerr effect (MOKE) angle.
125	Non-linear optical second harmonic generation.

130	Output matrix elements of the type $\langle\Psi_{i\mathbf{k}+\mathbf{q}}\vert\exp[i(\mathbf{G}+\mathbf{q})\cdot\mathbf{r}]\vert\Psi_{j\mathbf{k}}\rangle$.
135	Output all wavefunctions expanded in the plane wave basis up to a cut-off defined by `rgkmax`.
140	Energy loss near edge structure (ELNES).
141, 142, 143	1/2/3D plot of the electric field $\mathbf{E}(\mathbf{r})\equiv\nabla V_{\mathrm{C}}(\mathbf{r})$.
151, 152, 153	1/2/3D plot of $\mathbf{m}(\mathbf{r})\times\mathbf{B}_{\mathrm{xc}}(\mathbf{r})$.
162	Scanning-tunneling microscopy (STM) image.
180	Generate the RPA inverse dielectric function with local contributions and write it to file.
185	Write the Bethe-Salpeter equation (BSE) Hamiltonian to file.
186	Diagonalise the BSE Hamiltonian and write the eigenvectors and eigenvalues to file.
187	Output the BSE dielectric response function.
190	Write the atomic geometry to file for plotting with XCrySDen and V_Sim.
195	Calculation of X-ray density structure factors.
196	Calculation of magnetic structure factors.
200	Calculation of phonon dynamical matrices on a q-point set defined by `ngridq` using the supercell method.
202	Phonon dry run: just produce a set of empty DYN files.
205	Calculation of phonon dynamical matrices using density functional perturbation theory (DFPT).
210	Phonon density of states.
220	Phonon dispersion plot.
230	Phonon frequencies and eigenvectors for an arbitrary q-point.
240, 241	Generate the **q**-dependent phonon linewidths and electron-phonon coupling constants and write them to file.
245	Phonon linewidths plot.
250	Eliashberg function $\alpha^2F(\omega)$, electron-phonon coupling constant λ, and the McMillan-Allen-Dynes critical temperature T_c.
300	Reduced density matrix functional theory (RDMFT) calculation.
320	Time-dependent density functional theory (TDDFT) calculation of the dielectric response function including microscopic contributions.
330, 331	TDDFT calculation of the spin-polarised response function for arbitrary **q**-vectors. Task 331 writes the entire response function $\overleftrightarrow{\chi}(\mathbf{G},\mathbf{G}',q,\omega)$ to file.
400	Calculation of tensor moments and corresponding DFT+U Hartree-Fock energy contributions.
450	Generates a laser pulse in the form of a time-dependent vector potential $\mathbf{A}(t)$ and writes it to AFIELDT.OUT.
460	Time evolution run using TDDFT under the influence of $\mathbf{A}(t)$.

5.109 tau0atp

`tau0atp`	the step size to be used for atomic position optimisation	real	0.25

The position of atom α is updated on step m of a geometry optimisation run using

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

where τ_α is set to `tau0atp` for $m=0$, and incremented by the same amount if the atom

is moving in the same direction between steps. If the direction changes then τ_α is reset to `tau0atp`.

5.110 tau0latv

`tau0latv`	the step size to be used for lattice vector optimisation	real	0.25

This parameter is used for lattice vector optimisation in a procedure identical to that for atomic position optimisation. See `tau0atp` and `latvopt`.

5.111 tauoep

`tauoep`	step length and starting fraction for the OEP iterative solver	real(2)	$(0.05, 0.95)$

The optimised effective potential is determined using an interative method [Phys. Rev. Lett. 98, 196405 (2007)]. This variable sets the step length described in the article. The second number defines the fraction of the existing potential to be used as the starting point of the iterative procedure. These parameters are also used for inverting the Kohn-Sham equations as required by the self-consistent density GW method. See `maxitoep`.

5.112 taufsm

`taufsm`	the step size to be used when finding the effective magnetic field in fixed spin moment calculations	real	0.01

An effective magnetic field, $\mathbf{B}_{\rm FSM}$, is required for fixing the spin moment to a given value, $\mathbf{M}_{\rm FSM}$. This is found 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:

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

where λ is proportional to `taufsm`. See also `fsmtype`, `momfix` and `spinpol`.

5.113 tempk

`tempk`	temperature T of the electronic system in kelvin	real	-

Assigning a value to this variable sets `stype` to 3 (Fermi-Dirac) and the smearing width to $k_{\rm B}T$.

5.114 tforce

`tforce`	set to `.true.` if the force should be calculated at the end of the self-consistent cycle	logical	`.false.`

This variable is automatically set to `.true.` when performing geometry optimisation.

5.115 tefvit

`tefvit`	set to `.true.` if the first-variational eigenvalue equation should be solved iteratively	logical	`.false.`

5.116 tefvr

`tefvr`	set to `.true.` if a real symmetric eigenvalue solver should be used for crystals which have inversion symmetry	logical	`.true.`

For crystals with inversion symmetry, the first-variational Hamiltonian and overlap matrices can be made real by using appropriate transformations. In this case, a real symmetric (instead of complex Hermitian) eigenvalue solver can be used. This makes the calculation about three times faster.

5.117 tmomfix

`ntmfix`	number of tensor moments (TM) to be fixed	integer	0
`is(i)`	species number for entry i	integer	-
`ia(i)`	atom number	integer	-
`(l, n)(i)`	l and n indices of TM	integer	-
`(k, p, x, y)(i)` or `(k, p, r, t)(i)`	indices for the 2-index or 3-index TM, respectively	integer	-
`z(i)`	complex TM value	complex	-
`p(i)`	parity of spatial rotation	integer	-
`aspl(i)`	Euler angles of spatial rotation	real(3)	-
`aspn(i)`	Euler angles of spin rotation	real(3)	-

This block sets up the fixed tensor moment (FTM). There should be as many TM entries as `ntmfix`. See *Phys. Rev. Lett.* **103**, 107202 (2009) for the tensor moment indexing convention. This is a highly complicated feature of the code, and should only be attempted with a full understanding of tensor moments.

5.118 tmwrite

`tmwrite`	set to `.true.` if the tensor moments and the corresponding decomposition of DFT+U energy should be calculated at every loop of the self-consistent cycle	logical	`.false.`

This variable is useful to check the convergence of the tensor moments in DFT+U cacu-lations. Alternatively, with `task` equal to 400, one can calculate the tensor moments and corresponding DFT+U energy contributions from a given density matrix and set of Slater parameters at the end of the self-consistent cycle.

5.119 tsediag

`tsediag`	set to `.true.` if the self-energy matrix should be treated as diagonal	logical	`.true.`

When this variable is `.true.`, the self-energy used in a GW calculation $\Sigma_{ij}(\mathbf{k},\omega)$ is taken to be diagonal in the Kohn-Sham state indices i and j. When `tsediag` is `.false.`, the entire matrix is used. See also `twdiag`.

5.120 tshift

`tshift`	set to `.true.` if the crystal can be shifted so that the atom closest to the origin is exactly at the origin	logical	`.true.`

5.121 tstime

`tstime`	total simulation time of time evolution run	real	1000.0

See also `dtimes`.

5.122 twdiag

`twdiag`	set to `.true.` if the screened interaction matrix should be treated as diagonal	logical	`.false.`

When this variable is `.true.`, the screened interaction used in a GW calculation $W(\mathbf{G},\mathbf{G}',\mathbf{q},\omega)$ is taken to be diagonal in the plane wave indices $\mathbf{G}$ and $\mathbf{G}'$. See also `tsediag`.

5.123 vhmat

`vhmat(1)`	matrix row 1	real(3)	$(1.0, 0.0, 0.0)$
`vhmat(2)`	matrix row 2	real(3)	$(0.0, 1.0, 0.0)$
`vhmat(3)`	matrix row 3	real(3)	$(0.0, 0.0, 1.0)$

This is the transformation matrix M applied to every vector $\mathbf{H}$ in the structure factor output files `SFACRHO.OUT` and `SFACMAG.OUT`. It is stored in the usual row-column setting and applied directly as $\mathbf{H}' = M\mathbf{H}$ to every vector but *only* when writing the output files. See also `hmaxvr` and `reduceh`.

5.124 vhighq

`vhighq`	`.true.` if a very high quality parameter set should be used	logical	`.false.`

Setting this to `.true.` results in some default parameters being changed to ensure excellent convergence in most situations. See also `highq`.

5.125 vklem

`vklem`	the k-point in lattice coordinates at which to compute the effective mass tensors	real(3)	$(0.0, 0.0, 0.0)$

See `deltaem` and `ndspem`.

5.126 vkloff

`vkloff`	the k-point offset vector in lattice coordinates	real(3)	$(0.0, 0.0, 0.0)$

See `ngridk`.

5.127 vqlss

`vqlss`	the $\mathbf{q}$-vector of the spin-spiral state in lattice coordinates	real(3)	$(0.0, 0.0, 0.0)$

Spin-spirals arise from spinor states assumed to be of the form

$$\Psi^{\mathbf{q}}_{\mathbf{k}}(\mathbf{r}) = \begin{pmatrix} U^{\mathbf{q}\uparrow}_{\mathbf{k}}(\mathbf{r})e^{i(\mathbf{k}+\mathbf{q}/\mathbf{2})\cdot\mathbf{r}} \\ U^{\mathbf{q}\downarrow}_{\mathbf{k}}(\mathbf{r})e^{i(\mathbf{k}-\mathbf{q}/\mathbf{2})\cdot\mathbf{r}} \end{pmatrix}.$$

These are determined using a second-variational approach, and give rise to a magnetisation density of the form

$$\mathbf{m}^{\mathbf{q}}(\mathbf{r}) = (m_x(\mathbf{r})\cos(\mathbf{q}\cdot\mathbf{r}), m_y(\mathbf{r})\sin(\mathbf{q}\cdot\mathbf{r}), m_z(\mathbf{r})),$$

where m_x, m_y and m_z are lattice periodic. See also `spinsprl`.

5.128 wmaxgw

`wmaxgw`	maximum Matsubara frequency for GW calculations	real	-5.0

This defines the cut-off of the Matsubara frequencies on the imaginary axis for calculating the GW self-energy and solving the Dyson equation. If this number is negative then the cut-off is taken to be $|$`wmaxgw`$| \times \Delta\epsilon$, where $\Delta\epsilon$ is the difference between the largest and smallest Kohn-Sham valence eigenvalues.

5.129 wplot

`nwplot`	number of frequency/energy points in the DOS or optics plot	integer	500
`ngrkf`	fine k-point grid size used for integrating functions in the Brillouin zone	integer	100
`nswplot`	level of smoothing applied to DOS/optics output	integer	1
`wplot`	frequency/energy window for the DOS or optics plot	real(2)	$(-0.5, 0.5)$

DOS and optics plots require integrals of the kind

$$g(\omega_i) = \frac{\Omega}{(2\pi)^3}\int_{\mathrm{BZ}} f(\mathbf{k})\delta(\omega_i - e(\mathbf{k}))d\mathbf{k}.$$

These are calculated by first interpolating the functions $e(\mathbf{k})$ and $f(\mathbf{k})$ with the trilinear method on a much finer mesh whose size is determined by `ngrkf`. Then the ω-dependent histogram of the integrand is accumulated over the fine mesh. If the output function is noisy then either `ngrkf` should be increased or `nwplot` decreased. Alternatively, the output function can be artificially smoothed up to a level given by `nswplot`. This is the number of successive 3-point averages to be applied to the function g.

5.130 wsfac

`wsfac`	energy window to be used when calculating density or magnetic structure factors	real(2)	$(-10^6, 10^6)$

Only those states with eigenvalues within this window will contribute to the density or magnetisation. See also `hmaxvr` and `vhmat`.

5.131 xctype

`xctype`	integers defining the type of exchange-correlation functional to be used	integer(3)	$(3, 0, 0)$

Normally only the first value is used to define the functional type. The other value may be used for external libraries. Currently implemented are:

$-n$	Exact-exchange optimised effective potential (EXX-OEP) method with correlation energy and potential given by functional number n
1	No exchange-correlation funtional ($E_{\rm xc} \equiv 0$)
2	LDA, Perdew-Zunger/Ceperley-Alder, *Phys. Rev. B* **23**, 5048 (1981)
3	LSDA, Perdew-Wang/Ceperley-Alder, *Phys. Rev. B* **45**, 13244 (1992)
4	LDA, X-alpha approximation, J. C. Slater, *Phys. Rev.* **81**, 385 (1951)
5	LSDA, von Barth-Hedin, *J. Phys. C* **5**, 1629 (1972)
20	GGA, Perdew-Burke-Ernzerhof, *Phys. Rev. Lett.* **77**, 3865 (1996)
21	GGA, Revised PBE, Zhang-Yang, *Phys. Rev. Lett.* **80**, 890 (1998)
22	GGA, PBEsol, Phys. Rev. Lett. 100, 136406 (2008)
26	GGA, Wu-Cohen exchange (WC06) with PBE correlation, *Phys. Rev. B* **73**, 235116 (2006)
30	GGA, Armiento-Mattsson (AM05) spin-unpolarised functional, *Phys. Rev. B* **72**, 085108 (2005)
100	Libxc functionals; the second and third values of `xctype` define the exchange and correlation functionals in the Libxc library, respectively

6 Contributing to Elk

Please bear in mind when writing code for the Elk project that it should be an exercise in physics and not software engineering. All code should therefore be kept as simple and concise as possible, and above all it should be easy for anyone to locate and follow the Fortran representation of the original mathematics. We would also appreciate the following conventions being adhered to:

- Strict Fortran 2003 should be used. Features which are marked as obsolescent in Fortran 2003 should be avoided. These include assigned format specifiers, labeled do-loops, computed goto statements and statement functions.
- Modules should be used in place of common blocks for declaring global variables. Use the existing modules to declare new global variables.
- Any code should be written in lower-case free form style, starting from column one. Try and keep the length of each line to fewer than 80 characters using the & character for line continuation.
- Every function or subroutine, no matter how small, should be in its own file named `routine.f90`, where `routine` is the function or subroutine name. It is recommended that the routines are named so as to make their purpose apparent from the name alone.
- Use of `implicit none` is mandatory. Remember also to define the `intent` of any passed arguments.
- Local allocatable arrays must be deallocated on exit of the routine to prevent memory leakage. Use of automatic arrays should be limited to arrays of small size.
- Every function or subroutine must be documented with the Protex source code documentation system. This should include a short LaTeX description of the algorithms and methods involved. Equations which need to be referenced should be labeled with

`routine_1`, `routine_2`, etc. The authorship of each new piece of code or modification should be indicated in the `REVISION HISTORY` part of the header. See the Protex documentation for details.

- Ensure as much as possible that a routine will terminate the program when given improper input instead of continuing with erroneous results. Specifically, functions should have a well-defined domain for which they return accurate results. Input outside that domain should result in an error message and termination.
- Report errors prior to termination with a short description, for example:

```
write(*,*)
write(*,'("Error(readinput): natoms <= 0 : ",I8)') natoms(is)
write(*,'(" for species ",I4)') is
write(*,*)
stop
```

- Wherever possible, real numbers outputted as ASCII data should be formatted with the `G18.10` specifier.
- Avoid redundant or repeated code: check to see if the routine you need already exists, before writing a new one.
- All reading in of ASCII data should be done in the subroutine `readinput`. For binary data, separate routines for reading and writing should be used (for example `writestate` and `readstate`).
- Input filenames should be in lowercase and have the extension `.in` . All output filenames should be in uppercase with the extension `.OUT` .
- All internal units should be atomic. Input and output units should be atomic by default and clearly stated otherwise. Rydbergs should not be used under any circumstances.

6.1 Licensing

Routines which constitute the main part of the code are released under the GNU General Public License (GPL). Library routines are released under the less restrictive GNU Lesser General Public License (LGPL). Both licenses are contained in the file `COPYING`. Any contribution to the code must be licensed at the authors' discretion under either the GPL or LGPL. Author(s) of the code retain the copyrights. Copyright and (L)GPL information must be included at the beginning of every file, and no code will be accepted without this.

7 Routine/Function Prologues

7.1 allatoms (Source File: allatoms.f90)

INTERFACE:

```
subroutine allatoms
```

USES:

```
use modmain
use modxcifc
use modomp
```

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)
```

7.2 atom (Source File: atom.f90)

INTERFACE:

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

USES:

```
use modxcifc
```

INPUT/OUTPUT PARAMETERS:

```
sol    : speed of light in atomic units (in,real)
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(3))
```

```
xcgrad : 1 for GGA functional, 0 otherwise (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. 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)
```

7.3 atpstep (Source File: atpstep.f90)

INTERFACE:

```
subroutine atpstep
```

USES:

```
use modmain
use modmpi
```

DESCRIPTION:

Makes a geometry optimisation step and 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)
```

7.4 axangrot (Source File: axangrot.f90)

INTERFACE:

```
pure subroutine axangrot(v,th,rot)
```

INPUT/OUTPUT PARAMETERS:

```
v   : axis vector (in,real)
th  : rotation angle (in,real)
rot : rotation matrix (out,real(3,3))
```

DESCRIPTION:

Determines the 3×3 rotation matrix of a rotation specified by an axis-angle pair following the 'right-hand rule'. The axis vector need not be normalised. See `rotaxang` for details.

