Implementation of Bader theory in WIEN package

J. O. Sofo and G. N. García

October 24, 2000

1 Representation of the Charge Density in the Package WIEN97

This section summarizes the way the charge density is calculated by lapw5 from the coefficients stored in clmsum or clmval.

The charge density is represented by a plane wave expansion in the interstitial region (I) and as the combination of a radial function times spherical harmonics inside the muffin-tin spheres, in this way,

$$\rho(\vec{r}) = \begin{cases} \sum_{\vec{G}} \rho_{\vec{G}}\ e^{i\vec{G}\cdot\vec{r}} & \vec{r} \in \text{I} \\ \sum_{lm} \rho_{lm}(r)\ Y_{lm}(\theta,\phi) & \vec{r} \notin \text{I} \end{cases} \tag{1}$$

The subroutine `main1` [1] reads the mesh where the charge density is going to be calculated from `case.in5`. The coefficients of the charge density expansion are stored in `case.clmsum`. The first part of this file contains the coefficients of the spherical expansion and the last part the representative reciprocal lattice vector of each star and the corresponding coefficient. This former part is read in `outin`, where the stars are also rebuilt.

1.1 The charge density calculation in the interstitial region

When the point $\vec{r}$ is in the interstitial region the charge density is calculated as the Fourier expansion shown in Eq. (1) by the routine `rhoout`. This is a very simple routine that performs the summation over $\vec{G}$ space of the coefficients.

The summation over $\vec{G}$ is done over stars of $\vec{G}$. In the file clmsum, after the coefficients of the expansion inside the spheres, `NK` lines are stored with the $\vec{G}$ and the corresponding $\rho_{\vec{G}}$. These are not all the $\vec{G}$ included in the summation,

[1] All the subroutine names refer to the files located in SRC_lapw5 of the WIEN97 distribution and are written in `typewriter` font.

these are the representatives of each star. When these lines are read in `outin` the star for each one of these representatives is built by `stern`.

The stars are built applying each of the rotations in the symmetry group (`COMMON /SYM2/`) to the representative $\vec{G}$. In this way, `INST(I)` new $\vec{G}$ are created and stored in `KREC` (first member of `COMMON /OUT/`). In this process of creating the starts, some symmetry operations map the representative onto the same star member, for these symmetry operations the summation $\tau(\vec{G})$ has to be done as

$$\tau(\vec{G}') = \frac{1}{\texttt{INST(I)}} \sum_R e^{i\vec{G}'\cdot\vec{t}_R} \ ,$$

where the summation is done over all the symmetry operations that map $\vec{G}$ onto the same $\vec{G}'$ and the normalization with the number of elements of the star `INST(I)` is included here. These $\tau(\vec{G}')$ are stored in `TAUK` (fourth member of `COMMON /OUT/`).

With the stars rebuilt the summation of the Fourier series is done as

$$\rho(\vec{r}) = \sum_{i\in\text{stars}} \rho_i \sum_{\vec{G}\in\text{star i}} e^{i\vec{G}\cdot\vec{r}}\tau(\vec{G}) \ . \tag{2}$$

1.2 The charge density calculation inside the muffin-tin spheres

When `main1` determines that the point $\vec{r}$ where the charge density is to be calculated falls inside a muffin-tin sphere (`inter` is false) the following steps are performed:

- The point $\vec{r}$ is rotated using the symmetry operation that maps the atom where $\vec{r}$ fell close to the representative atom. This is done taking car of the ortho switch.
- The point $\vec{r}$ is reduced to the smallest possible with `reduc`. (No rotation performed here)
- The local rotation matrix is applied to the point.
- The index i_r of $r = |\vec{r}|$ in the logarithmic radial grid is calculated through

$$i_r = 1 + \frac{\ln\left(\frac{r}{R_0(j)}\right)}{\Delta X(j)} \ . \tag{3}$$

  Here j is the index of the inequivalent atom, $\Delta X(j)$ the mesh separation given by

$$\Delta X(j) = \frac{\ln\left(\frac{R_{MT}(j)}{R_0(j)}\right)}{n-1} \ , \tag{4}$$

  where $R_0(j)$ is the first radial mesh point, $R_{MT}(j)$ the muffin-tin radius, and n the number of radial mesh points for atom j as read form the `struct` file.

