Notes on the CRPA calculation in Exciting Plus

Robert G. Van Wesep

(Dated: October 26, 2016)

CONTENTS

I. INTRODUCTION

The constrained random-phase approximation is a way to calculate a model from first-principles for a subset of the total (infinite) number of degrees of freedom present in the solid-state problem. The degrees of freedom for which the model is calculated are called the *target* and all other degrees of freedom are called the *rest*. The target and rest are defined by single-particle states that span the respective subspaces.

One way to visualize the target and rest is to examine the single-particle band structure. Shown in Fig. 1 is the band structure of FeTe shown over a large energy range. The red and blue lines represent the eigenvalues found from solving the groundstate problem for **k** points along high symmetry lines in the first Brillouin zone. The blue lines mark the eigenvalues for those degrees of freedom that are in the target while the red are for those in the rest. The energy range shown is one typical for a converged CRPA calculation.

The goal of the CRPA calculation is to calculate the interactions between the target degrees of freedom which are screened by the rest. Though the physics is quite current and sophisticated, the mathematics is mostly basic linear algebra and the algorithm is fairly uncomplicated. In these notes, I will focus on the algorithmic implementation of the method and include descriptions of the physics only where necessary or when I cannot help myself.

I will structure these notes as follows. Each section will focus on a particular subroutine.

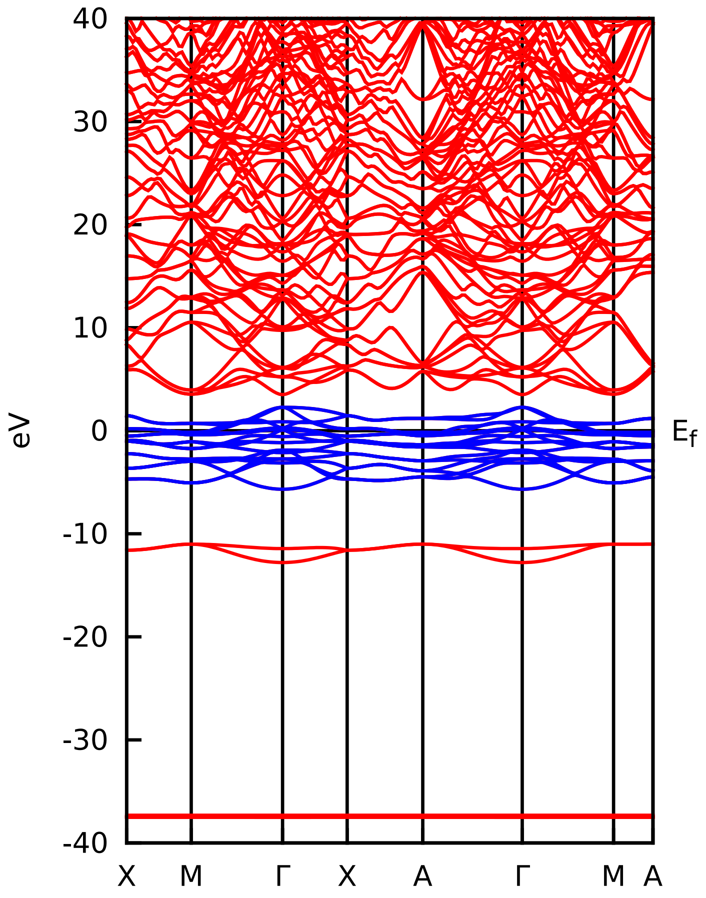

FIG. 1. Band structure for FeTe with the target highlighted in blue and the rest in red.

I will try to include all of the important ones and more may be added as I revise these notes. I will start with the highest level subroutine and then get progressively deeper to the lower level routines called by the high level ones. In discussing the top level routine I hope to give a global overview of the method with the essential equations. These notes will be updated and more added over time, so please stay tuned!

II. CRPA() IN ADDONS/CRPA.F90

The highest level subroutine for the CRPA calculation is `crpa()`. It is called from the big `select` statement in `main()`:

```
case(801)
  call crpa
```

This is how all of the calculations in Exciting Plus are accessed. For any number listed in the `tasks` input block in `elk.in`, the corresponding subroutine is called.