REVISION HISTORY:

```
Created February 2014 (JKD)
```

7.5 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)
```

7.6 bandstr (Source File: bandstr.f90)

INTERFACE:

```
subroutine bandstr
```

USES:

```
use modmain
use modomp
```

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)
```

7.7 brzint (Source File: brzint.f90)

INTERFACE:

```
subroutine brzint(nsm,ngridk,nsk,ivkik,nw,wint,n,ld,e,f,g)
```

USES:

```
use modomp
```

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))
ivkik  : map from (i1,i2,i3) to k-point index
         (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

$$g(\omega_i) = \frac{\Omega}{(2\pi)^3} \int_{\text{BZ}} f(\mathbf{k}) \delta(\omega_i - e(\mathbf{k})) d\mathbf{k},$$

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)
Added parallelism, March 2020 (JKD)
```

7.8 charge (Source File: charge.f90)

INTERFACE:

```
subroutine charge
```

USES:

```
use modmain
use modtest
```

DESCRIPTION:

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

REVISION HISTORY:

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

7.9 checkmt (Source File: checkmt.f90)

INTERFACE:

```
subroutine checkmt
```

USES:

```
use modmain
use modmpi
use modvars
```

DESCRIPTION:

Checks for muffin-tins which are too close together or intersecting. If any such muffin-tins are found then the radii of their associated atomic species are adjusted so that the minimum distance between their surfaces is `rmtdelta`.

REVISION HISTORY:

```
Created May 2003 (JKD)
Modified, October 2011 (JKD)
```

7.10 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-Gordon 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 $J_i \leq 50$. See `wigner3j`.

REVISION HISTORY:

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

7.11 dielectric (Source File: dielectric.f90)

INTERFACE:

```
subroutine dielectric
```

USES:

```
use modmain
use modtest
use modomp
```

DESCRIPTION:

Computes the dielectric tensor, optical conductivity and plasma frequency. The formulae are taken from *Physica Scripta* **T109**, 170 (2004).

REVISION HISTORY:

```
Created November 2005 (SS and JKD)
Added plasma frequency and intraband contribution (S. Lebegue)
Complete rewrite, 2008 (JKD)
Fixed problem with plasma frequency, 2009 (Marty Blaber and JKD)
Parallelised, 2009 (M. Blaber)
```

7.12 dos (Source File: dos.f90)

INTERFACE:

```
subroutine dos(fext,tocc,occsvp)
```

USES:

```
use modmain
use modomp
use modtest
```

INPUT/OUTPUT PARAMETERS:

```
fext   : filename extension (in,character(*))
tocc   : .true. if just the occupied orbitals should contribute to the DOS
         (in,logical)
occsvp : occupation numbers of second-variational orbitals
         (in,real(nstsv,nkpt))
```

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 `gendmatk` and `brzint`.

REVISION HISTORY:

```
Created January 2004 (JKD)
Parallelised and included sum over m, November 2009 (F. Cricchio)
```

7.13 elfplot (Source File: elfplot.f90)

INTERFACE:

```
subroutine elfplot
```

USES:

```
use modmain
```

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)
```

7.14 eliashberg (Source File: eliashberg.f90)

INTERFACE:

```
subroutine eliashberg
```

USES:

```
use modmain
use modphonon
use modomp
```

DESCRIPTION:

Calculates the superconducting gap within Eliashberg theory. This implementation is isotropic and assumes a flat density of states. The Eliashberg function α^2F is required as input for this calculation.

REVISION HISTORY:

```
Created December 2010 (Antonio Sanna)
Modified, June 2011 (JKD)
```

7.15 energy (Source File: energy.f90)

INTERFACE:

```
subroutine energy
```

USES:

```
use modmain
use moddftu
use modtest
```

DESCRIPTION:

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

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

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 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

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

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)

7.16 energyfdu (Source File: energyfdu.f90)

INTERFACE:

```
subroutine energyfdu
```

USES:

```
use modmain
use moddftu
use modmpi
```

DESCRIPTION:

Calculates the energies of radial functions to be used to calculate the Slater integrals. By convention those energies are chosen to be the ones at the center of the band.

REVISION HISTORY:

Created April 2008 (F. Cricchio)

7.17 erf (Source File: erf.f90)

INTERFACE:

```
pure 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)

7.18 eulerrot (Source File: eulerrot.f90)

INTERFACE:

```
pure subroutine eulerrot(ang,rot)
```

INPUT/OUTPUT PARAMETERS:

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

DESCRIPTION:

Given a set of Euler angles, (α, β, γ), this routine determines the corresponding 3×3 rotation matrix. The so-called 'y-convention' is taken for the Euler angles. See the routine `roteuler` for details.

REVISION HISTORY:

```
Created January 2014 (JKD)
```

7.19 eveqn (Source File: eveqn.f90)

subroutine eveqn(ik,evalfv,evecfv,evecsv) *USES:*

```
use modmain
use modulr
```

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 eigenvalue equations. See routines `match`, `eveqnfv`, `eveqnss` and `eveqnsv`.

REVISION HISTORY:

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

7.20 eveqnfv (Source File: eveqnfv.f90)

INTERFACE:

```
subroutine eveqnfv(nmatp,ngp,igpig,vpc,vgpc,apwalm,evalfv,evecfv)
```

USES:

```
use modmain
use modomp
```

INPUT/OUTPUT PARAMETERS:

```
nmatp  : order of overlap and Hamiltonian matrices (in,integer)
ngp    : number of G+p-vectors (in,integer)
igpig  : index from G+p-vectors to G-vectors (in,integer(ngkmax))
vpc    : p-vector in Cartesian coordinates (in,real(3))
vgpc   : G+p-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 eigenvalue equation,

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

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

REVISION HISTORY:

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

7.21 eveqnfvr (Source File: eveqnfvr.f90)

INTERFACE:

```
subroutine eveqnfvr(nmatp,ngp,vpc,h,o,evalfv,evecfv)
```

USES:

```
use modmain
use modomp
```

INPUT/OUTPUT PARAMETERS:

```
nmatp  : order of overlap and Hamiltonian matrices (in,integer)
ngp    : number of G+p-vectors (in,integer)
vpc    : p-vector in Cartesian coordinates (in,real(3))
h,o    : Hamiltonian and overlap matrices in upper triangular form
         (in,complex(*))
evalfv : first-variational eigenvalues (out,real(nstfv))
evecfv : first-variational eigenvectors (out,complex(nmatmax,nstfv))
```

DESCRIPTION:

This routine solves the first-variational eigenvalue equation for the special case when inversion symmetry is present. In this case the Hamiltonian and overlap matrices can be made real by using appropriate linear combinations of the local-orbitals for atoms related by inversion symmetry. These are derived from the effect of parity and complex conjugation on the spherical harmonics: $PY_{lm} = (-1)^l Y_{lm}$ and $(Y_{lm})^* = (-1)^m Y_{l-m}$.

REVISION HISTORY:

```
Created May 2011 (JKD)
```

7.22 factnm (Source File: factnm.f90)

INTERFACE:

```
elemental real(8) function factnm(n,m)
```

INPUT/OUTPUT PARAMETERS:

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

DESCRIPTION:

Returns the multifactorial

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

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

REVISION HISTORY:

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

7.23 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)
```

7.24 fderiv (Source File: fderiv.f90)

INTERFACE:

```
subroutine fderiv(m,n,x,f,g)
```

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))
```

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 $m = -1$ the anti-derivative of f given by

$$g(x_i) = \int_{x_1}^{x_i} f(x)\,dx$$

is calculated. Both derivatives and integrals are computed by first fitting the function to a clamped cubic spline.

REVISION HISTORY:

```
Created May 2002 (JKD)
```

7.25 findband (Source File: findband.f90)

INTERFACE:

```
subroutine findband(sol,l,nr,r,vr,eps,demax,e,fnd)
```

INPUT/OUTPUT PARAMETERS:

```
sol   : speed of light in atomic units (in,real)
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))
eps   : energy search tolerance (in,real)
demax : maximum allowed change from the input energy; enforced only if e < 0
        (in,real)
e     : input energy and returned band energy (inout,real)
fnd   : set to .true. if the band energy is found (out,logical)
```

DESCRIPTION:

Finds the band energies for a given radial potential and angular momentum. This is done by first searching upwards in energy, starting from the input energy plus the offset 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)
Added two-pass loop, October 2013 (JKD)
```

7.26 findlambdadu (Source File: findlambdadu.f90)

INTERFACE:

```
subroutine findlambdadu(is,l,ufix,lambda0,lambda)
use modmpi
```

INPUT/OUTPUT PARAMETERS:

```
is      : species type (in,integer)
l       : angular momentum (in,integer)
ufix    : fixed U (in,integer)
lambda0 : starting value for screening length  (inout,real)
lambda  : screening length corresponding to fixed U (out,real)
```

DESCRIPTION:

Find the screening length corresponding to a fixed value of U by using the half-interval method in the first few steps and then the more efficient secant method. For $U=0$ the code automatically sets the screening length to lambdamax $=50$. This value is enough to get $F^{(k)}\sim 10^{-3}$ corresponding to $U\sim 0$ (that perfectly mimics a bare DFT calculation). However the code is stable up to lambdamax ~ 200 thanks to improvement of `spline`.

REVISION HISTORY:

```
Created July 2009 (Francesco Cricchio)
```

7.27 findngkmax (Source File: findngkmax.f90)

INTERFACE:

```
pure subroutine findngkmax(nkpt,vkc,nspnfv,vqcss,ngv,vgc,gkmax,ngkmax)
```

INPUT/OUTPUT PARAMETERS:

```
nkpt   : number of k-points (in,integer)
vkc    : k-point vectors in Cartesian coordinates (in,real(3,nkpt))
nspnfv : number of first-variational spin components: 1 normal case, 2 for
         spin-spiral case (in,integer)
vqcss  : spin-spiral q-vector, not referenced if nspnfv=1 (in,integer)
ngv    : number of G-vectors (in,integer)
vgc    : G-vectors in Cartesian coordinates (in,real(3,ngv))
gkmax  : maximum allowed |G+k| (in,real)
ngkmax : maximum number of G+k-vectors over all k-points (out,integer)
```

DESCRIPTION:

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

REVISION HISTORY:

```
Created October 2004 (JKD)
Modified, August 2012 (JKD)
Removed modmain and added arguments, September 2012 (JKD)
```

7.28 findprimcell (Source File: findprimcell.f90)

INTERFACE:

```
subroutine findprimcell
```

USES:

```
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)
```

7.29 findswidth (Source File: findswidth.f90)

INTERFACE:

```
subroutine findswidth
```

USES:

```
use modmain
```

DESCRIPTION:

Calculates the smearing width from the k-point density, $V_{\rm BZ}/n_k$; the valence band width, W; and an effective mass parameter, m^*; according to

$$\sigma = \frac{\sqrt{2W}}{m^*}\left(\frac{3}{4\pi}\frac{V_{\rm BZ}}{n_k}\right)^{1/3}.$$

The valence bandwidth is determined by stepping down in energy from the Fermi level until a gap larger than a given tolerance is found. This method was presented in T. Björkman and O. Gråsäs, *Int. J. Quant. Chem.* DOI: 10.1002/qua.22476.

REVISION HISTORY:

```
Created April 2010 (Torbjorn Bjorkman and JKD)
```

7.30 findsymcrys (Source File: findsymcrys.f90)

INTERFACE:

```
subroutine findsymcrys
```

USES:

```
use modmain
use modmpi
use modtest
```

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)
```

7.31 findsym (Source File: findsym.f90)

INTERFACE:

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

USES:

```
use modmain
use moddftu
```

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)
```

7.32 findsymlat (Source File: findsymlat.f90)

INTERFACE:

```
subroutine findsymlat
```

USES:

```
use modmain
use modtddft
```

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

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

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

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

The first matrix in the set returned is the identity.

REVISION HISTORY:

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

7.33 force (Source File: force.f90)

INTERFACE:

```
subroutine force
```

USES:

```
use modmain
use modtddft
use modtest
use modmpi
use modomp
```

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 Kohn-Sham potential v_s. Explicitly this is given by

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

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:

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

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.

REVISION HISTORY:

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

7.34 forcek (Source File: forcek.f90)

INTERFACE:

```
subroutine forcek(ik)
```

USES:

```
use modmain
use modomp
```

INPUT/OUTPUT PARAMETERS:

```
ik : reduced k-point number (in,integer)
```

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)
```

7.35 fsmbfield (Source File: fsmbfield.f90)

INTERFACE:

```
subroutine fsmbfield
```

USES:

```
use modmain
```

DESCRIPTION:

Updates the effective magnetic field, $\mathbf{B}_{\rm FSM}$, required for fixing the spin moment to a given value, $\mathbf{M}_{\rm FSM}$. 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:

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

where λ is a scaling factor.

REVISION HISTORY:

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

7.36 fsmooth (Source File: fsmooth.f90)

INTERFACE:

```
pure subroutine fsmooth(m,n,f)
```

INPUT/OUTPUT PARAMETERS:

```
m  : number of 3-point running averages to perform (in,integer)
n  : number of point (in,integer)
f  : function array (inout,real(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)
```

7.37 fyukawa0 (Source File: fyukawa0.f90)

INTERFACE:

```
real(8) function fyukawa0(is,l1,k)
```

USES:

```
use modmain
use moddftu
```

INPUT/OUTPUT PARAMETERS:

```
is : species type (in,integer)
l  : an angular momentum (in,integer)
k  : order of Slater parameter (in,integer)
```

DESCRIPTION:

Calculates the Slater parameters in the unscreened case. See *Phys. Rev. B* **52**, 1421 (1995) and *Phys. Rev. B* **80**, 035121 (2009).

REVISION HISTORY:

```
Created April 2008 (LN)
Modified and tested July 2008 (LN and FC)
```

7.38 fyukawa (Source File: fyukawa.f90)

INTERFACE:

```
real(8) function fyukawa(is,l,k,lambda)
```

USES:

```
use modmain
use moddftu
```

INPUT/OUTPUT PARAMETERS:

```
is     : species type (in,integer)
l      : an angular momentum (in,integer)
k      : order of Slater parameter (in,integer)
lambda : screening length of Yukawa potential (in,real)
```

DESCRIPTION:

Calculates the Slater parameters using a screened Yukawa potential. See *Phys. Rev. B* **52**, 1421 (1995) and *Phys. Rev. B* **80**, 035121 (2009).

REVISION HISTORY:

```
Created April 2008 (Lars Nordstrom)
Modified and tested July 2008 (LN and FC)
```

7.39 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)
```

7.40 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)
```

7.41 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)
```

7.42 genafieldt (Source File: genafieldt.f90)

INTERFACE:

```
subroutine genafieldt
```

USES:

```
use modmain
use modtddft
```

DESCRIPTION:

Generates a time-dependent vector potential, $\mathbf{A}(t)$, representing a laser pulse and stores it in `AFIELDT.OUT`. The vector potential is constructed from a sum of sinusoidal waves, each modulated with a Gaussian envelope function:

$$\mathbf{A}(t)=\mathbf{A}_0\frac{e^{-(t-t_0)^2/2\sigma^2}}{\sigma\sqrt{2\pi}}\sin(\omega(t-t_0)+\phi).$$

Seven real numbers have to be specified for each pulse, namely the vector amplitude $\mathbf{A}_0$, peak time t_0, full-width at half-maximum $d=2\sqrt{2\ln 2}\sigma$, frequency ω and phase ϕ.

REVISION HISTORY:

```
Created May 2012 (K. Krieger)
Modified, January 2014 (S. Sharma)
Modified, February 2014 (JKD)
```

7.43 genapwfr (Source File: genapwfr.f90)

INTERFACE:

```
subroutine genapwfr
```

USES:

```
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 Kohn-Sham 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)
Copied to equivalent atoms, February 2010 (A. Kozhevnikov and JKD)
```

7.44 gencfun (Source File: gencfun.f90)

INTERFACE:

```
subroutine gencfun
```

USES:

```
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)
```

7.45 gencore (Source File: gencore.f90)

INTERFACE:

```
subroutine gencore
```

USES:

```
use modmain
use modomp
```

DESCRIPTION:

Computes the core radial wavefunctions, eigenvalues and densities. The radial Dirac equation is solved in the spherical part of the Kohn-Sham potential to which the atomic potential has been appended for $r > R_{\rm MT}$. In the case of spin-polarised calculations, and when `spincore` is set to `.true.`, the Dirac equation is solved in the spin-up and -down potentials created from the Kohn-Sham scalar potential and magnetic field magnitude, with the occupancy divided equally between up and down. The up and down densities determined in this way are added to both the scalar density and the magnetisation in the routine `rhocore`. Note that this procedure is a simple, but inexact, approach to solving the radial Dirac equation in a magnetic field.

REVISION HISTORY:

```
Created April 2003 (JKD)
Added polarised cores, November 2009 (JKD)
```

7.46 genfdu (Source File: genfdu.f90)

INTERFACE:

```
subroutine genfdu(i,u,j,f)
```

USES:

```
use moddftu
use modmpi
```

INPUT/OUTPUT PARAMETERS:

```
i : DFT+U entry (in,integer)
u : parameter U (inout,real)
j : parameter J (inout,real)
f : Slater parameters (inout,real)
```

DESCRIPTION:

Calculate the Slater parameters for DFT+U calculation with different approaches, see *Phys. Rev. B* **80**, 035121 (2009). The relations among Slater and Racah parameters are from E. U. Condon and G. H. Shortley, *The Theory of Atomic Spectra*, The University Press, Cambridge (1935).

REVISION HISTORY:

```
Created July 2008 (Francesco Cricchio)
```

7.47 genfdufr (Source File: genfdufr.f90)

INTERFACE:

```
subroutine genfdufr
```

USES:

```
use modmain
use moddftu
```

DESCRIPTION:

Generates the radial functions used to calculate the Slater integrals through a Yukawa potential.

REVISION HISTORY:

```
Created April 2008 from genapwfr (Francesco Cricchio)
```

7.48 gengclq (Source File: gengclq.f90)

INTERFACE:

```
subroutine gengclq
```

USES:

```
use modmain
use modtest
```

DESCRIPTION:

The Fock matrix elements

$$V_{ij\mathbf{k}} \equiv \sum_{l\mathbf{k}'} \int \frac{\Psi^{\dagger}_{i\mathbf{k}}(\mathbf{r}) \cdot \Psi_{l\mathbf{k}'}(\mathbf{r}) \Psi^{\dagger}_{l\mathbf{k}'}(\mathbf{r}') \cdot \Psi_{j\mathbf{k}}(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} \, d^3r \, d^3r'$$

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, N_q, is very slow. This routine computes the regularised Coulomb Green's function

$$g(\mathbf{q}_i) = \frac{4\pi}{V} \int_{V_i} \frac{1}{q^2} \, d^3q, \tag{1}$$

where the integral is over the small parallelepiped with volume $V = \Omega_{\rm BZ}/N_q$ and centered on the discrete point $\mathbf{q}_i$. This dramatically increases the rate of convergence of methods which involve a summation over the $1/q^2$ part of the Coulomb interaction. The above integral is evaluated numerically on increasingly finer grids and then extrapolated to the continuum.

REVISION HISTORY:

```
Created August 2004 (JKD,SS)
Changed from genwiq2, July 2017 (JKD)
```

7.49 gengkvec (Source File: gengkvec.f90)

INTERFACE:

```
pure subroutine gengkvec(ngv,ivg,vgc,vkl,vkc,gkmax,ngkmax,ngk,igkig,vgkl,vgkc, &
 gkc)
```

INPUT/OUTPUT PARAMETERS:

```
ngv    : number of G-vectors (in,integer)
ivg    : G-vector integer coordinates (in,integer(3,ngv))
vgc    : G-vectors in Cartesian coordinates (in,real(3,ngv))
vkl    : k-point vector in lattice coordinates (in,real(3))
vkc    : k-point vector in Cartesian coordinates (in,real(3))
gkmax  : G+k-vector cut-off (in,real)
ngkmax : maximum number of G+k-vectors (in,integer)
ngk    : number of G+k-vectors returned (out,integer)
igkig  : index from G+k-vectors to G-vectors (out,integer(ngkmax))
vgkl   : G+k-vectors in lattice coordinates (out,real(3,ngkmax))
vgkc   : G+k-vectors in Cartesian coordinates (out,real(3,ngkmax))
gkc    : length of G+k-vectors (out,real(ngkmax))
```

DESCRIPTION:

Generates a set of $\mathbf{G}+\mathbf{k}$-vectors for the input k-point with length less than `gkmax`.

REVISION HISTORY:

```
Created April 2003 (JKD)
Removed spherical coordinate generation, May 2010 (JKD)
Removed modmain and added arguments, September 2012 (JKD)
```

7.50 gengvec (Source File: gengvec.f90)

INTERFACE:

```
subroutine gengvec
```

USES:

```
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)
```

7.51 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)
```

7.52 genjlgprmt (Source File: genjlgprmt.f90)

INTERFACE:

```
subroutine genjlgprmt(lmax,ngp,gpc,ld,jlgprmt)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
lmax    : angular momentum cut-off (in,integer)
ngp     : number of G+p-vectors (in,integer)
gpc     : length of G+p-vectors (in,real(ngkmax))
ld      : leading dimension (in,integer)
jlgprmt : spherical Bessel functions (out,real(0:lmax,ld,nspecies))
```

DESCRIPTION:

Calculates and stores the spherical Bessel functions $j_l(|\mathbf{G}+\mathbf{p}|\mathbf{R}_{\rm MT})$ for all input $\mathbf{G}+\mathbf{p}$ vectors and the muffin-tin radii $\mathbf{R}_{\rm MT}$ of every atomic species.

REVISION HISTORY:

```
Created April 2002 (JKD)
```

7.53 genkmat (Source File: genkmat.f90)

INTERFACE:

```
subroutine genkmat(tfv,tvclcr)
```

USES:

```
use modmain
use modmpi
use modomp
```

INPUT/OUTPUT PARAMETERS:

```
tfv    : .true. if the matrix elements are to be expressed in the
         first-variational basis; second-variational otherwise (in,logical)
tvclvr : .true. if the non-local Coulomb potential from the core states is
         to be included in the kinetic matrix elements (in,logical)
```

DESCRIPTION:

Computes the kinetic matrix elements in the first- or second-variational basis and stores them in the file `KMAT.OUT`. See routine `putkmat`.

REVISION HISTORY:

```
Created January 2007 (JKD)
```

7.54 genlofr (Source File: genlofr.f90)

INTERFACE:

```
subroutine genlofr
```

USES:

```
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 Kohn-Sham 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)
Copied to equivalent atoms, February 2010 (A. Kozhevnikov and JKD)
```

7.55 genpmatk (Source File: genpmatk.f90)

INTERFACE:

```
subroutine genpmatk(ngp,igpig,vgpc,wfmt,wfgp,pmat)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
ngp   : number of G+p-vectors (in,integer(nspnfv))
igpig : index from G+p-vectors to G-vectors (in,integer(ngkmax,nspnfv))
vgpc  : G+p-vectors in Cartesian coordinates (in,real(3,ngkmax,nspnfv))
wfmt  : muffin-tin wavefunction in spherical harmonics
        (in,complex(npcmtmax,natmtot,nspinor,nstsv))
wfgp  : interstitial wavefunction in plane wave basis
        (in,complex(ngkmax,nspinor,nstsv))
pmat  : momentum matrix elements (out,complex(nstsv,nstsv,3))
```

DESCRIPTION:

Calculates the momentum matrix elements

$$P_{ij} = \int d^3r\, \Psi^*_{i\mathbf{k}}(\mathbf{r}) \left(-i\nabla + \frac{1}{4c^2} [\vec{\sigma} \times \nabla V_s(\mathbf{r})] \right) \Psi_{j\mathbf{k}}(\mathbf{r}),$$

where V_s is the Kohn-Sham effective potential. The second term in the brackets is only calculated if spin-orbit coupling is enabled. See Rathgen and Katsnelson, *Physica Scripta* **T109**, 170 (2004).

REVISION HISTORY:

```
Created November 2003 (Sharma)
Fixed bug found by Juergen Spitaler, September 2006 (JKD)
Added spin-orbit correction, July 2010 (JKD)
Fixed bug found by Koichi Kitahara, January 2014 (JKD)
```

7.56 genppts (Source File: genppts.f90)

INTERFACE:

```
subroutine genppts(tfbz,nsym,sym,ngridp,npptnr,epslat,bvec,boxl,nppt,ipvip, &
 ipvipnr,ivp,vpl,vpc,wppt,wpptnr)
```

INPUT/OUTPUT PARAMETERS:

```
tfbz    : .true. if vpl and vpc should be mapped to the first Brillouin zone
          (in,logical)
nsym    : number of point group symmetries used for reduction, set to 1 for
          no reduction (in,integer)
sym     : symmetry matrices in lattice coordinates (in,integer(3,3,*))
ngridp  : p-point grid sizes (in,integer(3))
npptnr  : number of non-reduced p-points: ngridp(1)*ngridp(2)*ngridp(3)
          (in,integer)
epslat  : tolerance for determining identical vectors (in,real)
bvec    : reciprocal lattice vectors (in,real(3,3))
boxl    : corners of box containing p-points in lattice coordinates, the
          zeroth vector is the origin (in,real(3,0:3))
nppt    : total number of p-points (out,integer)
ipvip   : map from (i1,i2,i3) to reduced p-point index
          (out,integer(0:ngridp(1)-1,0:ngridp(2)-1,0:ngridp(3)-1))
ipvipnr : map from (i1,i2,i3) to non-reduced 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 reduced p-point
          (out,real(ngridp(1)*ngridp(2)*ngridp(3)))
wpptnr  : weight of each non-reduced p-point (out,real)
```

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. 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 each `vpl` vector is mapped to the first Brillouin zone. If `tfbz` is `.false.` and then the coordinates of each `vpl` are mapped to the $[0, 1)$ interval. The p-point weights are stored in `wppt` and the array `ipvip` contains the map from the integer coordinates to the reduced index.

REVISION HISTORY:

```
Created August 2002 (JKD)
Updated April 2007 (JKD)
Added mapping to the first Brillouin zone, September 2008 (JKD)
Made independent of modmain, February 2010 (JKD)
```

7.57 genrlmv (Source File: genrlmv.f90)

INTERFACE:

```
subroutine genrlmv(lmax,v,rlm)
```

INPUT/OUTPUT PARAMETERS:

```
lmax : maximum angular momentum (in,integer)
v    : input vector (in,real(3))
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 `genylmv`.

REVISION HISTORY:

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

7.58 genrmesh (Source File: genrmesh.f90)

INTERFACE:

```
subroutine genrmesh
```

USES:

```
use modmain
use modvars
```

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`.

REVISION HISTORY:

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

7.59 gensdmat (Source File: gensdmat.f90)

INTERFACE:

```
pure 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)
```

7.60 gensfacgp (Source File: gensfacgp.f90)

INTERFACE:

```
pure 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)
```

7.61 genshtmat (Source File: genshtmat.f90)

INTERFACE:

```
subroutine genshtmat
```

USES:

```
use modmain
```

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)
```

7.62 genspchi0 (Source File: genspchi0.f90)

INTERFACE:

```
subroutine genspchi0(ik,lock,ssr,vqpl,jlgqr,ylmgq,sfacgq,chi0)
```

USES:

```
use modmain
use modomp
```

INPUT/OUTPUT PARAMETERS:

```
ik     : k-point from non-reduced set (in,integer)
lock   : OpenMP locks for frequency index of chi0 (in,integer(nwrf))
ssr    : scissor correction (in,real)
vqpl   : input q-point in lattice coordinates (in,real(3))
jlgqr  : spherical Bessel functions evaluated on the coarse radial mesh for
         all species and G+q-vectors (in,real(njcmax,nspecies,ngrf))
ylmgq  : spherical harmonics of the G+q-vectors (in,complex(lmmaxo,ngrf))
sfacgq : structure factors of G+q-vectors (in,complex(ngrf,natmtot))
chi0   : spin-dependent Kohn-Sham response function in G-space
         (out,complex(ngrf,4,ngrf,4,nwrf))
```

DESCRIPTION:

Computes the spin-dependent Kohn-Sham response function:

$$\chi_{\alpha\beta,\alpha'\beta'}(\mathbf{r},\mathbf{r}',\omega) \equiv \frac{\delta\rho_{\alpha\beta}(\mathbf{r},\omega)}{\delta v_{\alpha'\beta'}(\mathbf{r}',\omega)}$$
$$= \frac{1}{N_k}\sum_{i\mathbf{k},j\mathbf{k}'}(f_{i\mathbf{k}}-f_{j\mathbf{k}'})\frac{\langle i\mathbf{k}|\hat{\rho}_{\beta\alpha}(\mathbf{r})|j\mathbf{k}'\rangle\langle j\mathbf{k}'|\hat{\rho}_{\alpha'\beta'}(\mathbf{r}')|i\mathbf{k}\rangle}{w+(\varepsilon_{i\mathbf{k}}-\varepsilon_{j\mathbf{k}'})+i\eta},$$

where α and β are spin-coordinates, N_k is the number of k-points, $f_{i\mathbf{k}}$ are the occupancies, v is the Kohn-Sham potential and $\hat{\rho}$ is the spin-density operator. With translational symmetry in mind, we adopt the following convention for its Fourier transform:

$$\chi_{\alpha\beta,\alpha'\beta'}(\mathbf{G},\mathbf{G}',\mathbf{q},\omega) = \frac{1}{\Omega}\int d^3r\, d^3r'\, e^{-i(\mathbf{G}+\mathbf{q})\cdot\mathbf{r}} e^{i(\mathbf{G}'+\mathbf{q})\cdot\mathbf{r}'}\chi_{\alpha\beta,\alpha'\beta'}(\mathbf{r},\mathbf{r}',\omega).$$

Let

$$Z^{\alpha\beta}_{i\mathbf{k},j\mathbf{k}+\mathbf{q}}(\mathbf{G}) \equiv \int d^3r\, e^{i(\mathbf{G}+\mathbf{q})\cdot\mathbf{r}}\varphi^*_{j\mathbf{k}+\mathbf{q},\alpha}(\mathbf{r})\varphi_{i\mathbf{k},\beta}(\mathbf{r})$$

then the response function in G-space can be written

$$\chi_{\alpha\beta,\alpha'\beta'}(\mathbf{G},\mathbf{G}',\mathbf{q},\omega) = \frac{1}{N_k\Omega}\sum_{i\mathbf{k},j\mathbf{k}+\mathbf{q}}(f_{i\mathbf{k}}-f_{j\mathbf{k}})\frac{\left[Z^{\alpha\beta}_{i\mathbf{k},j\mathbf{k}+\mathbf{q}}(\mathbf{G})\right]^* Z^{\alpha'\beta'}_{i\mathbf{k},j\mathbf{k}+\mathbf{q}}(\mathbf{G}')}{w+(\varepsilon_{i\mathbf{k}}-\varepsilon_{j\mathbf{k}+\mathbf{q}})+i\eta}.$$

REVISION HISTORY:

```
Created March 2012 (SS and JKD)
```

7.63 genvclijji (Source File: genvclijji.f90)

INTERFACE:

```
subroutine genvclijji(ikp,vclijji)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
ikp     : k-point from non-reduced set (in,integer)
vclijji : Coulomb matrix elements (out,real(nstsv,nstsv,nkpt))
```

DESCRIPTION:

Calculates the Coulomb matrix elements of the type $(i - jj - i)$.

REVISION HISTORY:

```
Created June 2008 (Sharma)
```

7.64 genvclijjk (Source File: genvclijjk.f90)

INTERFACE:

```
subroutine genvclijjk(ikp,vclijjk)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
ikp     : k-point from non-reduced set (in,integer)
vclijjk : Coulomb matrix elements (out,complex(nstsv,nstsv,nstsv,nkpt))
```

DESCRIPTION:

Calculates Coulomb matrix elements of the type $(i - jj - k)$.

REVISION HISTORY:

```
Created 2008 (Sharma)
```

7.65 genveedu (Source File: genveedu.f90)

INTERFACE:

```
subroutine genveedu(i,u,j,vee)
```

USES:

```
use modmain
use moddftu
```

INPUT/OUTPUT PARAMETERS:

```
i   : DFT+U entry (in,integer)
u   : parameter U (out,real)
j   : parameter J (out,real)
vee : Coulomb matrix elements (out,real(-lmaxdm:lmaxdm,-lmaxdm:lmaxdm,
       -lmaxdm:lmaxdm,-lmaxdm:lmaxdm))
```

DESCRIPTION:

Calculates the Coulomb matrix elements used in DFT+U calculations. See *Phys. Rev. B* **52**, 5467 (1995).

REVISION HISTORY:

```
Created November 2007 (FC,JKD,FB,LN)
Modified July 2009 (FC)
```

7.66 genvmatmt (Source File: genvmatmt.f90)

INTERFACE:

```
subroutine genvmatmt
```

USES:

```
use modmain
use moddftu
```

DESCRIPTION:

Calculate the DFT+U potential matrix to be used in the second-variational step. See *Phys. Rev. B* **52**, 5467 (1995) and *Phys. Rev. B* **80**, 035121 (2009).

REVISION HISTORY:

```
Created November 2007 (FC,FB,LN,JKD)
Fixed bug for dftu=3, January 2021 (JKD)
```

7.67 genvsig (Source File: genvsig.f90)

INTERFACE:

```
subroutine genvsig
```

USES:

```
use modmain
```

DESCRIPTION:

Generates the Fourier transform of the Kohn-Sham effective potential in the interstitial 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)

7.68 genwfsv (Source File: genwfsv.f90)

INTERFACE:

```
subroutine genwfsv(tsh,tgp,nst,idx,ngdg,igf,ngp,igpig,apwalm,evecfv,evecsv, &
 wfmt,ld,wfir)
```

USES:

```
use modmain
use modomp
```

INPUT/OUTPUT PARAMETERS:

```
tsh    : .true. if wfmt should be in spherical harmonic basis (in,logical)
tgp    : .true. if wfir should be in G+p-space, otherwise in real-space
         (in,logical)
nst    : number of states to be calculated (in,integer)
idx    : index to states which are to be calculated (in,integer(nst))
ngdg   : G-vector grid sizes (in,integer(3))
igf    : map from G-vector index to FFT array (in,integer(*))
ngp    : number of G+p-vectors (in,integer(nspnfv))
igpig  : index from G+p-vectors to G-vectors (in,integer(ngkmax,nspnfv))
apwalm : APW matching coefficients
         (in,complex(ngkmax,apwordmax,lmmaxapw,natmtot,nspnfv))
evecfv : first-variational eigenvectors (in,complex(nmatmax,nstfv,nspnfv))
evecsv : second-variational eigenvectors (in,complex(nstsv,nstsv))
wfmt   : muffin-tin part of the wavefunctions for every state in spherical
         coordinates (out,complex(npcmtmax,natmtot,nspinor,nst))
ld     : leading dimension of wfir (in,integer)
wfir   : interstitial part of the wavefunctions for every state
         (out,complex(ld,nspinor,nst))
```

DESCRIPTION:

Calculates the second-variational spinor wavefunctions in both the muffin-tin and interstitial regions for every state of a particular k-point. A coarse radial mesh is assumed in the muffin-tins with angular momentum cut-off of `lmaxo`.