- The module `charge` is called, where the summation over lm is done as

$$\rho(r,\theta,\phi) = \sum_{lm=1}^{\texttt{LMMAX}} \rho_{lm}(r)\ \Lambda_{lm}(\theta,\phi)\ , \tag{5}$$

  with $\rho(r,\theta,\phi)$ stored in `CHG`, $\rho_{lm}(r)$ stored in `RHO(ILM)`, and $\Lambda_{lm}(\theta,\phi)$ stored in `ANG(ILM)`. To perform this sum the code follows this steps:

  - The spherical harmonics $Y_l^m(\theta,\phi)$ are calculated in `ylm` using a recursion method and stored in `YL`$(l(l+1)+m+1)$.
  - For each lm pair $\rho_{lm}(r)$ is calculated by `radial` interpolating the `CLM` read from `clmsum` and dividing by r^2.
  - In the same loop $\Lambda_{lm}(\theta,\phi)$ is calculated as

$$\Lambda_{lm} = \begin{cases} Y_l^m & \text{if } m = 0\ , \\ \dfrac{i\left((-1)^{m+1}Y_l^m + Y_l^{-m}\right)}{\sqrt{2}} & \text{if } m \neq 0 \text{ and } l < 0\ , \\ \dfrac{\left((-1)^{m}Y_l^m + Y_l^{-m}\right)}{\sqrt{2}} & \text{if } m \neq 0 \text{ and } l > 0\ , \end{cases}$$

    where l and m are stored in `LM`$(1,$`ILM`$,j)$ and `LM`$(2,$`ILM`$,j)$ respectively, and read from `in2`.
  - Finally the summation of Eq. (5) is performed talking care if the local symmetry of the atom is cubic or not.

- With the charge density stored in `CHG`, `main1` writes it to a temporary unformatted file (unit 10).

2 Calculation of the charge density gradient

Using `lapw5` as our starting point, we have written a program, called `bader`, which adds to the functionality of `lapw5` a switch `GRAD` to calculate the charge density gradient. In this section we describe the details of the implementations of this switch.

The input files are read by `main1` as before and the decision is made if the point where the charge density or gradient are to be calculated falls inside or outside the muffin tins. If the point is interstitial, $\nabla\rho(\vec{r})$ is calculated inside `grhoinst`, if the point is inside a muffin tin, the calculation is done in `grhosphe`. These routines are described in the following sub-sections. After the gradient is returned, it is projected on the plane where $\vec{r}$ is constrained.

2.1 Gradient of the charge density in the interstitial region

Before calling `grhoinst` to calculate $\nabla\rho(\vec{r})$ in the interstitial region, `main1` takes care of the normalization difference between `ortho` false and true. If `ortho` is true, $\vec{r}$ is given to `grhoinst` in units of the lattice constants and the $\vec{G}$'s in units of the inverse of lattice constants. On the other hand, if `ortho` is false, $\vec{r}$ is given in Bohr and the $\vec{G}$'s in Bohr^{-1}. This difference in the treatment has to be taken into account to correct the units once `grhoinst` returns the gradient.

In `grhoinst` the calculation of the gradient is very simple, the derivative of the charge density in the interstitial region is given by the gradient of ρ given by Eq. (2), as

$$\nabla\rho(\vec{r}) = \sum_{i\in\text{stars}} \rho_i \sum_{\vec{G}\in\text{star i}} i\vec{G}\, e^{i\vec{G}\cdot\vec{r}}\, \tau(\vec{G})\ . \tag{6}$$

In case of a real calculation, with inversion symmetry, the charge density is calculated using the real part of Eqs. (2) and (6). This saves the space required for complex storage.

2.2 Gradient of the charge density inside the spheres

In the case of the charge density inside the spheres, before calling `grhosphe` the vector $\vec{r}$ is rotated twice. First, a symmetry rotation is applied that maps the atom where $\vec{r}$ fell close to, onto the representative atom. Second, the local rotation matrix for that atom is applied. After the vector $\nabla\rho$ is returned by `grhosphe` this rotations have to be reversed, this is done by `rotat_back` for the local rotation matrix, and by `rotate_back` for the symmetry rotation.

The calculation of the gradient charge density in `grhosphe` is done in four parts: the initialization part, a loop over `ilm` with the calculation of the radial and angular parts of the expansion and its derivatives, the summation over ilm, and the transformation to Cartesian coordinates.

During the initialization part we calculate the spherical harmonics with a call to `ylm`, the derivative of the spherical harmonics with respect to θ in `dtylm`, and the matrix `change` that maps the derivatives of ρ with respect to r, θ, and ϕ to its derivatives respect to x, y, and z. Details on the calculation of $\partial_\theta Y_l^m(\theta,\phi)$ are given in Appendix A. The storage of $\partial_\theta Y_l^m(\theta,\phi)$ is similar to the one used to store the Y_l^m, i.e. $\partial_\theta Y_l^m(\theta,\phi)$ is stored in `dtyl`$(l(l+1)+m+1)$.