Now I will move step by step and highlight important parts of the subroutine. I will not go over every line, but if anything is not included that you would like to know about, please let me know. First, all of the infrastructure that is used by virtually all Exciting Plus calculations (global variables, atomic information, **k**-mesh, etc.) must be initialized:

```
call init0
call init1
```

A (perhaps *the*) major quantity in the CRPA calculation is the constrained, screened interaction $W^{\mathrm{c}}_{\mathbf{GG}'}(\mathbf{q},\omega)$. I will have much more to say about this quantity throughout the discussion in these notes, but the first thing to say is that the point $\mathbf{q}=0$ must be treated carefully because there is a singularity:

$$\lim_{\mathbf{q}\to\mathbf{0}} W^{\mathrm{c}}_{\mathbf{00}}(\mathbf{q},\omega) \to \infty. \tag{1}$$

This divergence is completely physical and related to the long-range behavior of the Coulomb interaction. As long as it is treated with care it should not cause a problem.

The first sign on this careful treatment are in the lines:

```
if (screenu4) then
  call init_q_mesh(8)
else
  call init_q_mesh(1)
endif
```

The first thing to note is that `screenu4` will almost always be set to `.true.`. This flag tells the code to calculate the constrained screened interaction W^{c} rather than use the bare Coulomb interaction. Warning: `screenu4=.false.` may not work properly, so please contact me if you would like to use this flag.

Thus, `init_q_mesh(8)` will be called. Passing `8` to this subroutine tells it to create a small box around $\mathbf{q}=0$ and add the **q**-points at the corners of these boxes to the list of **q**-points for which $W^{\mathrm{c}}_{\mathbf{GG}'}(\mathbf{q},\omega)$ will be calculated.

The CRPA calculation depends on the calculation of the groundstate. The groundstate is run using `0` or `1` in the `tasks` block of `elk.in`. Use `0` if you are starting a groundstate from scratch and `1` if you are restarting a previous groundstate calculation. Some inputs can be changed when restarting a groundstate and some cannot. Please ask me if you need details. You do not have to run the groundstate in the same run as the CRPA calculation and you probably should not. The groundstate only has to have been run and the output files placed in the same directory as the CRPA calculation is being run in. This is because the potential and eigenvectors and eigenvalues are read into the CRPA calculation:

```
! read the density and potentials from file
call readstate
call genradf
...
wproc=wproc1
inquire(file="wfnrkp.hdf5",exist=exist)
if (exist) then
  call timer_start(1,reset=.true.)
  call drc_read_wf
  call timer_stop(1)
  if (wproc1) then
    write(151,*)
    write(151,'("drc_read_wf done in ",F8.2," seconds")')timer_get_value(1)
    call flushifc(151)
  endif
else
! generate wave-functions for entire BZ
  call genwfnr(151,tq0bz)
endif
```

The potential and electron density are read by `readstate` and `genradf` generates radial functions that are used to construct the LAPW basis in which the eigenvectors are expressed. As it stands now, there will not be a `wfnrkp.hdf5` file in the working directory. I believe this is part of an improved I/O implementation by Anton (Kozhevnikov), but I do not know the status of it. In the current state of the calculation, `genwfnr` will be called.

The CRPA algorithm does not stop with the calculation of $W^{\mathrm{c}}_{\mathbf{G}\mathbf{G}'}(\mathbf{q},\omega)$. Instead, it calculates this quantity in a basis of localized orbitals, called Wannier functions. What is important for the CRPA calculation are the matrix elements of $\mathrm{e}^{-\mathrm{i}(\mathbf{q}+\mathbf{G})\cdot\mathbf{x}}$ in the Wannier basis. These are given formally by

$$A_{\lambda=\{\nu\nu';\mathbf{R}\}}(\mathbf{q}+\mathbf{G}) = \int_{\Omega} \mathrm{d}^3x\, w^*_{\nu}(\mathbf{x})\mathrm{e}^{-\mathrm{i}(\mathbf{q}+\mathbf{G})\cdot\mathbf{x}} w_{\nu'}(\mathbf{x}-\mathbf{R}), \tag{2}$$

where ν and ν' are indices that run over degrees of freedom in the unit cell (site, orbital, spin) and $\mathbf{R}$ is a lattice vector. These are gathered into a compound index λ that enumerates the various transitions between Wannier orbitals. The infrastructure for keeping track, accessing, finding, etc. these compound indices is set up with the call

```
call genwantran(megqwantran,megqwan_mindist,megqwan_maxdist,allwt=.true.)
```