REVISION HISTORY:

```
Created November 2004 (Sharma)
Updated for spin-spirals, June 2010 (JKD)
Packed muffin-tins, April 2016 (JKD)
```

7.69 genylmg (Source File: genylmg.f90)

INTERFACE:

```
subroutine genylmg
```

USES:

```
use modmain
```

DESCRIPTION:

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

REVISION HISTORY:

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

7.70 genylmv (Source File: genylmv.f90)

INTERFACE:

```
pure subroutine genylmv(lmax,v,ylm)
```

INPUT/OUTPUT PARAMETERS:

```
lmax : maximum angular momentum (in,integer)
v    : input vector (in,real(3))
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 $j = l(l+1) + m + 1$. 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)
Changed algorithm, June 2019 (JKD)
```

7.71 genzrho (Source File: genzrho.f90)

INTERFACE:

```
subroutine genzrho(tsh,tspc,ngt,wfmt1,wfir1,wfmt2,wfir2,zrhomt,zrhoir)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
tsh    : .true. if the muffin-tin density is to be in spherical harmonics
         (in,logical)
tspc   : .true. if the density should be contracted over spin (in,logical)
ngt    : total number of interstitial grid points (in,integer)
wfmt1  : muffin-tin part of wavefunction 1 in spherical coordinates
         (in,complex(npcmtmax,natmtot,*))
wfir1  : interstitial wavefunction 1 (in,complex(ngt,*))
wfmt2  : muffin-tin part of wavefunction 2 in spherical coordinates
         (in,complex(npcmtmax,natmtot,*))
wfir2  : interstitial wavefunction 2 (in,complex(ngt,*))
zrhomt : muffin-tin charge density in spherical harmonics/coordinates
         (out,complex(npcmtmax,natmtot))
zrhoir : interstitial charge density (out,complex(ngt))
```

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.

REVISION HISTORY:

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

7.72 getevecfv (Source File: getevecfv.f90)

INTERFACE:

```
subroutine getevecfv(fext,ikp,vpl,vgpl,evecfv)
```

USES:

```
use modmain
use modramdisk
```

INPUT/OUTPUT PARAMETERS:

```
fext   : filename extension (in,character(*))
ikp    : p-point vector index (in,integer)
vpl    : p-point vector in lattice coordinates (in,real(3))
vgpl   : G+p-vectors in lattice coordinates (out,real(3,ngkmax,nspnfv))
evecfv : first-variational eigenvectors (out,complex(nmatmax,nstfv,nspnfv))
```

DESCRIPTION:

Reads in a first-variational eigenvector from file. If the input k-point, $\mathbf{p}$, is not in the reduced set, then the eigenvector of the equivalent point is read in and the required rotation/translation operations applied.

REVISION HISTORY:

```
Created Feburary 2007 (JKD)
Fixed transformation error, October 2007 (JKD, Anton Kozhevnikov)
Fixed l.o. rotation, June 2010 (A. Kozhevnikov)
```

7.73 getvclijji (Source File: getvclijji.f90)

INTERFACE:

```
subroutine getvclijji(ikp,vclijji)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
ikp     : k-point from non-reduced set (in,integer)
vclijji : Coulomb matrix elements (out,real(nstsv,nstsv,nkpt))
```

DESCRIPTION:

Retrieves Coulomb matrix elements of the type $(i-jj-i)$ from the file `VCLIJJI.OUT`.

REVISION HISTORY:

```
Created 2009 (Sharma)
```

7.74 getvclijjk (Source File: getvclijjk.f90)

INTERFACE:

```
subroutine getvclijjk(ikp,vclijjk)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
ikp     : k-point from non-reduced set (in,integer)
vclijjk : Coulomb matrix elements (out,complex(nstsv,nstsv,nstsv,nkpt))
```

DESCRIPTION:

Retrieves Coulomb matrix elements of the type $(i-jj-k)$ from the file `VCLIJJK.OUT`.

REVISION HISTORY:

```
Created 2009 (Sharma)
```

7.75 ggair_1 (Source File: ggair_1.f90)

INTERFACE:

```
subroutine ggair_1(rho,grho,g2rho,g3rho)
```

USES:

```
use modmain
```

DESCRIPTION:

Spin-unpolarised version of `ggair_sp_1`.

REVISION HISTORY:

```
Created November 2009 (JKD)
```

7.76 ggair_2a (Source File: ggair_2a.f90)

INTERFACE:

```
subroutine ggair_2a(rho,g2rho,gvrho,grho2)
```

USES:

```
use modmain
```

DESCRIPTION:

Spin-unpolarised version of ggair_sp_2a.

REVISION HISTORY:

```
Created November 2009 (JKD and TMcQ)
```

7.77 ggair_2b (Source File: ggair_2b.f90)

INTERFACE:

```
subroutine ggair_2b(g2rho,gvrho,vx,vc,dxdgr2,dcdgr2)
```

USES:

```
use modmain
```

DESCRIPTION:

Spin-unpolarised version of ggair_sp_2b.

REVISION HISTORY:

```
Created November 2009 (JKD and TMcQ)
```

7.78 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:

```
rhoup : spin-up density (in,real(ngtot))
rhodn : spin-down density (in,real(ngtot))
grho  : |grad rho| (out,real(ngtot))
gup   : |grad rhoup| (out,real(ngtot))
gdn   : |grad rhodn| (out,real(ngtot))
g2up  : grad^2 rhoup (out,real(ngtot))
g2dn  : grad^2 rhodn (out,real(ngtot))
g3rho : (grad rho).(grad |grad rho|) (out,real(ngtot))
g3up  : (grad rhoup).(grad |grad rhoup|) (out,real(ngtot))
g3dn  : (grad rhodn).(grad |grad rhodn|) (out,real(ngtot))
```

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 `potxc` and `modxcifc`.

REVISION HISTORY:

```
Created October 2004 (JKD)
Simplified and improved, October 2009 (JKD)
```

7.79 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 modmain
```

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 and meta-GGA. See `ggamt_sp_2a` for details.

REVISION HISTORY:

```
Created November 2009 (JKD and TMcQ)
```

7.80 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)
```

USES:

```
use modmain
```

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)
```

7.81 ggamt_1 (Source File: ggamt_1.f90)

INTERFACE:

```
subroutine ggamt_1(tsh,is,np,rho,grho,g2rho,g3rho)
```

USES:

```
use modmain
```

DESCRIPTION:

Spin-unpolarised version of ggamt_sp_1.

REVISION HISTORY:

Created November 2009 (JKD)

7.82 ggamt_2a (Source File: ggamt_2a.f90)

INTERFACE:

```
subroutine ggamt_2a(tsh,is,np,rho,g2rho,gvrho,grho2)
```

USES:

```
use modmain
```

DESCRIPTION:

Spin-unpolarised version of ggamt_sp_2a.

REVISION HISTORY:

Created November 2009 (JKD and TMcQ)

7.83 ggamt_2b (Source File: ggamt_2b.f90)

INTERFACE:

```
subroutine ggamt_2b(is,np,g2rho,gvrho,vx,vc,dxdgr2,dcdgr2)
```

USES:

```
use modmain
```

DESCRIPTION:

Spin-unpolarised version of ggamt_sp_2b.

REVISION HISTORY:

Created November 2009 (JKD and TMcQ)

7.84 ggamt_sp_1 (Source File: ggamt_sp_1.f90)

INTERFACE:

```
subroutine ggamt_sp_1(is,np,rhoup,rhodn,grho,gup,gdn,g2up,g2dn,g3rho,g3up,g3dn)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
is    : species number (in,integer)
np    : number of muffin-tin points (in,integer)
rhoup : spin-up density in spherical coordinates (in,real(np))
rhodn : spin-down density (in,real(np))
grho  : |grad rho| (out,real(np))
gup   : |grad rhoup| (out,real(np))
gdn   : |grad rhodn| (out,real(np))
g2up  : grad^2 rhoup (out,real(np))
g2dn  : grad^2 rhodn (out,real(np))
g3rho : (grad rho).(grad |grad rho|) (out,real(np))
g3up  : (grad rhoup).(grad |grad rhoup|) (out,real(np))
g3dn  : (grad rhodn).(grad |grad rhodn|) (out,real(np))
```

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)
```

7.85 ggamt_sp_2a (Source File: ggamt_sp_2a.f90)

INTERFACE:

```
subroutine ggamt_sp_2a(is,np,rhoup,rhodn,g2up,g2dn,gvup,gvdn,gup2,gdn2,gupdn)
```

USES:

```
use modmain
```

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

$$\begin{aligned}V_{xc}^\uparrow(\mathbf{r})=&\frac{\partial\hat{\epsilon}_{xc}}{\partial\rho^\uparrow(\mathbf{r})}-2\left(\nabla\frac{\partial\hat{\epsilon}_{xc}}{\partial(\nabla\rho^\uparrow)^2}\right)\cdot\nabla\rho^\uparrow-2\frac{\hat{\epsilon}_{xc}}{\partial(\nabla\rho^\uparrow)^2}\nabla^2\rho^\uparrow\\&-\left(\nabla\frac{\hat{\epsilon}_{xc}}{\partial(\nabla\rho^\uparrow\cdot\nabla\rho^\downarrow)}\right)\cdot\nabla\rho^\downarrow-\frac{\partial\hat{\epsilon}_{xc}}{\partial(\nabla\rho^\uparrow\cdot\nabla\rho^\downarrow)}\nabla^2\rho^\downarrow,\end{aligned}$$

and similarly for $V_{xc}^\downarrow$.

REVISION HISTORY:

Created November 2009 (JKD and TMcQ)

7.86 ggamt_sp_2b (Source File: ggamt_sp_2b.f90)

INTERFACE:

```
subroutine ggamt_sp_2b(is,np,g2up,g2dn,gvup,gvdn,vxup,vxdn,vcup,vcdn,dxdgu2, &
 dxdgd2,dxdgud,dcdgu2,dcdgd2,dcdgud)
```

USES:

```
use modmain
```

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)

7.87 gndstate (Source File: gndstate.f90)

INTERFACE:

```
subroutine gndstate
```

USES:

```
use modmain
use moddftu
use modulr
use modmpi
use modomp
```

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`. The density, magnetisation, Kohn-Sham potential and magnetic field are written to `STATE.OUT`.

REVISION HISTORY:

```
Created October 2002 (JKD)
Added MPI, August 2010 (JKD)
```

7.88 gradrfmt (Source File: gradrfmt.f90)

INTERFACE:

```
subroutine gradrfmt(nr,nri,ri,wcr,rfmt,ld,grfmt)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
nr    : number of radial mesh points (in,integer)
nri   : number of points on inner part of muffin-tin (in,integer)
ri    : 1/r on the radial mesh (in,real(nr))
wcr   : weights for spline coefficients on radial mesh (in,real(12,nr))
rfmt  : real muffin-tin function (in,real(*))
ld    : leading dimension (in,integer)
grfmt : gradient of rfmt (out,real(ld,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)
```

7.89 gradzfmt (Source File: gradzfmt.f90)

INTERFACE:

```
subroutine gradzfmt(nr,nri,ri,wcr,zfmt,ld,gzfmt)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
nr    : number of radial mesh points (in,integer)
nri   : number of points on inner part of muffin-tin (in,integer)
ri    : 1/r on the radial mesh (in,real(nr))
wcr   : weights for spline coefficients on radial mesh (in,real(12,nr))
zfmt  : complex muffin-tin function (in,complex(*))
ld    : leading dimension (in,integer)
gzfmt : gradient of zfmt (out,complex(ld,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 gradient formula (see, for example, V. Devanathan, *Angular Momentum Techniques In Quantum Mechanics*)

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

where the vector spherical harmonics are determined from Clebsch-Gordan coefficients as follows:

$$\mathbf{Y}_{lm}^{l'}(\hat{\mathbf{r}}) = \sum_{m'\mu} \begin{bmatrix} l' & 1 & l \\ m' & \mu & m \end{bmatrix} Y_{lm}(\hat{\mathbf{r}})\hat{\mathbf{e}}^{\mu}$$

and the (contravariant) spherical unit vectors are given by

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

REVISION HISTORY:

```
Rewritten May 2009 (JKD)
Modified, February 2020 (JKD)
```

7.90 gridsize (Source File: gridsize.f90)

INTERFACE:

```
subroutine gridsize(avec,gmaxvr,ngridg,ngtot,intgv)
```

INPUT/OUTPUT PARAMETERS:

```
avec   : lattice vectors (in,real(3,3))
gmaxvr : G-vector cut-off (in,real)
ngridg : G-vector grid sizes (out,integer(3))
ngtot  : total number of G-vectors (out,integer)
intgv  : integer grid intervals for each direction (out,integer(2,3))
```

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)
Removed modmain and added arguments, September 2012 (JKD)
```

7.91 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)
```

7.92 hmlaa (Source File: hmlaa.f90)

INTERFACE:

```
subroutine hmlaa(thr,is,ias,ngp,apwalm,ld,h)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
thr    : .true. if the matrix h is real valued (in,logical)
is     : species number (in,integer)
ias    : joint atom and species number (in,integer)
ngp    : number of G+p-vectors (in,integer)
apwalm : APW matching coefficients (in,complex(ngkmax,apwordmax,lmmaxapw))
ld     : leading dimension of h (in,integer)
h      : Hamiltonian matrix (inout,complex(*))
```

DESCRIPTION:

Calculates the APW-APW contribution to the Hamiltonian matrix.

REVISION HISTORY:

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

7.93 hmlistl (Source File: hmlistl.f90)

INTERFACE:

```
pure subroutine hmlistl(ngp,igpig,vgpc,ld,h)
```

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))
vgpc  : G+p-vectors in Cartesian coordinates (in,real(3,ngkmax))
ld    : leading dimension of h (in,integer)
h     : Hamiltonian matrix (inout,complex(*))
```

DESCRIPTION:

Computes the interstitial contribution to the Hamiltonian matrix for the APW basis functions. The Hamiltonian is given by

$$H^{\rm I}(\mathbf{G}+\mathbf{k},\mathbf{G}'+\mathbf{k})=\frac{1}{2}(\mathbf{G}+\mathbf{k})\cdot(\mathbf{G}'+\mathbf{k})\tilde{\Theta}(\mathbf{G}-\mathbf{G}')+V_s(\mathbf{G}-\mathbf{G}'),$$

where V_s is the interstitial Kohn-Sham potential and $\tilde{\Theta}$ is the characteristic function. See routine `gencfun`.

REVISION HISTORY:

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

7.94 hmlrad (Source File: hmlrad.f90)

INTERFACE:

```
subroutine hmlrad
```

USES:

```
use modmain
use modomp
```

DESCRIPTION:

Calculates the radial Hamiltonian integrals of the APW and local-orbital basis functions. In other words, for atom α, it computes integrals of the form

$$h^{\alpha}_{qq';ll'l''m''}=\begin{cases}\int_0^{R_i}u^{\alpha}_{q;l}(r)Hu^{\alpha}_{q';l'}(r)r^2dr & l''=0\\ \int_0^{R_i}u^{\alpha}_{q;l}(r)V^{\alpha}_{l''m''}(r)u^{\alpha}_{q';l'}(r)r^2dr & l''>0\end{cases},$$

where $u^{\alpha}_{q;l}$ is the qth APW radial function for angular momentum l; H is the Hamiltonian of the radial Schrödinger equation; and $V^{\alpha}_{l''m''}$ is the muffin-tin Kohn-Sham potential. Similar integrals are calculated for APW-local-orbital and local-orbital-local-orbital contributions.

REVISION HISTORY:

```
Created December 2003 (JKD)
Updated for compressed muffin-tin functions, March 2016 (JKD)
```

7.95 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)
```

7.96 i3mtv (Source File: i3mtv.f90)

INTERFACE:

```
pure 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)
```

7.97 init0 (Source File: init0.f90)

INTERFACE:

```
subroutine init0
```

USES:

```
use modmain
use modxcifc
use moddftu
use modtddft
use modphonon
use modulr
use modtest
use modvars
use modmpi
use modomp
```

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)
```

7.98 init1 (Source File: init1.f90)

INTERFACE:

```
subroutine init1
```

USES:

```
use modmain
use moddftu
use modulr
use modtddft
use modtest
use modvars
use modstore
```

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)
```

7.99 linengy (Source File: linengy.f90)

INTERFACE:

```
subroutine linengy
```

USES:

```
use modmain
use modmpi
```

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)
```

7.100 lopzflm (Source File: lopzflm.f90)

INTERFACE:

```
pure 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 $\hat{\mathbf{L}}$ to a function expanded in terms of complex spherical harmonics. This makes use of the identities

$$(L_x + iL_y)Y_{lm}(\theta,\phi) = \sqrt{(l-m)(l+m+1)}Y_{lm+1}(\theta,\phi)$$
$$(L_x - iL_y)Y_{lm}(\theta,\phi) = \sqrt{(l+m)(l-m+1)}Y_{lm-1}(\theta,\phi)$$
$$L_z Y_{lm}(\theta,\phi) = mY_{lm}(\theta,\phi).$$

REVISION HISTORY:

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

7.101 massnucl (Source File: massnucl.f90)

INTERFACE:

```
elemental real(8) function massnucl(z)
```

INPUT/OUTPUT PARAMETERS:

```
z : atomic number (in,real)
```

DESCRIPTION:

Computes an approximate nuclear mass from the atomic number Z. The nuclear mass number, A, is first estimated using

$$A = 4.467 \times 10^{-3} Z^2 + 2.163 Z - 1.168,$$

[D. Andrae in *Relativistic Electronic Structure Theory - Fundamentals* **11**, 203 (2002)]. Then the nuclear mass can be determined from:

$$M = Z m_p + N m_n - \frac{B}{c^2},$$

where m_p is the proton mass, m_n is the neutron mass and B is the nuclear binding energy. The binding energy is approximated by the Weizsäcker formula:

$$B = a_V A - a_S A^{2/3} - a_C Z^2 A^{-1/3} - a_{\rm sym}(Z-N)^2 A^{-1} + B_p + B_{\rm shell}.$$

See F. Yang and J. H. Hamilton in *Modern Atomic and Nuclear Physics*, Revised Edition 2010, for details on the quantities in this formula. In this implementation, B_p and $B_{\rm shell}$ are set to zero.

REVISION HISTORY:

```
Created February 2014 (JKD)
```

7.102 match (Source File: match.f90)

INTERFACE:

```
subroutine match(ngp,vgpc,gpc,sfacgp,apwalm)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
ngp    : number of G+p-vectors (in,integer)
vgpc   : G+p-vectors in Cartesian coordinates (in,real(3,ngkmax))
gpc    : length of G+p-vectors (in,real(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 $(M^{\alpha}_{l}-1)$th derivative across the muffin-tin boundary therefore requires that the matching coefficients satisfy

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

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)
```

7.103 mixadapt (Source File: mixadapt.f90)

INTERFACE:

```
pure subroutine mixadapt(iscl,beta0,betamax,n,nu,mu,beta,f,d)
```

INPUT/OUTPUT PARAMETERS:

```
iscl    : self-consistent loop number (in,integer)
beta0   : mixing parameter (in,real)
betamax : maximum mixing parameter (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 $\beta_j^{i+1} = \beta_j^i + \beta_0$ 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 $\beta_j^{i+1} = (\beta_j^i + \beta_0)/2$. 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 thread-safe so long as each thread has its own independent work arrays. Complex arrays may be passed as real arrays with n doubled.