In the loop over lm the values of ρ_{lm}, $\partial_r\rho_{lm}$, Λ_{lm}, $\partial_\theta\Lambda_{lm}$, and $\partial_\phi\Lambda_{lm}$ are obtained and stored in `rho(ilm)`, `drrho(ilm)`, `ang(ilm)`, `dtang(ilm)`, and `dfang(ilm)` respectively.

The sum over lm is done to calculate the partial derivatives of the charge density respect to r. θ, and ϕ as,

$$\partial_r\rho \;=\; \sum_{lm=1}^{\texttt{LMMAX}} \partial_r\rho_{lm}(r)\ \Lambda(\theta,\phi) \tag{7}$$

$$\partial_\theta \rho = \sum_{lm=1}^{\texttt{LMMAX}} \rho_{lm}(r)\ \partial_\theta \Lambda(\theta,\phi) \tag{8}$$

$$\partial_\phi \rho = \sum_{lm=1}^{\texttt{LMMAX}} \rho_{lm}(r)\ \partial_\phi \Lambda(\theta,\phi)\ , \tag{9}$$

the partial derivatives of ρ in spherical coordinates are stored in the vector `dscrho`.

Finally, the transformation to Cartesian coordinates is done. If we call $u_1 = r$, $u_2 = \theta$, $u_3 = \phi$, $x_1 = x$, $x_2 = y$, and $x_3 = z$, the components of the gradient in Cartesian coordinates $\partial\rho/\partial x_i$ are obtained as

$$\frac{\partial \rho}{\partial x_i} = \sum_{j=1}^{3} \frac{\partial \rho}{\partial u_j}\frac{\partial u_j}{\partial x_i}\ ,$$

The terms $\partial u_j/\partial x_i$ are stored in `change(j,i)` and are given by

$$\frac{\partial r}{\partial x} = \sin\theta\cos\phi \quad \frac{\partial \theta}{\partial x} = \frac{\cos\theta\cos\phi}{r} \quad \frac{\partial \phi}{\partial x} = -\frac{\sin\phi}{r\sin\theta}$$

$$\frac{\partial r}{\partial y} = \sin\theta\sin\phi \quad \frac{\partial \theta}{\partial y} = \frac{\cos\theta\sin\phi}{r} \quad \frac{\partial \phi}{\partial y} = \frac{\cos\phi}{r\sin\theta}$$

$$\frac{\partial r}{\partial z} = \cos\theta \quad \frac{\partial \theta}{\partial z} = \frac{\sin\theta}{r} \quad \frac{\partial \phi}{\partial z} = 0$$

This expressions are coded in `gen_change`, where `change` is loaded.

A Derivatives of the spherical harmonics

The expression for the spherical harmonics is [1]

$$Y_l^m(\theta,\phi) = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}\ P_l^m(\cos\theta)\ e^{im\phi}\ .$$

The derivative of the spherical harmoinics with respect to ϕ is just

$$\partial_\phi Y_l^m(\theta,\phi) = imY_l^m(\theta,\phi)$$

and does not need any special consideration.

The derivative respect to θ is essentialy the derivative of the associated Legendre polynomial

$$\partial_\theta Y_l^m(\theta,\phi) = -\sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}\ \left(\frac{dP_l^m(x)}{dx}\right)_{x=\cos\theta} \sin\theta\ e^{im\phi}\ ,$$

and from the definition of these polynomials

$$P_l^m(x) = (-1)^m \left(1-x^2\right)^{m/2} \frac{d^m}{dx^m} P_l^0(x)$$

the derivative can be evaluated as

$$\frac{dP_l^m(x)}{dx} = -\frac{mx}{1-x^2} P_l^m(x) - \frac{1}{(1-x^2)^{1/2}} P_l^{m+1}(x) \ .$$

Replacing this derivative in the expresion for the derivative of the spherical harmonics we get

$$\partial_\theta Y_l^m(\theta,\phi) = m\frac{\cos\theta}{\sin\theta} Y_l^m(\theta,\phi) + e^{-i\phi}\sqrt{l(l+1)-m(m+1)}\, Y_l^{m+1}(\theta,\phi) \ .$$

This is the expresion we coded in `dtylm`. In this expresion, there is a detail to be taken into account regrading the limit when θ is zero. In this limit the second member on the right is zero, because the spherical harmonics is zero. The first member instead has a non-zero limit if $|m| = 1$ and zero otherwise. In the case $\theta = 0$, the expression

$$\partial Y_l^m(0,0) = \begin{cases} -m\dfrac{\sqrt{l(l+1)(2l+1)}}{4\sqrt{\pi}} & \text{if } |m| = 1 \ , \\ \\ 0 & \text{other case} \end{cases}$$

is used.

References

[1] W. H. Press, S. A. Teuklosky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing* (Cambridge University Press, Cambridge, UK, 1992) p. 252.