The MPI grid for the calculation is two-dimensional. At this top level, the frequency points ω are distributed over the first dimension and the **q**-points in the first Brillouin zone are distributed over the second dimension:

```
! distribute frequency points over 1-st dimension
nwloc=mpi_grid_map(lr_nw,dim_k)
! distribute q-vectors along 2-nd dimention
nvqloc=mpi_grid_map(nvq,dim_q)
```

However, the first dimension of the grid is actually reused in a lower level subroutine. I will point that out when discussing that subroutine, but I will say here that another set of points in the first Brillouin zone, the **k**-points are distributed over that dimension.

Next is the heart of the algorithm, the main loop over **q**-points:

```
! main loop over q-points
if (mpi_grid_root()) then
  write(*,*)'Begin q loop'
endif
do iqloc=1,nvqloc
  iq=mpi_grid_map(nvq,dim_q,loc=iqloc)
  call genmegq(iq,.true.,.true.,.false.)
  call genu4(iq,nwloc)
```

```
if (mpi_grid_root()) then
  write(*,'("iq=",I4," Complete")')iq
endif
enddo
```

The call to genmegq is necessary for the calculation of the matrix elements in Eq. (2). The integral in Eq. (2) is not calculated directly. Instead, we use the formula

$$A_\lambda(\mathbf{q}+\mathbf{G}) = \frac{1}{N}\sum_{\mathbf{k}}^{\mathrm{BZ}} \mathrm{e}^{-\mathrm{i}(\mathbf{k}+\mathbf{q})\cdot\mathbf{R}} \sum_{jj'} a^*_{\nu j}(\mathbf{k}) a_{\nu' j'}(\mathbf{k}+\mathbf{q}) \langle \mathbf{k}, j | \mathrm{e}^{-\mathrm{i}(\mathbf{q}+\mathbf{G})\cdot\mathbf{x}} | \mathbf{k}+\mathbf{q}, j' \rangle, \tag{3}$$

where the quantities $\langle \mathbf{k}, j | \mathrm{e}^{-\mathrm{i}(\mathbf{q}+\mathbf{G})\cdot\mathbf{x}} | \mathbf{k}+\mathbf{q}, j' \rangle$ are the matrix elements in the basis of Bloch states, i.e. in the eigenbasis that comes from the groundstate calculation. The $a_{\nu j}(\mathbf{k})$ coefficients come from the definition of the Wannier functions:

$$w_\nu(\mathbf{x}-\mathbf{R}) = \frac{1}{\sqrt{N}}\sum_{\mathbf{k}}^{\mathrm{BZ}} \mathrm{e}^{-\mathrm{i}\mathbf{k}\cdot\mathbf{R}} \sum_{j} a_{\nu j}(\mathbf{k}) \phi_{\mathbf{k}j}(\mathbf{x}). \tag{4}$$

The sums over j run over the bands in our target space, like those highlighted in blue in Fig. 1. N is the number of **k**-points in our **k**-mesh (inside the first Brillouin zone).

The call to genu4 generates the constrained screened interaction in the Wannier basis, given by the equation

$$W^{\mathrm{c}}_{\nu_1\nu_2\nu_3\nu_4;\,\mathbf{R}_1\mathbf{R}_2\mathbf{R}_3}(\omega) = \frac{1}{\Omega}\sum_{\mathbf{q}}^{\mathrm{BZ}} \mathrm{e}^{-\mathrm{i}\mathbf{q}\cdot\mathbf{R}_1} \sum_{\mathbf{G}_1\mathbf{G}_2} A^*_{\lambda=\{\nu_1\nu_2;\mathbf{R}_2\}}(\mathbf{q}+\mathbf{G}_1) W^{\mathrm{c}}_{\mathbf{G}_1\mathbf{G}_2}(\mathbf{q},\omega) A_{\lambda'=\{\nu_3\nu_4;\mathbf{R}_3\}}(\mathbf{q}+\mathbf{G}_2), \tag{5}$$

where Ω is the volume of the "macrocrystal" i.e. the total volume of our simulated crystal. It is given by $\Omega = N\Omega^{\mathrm{UC}}$, where Ω^{UC} is the volume of the unit cell. Each MPI rank only calculates the part of the sum over **q** in Eq. (5) for the **q**-points assigned to that rank. The full $W^{\mathrm{c}}_{\nu_1\nu_2\nu_3\nu_4;\,\mathbf{R}_1\mathbf{R}_2\mathbf{R}_3}(\omega)$ is then calculated with an MPI reduce:

```
do iwloc=1,nwloc
  do it=1,megqwantran%ntr
    call mpi_grid_reduce(u4(1,1,it,iwloc),megqwantran%nwt*megqwantran%nwt, &
                        &dims=(/dim_q/))
  enddo
enddo
```

The array `u4(:,:,:,:)` resides in the `mod_linresp` module in `addons/linresp/mod_linresp.f90`.