REVISION HISTORY:

```
Created March 2003 (JKD)
Modified, September 2008 (JKD)
Modified, August 2011 (JKD)
```

7.104 randomu (Source File: modrandom.f90)

INTERFACE:

```
real(8) function randomu()
```

DESCRIPTION:

Generates random numbers with a uniform distribution in the interval $[0, 1]$ using the fifth-order multiple recursive generator of P. L'Ecuyer, F. Blouin, and R. Coutre, *ACM Trans. Modeling Comput. Simulation* **3**, 87 (1993). The sequence of numbers r_i is produced from

$$x_i = (a_1 x_{i-1} + a_5 x_{i-5}) \mod m$$

with $r_i = x_i/m$. The period is about 2^{155}.

REVISION HISTORY:

```
Created January 2012 (JKD)
Changed initial state, April 2017 (JKD)
```

7.105 xcifc (Source File: modxcifc.f90)

INTERFACE:

```
subroutine xcifc(xctype,n,c_tb09,tempa,rho,rhoup,rhodn,grho,gup,gdn,g2rho, &
 g2up,g2dn,g3rho,g3up,g3dn,grho2,gup2,gdn2,gupdn,tau,tauup,taudn,ex,ec,vx,vc, &
 vxup,vxdn,vcup,vcdn,dxdgr2,dxdgu2,dxdgd2,dxdgud,dcdgr2,dcdgu2,dcdgd2,dcdgud, &
 dxdg2r,dxdg2u,dxdg2d,dcdg2r,dcdg2u,dcdg2d,wx,wxup,wxdn,wc,wcup,wcdn)
```

INPUT/OUTPUT PARAMETERS:

```
    xctype : type of exchange-correlation functional (in,integer(3))
    n      : number of density points (in,integer)
    c_tb09 : Tran-Blaha '09 constant c (in,real,optional)
    tempa  : temperature in atomic units (in,real,optional)
    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)
    tau    : kinetic energy density (in,real(n),optional)
    tauup  : spin-up kinetic energy density (in,real(n),optional)
    taudn  : spin-down kinetic energy density (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)
    dxdgr2 : 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)
    dcdgr2 : 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)
    dxdg2r : de_x/d(grad^2 rho) (out,real(n),optional)
    dxdg2u : de_x/d(grad^2 rhoup) (out,real(n),optional)
    dxdg2d : de_x/d(grad^2 rhodn) (out,real(n),optional)
```

```
dcdg2r : de_c/d(grad^2 rho) (out,real(n),optional)
dcdg2u : de_c/d(grad^2 rhoup) (out,real(n),optional)
dcdg2d : de_c/d(grad^2 rhodn) (out,real(n),optional)
wx     : de_x/dtau (out,real(n),optional)
wxup   : de_x/dtauup (out,real(n),optional)
wxdn   : de_x/dtaudn (out,real(n),optional)
wc     : de_c/dtau (out,real(n),optional)
wcup   : de_c/dtauup (out,real(n),optional)
wcdn   : de_c/dtaudn (out,real(n),optional)
```

DESCRIPTION:

Interface to the exchange-correlation routines. In the most general case (meta-GGA), the exchange-correlation energy is given by

$$E_{xc}[\rho^{\uparrow},\rho^{\downarrow}] = \int d^3r\, \rho(\mathbf{r})\, \varepsilon_{xc}(\rho^{\uparrow},\rho^{\downarrow},|\nabla\rho|,|\nabla\rho^{\uparrow}|,|\nabla\rho^{\downarrow}|,\nabla^2\rho^{\uparrow},\nabla^2\rho^{\downarrow},\tau),$$

where $\rho(\mathbf{r}) = \rho^{\uparrow}(\mathbf{r}) + \rho^{\downarrow}(\mathbf{r})$ is the density;

$$\tau(\mathbf{r}) \equiv \sum_{i\ \mathrm{occ}} \nabla\psi(\mathbf{r}) \cdot \nabla\psi(\mathbf{r})$$

is twice the spin-contracted kinetic energy density; and ε_{xc} is the exchange-correlation energy per electron.

REVISION HISTORY:

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

7.106 getxcdata (Source File: modxcifc.f90)

INTERFACE:

```
subroutine getxcdata(xctype,xcdescr,xcspin,xcgrad,hybrid,hybridc)
```

INPUT/OUTPUT PARAMETERS:

```
xctype  : type of exchange-correlation functional (in,integer(3))
xcdescr : description of functional (out,character(512))
xcspin  : spin treatment (out,integer)
xcgrad  : gradient treatment (out,integer)
hybrid  : .true. if functional a hybrid (out,logical)
hybridc : hybrid exact exchange mixing coefficient (out,real(8))
```

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)
```

7.107 moment (Source File: moment.f90)

INTERFACE:

```
subroutine moment
```

USES:

```
use modmain
use modtest
```

DESCRIPTION:

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

REVISION HISTORY:

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

7.108 mossbauer (Source File: mossbauer.f90)

INTERFACE:

```
subroutine mossbauer
```

USES:

```
use modmain
use modmpi
use modtest
```

DESCRIPTION:

Computes the contact charge density and magnetic hyperfine field for each atom and outputs the data to the file `MOSSBAUER.OUT`. See S. Blügel, H. Akai, R. Zeller, and P. H. Dederichs, *Phys. Rev. B* **35**, 3271 (1987).

REVISION HISTORY:

```
Created May 2004 (JKD)
Contact hyperfine field evaluated at the nuclear radius rather than averaged
over the Thomson sphere, June 2019 (JKD)
Added spin and orbital dipole terms, July 2019 (JKD)
```

7.109 mtdmin (Source File: mtdmin.f90)

INTERFACE:

```
pure subroutine mtdmin(is,js,dmin)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
is, js : species numbers (out,integer)
dmin   : minimum distance between muffin-tin surfaces (out,real)
```

DESCRIPTION:

Finds the atomic species pair for which the distance between the muffin-tin surfaces is a minimum. This distance may be negative if the muffin-tins overlap.

REVISION HISTORY:

```
Created October 2011 (JKD)
```

7.110 nfftifc (Source File: nfftifc.f90)

INTERFACE:

```
subroutine nfftifc(np,n)
```

INPUT/OUTPUT PARAMETERS:

```
np : number of allowed primes (in,integer)
n  : required/avalable grid size (inout,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)
```

7.111 nonlinopt (Source File: nonlinopt.f90)

INTERFACE:

```
subroutine nonlinopt
```

USES:

```
use modmain
use modtest
use modomp
```

DESCRIPTION:

Calculates susceptibility tensor for non-linear optical second-harmonic generation (SHG). The terms (ztm) are numbered according to Eqs. (49)-(51) of the article *Physica Scripta* **T109**, 128 (2004). Other good references are *Phys. Rev. B* **48**, 11705 (1993) and *Phys. Rev. B* **53**, 10751 (1996).

REVISION HISTORY:

```
Rewrote earlier version, June 2010 (Sharma)
Improved parallelism, January 2020 (R. Cohen)
```

7.112 occupy (Source File: occupy.f90)

INTERFACE:

```
subroutine occupy
```

USES:

```
use modmain
use modtest
```

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)
Added gap estimation, November 2009 (F. Cricchio)
Added adaptive smearing width, April 2010 (T. Bjorkman)
```

7.113 olpistl (Source File: olpistl.f90)

INTERFACE:

```
pure subroutine olpistl(ngp,igpig,ld,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))
ld    : leading dimension of o (in,integer)
o     : overlap matrix (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)
```

7.114 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 atom α, it computes integrals of the form

$$o^{\alpha}_{qp}=\int_0^{R_i}u^{\alpha}_{q;l_p}(r)v^{\alpha}_{p}(r)r^2dr$$

and

$$o^{\alpha}_{pp'}=\int_0^{R_i}v^{\alpha}_{p}(r)v^{\alpha}_{p'}(r)r^2dr,\quad l_p=l_{p'}$$

where $u^{\alpha}_{q;l}$ is the qth APW radial function for angular momentum l; and v^{α}_{p} is the pth local-orbital radial function and has angular momentum l_p.

REVISION HISTORY:

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

7.115 pade (Source File: pade.f90)

INTERFACE:

```
subroutine pade(ni,zi,ui,no,zo,uo)
```

INPUT/OUTPUT PARAMETERS:

```
ni : number of input points (in,integer)
zi : input points (in,complex(ni))
ui : input function values (in,complex(ni))
no : number of output points (in,integer)
zo : output points (in,complex(no))
uo : output function values (out,complex(no))
```

DESCRIPTION:

Calculates a Padé approximant of a function, given the function evaluated on a set of points in the complex plane. The function is returned for a set of complex output points. The algorithm from H. J. Vidberg and J. W. Serene *J. Low Temp. Phys.* **29**, 179 (1977) is used.

REVISION HISTORY:

```
Created December 2010 (Antonio Sanna)
```

7.116 plot1d (Source File: plot1d.f90)

INTERFACE:

```
subroutine plot1d(fnum1,fnum2,nf,rfmt,rfir)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
fnum1 : plot file number (in,integer)
fnum2 : vertex location file number (in,integer)
nf    : number of functions (in,integer)
rfmt  : real muffin-tin function (in,real(npmtmax,natmtot,nf))
rfir  : real intersitial function (in,real(ngtot,nf))
```

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 `rfplot`.

REVISION HISTORY:

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

7.117 plot2d (Source File: plot2d.f90)

INTERFACE:

```
subroutine plot2d(tproj,fnum,nf,rfmt,rfir)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
tproj : .true. if nf=3 and the vector function should be projected onto the
        2D plotting plane axes (in,logical)
fnum  : plot file number (in,integer)
nf    : number of functions (in,integer)
rfmt  : real muffin-tin function (in,real(npmtmax,natmtot,nf))
rfir  : real intersitial function (in,real(ngtot,nf))
```

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 `rfplot`.

REVISION HISTORY:

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

7.118 plot3d (Source File: plot3d.f90)

INTERFACE:

```
subroutine plot3d(fnum,nf,rfmt,rfir)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
fnum : plot file number (in,integer)
nf   : number of functions (in,integer)
rfmt : real muffin-tin function (in,real(npmtmax,natmtot,nf))
rfir : real intersitial function (in,real(ngtot,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)
```

7.119 plotpt1d (Source File: plotpt1d.f90)

INTERFACE:

```
subroutine plotpt1d(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.

REVISION HISTORY:

```
Created June 2003 (JKD)
Improved September 2007 (JKD)
Improved again, July 2010 (T. McQueen and JKD)
```

7.120 polar (Source File: polar.f90)

INTERFACE:

```
subroutine polar(pvl)
```

USES:

```
use modmain
use modstore
use modmpi
use modomp
```

INPUT/OUTPUT PARAMETERS:

```
pvl : polarisation vector modulo $2\pi$ (out,real(8))
```

DESCRIPTION:

Calculates the polarisation vector modulo 2π in lattice coordinates using the formula of R. D. King-Smith and David Vanderbilt [Phys. Rev. B **47**, 1651(R) (1993)], namely

$$P_l = \sum_{\mathbf{k}} \mathrm{Im} \ln \det \left(\langle u_{i\mathbf{k}+\Delta\mathbf{k}_l} | u_{j\mathbf{k}} \rangle \right),$$

where $\Delta\mathbf{k}_l = (1/n_l)\mathbf{B}_l$ and $\mathbf{B}_l$ is a reciprocal lattice vector. The number of points n_l is equal to that of the original k-point grid in direction of $\mathbf{B}_l$, multiplied by `nskpolar`. See also the routines `polark` and `bornechg`.

REVISION HISTORY:

```
Created May 2020 (JKD)
```

7.121 polynm (Source File: polynm.f90)

INTERFACE:

```
pure real(8) function polynm(m,np,xa,ya,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))
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)
```

7.122 potcoul (Source File: potcoul.f90)

INTERFACE:

```
subroutine potcoul
```

USES:

```
use modmain
use modomp
```

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 coverted to complex representations and passed to the routine `zpotcoul`.

REVISION HISTORY:

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

7.123 potks (Source File: potks.f90)

INTERFACE:

```
subroutine potks(txc)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
txc : .true. if the exchange-correlation energy density and potentials
      should be calculated (in,logical)
```

DESCRIPTION:

Computes the Kohn-Sham effective potential by adding together the Coulomb and exchange-correlation potentials. Also computes the effective magnetic field. See routines `potcoul` and `potxc`.

REVISION HISTORY:

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

7.124 potnucl (Source File: potnucl.f90)

INTERFACE:

```
subroutine potnucl(ptnucl,nr,r,zn,vn)
```

INPUT/OUTPUT PARAMETERS:

```
ptnucl : .true. if the nucleus is a point charge (in,logical)
nr     : number of radial mesh points (in,integer)
r      : radial mesh (in,real(nr))
zn     : nuclear charge (in,real)
vn     : potential on radial mesh (out,real(nr))
```

DESCRIPTION:

Computes the nuclear Coulomb potential on a radial mesh. The nuclear radius R is estimated from the nuclear charge Z and the potential is given by

$$V(r) = \begin{cases} Z(3R^2 - r^2)/2R^3 & r < R \\ Z/r & r \geq R \end{cases}$$

assuming that the nucleus is a uniformly charged sphere. If `ptnucl` is `.true.` then the nucleus is treated as a point particle.

REVISION HISTORY:

Created January 2009 (JKD)

7.125 potplot (Source File: potplot.f90)

INTERFACE:

```
subroutine potplot
```

USES:

```
use modmain
```

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)

7.126 pottm2 (Source File: pottm2.f90)

INTERFACE:

```
subroutine pottm2(i,k1,p,vh,vx)
```

USES:

```
use moddftu
```

INPUT/OUTPUT PARAMETERS:

```
i  : DFT+U entry (in,integer)
k1 : k-index of tensor moment (in,integer)
p  : p-index of tensor moment  (in,integer)
vh : Hartree potential energy (out,real)
vx : exchange potential energy (out,real)
```

DESCRIPTION:

Calculates the DFT+U Hartree and exchange potential energies for a 2-index tensor moment component. See `pottm3`.

REVISION HISTORY:

```
Created April 2008 (F. Cricchio and L. Nordstrom)
Modified, January 2014 (JKD)
```

7.127 pottm3 (Source File: pottm3.f90)

INTERFACE:

```
subroutine pottm3(i,k1,p,r,vh,vx)
```

USES:

```
use moddftu
```

INPUT/OUTPUT PARAMETERS:

```
i  : DFT+U entry (in,integer)
k1 : k-index of tensor moment (in,integer)
p  : p-index of tensor moment (in,integer)
r  : r-index of tensor moment (in,integer)
vh : Hartree potential energy (out,real)
vx : exchange potential energy (out,real)
```

DESCRIPTION:

Calculates the DFT+U Hartree and exchange potential energies for a 3-index tensor moment component. See Eq. (28) in *Phys. Rev. B* **80**, 035121 (2009); and Eqs. (3), (4) in *Phys. Rev. B* **78**, 100404 (2008).

REVISION HISTORY:

```
Created April 2008 (F. Cricchio and L. Nordstrom)
Modified and fixed bug, January 2014 (JKD)
```

7.128 potxc (Source File: potxc.f90)

INTERFACE:

```
subroutine potxc(tsh,xctype_,rhomt_,rhoir_,magmt_,magir_,taumt_,tauir_,exmt_, &
 exir_,ecmt_,ecir_,vxcmt_,vxcir_,bxcmt_,bxcir_,wxcmt_,wxcir_)
```

USES:

```
use modmain
use modomp
```

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)
```

7.129 r3cross (Source File: r3cross.f90)

INTERFACE:

```
pure 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)
```

7.130 r3frac (Source File: r3frac.f90)

INTERFACE:

```
pure subroutine r3frac(eps,v)
```

INPUT/OUTPUT PARAMETERS:

```
eps : zero component tolerance (in,real)
v   : input vector (inout,real(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 $(1 - \epsilon, 1]$.

REVISION HISTORY:

```
Created January 2003 (JKD)
Removed iv, September 2011 (JKD)
```

7.131 r3mdet (Source File: r3mdet.f90)

INTERFACE:

```
pure 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)
```

7.132 r3minv (Source File: r3minv.f90)

INTERFACE:

```
subroutine r3minv(a,b)
```

INPUT/OUTPUT PARAMETERS:

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

DESCRIPTION:

Computes the inverse of a real 3×3 matrix.

REVISION HISTORY:

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

7.133 r3mm (Source File: r3mm.f90)

INTERFACE:

```
pure 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)
```

7.134 r3mmt (Source File: r3mmt.f90)

INTERFACE:

```
pure 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)
```

7.135 r3mtm (Source File: r3mtm.f90)

INTERFACE:

```
pure 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)
```

7.136 r3mtv (Source File: r3mtv.f90)

INTERFACE:

```
pure 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)
```

7.137 r3mv (Source File: r3mv.f90)

INTERFACE:

```
pure 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)
```

7.138 radnucl (Source File: radnucl.f90)

INTERFACE:

```
elemental real(8) function radnucl(z)
```

INPUT/OUTPUT PARAMETERS:

```
z : atomic number (in,real)
```

DESCRIPTION:

Computes an approximate nuclear charge radius from the atomic number Z. The nuclear mass number, A, is estimated using

$$A = 4.467 \times 10^{-3} Z^2 + 2.163 Z - 1.168,$$

[D. Andrae in *Relativistic Electronic Structure Theory - Fundamentals* **11**, 203 (2002)], and the nuclear charge radius can be determined from

$$r = \left(r_0 + \frac{r_1}{A^{2/3}} + \frac{r_2}{A^{4/3}} \right) A^{1/3},$$

where $r_0 = 0.9071$, $r_1 = 1.105$ and $r_2 = -0.548$ [I. Angeli, *Atomic Data and Nuclear Data Tables* **87**, 185 (2004)].

REVISION HISTORY:

```
Created October 2011 (JKD)
```

7.139 rdirac (Source File: rdirac.f90)

INTERFACE:

```
subroutine rdirac(sol,n,l,k,nr,r,vr,eval,g0,f0)
```

INPUT/OUTPUT PARAMETERS:

```
sol  : speed of light in atomic units (in,real)
n    : principal quantum number (in,integer)
l    : quantum number l (in,integer)
k    : quantum number k (l or l+1) (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)
```

7.140 rdiracint (Source File: rdiracint.f90)

INTERFACE:

```
pure subroutine rdiracint(sol,kpa,e,nr,r,vr,nn,g0,g1,f0,f1)
```

INPUT/OUTPUT PARAMETERS:

```
sol  : speed of light in atomic units (in,real)
kpa  : quantum number kappa (in,integer)
e    : energy (in,real)
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:

Integrates 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)

$$\left(\frac{d}{dr}+\frac{\kappa}{r}\right)G_\kappa = \frac{1}{c}\{2E_0+E-V\}F_\kappa$$
$$\left(\frac{d}{dr}-\frac{\kappa}{r}\right)F_\kappa = -\frac{1}{c}\{E-V\}G_\kappa,$$

where $G_\kappa = rg_\kappa$ and $F_\kappa = rf_\kappa$ are 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 $j = l + \frac{1}{2}$.