Finally, $W^{\mathrm{c}}_{\nu_1\nu_2\nu_3\nu_4;\,\mathbf{R}_1\mathbf{R}_2\mathbf{R}_3}(\omega)$ is written to file using HDF5:

```
if (mpi_grid_side(dims=(/dim_k/)).and.nwloc.gt.0) then
  write(fu4,'("u4_",I4.4,".hdf5")')mpi_grid_dim_pos(dim_k)
  call hdf5_create_file(trim(fu4))
...
      call hdf5_write(fu4,"/iwloc/"//c1//"/"//c2,"u4",u4(1,1,it,iwloc),&
        &(/megqwantran%nwt,megqwantran%nwt/))
    enddo
  enddo
endif
```

If there are several frequency points distributed over the first MPI dimension, then each MPI rank on the side will output its own file. If we are only concerned with $W^{\mathrm{c}}_{\nu_1\nu_2\nu_3\nu_4;\,\mathbf{R}_1\mathbf{R}_2\mathbf{R}_3}(\omega = 0)$, then only one file will be written.

III. GENMEGQ(IQ,TOUT,TG0Q,ALLIBT) IN ADDONS/EXPIGQR/MOD_EXPIGQR.F90

This subroutine calculates the matrix elements that are required both for the calculation of $W^{\mathrm{c}}_{\nu_1\nu_2\nu_3\nu_4;\,\mathbf{R}_1\mathbf{R}_2\mathbf{R}_3}(\omega)$ in Eq. (5) and for the calculation of $\chi^{\mathrm{KS,\,r}}_{\mathbf{G}_1\mathbf{G}_2}(\mathbf{q},\omega)$ used in Eq. (15). First is the initialization of various arrays and bookkeeping tools that will be useful:

```
! initialize G+q vector arays
call init_gq(iq,lmaxexp,lmmaxexp,tg0q)
! initialize k+q array
call init_kq(iq)
! initialize interband transitions
call init_band_trans(allibt)
! initialize Gaunt-like coefficients
call init_gntuju(iq,lmaxexp)
```

There are a couple of things to note here. First, the call to `init_band_trans` sets up translation arrays between the index enumerating pairs $\{j, j'\}$ of bands and decides which pairs will be included. This includes matrix elements in the target space even though these do not play a role in the calculation of $\chi^{\mathrm{KS,r}}_{\mathbf{G}_1\mathbf{G}_2}(\mathbf{q},\omega)$ because they are used in calculating Wannier matrix elements in Eq. (3). Much of this work is done with a call to `getmeidx` from `init_band_trans`. Second, the calculation uses Gaunt coefficients set up with the call to `init_gntuju` to calculate the integrals involved in calculating $\langle\mathbf{k},j|\mathrm{e}^{-\mathrm{i}(\mathbf{q}+\mathbf{G})\cdot\mathbf{x}}|\mathbf{k}+\mathbf{q},j'\rangle$.

Next comes the calculation of the matrix elements $\langle\mathbf{k},j|\mathrm{e}^{-\mathrm{i}(\mathbf{q}+\mathbf{G})\cdot\mathbf{x}}|\mathbf{k}+\mathbf{q},j'\rangle$:

```
do ikstep=1,nkstep
! transmit wave-functions
  call timer_start(1)
  call getwfkq(ikstep,ngknr_jk,igkignr_jk,wfsvmt_jk,wfsvit_jk)
  call timer_stop(1)
! compute matrix elements
  call timer_start(2)
  if (ikstep.le.nkptnrloc) then
    call genmegqblh(iq,ikstep,ngknr(ikstep),ngknr_jk,igkignr(1,ikstep),&
      igkignr_jk,wfsvmtnrloc(1,1,1,1,1,ikstep),wfsvmt_jk,&
      wfsvitnrloc(1,1,1,ikstep),wfsvit_jk)
  endif !ikstep.le.nkptnrloc
  call timer_stop(2)
enddo !ikstep
```

Since the Bloch eigenstates $|\mathbf{k}+\mathbf{q},j'\rangle$ are required in addition to $|\mathbf{k},j\rangle$ and these may reside on a different MPI rank, some communication must occur to put both eigenstates on the same rank. This is done with the call to `getwfkq`. Only the matrix elements needed for that rank are stored on that rank to allow for distributed memory usage.

Next, the matrix elements are calculated in `genmegqblh` with the angular integrals over the spherical harmonics being computed analytically and the radial integrals numerically. I will describe the algorithm inside `genmegqblh` in more detail in Section IV. The radial integrals are calculated in `gengntuju` which you can find in

```
addons/expigqr/gengntuju.f90
```

That calculation occurs inside of the atomic spheres. In the interstitial the integrals over planewaves are calculated analytically.

Next comes the calculation of the Wannier matrix elements:

```
! compute matrix elements of e^{-i(G+q)x} in the basis of Wannier functions
  call genmegqwan(iq)
! sum over all k-points and interband transitions to get <n,T=0|e^{-i(G+q)x}|n',T'>
  call mpi_grid_reduce(megqwan(1,1),megqwantran%nwt*ngq(iq),&
    &dims=(/dim_k/),all=.true.)
  megqwan=megqwan/nkptnr
```

This is done by calculating each term in the sum over $\mathbf{k}$ in Eq. (3) and accumulating them locally on each MPI rank. This is done in genmegqwan. Then the sum over $\mathbf{k}$ with an MPI reduce.

Finally, the matrix elements for the points in the Brillouin zone that are part of the small box that we placed around $\mathbf{q} = 0$ must be treated specially:

```
megqblh(:,ig,ikloc)=zzero
        do i=1,nmegqblh(ikloc)
          ist1=bmegqblh(1,i,ikloc)
          ist2=bmegqblh(2,i,ikloc)
          t1=evalsvnr(ist2,ik)-evalsvnr(ist1,ik)
! Problem with wannier matrix elements if intraband part is included
          if (ist1.eq.ist2.and..not.wannier_megq) megqblh(i,ig,ikloc)=zone
          if (abs(t1).gt.1d-8) then
...
            megqblh(i,ig,ikloc)=megqblh(i,ig,ikloc)-&
              &dot_product(vqc(:,iq),pmatnrloc(:,ist1,ist2,ikloc))/t1
          endif
        enddo
```

This is essentially an implementation of an expansion of the matrix elements for small $\mathbf{q}$ to first order in $\mathbf{q}$. All other terms in the expansion drop away in the $\mathbf{q} \to 0$ limit.

IV. GENMEGQBLH(IQ,IKLOC,NGKNR1,NGKNR2,IGKIGNR1,IGKIGNR2, WFSVMT1,WFSVMT2,WFSVIT1,WFSVIT2) IN ADDONS/EXPIGQR/GENMEGQBLH.F90

As mentioned in Section III, `genmegqblh` computes the matrix elements $\langle \mathbf{k}, j|\mathrm{e}^{-\mathrm{i}(\mathbf{q}+\mathbf{G})\cdot\mathbf{x}}|\mathbf{k}+\mathbf{q}, j'\rangle$ used in the calculation of both the Wannier matrix elements in Eq. (3) and the calculation of $\chi^{\mathrm{KS,\,r}}_{\mathbf{G}_1\mathbf{G}_2}(\mathbf{q},\omega)$ used in Eq. (15). The majority of them are used in the latter computation as there are many more pairs of bands j, j' in the rest versus the target.

Using periodicity and Bloch's theorem, the matrix elements can be written as an integral over the unit cell:

$$\langle \mathbf{k}, j|\mathrm{e}^{-\mathrm{i}(\mathbf{q}+\mathbf{G})\cdot\mathbf{x}}|\mathbf{k}+\mathbf{q}, j'\rangle = \int_{\Omega_{\mathrm{UC}}} \mathrm{d}^3x\, \phi^*_{\mathbf{k},j}(\mathbf{x})\mathrm{e}^{-\mathrm{i}(\mathbf{q}+\mathbf{G})\cdot\mathbf{x}}\phi_{\mathbf{k}+\mathbf{q},j'}(\mathbf{x}), \tag{6}$$

where $\phi_{\mathbf{k},j}(\mathbf{x})$ is the spatial representation of the eigenfunction produced by the groundstate calculation and Ω_{UC} represents the unit cell volume. However, this integral is not calculated directly in real space and instead we use the representation of the eigenfunctions in the LAPW basis. Thus, it is necessary to give a short introduction to that basis.