REVISION HISTORY:

```
Created September 2002 (JKD)
Polynomial order fixed to 3, September 2013 (JKD)
```

7.141 rdmdedc (Source File: rdmdedc.f90)

INTERFACE:

```
subroutine rdmdedc(dedc)
```

USES:

```
use modmain
use modrdm
use modomp
```

INPUT/OUTPUT PARAMETERS:

```
dedc : energy derivative (out,complex(nstsv,nstsv,nkpt))
```

DESCRIPTION:

Calculates the derivative of the total energy w.r.t. the second-variational coefficients `evecsv`.

REVISION HISTORY:

```
Created 2008 (Sharma)
```

7.142 rdmdedn (Source File: rdmdedn.f90)

INTERFACE:

```
subroutine rdmdedn(dedn)
```

USES:

```
use modmain
use modrdm
use modomp
```

INPUT/OUTPUT PARAMETERS:

```
dedn : free energy derivative (out,real(nstsv,nkpt))
```

DESCRIPTION:

Calculates the negative of the derivative of total free energy w.r.t. occupation numbers.

REVISION HISTORY:

```
Created 2008 (Sharma)
```

7.143 rdmdexcdc (Source File: rdmdexcdc.f90)

INTERFACE:

```
subroutine rdmdexcdc(dedc)
```

USES:

```
use modmain
use modrdm
```

INPUT/OUTPUT PARAMETERS:

```
dedc : energy derivative (inout,complex(nstsv,nstsv,nkpt))
```

DESCRIPTION:

Calculates the derivative of the exchange-correlation energy w.r.t. `evecsv` and adds the result to the total.

REVISION HISTORY:

```
Created 2008 (Sharma)
```

7.144 rdmdexcdn (Source File: rdmdexcdn.f90)

INTERFACE:

```
subroutine rdmdexcdn(dedn)
```

USES:

```
use modmain
use modrdm
```

INPUT/OUTPUT PARAMETERS:

```
dedn : energy derivative (inout,real(nstsv,nkpt))
```

DESCRIPTION:

Calculates the derivative of the exchange-correlation energy w.r.t. occupation numbers and adds the result to the total.

REVISION HISTORY:

```
Created 2008 (Sharma)
```

7.145 rdmdkdc (Source File: rdmdkdc.f90)

INTERFACE:

```
subroutine rdmdkdc
```

USES:

```
use modmain
use modrdm
use modomp
```

DESCRIPTION:

Calculates the derivative of kinetic energy w.r.t. the second-variational coefficients `evecsv`.

REVISION HISTORY:

```
Created October 2008 (Sharma)
```

7.146 rdmdtsdn (Source File: rdmdtsdn.f90)

INTERFACE:

```
subroutine rdmdtsdn(dedn)
```

USES:

```
use modmain
use modrdm
```

INPUT/OUTPUT PARAMETERS:

```
dedn : energy derivative (inout,real(nstsv,nkpt))
```

DESCRIPTION:

Calculates the derivative of the entropic contribution to the free energy w.r.t. occupation numbers and adds it to the total.

REVISION HISTORY:

```
Created 2008 (Baldsiefen)
```

7.147 rdmenergy (Source File: rdmenergy.f90)

INTERFACE:

```
subroutine rdmenergy
```

USES:

```
use modmain
use modrdm
use modtest
```

DESCRIPTION:

Calculates RDMFT total energy (free energy for finite temperatures).

REVISION HISTORY:

```
Created 2008 (Sharma)
Updated for free energy 2009 (Baldsiefen)
```

7.148 rdmengyxc (Source File: rdmengyxc.f90)

INTERFACE:

```
subroutine rdmengyxc
```

USES:

```
use modmain
use modrdm
```

DESCRIPTION:

Calculates RDMFT exchange-correlation energy.

REVISION HISTORY:

```
Created 2008 (Sharma)
```

7.149 rdmentropy (Source File: rdmentropy.f90)

INTERFACE:

```
subroutine rdmentropy
```

USES:

```
use modmain
use modrdm
```

DESCRIPTION:

Calculates RDMFT entropy $S = -\sum_i n_i \log(n_i/n_{\max}) + (n_{\max} - n_i) \log(1 - n_i/n_{\max})$, where $n_{\max}$ is the maximum allowed occupancy (1 or 2).

REVISION HISTORY:

```
Created 2008 (Baldsiefen)
```

7.150 rdmeval (Source File: rdmeval.f90)

INTERFACE:

```
subroutine rdmeval
```

USES:

```
use modmain
use modrdm
```

DESCRIPTION:

RDMFT eigenvalues are determined by calculating the derivative of the total energy with respect to the occupation number at half the maximum occupancy ($n_{\max}/2$).

REVISION HISTORY:

```
Created 2009 (Sharma)
```

7.151 rdmft (Source File: rdmft.f90)

INTERFACE:

```
subroutine rdmft
```

USES:

```
use modmain
use modrdm
use modmpi
```

DESCRIPTION:

Main routine for one-body reduced density matrix functional theory (RDMFT).

REVISION HISTORY:

Created 2008 (Sharma)

7.152 rdmminc (Source File: rdmminc.f90)

INTERFACE:

```
subroutine rdmminc
```

USES:

```
use modmain
use modrdm
use modmpi
```

DESCRIPTION:

Minimizes the total energy with respect to the second-variational coefficients `evecsv`. The steepest-descent algorithm is used.

REVISION HISTORY:

Created 2008 (Sharma)

7.153 rdmminn (Source File: rdmminn.f90)

INTERFACE:

```
subroutine rdmminn
```

USES:

```
use modmain
use modrdm
use modmpi
```

DESCRIPTION:

Minimizes the total energy w.r.t. occupation numbers. The steepest-descent algorithm is used.

REVISION HISTORY:

Created 2008 (Sharma)

7.154 rdmvaryc (Source File: rdmvaryc.f90)

INTERFACE:

```
subroutine rdmvaryc
```

USES:

```
use modmain
use modrdm
```

DESCRIPTION:

Calculates new `evecsv` from old by using the derivatives of the total energy w.r.t. `evecsv`. A single step of steepest-descent is made.

REVISION HISTORY:

```
Created 2009 (Sharma)
```

7.155 rdmvaryn (Source File: rdmvaryn.f90)

INTERFACE:

```
subroutine rdmvaryn
```

USES:

```
use modmain
use modrdm
use modmpi
```

DESCRIPTION:

Calculates new occupation numbers from old by using the derivatives of the total energy: $n_i^{\rm new} = n_i^{\rm old} - \tau\gamma_i$, where τ is chosen such that $0 \le n_i \le n_{\rm max}$ with

$$\gamma_i = \begin{cases} g_i(n_{\rm max} - n_i) & g_i > 0 \\ g_i n_i & g_i \le 0 \end{cases}$$

where $g_i = \partial E/\partial n_i - \kappa$, and κ is chosen such that $\sum_i \gamma_i = 0$.

REVISION HISTORY:

```
Created 2009 (JKD,Sharma)
```

7.156 rdmwritededn (Source File: rdmwritededn.f90)

INTERFACE:

```
subroutine rdmwritededn(dedn)
```

USES:

```
use modmain
use modrdm
```

INPUT/OUTPUT PARAMETERS:

```
dedn : derivative of energy (in,real(nstsv,nkpt))
```

DESCRIPTION:

Writes the derivative of total energy with respect to occupation numbers to file `RDM_DEDN.OUT`.

REVISION HISTORY:

```
Created 2008 (Sharma)
```

7.157 rdmwriteengy (Source File: rdmwriteengy.f90)

INTERFACE:

```
subroutine rdmwriteengy(fnum)
```

USES:

```
use modmain
use modrdm
```

INPUT/OUTPUT PARAMETERS:

```
fnum : file number for writing output (in,integer)
```

DESCRIPTION:

Writes all contributions to the total energy to file.

REVISION HISTORY:

```
Created 2008 (Sharma)
```

7.158 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)
```

7.159 readinput (Source File: readinput.f90)

INTERFACE:

```
subroutine readinput
```

USES:

```
use modmain
use moddftu
use modrdm
use modphonon
use modtest
use modrandom
use modpw
use modtddft
use modulr
use modvars
use modgw
use modbog
use modw90
use modmpi
use modomp
use modramdisk
```

DESCRIPTION:

Reads in the input parameters from the file `elk.in`. Also sets default values for the input parameters.

REVISION HISTORY:

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

7.160 readstate (Source File: readstate.f90)

INTERFACE:

```
subroutine readstate
```

USES:

```
use modmain
use moddftu
```

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)
```

7.161 reciplat (Source File: reciplat.f90)

INTERFACE:

```
subroutine reciplat(avec,bvec,omega,omegabz)
```

INPUT/OUTPUT PARAMETERS:

```
avec    : lattice vectors (in,real(3,3))
bvec    : reciprocal lattice vectors (out,real(3,3))
omega   : unit cell volume (out,real)
omegabz : Brillouin zone volume (out,real)
```

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 $s = \mathbf{a}_1 \cdot (\mathbf{a}_2 \times \mathbf{a}_3)$, and the Brillouin zone volume $\Omega_{\rm BZ} = (2\pi)^3/\Omega$.

REVISION HISTORY:

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

7.162 rfinp (Source File: rfinp.f90)

INTERFACE:

```
real(8) function rfinp(rfmt1,rfir1,rfmt2,rfir2)
```

USES:

```
use modmain
use modomp
```

INPUT/OUTPUT PARAMETERS:

```
rfmt1 : first function in real spherical harmonics for all muffin-tins
        (in,real(npmtmax,natmtot))
rfir1 : first real interstitial function in real-space (in,real(ngtot))
rfmt2 : second function in real spherical harmonics for all muffin-tins
        (in,real(npmtmax,natmtot))
rfir2 : second real interstitial function in real-space (in,real(ngtot))
```

DESCRIPTION:

Calculates the inner product of two real functions over the entire unit cell. The input muffin-tin functions should have angular momentum cut-off `lmaxo`. In the interstitial 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)
```

7.163 rfinterp (Source File: rfinterp.f90)

INTERFACE:

```
subroutine rfinterp(ni,xi,wci,fi,no,xo,fo)
```

INPUT/OUTPUT PARAMETERS:

```
ni  : number of input points (in,integer)
xi  : input abscissa array (in,real(ni))
wci : input spline coefficient weights (in,real(12,ni))
fi  : input data array (in,real(ni))
no  : number of output points (in,integer)
xo  : output abscissa array (in,real(no))
fo  : output interpolated function (out,real(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)
Arguments changed, April 2016 (JKD)
```

7.164 rfmtctof (Source File: rfmtctof.f90)

INTERFACE:

```
subroutine rfmtctof(rfmt)
```

USES:

```
use modmain
use modomp
```

INPUT/OUTPUT PARAMETERS:

```
rfmt : real muffin-tin function (in,real(npmtmax,natmtot))
```

DESCRIPTION:

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

REVISION HISTORY:

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

7.165 rfmtinp (Source File: rfmtinp.f90)

INTERFACE:

```
pure real(8) function rfmtinp(nr,nri,wr,rfmt1,rfmt2)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
nr    : number of radial mesh points (in,integer)
nri   : number of radial mesh points on the inner part of the muffin-tin
        (in,integer)
wr    : weights for integration on radial mesh (in,real(nr))
rfmt1 : first real function inside muffin-tin (in,real(*))
rfmt2 : second real function inside muffin-tin (in,real(*))
```

DESCRIPTION:

Calculates the inner product of two real functions 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

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

REVISION HISTORY:

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

7.166 rhocore (Source File: rhocore.f90)

INTERFACE:

```
subroutine rhocore
```

USES:

```
use modmain
```

DESCRIPTION:

Adds the core density and magnetisation to the muffin-tin functions. Also computes the amount of leakage of core charge from the muffin-tin spheres into the interstitial.

REVISION HISTORY:

```
Created April 2003 (JKD)
Fixed core moment direction, October 2012 (M. Meinert)
```

7.167 rhoinit (Source File: rhoinit.f90)

INTERFACE:

```
subroutine rhoinit
```

USES:

```
use modmain
use modomp
```

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)
```

7.168 rhomagk (Source File: rhomagk.f90)

INTERFACE:

```
subroutine rhomagk(ngp,igpig,lock,wppt,occsvp,apwalm,evecfv,evecsv)
```

USES:

```
use modmain
use modomp
```

INPUT/OUTPUT PARAMETERS:

```
ngp    : number of G+p-vectors (in,integer(nspnfv))
igpig  : index from G+p-vectors to G-vectors (in,integer(ngkmax,nspnfv))
lock   : OpenMP lock for each atom (in,integer(natmtot))
wppt   : weight of input p-point (in,real)
occsvp : occupation number for each state (in,real(nstsv))
apwalm : APW matching coefficients
         (in,complex(ngkmax,apwordmax,lmmaxapw,natmtot,nspnfv))
evecfv : first-variational eigenvectors (in,complex(nmatmax,nstfv,nspnfv))
evecsv : second-variational eigenvectors (in,complex(nstsv,nstsv))
```

DESCRIPTION:

Generates the partial valence charge density and magnetisation from the eigenvectors at a particular k-point. 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. A similar proccess is used for the intersitial density in which the wavefunction in real-space is obtained from a Fourier transform of the APW functions. See routines `wavefmt`, `genshtmat` and `eveqn`.

REVISION HISTORY:

```
Created April 2003 (JKD)
Removed conversion to spherical harmonics, January 2009 (JKD)
Partially de-phased the muffin-tin magnetisation for spin-spirals,
 February 2009 (FC, FB & LN)
Optimisations, July 2010 (JKD)
```

7.169 rhomagsh (Source File: rhomagsh.f90)

INTERFACE:

```
subroutine rhomagsh
```

USES:

```
use modmain
use modomp
```

DESCRIPTION:

Converts the muffin-tin density and magnetisation from spherical coordinates to a spherical harmonic expansion. See `rhomagk`.

REVISION HISTORY:

```
Created January 2009 (JKD)
```

7.170 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)
```

7.171 rhoplot (Source File: rhoplot.f90)

INTERFACE:

```
subroutine rhoplot
```

USES:

```
use modmain
```

DESCRIPTION:

Outputs the charge density, read in from `STATE.OUT`, for 1D, 2D or 3D plotting.

REVISION HISTORY:

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

7.172 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. The rotation convention follows the 'right-hand rule'.

REVISION HISTORY:

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

7.173 roteuler (Source File: roteuler.f90)

INTERFACE:

```
subroutine roteuler(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)
Fixed problem thanks to Frank Wagner, June 2013 (JKD)
```

7.174 rotrflm (Source File: rotrfmt.f90)

INTERFACE:

```
subroutine rotrflm(rot,lmax,n,ld,rflm1,rflm2)
```

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)
rflm1 : coefficients of the real spherical harmonic expansion for each
        function (in,real(ld,n))
rflm2 : coefficients of rotated functions (out,complex(ld,n))
```

DESCRIPTION:

Rotates a set of real functions

$$f_i(\mathbf{r}) = \sum_{lm} f_{lm}^i R_{lm}(\hat{\mathbf{r}})$$

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 `roteuler`) and then applying the spherical harmonic rotation matrix generated by the routine `rlmrot`.