When using the LAPW basis, the eigenfunctions are written as

$$\phi_{\mathbf{k}j}(\mathbf{x}) = \sum_{\mathbf{G}} c_{\mathbf{k}j,\mathbf{G}}\, \zeta_{\mathbf{k}+\mathbf{G}}(\mathbf{x}) + \sum_{\alpha\ell m}\sum_{\lambda=2}^{N_\lambda(\alpha)} c_{\mathbf{k}j,\lambda\alpha\ell m} \sum_{\mathbf{R}} \mathrm{e}^{\mathrm{i}\mathbf{k}\cdot\mathbf{R}}\zeta_{\lambda\alpha\ell m}(\mathbf{x}-\mathbf{R}). \tag{7}$$

There are two types of functions in this expansion. Both involve the partitioning of the unit cell volume into two kinds of regions. The first type is a collection of spheres centered on atomic sites. These are denoted by MT_α for site α ("MT" stands for "muffin-tin") and the positions of the atomic sites are denoted by $\mathbf{x}_\alpha$. The second type of region is the interstitial which is all of the volume not taken up by the atomic spheres.

The first type of basis function are the APW functions

$$\zeta_{\mathbf{k}+\mathbf{G}}(\mathbf{x}) = \begin{cases} \dfrac{1}{\sqrt{\Omega_{\mathrm{UC}}}}\mathrm{e}^{\mathrm{i}(\mathbf{k}+\mathbf{G})\cdot\mathbf{x}}, & \mathbf{x}\in \mathrm{I} \\ \displaystyle\sum_{\ell=0}^{\ell_{\max}}\sum_{m=-\ell}^{\ell} A^{\alpha\ell m}_{\mathbf{k}+\mathbf{G}} f_{\lambda=1,\alpha\ell}(r)Y_{\ell m}(\hat{\mathbf{r}}), & \mathbf{x}=\mathbf{x}_\alpha+\mathbf{r}\in \mathrm{MT}_\alpha \end{cases} \tag{8}$$

where "I" denotes the interstitial, $Y_{\ell m}(\hat{\mathbf{r}})$ are spherical harmonics and $f_{\lambda\alpha\ell}(r)$ are radial functions found by solving a linearized Schrödinger equation. For the purposes of these notes, the $f_{\lambda\alpha\ell}(r)$ functions can be considered as simply given.

The matching coefficients $A^{\alpha\ell m}_{\mathbf{k}+\mathbf{G}}$ are not independent and are instead given by the condition that the LAPW basis functions should be continuous at boundary of the atomic spheres. This means that the $A^{\alpha\ell m}_{\mathbf{k}+\mathbf{G}}$ coefficients must satisfy

$$f_{\alpha\ell}(R_\alpha)A^{\alpha\ell m}_{\mathbf{k}+\mathbf{G}} = \frac{4\pi}{\sqrt{\Omega}}\mathrm{e}^{\mathrm{i}(\mathbf{k}+\mathbf{G})\cdot\mathbf{x}_\alpha}\mathrm{i}^\ell j_\ell(|\mathbf{k}+\mathbf{G}|R_\alpha)Y^*_{\ell m}(\widehat{\mathbf{k}+\mathbf{G}}), \tag{9}$$

where R_α is the radius of muffin-tin α and $j_\ell(|\mathbf{k}+\mathbf{G}|R_\alpha)$ is a spherical Bessel function. This equation only has a non-trivial solution if $f_{\alpha\ell}(R_\alpha) \neq 0$. The second type of functions in the expansion Eq. (7) are the local orbitals:

$$\zeta_{\mathbf{k},\lambda\alpha\ell m}(\mathbf{x}) = \begin{cases} 0, & \mathbf{x} \in \mathrm{I} \\ f_{\lambda\alpha\ell}(r)Y_{\ell m}(\hat{\mathbf{r}}), & \mathbf{x} = \mathbf{x}_\alpha + \mathbf{r} \in \mathrm{MT}_\alpha \end{cases}, \tag{10}$$

where the λ index enumerates the radial functions $f_{\lambda\alpha\ell}(r)$. The local orbitals are given by $\lambda > 1$.