REVISION HISTORY:

```
Created December 2008 (JKD)
```

7.175 rlmrot (Source File: rotrfmt.f90)

INTERFACE:

```
subroutine rlmrot(p,ang,lmax,ld,d)
```

INPUT/OUTPUT PARAMETERS:

```
p    : if p=-1 then the rotation matrix is improper (in,integer)
ang  : Euler angles; alpha, beta, gamma (in,real(3))
lmax : maximum angular momentum (in,integer)
ld   : leading dimension (in,integer)
d    : real spherical harmonic rotation matrix (out,real(ld,*))
```

DESCRIPTION:

Returns the rotation matrix in the basis of real spherical harmonics given the three Euler angles, (α, β, γ), and the parity, p, of the rotation. The matrix is determined using the formula of V. V. Nechaev, [*J. Struct. Chem.* **35**, 115 (1994)], suitably modified for our definition of the real spherical harmonics ($m_1 > 0$, $m_2 > 0$):

$$\Delta^l_{00} = d^l_{00},$$
$$\Delta^l_{m_1 0} = \sqrt{2}\,(-1)^{m_1} d^l_{0m_1} \cos(m_1\alpha),$$
$$\Delta^l_{0m_2} = \sqrt{2}\,(-1)^{m_2} d^l_{m_2 0} \cos(m_2\gamma),$$
$$\Delta^l_{-m_1 0} = -\sqrt{2}\, d^l_{0m_1} \sin(m_1\alpha),$$
$$\Delta^l_{0-m_2} = \sqrt{2}\, d^l_{m_2 0} \sin(m_2\gamma),$$
$$\Delta^l_{m_1 m_2} = (-1)^{m_1}(-1)^{m_2}\{\cos(m_1\alpha)\cos(m_2\gamma)[d_A + d_B] - \sin(m_1\alpha)\sin(m_2\gamma)[d_A - d_B]\},$$
$$\Delta^l_{m_1 -m_2} = (-1)^{m_1}\{\sin(m_1\alpha)\cos(m_2\gamma)[d_A - d_B] + \cos(m_1\alpha)\sin(m_2\gamma)[d_A + d_B]\},$$
$$\Delta^l_{-m_1 m_2} = -(-1)^{m_2}\{\sin(m_1\alpha)\cos(m_2\gamma)[d_A + d_B] + \cos(m_1\alpha)\sin(m_2\gamma)[d_A - d_B]\},$$
$$\Delta^l_{-m_1 -m_2} = \cos(m_1\alpha)\cos(m_2\gamma)[d_A - d_B] - \sin(m_1\alpha)\sin(m_2\gamma)[d_A + d_B],$$

where $d_A \equiv d^l_{-m_1 -m_2}$, $d_B \equiv (-1)^{m_1} d^l_{m_1 -m_2}$ and d is the rotation matrix about the y-axis for complex spherical harmonics. See the routines `genrlm`, `roteuler` and `ylmroty`.

REVISION HISTORY:

```
Created December 2008 (JKD)
```

7.176 rotzflm (Source File: rotzflm.f90)

INTERFACE:

```
subroutine rotzflm(rot,lmin,lmax,lmmax,n,ld,zflm1,zflm2)
```

INPUT/OUTPUT PARAMETERS:

```
rot   : rotation matrix (in,real(3,3))
lmin  : minimum angular momentum (in,integer)
lmax  : maximum angular momentum (in,integer)
lmmax : (lmax+1)^2 or larger (in,integer)
n     : number of functions to rotate (in,integer)
ld    : leading dimension (in,integer)
zflm1 : coefficients of the 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 complex 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 `roteuler`) and then applying the spherical harmonic rotation matrix generated by the routine `ylmrot`.

REVISION HISTORY:

```
Created April 2003 (JKD)
Modified, December 2008 (JKD)
```

7.177 rschrodint (Source File: rschrodint.f90)

INTERFACE:

```
pure subroutine rschrodint(sol,l,e,nr,r,vr,nn,p0,p1,q0,q1)
```

INPUT/OUTPUT PARAMETERS:

```
sol : speed of light in atomic units (in,real)
l   : angular momentum quantum number (in,integer)
e   : energy (in,real)
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:

Integrates 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)

$$\frac{d}{dr}P_l = 2MQ_l + \frac{1}{r}P_l$$
$$\frac{d}{dr}Q_l = -\frac{1}{r}Q_l + \left[\frac{l(l+1)}{2Mr^2} + (V-E)\right]P_l,$$

where V is the external potential, E is the eigen energy and $M = 1+(E-V)/2c^2$. 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`).

REVISION HISTORY:

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

7.178 rtozflmn (Source File: rtozfmt.f90)

INTERFACE:

```
pure subroutine rtozflmn(lmax,n,ld,rflm,zflm)
```

INPUT/OUTPUT PARAMETERS:

```
lmax : maximum angular momentum (in,integer)
n    : number of functions to convert (in,integer)
ld   : leading dimension (in,integer)
rflm : coefficients of real spherical harmonic expansion (in,real(ld,n))
zflm : coefficients of complex spherical harmonic expansion
       (out,complex(ld,n))
```

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)
```

7.179 rvfcross (Source File: rvfcross.f90)

INTERFACE:

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

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

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

7.180 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 $x \geq 1$ 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 $l \leq 30$ and $m \leq 6$.

REVISION HISTORY:

```
Created March 2003 (JKD)
Modified to return an array of values, October 2004 (JKD)
```

7.181 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)
```

7.182 sdelta (Source File: sdelta.f90)

INTERFACE:

```
real(8) 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

5. Lorentzian

See routines `stheta`, `sdelta_mp`, `sdelta_fd` and `sdelta_sq`.

REVISION HISTORY:

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

7.183 getsdata (Source File: sdelta.f90)

INTERFACE:

```
subroutine getsdata(stype,sdescr)
```

INPUT/OUTPUT PARAMETERS:

```
stype  : smearing type (in,integer)
sdescr : smearing scheme description (out,character(*))
```

DESCRIPTION:

Returns a description of the smearing scheme as string `sdescr` up to 256 characters long.

REVISION HISTORY:

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

7.184 sdelta_fd (Source File: sdelta_fd.f90)

INTERFACE:

```
elemental 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)
```

7.185 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)
```

7.186 sdelta_sq (Source File: sdelta_sq.f90)

INTERFACE:

```
elemental 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)
```

7.187 sfacmag (Source File: sfacmag.f90)

INTERFACE:

```
subroutine sfacmag
```

USES:

```
use modmain
use modpw
use modtest
```

DESCRIPTION:

Outputs magnetic structure factors, i.e. the Fourier transform coefficients of each component j of magnetization $\mathbf{m}(\mathbf{r})$,

$$F_j(\mathbf{H}) = \int_\Omega d^3r\, m_j(\mathbf{r}) e^{i\mathbf{H}\cdot\mathbf{r}},$$

to the files `SFACMAG_j.OUT`. The lattice coordinates (h,k,l) of $\mathbf{H}$-vectors in this file are transformed by the matrix `vhmat`. See also routines `zftrf` and `genhvec`.

REVISION HISTORY:

```
Created July 2010 (Alexey I. Baranov)
Added multiplicity of the H-vectors, Oct. 2010 (Alexey I. Baranov)
```

7.188 sfacrho (Source File: sfacrho.f90)

INTERFACE:

```
subroutine sfacrho
```

USES:

```
use modmain
use modpw
use modtest
```

DESCRIPTION:

Outputs X-ray structure factors, i.e. the Fourier transform coefficients of the total electron density

$$F(\mathbf{H}) = \int_\Omega d^3r\, \rho(\mathbf{r}) e^{i\mathbf{H}\cdot\mathbf{r}},$$

to the file `SFACRHO.OUT`. The lattice coordinates (h,k,l) of the $\mathbf{H}$-vectors in this file are transformed by the matrix `vhmat`. If and energy window is set using the variable `wsfac`, then only those states within the window are used to compute the density. See also routines `zftrf` and `genhvec`.

REVISION HISTORY:

```
Created July 2010 (Alexey I. Baranov)
Added multiplicity of the H-vectors, Oct. 2010 (Alexey I. Baranov)
```

7.189 sortidx (Source File: sortidx.f90)

INTERFACE:

```
subroutine sortidx(n,x,idx)
```

INPUT/OUTPUT PARAMETERS:

```
n   : number of elements in array (in,integer)
x   : real array (in,real(n))
idx : permutation index (out,integer(n))
```

DESCRIPTION:

Finds the permutation index `idx` which sorts the real array `x` into ascending order. No sorting of the array `x` itself is performed. Uses the heapsort algorthim.

REVISION HISTORY:

```
Created October 2002 (JKD)
Included tolerance eps, April 2006 (JKD)
```

7.190 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 spherical (θ, ϕ) coordinates are generated using the explicit 'golden section' formula:

$$\theta_k = \arccos\left[1 - \left(k - \tfrac{1}{2}\right)\delta z\right]$$
$$\phi_k = (k-1)\delta\phi,$$

where $\delta z = 2/n$ and $\delta\phi = \pi(1 - \sqrt{5})$.

REVISION HISTORY:

```
Created April 2008 (JKD)
Improved covering, October 2009 (JKD)
```

7.191 sphcrd (Source File: sphcrd.f90)

INTERFACE:

```
pure 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)
```

7.192 spline (Source File: spline.f90)

INTERFACE:

```
subroutine spline(n,x,f,cf)
```

INPUT/OUTPUT PARAMETERS:

```
n  : number of points (in,integer)
x  : abscissa array (in,real(n))
f  : input data array (in,real(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 $x \in [x_i, x_{i+1})$. The coefficients are determined piecewise by fitting a cubic polynomial to adjacent points.

REVISION HISTORY:

```
Created November 2011 (JKD)
```

7.193 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)
```

7.194 stheta_fd (Source File: stheta_fd.f90)

INTERFACE:

```
elemental 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)
```

7.195 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)
```

7.196 stheta_sq (Source File: stheta_sq.f90)

INTERFACE:

```
elemental 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)
```

7.197 sumrule (Source File: sumrule.f90)

INTERFACE:

```
subroutine sumrule(dynq)
```

INPUT/OUTPUT PARAMETERS:

```
dynq : dynamical matrices on q-point set (in,real(nbph,nbph,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)
```

7.198 symrf (Source File: symrf.f90)

INTERFACE:

```
subroutine symrf(nr,nri,np,ngdg,ngt,ngv,igf,ld,rfmt,rfir)
```

USES:

```
use modmain
use modomp
```

INPUT/OUTPUT PARAMETERS:

```
nr   : number of radial points for each species (in,integer(nspecies))
nri  : number of radial points on the inner part (in,integer(nspecies))
np   : total number of points in each muffin-tin (in,integer(nspecies))
ngdg : G-vector grid sizes (in,integer(3))
ngt  : total number of G-vectors (in,integer)
ngv  : number of G-vectors within cut-off (in,integer)
igf  : map from G-vector index to FFT array (in,integer(ngv))
ld   : leading dimension (in,integer)
rfmt : real muffin-tin function (inout,real(ld,natmtot))
rfir : real intersitial function (inout,real(ngtot))
```

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.

REVISION HISTORY:

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

7.199 symrfir (Source File: symrfir.f90)

INTERFACE:

```
subroutine symrfir(ngdg,ngt,ngv,igf,rfir)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
ngdg : G-vector grid sizes (in,integer(3))
ngt  : total number of G-vectors (in,integer)
ngv  : number of G-vectors within cut-off (in,integer)
igf  : map from G-vector index to FFT array (in,integer(ngv))
rfir : real intersitial function (inout,real(ngt))
```

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)
```

7.200 symrvf (Source File: symrvf.f90)

INTERFACE:

```
subroutine symrvf(tspin,tnc,nr,nri,np,ngdg,ngt,ngv,igf,ld1,rvfmt,ld2,rvfir)
```

USES:

```
use modmain
use modomp
```

INPUT/OUTPUT PARAMETERS:

```
tspin : .true. if spin rotations should be used (in,logical)
tnc   : .true. if the vector field is non-collinear, otherwise it is
        collinear along the z-axis (in,logical)
nr    : number of radial points for each species (in,integer(nspecies))
nri   : number of radial points on the inner part (in,integer(nspecies))
np    : total number of points in each muffin-tin (in,integer(nspecies))
ngdg  : G-vector grid sizes (in,integer(3))
ngt   : total number of G-vectors (in,integer)
ngv   : number of G-vectors within cut-off (in,integer)
igf   : map from G-vector index to FFT array (in,integer(ngv))
ld1   : leading dimension (in,integer)
rvfmt : real muffin-tin vector field (in,real(ld1,natmtot,*))
ld2   : leading dimension (in,integer)
rvfir : real interstitial vector field (in,real(ld2,*))
```

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)
```

7.201 symrvfir (Source File: symrvfir.f90)

subroutine symrvfir(tspin,tnc,ngdg,ngt,ngv,igf,ld,rvfir) *USES:*

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
tspin : .true. if spin rotations should be used (in,logical)
tnc   : .true. if the vector field is non-collinear, otherwise it is
        collinear along the z-axis (in,logical)
ngdg  : G-vector grid sizes (in,integer(3))
```

```
ngt   : total number of G-vectors (in,integer)
ngv   : number of G-vectors within cut-off (in,integer)
igf   : map from G-vector index to FFT array (in,integer(ngv))
ld    : leading dimension (in,integer)
rvfir : real interstitial vector function (inout,real(ld,*))
```

DESCRIPTION:

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

REVISION HISTORY:

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

7.202 symveca (Source File: symveca.f90)

INTERFACE:

```
subroutine symveca(vca)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
vca : 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. Only the spatial part of each crystal symmetry is used.

REVISION HISTORY:

```
Created June 2004 (JKD)
```

7.203 timesec (Source File: timesec.f90)

INTERFACE:

```
subroutine timesec(ts)
```

INPUT/OUTPUT PARAMETERS:

```
ts : system time in seconds (out,real)
```

DESCRIPTION:

Outputs the system time in seconds.

REVISION HISTORY:

```
Created September 2010 (JKD)
```

7.204 vecfbz (Source File: vecfbz.f90)

INTERFACE:

```
subroutine vecfbz(eps,bvec,vpl)
```

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))
```

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)
```

7.205 vecplot (Source File: vecplot.f90)

INTERFACE:

```
subroutine vecplot
```

DESCRIPTION:

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)
```

7.206 wavefmt (Source File: wavefmt.f90)

INTERFACE:

```
subroutine wavefmt(lrstp,ias,ngp,apwalm,evecfv,wfmt)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
lrstp  : radial step length (in,integer)
ias    : joint atom and species number (in,integer)
ngp    : number of G+p-vectors (in,integer)
apwalm : APW matching coefficients (in,complex(ngkmax,apwordmax,lmmaxapw))
evecfv : first-variational eigenvector (in,complex(nmatmax))
wfmt   : complex muffin-tin wavefunction passed in as real array
         (out,real(2,*))
```

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 `eveqn`; $A^{\alpha}_{jlm}(\mathbf{G}+\mathbf{p})$ 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 `eveqn`.

REVISION HISTORY:

```
Created April 2003 (JKD)
Fixed description, October 2004 (C. Brouder)
Removed argument ist, November 2006 (JKD)
Changed arguments and optimised, December 2014 (JKD)
```

7.207 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 formulae for the $3j$-symbols, the following provides high accuracy for $j \leq 50$

$$\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} =$$

$$(-1)^{j1+j2+m3}\sqrt{\frac{(j_1+m_1)!\,(j_2+m_2)!\,(j_3+m_3)!\,(j_3-m_3)!\,(j_1-m_1)!\,(j_2-m_2)!}{(j_2-j_1+j_3)!\,(j_1-j_2+j_3)!\,(j_1+j_2-j_3)!\,(1+j_1+j_2+j_3)!}}\sum_k(-1)^k$$

$$\frac{(j_2-j_1+j_3)!\,(j_1-j_2+j_3)!\,(j_1+j_2-j_3)!}{(j_3-j_1-m_2+k)!\,(j_3-j_2+m_1+k)!\,(j_1+j_2-j_3-k)!\,k!\,(j_1-m_1-k)!\,(j_2+m_2-k)!},$$

where the sum is over all integers k for which the factorials in the summand are non-negative.

REVISION HISTORY:

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

7.208 wigner3jf (Source File: wigner3jf.f90)

INTERFACE:

```
real(8) function wigner3jf(j12,j22,j32,m12,m22,m32)
```

INPUT/OUTPUT PARAMETERS:

```
j12, j22, j32 : angular momentum quantum numbers times 2 (in,integer)
m12, m22, m32 : magnetic quantum numbers times 2 (in,integer)
```

DESCRIPTION:

Returns the Wigner $3j$-symbol for the case where the arguments may be fractional, i.e. multiples of $\frac{1}{2}$. The input parameters to this function are taken to be twice their actual values, which allows them to remain integers. The formula used is identical to that in `wigner3j`.

REVISION HISTORY:

```
Created January 2014 (JKD)
```

7.209 wigner6j (Source File: wigner6j.f90)

INTERFACE:

```
real(8) function wigner6j(j1,j2,j3,k1,k2,k3)
```

INPUT/OUTPUT PARAMETERS:

```
j1, j2, j3 : angular momentum quantum numbers (in,integer)
k1, k2, k3 : angular momentum quantum numbers (in,integer)
```

DESCRIPTION:

Returns the Wigner $6j$-symbol for integer arguments. This is computed using the Racah formula:

$$\begin{Bmatrix} j_1 & j_2 & j_3 \\ k_1 & k_2 & k_3 \end{Bmatrix} = \sqrt{\Delta(j_1\, j_2\, j_3)\Delta(j_1\, k_2\, k_3)\Delta(k_1\, j_2\, k_3)\Delta(k_1\, k_2\, j_3)} \sum_n \frac{(-1)^n (n+1)!}{f(n)},$$

where

$$\begin{aligned} f(n) =& (n-j_1-j_2-j_3)!\,(n-j_1-k_2-k_3)!\,(n-k_1-j_2-k_3)!\,(n-k_1-k_2-j_3)! \\ & \times (j_1+j_2+k_1+k_2-n)!\,(j_2+j_3+k_2+k_3-n)!\,(j_1+j_3+k_1+k_3-n)! \end{aligned}$$

and

$$\Delta(a,b,c) = \frac{(a+b-c)!\,(a-b+c)!\,(-a+b+c)!}{(a+b+c+1)!}$$

is the triangle coefficient, and where the sum is over all integers n for which the factorials in $f(n)$ have non-negative arguments. The Wigner-$6j$ function is zero unless each triad, $(j_1\, j_2\, j_3)$, $(j_1\, k_2\, k_3)$, $(k_1\, j_2\, k_3)$ and $(k_1\, k_2\, j_3)$, satifies the triangle inequalities:

$$|x-y| \le z \le x+y,$$

for triad (x,y,z). See, for example, A. Messiah, *Quantum Mechanics*, Vol. 2., 1061-1066 (1962).

REVISION HISTORY:

```
Created August 2009 (JKD)
```

7.210 writeefdu (Source File: writeefdu.f90)

INTERFACE:

```
subroutine writeefdu
```

USES:

```
use modmain
use moddftu
```

DESCRIPTION:

Writes to file the linearisation energies for all radial functions used to calculate the Slater integrals through a Yukawa potential.

REVISION HISTORY:

Created July 2008 (Francesco Cricchio)

7.211 writeefg (Source File: writeefg.f90)

INTERFACE:

```
subroutine writeefg
```

USES:

```
use modmain
use modtest
```

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)

7.212 writeeval (Source File: writeeval.f90)

INTERFACE:

```
subroutine writeeval
```

USES:

```
use modmain
```

DESCRIPTION:

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

REVISION HISTORY:

Created June 2003 (JKD)

7.213 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)
```

7.214 writegclq (Source File: writegclq.f90)

INTERFACE:

```
subroutine writegclq
```

USES:

```
use modmain
```

DESCRIPTION:

Outputs the volume-averaged integral of $4\pi/q^2$ in the small parallelepiped around each discrete q-point to the file `GCLQ.OUT`. These represent the regularised Coulomb Green's function in reciprocal space for small q. See the routine gengclq.

REVISION HISTORY:

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

7.215 writegeom (Source File: writegeom.f90)

INTERFACE:

```
subroutine writegeom(fnum)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
fnum : file number for writing output (in,integer)
```

DESCRIPTION:

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

REVISION HISTORY:

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

7.216 writeiad (Source File: writeiad.f90)

INTERFACE:

```
subroutine writeiad(fnum)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
fnum : file number for writing output (in,integer)
```

DESCRIPTION:

Outputs the interatomic distances to file.

REVISION HISTORY:

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

7.217 writeinfo (Source File: writeinfo.f90)

INTERFACE:

```
subroutine writeinfo(fnum)
```

USES:

```
use modmain
use moddftu
use modrdm
use modxcifc
use modmpi
```

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)
Updated with DFT+U quantities July 2009 (FC)
```

7.218 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)
```

7.219 writelinen (Source File: writelinen.f90)

INTERFACE:

```
subroutine writelinen
```

USES:

```
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)
```

7.220 writepmat (Source File: writepmat.f90)

INTERFACE:

```
subroutine writepmat
```

USES:

```
use modmain
use modmpi
```

DESCRIPTION:

Calculates the momentum matrix elements using routine `genpmat` and writes them to direct access file `PMAT.OUT`.

REVISION HISTORY:

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

7.221 writestate (Source File: writestate.f90)

INTERFACE:

```
subroutine writestate
```

USES:

```
use modmain
use moddftu
```

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)
```

7.222 writesym (Source File: writesym.f90)

INTERFACE:

```
subroutine writesym
```

USES:

```
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)
```

7.223 writetm2du (Source File: writetm2du.f90)

INTERFACE:

```
subroutine writetm2du(fnum)
```

USES:

```
use modmain
use moddftu
use modtest
```

DESCRIPTION:

Decompose the density matrix and the DFT+U Hartree-Fock energy in 2-index tensor moments components, see *Phys. Rev. B* **80**, 035121 (2009).

REVISION HISTORY:

```
Created October 2009 (F. Cricchio and L. Nordstrom)
```

7.224 writetm3du (Source File: writetm3du.f90)

INTERFACE:

```
subroutine writetm3du(fnum)
```

USES:

```
use modmain
use moddftu
use modtest
```

DESCRIPTION:

Decompose the density matrix and the DFT+U Hartree-Fock energy in 3-index tensor moments components, see *Phys. Rev. B* **80**, 035121 (2009).

REVISION HISTORY:

```
Created April 2008 (F. Cricchio and L. Nordstrom)
```

7.225 writevclijji (Source File: writevclijji.f90)

INTERFACE:

```
subroutine writevclijji
```

USES:

```
use modmain
use modmpi
use modomp
```

DESCRIPTION:

Generates Coulomb matrix elements of the type $(i-jj-i)$ and outputs them to the file VCLIJJI.OUT.

REVISION HISTORY:

Created 2008 (Sharma)

7.226 writevclijjk (Source File: writevclijjk.f90)

INTERFACE:

```
subroutine writevclijjk
```

USES:

```
use modmain
use modmpi
use modomp
```

DESCRIPTION:

Generates Coulomb matrix elements of the type $(i-jj-k)$ and outputs them to the file VCLIJJK.OUT. Also writes the real diagonal of this matrix, $(i-jj-i)$, to VCLIJJI.OUT.

REVISION HISTORY:

Created 2008 (Sharma)

7.227 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
```

7.228 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)
```

7.229 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)
```

7.230 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)
```

7.231 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)
```

7.232 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)
```

7.233 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)
Rewrote, October 2011 (JKD)
```

7.234 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)
```

7.235 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)
```

7.236 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)
```

7.237 ylmrot (Source File: ylmrot.f90)

INTERFACE:

```
subroutine ylmrot(p,ang,lmax,ld,d)
```

INPUT/OUTPUT PARAMETERS:

```
p    : if p=-1 then the rotation matrix is improper (in,integer)
ang  : Euler angles; alpha, beta, gamma (in,real(3))
lmax : maximum angular momentum (in,integer)
ld   : leading dimension (in,integer)
d    : complex spherical harmonic rotation matrix (out,complex(ld,*))
```

DESCRIPTION:

Returns the rotation matrix in the basis of complex spherical harmonics given the three Euler angles, (α, β, γ), and the parity, p, of the rotation. The matrix is given by the formula

$$D^l_{m_1 m_2}(\alpha, \beta, \gamma) = d^l_{m_1 m_2}(\beta) e^{-i(m_1 \alpha + m_2 \gamma)},$$

where d is the rotation matrix about the y-axis. For improper rotations, i.e. those which are a combination of a rotation and inversion, D is modified with $D^l_{m_1 m_2} \rightarrow (-1)^l D^l_{m_1 m_2}$. See the routines `roteuler` and `ylmroty`.

REVISION HISTORY:

```
Created December 2008 (JKD)
```

7.238 ylmroty (Source File: ylmroty.f90)

INTERFACE:

```
subroutine ylmroty(beta,lmax,ld,dy)
```

INPUT/OUTPUT PARAMETERS:

```
beta : rotation angle about y-axis (in,real)
lmax : maximum angular momentum (in,integer)
ld   : leading dimension (in,integer)
dy   : rotation matrix for complex spherical harmonics (out,real(ld,*))
```

DESCRIPTION:

Returns the rotation matrix in the basis of complex spherical harmonics for a rotation of angle β about the y-axis. This matrix is real and is given by the formula

$$d^l_{m_1m_2}(\beta) = [(l+m_1)!(l-m_1)!(l+m_2)!(l-m_2)!]^{1/2} \times \sum_k (-1)^k \frac{\left(\cos\frac{\beta}{2}\right)^{2(l-k)-m_2+m_1} \left(\sin\frac{\beta}{2}\right)^{2k+m_2-m_1}}{k!(l+m_1-k)!(l-m_2-k)!(m_2-m_1+k)!},$$

where k runs through all integer values for which the factorials exist.

REVISION HISTORY:

```
Created December 2008 (JKD)
```

7.239 z2mctm (Source File: z2mctm.f90)

INTERFACE:

```
pure 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)
```

7.240 z2mmct (Source File: z2mmct.f90)

INTERFACE:

```
pure 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)
```

7.241 z2mm (Source File: z2mm.f90)

INTERFACE:

```
pure 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)
```

7.242 zbessela (Source File: zbessela.f90)

INTERFACE:

```
subroutine zbessela(lmax,x,a)
```

INPUT/OUTPUT PARAMETERS:

```
lmax : maximum order of Bessel function (in,integer)
x    : real argument (in,real)
a    : array of returned values (out,real(0:lmax))
```

DESCRIPTION:

Computes variations of the spherical Bessel function, $a_l(x) = i^l j_l(ix)$, for real 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 upwards. For starting values there are

$$a_0(x) = \frac{\sinh(x)}{x}; \qquad a_1(x) = \frac{a_0(x) - \cosh(x)}{x}$$

. For $x \ll 1$ the asymtotic forms

$$a_l(x) \approx \frac{(-x)^l}{(2l+1)!!}$$

are used.

REVISION HISTORY:

```
Created April 2008 from sbessel routine (Lars Nordstrom)
```

7.243 zbesselb (Source File: zbesselb.f90)

INTERFACE:

```
subroutine zbesselb(lmax,x,b)
```

INPUT/OUTPUT PARAMETERS:

```
lmax : maximum order of Bessel function (in,integer)
x    : real argument (in,real)
b    : array of returned values (out,real(0:lmax))
```

DESCRIPTION:

Computes variations of the spherical Bessel function $b_l(x) = i^l h_l^{(1)}(ix)$, for real 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 upwards. For starting values there are

$$b_0(x) = -\frac{e^{-x}}{x}; \qquad b_1(x) = b_0(x) \left\{ 1 + \frac{1}{x} \right\}.$$

For $x \ll 1$ the asymtotic forms

$$b_l(x) \approx \frac{-(2l-1)!!}{(-x)^{l+1}}$$

are used.

REVISION HISTORY:

```
Created April 2008 from sbessel routine (Lars Nordstrom)
```

7.244 zbsht (Source File: zbsht.f90)

INTERFACE:

```
subroutine zbsht(nr,nri,zfmt1,zfmt2)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
nr    : number of radial mesh points (in,integer)
nri   : number of points on the inner part of the muffin-tin (in,integer)
zfmt1 : input complex muffin-tin function in spherical harmonics
        (in,complex(*))
zfmt2 : output complex muffin-tin function in spherical coordinates
        (out,complex(*))
```

DESCRIPTION:

Performs a backward spherical harmonic transform (SHT) on a complex muffin-tin function expressed in spherical harmonics to obtain a function in spherical coordinates. See also `genshtmat` and `zfsht`.

REVISION HISTORY:

```
Created October 2013 (JKD)
```

7.245 zfinp (Source File: zfinp.f90)

INTERFACE:

```
complex(8) function zfinp(zfmt1,zfir1,zfmt2,zfir2)
```

USES:

```
use modmain
use modomp
```

INPUT/OUTPUT PARAMETERS:

```
zfmt1 : first complex function in spherical harmonics/coordinates for all
        muffin-tins (in,complex(npcmtmax,natmtot))
zfir1 : first complex interstitial function in real-space
        (in,complex(ngtc))
zfmt2 : second complex function in spherical harmonics/coordinates for all
        muffin-tins (in,complex(npcmtmax,natmtot))
zfir2 : second complex interstitial function in real-space
        (in,complex(ngtc))
```

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. In the interstitial 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)
```

7.246 zflmnconj (Source File: zfmtconj.f90)

INTERFACE:

```
pure subroutine zflmnconj(lmax,n,ld,zflm1,zflm2)
```

INPUT/OUTPUT PARAMETERS:

```
lmax  : maximum angular momentum (in,integer)
n     : number of functions to conjugate (in,integer)
ld    : leading dimension (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).

REVISION HISTORY:

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

7.247 zfmtinp (Source File: zfmtinp.f90)

INTERFACE:

```
complex(8) function zfmtinp(nr,nri,wr,zfmt1,zfmt2)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
nr    : number of radial mesh points (in,integer)
nri   : number of points on the inner part of the muffin-tin (in,integer)
wr    : weights for integration on radial mesh (in,real(nr))
zfmt1 : first complex muffin-tin function in spherical harmonics
        (in,complex(*))
zfmt2 : second complex muffin-tin function (in,complex(*))
```

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

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

REVISION HISTORY:

```
Created November 2003 (Sharma)
Modified, September 2013 (JKD)
Modified for packed functions, June 2016 (JKD)
```

7.248 zfsht (Source File: zfsht.f90)

INTERFACE:

```
subroutine zfsht(nr,nri,zfmt1,zfmt2)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
nr    : number of radial mesh points (in,integer)
nri   : number of points on the inner part of the muffin-tin (in,integer)
zfmt1 : input complex muffin-tin function in spherical coordinates
        (in,complex(*))
zfmt2 : output complex muffin-tin function in spherical harmonics
        (out,complex(*))
```

DESCRIPTION:

Performs a forward spherical harmonic transform (SHT) on a complex muffin-tin function in spherical coordinates to obtain a function expressed in spherical harmonics. See also `genshtmat` and `zbsht`.

REVISION HISTORY:

```
Created October 2013 (JKD)
```

7.249 zftrf (Source File: zftrf.f90)

INTERFACE:

```
subroutine zftrf(npv,ivp,vpc,rfmt,rfir,zfp)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
npv  : number of P-vectors (in,integer)
ivp  : integer coordinates of the P-vectors (in,integer(3,npv))
vpc  : P-vectors in Cartesian coordinates (in,real(3,npv))
rfmt : real muffin-tin function (in,real(npmtmax,natmtot))
rfir : real interstitial function (in,real(ngtot))
zfp  : Fourier expansion coefficients of the real-space function
       (out,complex(npv))
```

DESCRIPTION:

Given a real function periodic in the unit cell, $f(\mathbf{r})$, this routine calculates its complex Fourier expansion coefficients:

$$f(\mathbf{P}) = \frac{1}{\Omega}\int d^3r\, f(\mathbf{r})\tilde{\Theta}(\mathbf{r})e^{-i\mathbf{P}\cdot\mathbf{r}} + \frac{4\pi}{\Omega}\sum_{\alpha} e^{-i\mathbf{P}\cdot\mathbf{R}_\alpha}\sum_{lm}(-i)^l Y_{lm}(\hat{\mathbf{P}})\int_0^{R_\alpha} dr\, r^2 j_l(|\mathbf{P}|r) f^\alpha_{lm}(r),$$

where $\tilde{\Theta}$ is the smooth characteristic function of the interstitial region, Ω is the unit cell volume and R_α is the muffin-tin radius of atom α.

REVISION HISTORY:

```
Created July 2010 (Alexey I. Baranov)
Modified, November 2010 (JKD)
```

7.250 zpotclmt (Source File: zpotclmt.f90)

INTERFACE:

```
pure subroutine zpotclmt(nr,nri,ld,rl,wpr,zrhomt,zvclmt)
```

USES:

```
use modmain
```

INPUT/OUTPUT PARAMETERS:

```
nr     : number of radial mesh points (in,integer)
nri    : number of points on inner part of muffin-tin (in,integer)
ld     : leading dimension (in,integer)
rl     : r^l on the radial mesh (in,real(ld,-lmaxo-1:lmaxo+2))
wpr    : weights for partial integration on radial mesh (in,real(4,nr))
zrhomt : muffin-tin charge density (in,complex(*))
zvclmt : muffin-tin Coulomb potential (out,complex(*))
```

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

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

where R is the muffin-tin radius.

REVISION HISTORY:

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

7.251 zpotcoul (Source File: zpotcoul.f90)

INTERFACE:

```
subroutine zpotcoul(nr,nri,np,npi,ld1,rl,ngdg,igf,ngp,gpc,gclgp,ld2,jlgprmt, &
 ylmgp,sfacgp,zrhoir,ld3,zvclmt,zvclir)
```

USES:

```
use modmain
use modphonon
```

INPUT/OUTPUT PARAMETERS:

```
nr      : number of radial points for each species (in,integer(nspecies))
nri     : number of radial points on inner part (in,integer(nspecies))
np      : total number of points in muffin-tins (in,integer(nspecies))
npi     : number of points on inner part (in,integer(nspecies))
ld1     : leading dimension (in,integer)
rl      : r^l on radial mesh for each species
          (in,real(ld1,-lmaxo-1:lmaxo+2,nspecies))
ngdg    : G-vector grid sizes (in,integer(3))
igf     : map from G-vector index to FFT array (in,integer(*))
ngp     : number of G+p-vectors (in,integer)
gpc     : G+p-vector lengths (in,real(ngp))
gclgp   : Coulomb Green's function in G+p-space (in,real(ngp))
ld2     : leading dimension (in,integer)
jlgprmt : spherical Bessel functions for evergy G+p-vector and muffin-tin
          radius (in,real(0:lnpsd,ld2,nspecies))
ylmgp   : spherical harmonics of the G+p-vectors (in,complex(lmmaxo,ngp))
sfacgp  : structure factors of the G+p-vectors (in,complex(ld2,natmtot))
zrhoir  : interstitial charge density (in,complex(*))
ld3     : leading dimension (in,integer)
zvclmt  : muffin-tin Coulomb potential, with the contribution from the
          isolated muffin-tin density precalculated and passed in
          (inout,complex(ld3,natmtot))
zvclir  : interstitial Coulomb potential (out,complex(*))
```

DESCRIPTION:

Calculates the Coulomb potential of a complex charge density by solving Poisson's equation using the method of M. Weinert, *J. Math. Phys.* **22**, 2433 (1981). First, the multipole moments of the muffin-tin charge are determined for the jth atom of the ith species by

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

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

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

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

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

where

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

and $N_i \approx \frac{1}{4}R_i G_{\max}$ 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)
```

7.252 ztorflmn (Source File: ztorfmt.f90)

INTERFACE:

```
pure subroutine ztorflmn(lmax,n,ld,zflm,rflm)
```

INPUT/OUTPUT PARAMETERS:

```
lmax : maximum angular momentum (in,integer)
n    : number of functions to convert (in,integer)
ld   : leading dimension (in,integer)
zflm : coefficients of complex spherical harmonic expansion
       (in,complex(ld,n))
rflm : coefficients of real spherical harmonic expansion (out,real(ld,n))
```

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)
```