Since the LAPW basis functions are defined with respect to a unit cell that has been partitioned into atomic spheres and the interstitial, it is natural to perform the integral in Eq. (6) for each sub-volume of the partition separately and sum the result. First, I will discuss the integrals over the atomic spheres. Examining the integral for a particular atomic sphere labeled by α, we can shift coordinates to the center of the sphere and rewrite the integral in spherical coordinates:

$$\begin{aligned}\int_{\mathrm{MT}_\alpha} & \mathrm{d}^3x\, \phi^*_{\mathbf{k},j}(\mathbf{x})\mathrm{e}^{-\mathrm{i}(\mathbf{q}+\mathbf{G})\cdot\mathbf{x}}\phi_{\mathbf{k}+\mathbf{q},j'}(\mathbf{x}) \\ &= \mathrm{e}^{-\mathrm{i}(\mathbf{q}+\mathbf{G})\cdot\mathbf{x}_\alpha}\sum_{\substack{\lambda\lambda' \\ \ell\ell' \\ mm'}} c^*_{\mathbf{k}j,\lambda\alpha\ell m}c_{\mathbf{k}+\mathbf{q},j';\lambda'\alpha\ell'm'}\sum_{\ell''m''}4\pi(-\mathrm{i})^{\ell''}Y_{\ell''m''}(\widehat{\mathbf{q}+\mathbf{G}}) \\ &\quad\times\int \mathrm{d}r\, f_{\lambda\alpha\ell}(r)j_{\ell''}(|\mathbf{q}+\mathbf{G}|r)f_{\lambda'\alpha\ell'}(r)\int \mathrm{d}\Omega\, Y^*_{\ell m}(\hat{r})Y^*_{\ell''m''}(\hat{r})Y_{\ell'm'}(\hat{r}).\end{aligned} \tag{11}$$

Here, all the sums over ℓ run from 0 to a cutoff $\ell_{\max}$ and the sums over m run from $-\ell$ to ℓ for their respective ℓ's. Also, $\int \mathrm{d}\Omega$ represents an integral over solid angles. Also

$$c_{\mathbf{k}j,\lambda\alpha\ell m} = \sum_{\mathbf{G}} c_{\mathbf{k}j,\mathbf{G}}A^{\alpha\ell m}_{\mathbf{k}+\mathbf{G}} \quad \text{if } \lambda = 1. \tag{12}$$

The computation is performed in two primary parts. First the integrals and inner sum over ℓ'', m'' is performed in `gengntuju`. The angular integrals are related to the Gaunt coefficients and the radial integrals are performed numerically.

The part of the integral over the interstitial is simpler and is given by

$$\int_{\mathrm{I}} \mathrm{d}^3x\, \phi^*_{\mathbf{k},j}(\mathbf{x}) \mathrm{e}^{-\mathrm{i}(\mathbf{q}+\mathbf{G})\cdot\mathbf{x}} \phi_{\mathbf{k}+\mathbf{q},j'}(\mathbf{x}) = \frac{1}{\Omega_{\mathrm{UC}}} \sum_{\mathbf{G}'\mathbf{G}''} c^*_{\mathbf{k}j,\mathbf{G}'} c_{\mathbf{k}+\mathbf{q},j;\mathbf{G}''} \int_{\mathrm{I}} \mathrm{d}^3x\, \mathrm{e}^{\mathrm{i}(\mathbf{G}''-\mathbf{G}'-\mathbf{G})\cdot\mathbf{x}}. \tag{13}$$

A note of caution however. The integral does not equal a Kronecker delta as it would if the integral were over the entire unit cell, because the integral is restricted to the interstitial region.

The way that this calculation is done in `genmegqblh` is as follows. First the sums over λ, ℓ and m in Eq. (11) are performed and collected in the `wftmp1` array:

```
 do ig=1,ngq(iq)
! precompute muffint-tin part of \psi_1^{*}(r)*e^{-i(G+q)r}
       do ias=1,natmtot
         b1=dconjg(wfsvmt1(:,ias,ispn1,ist1)*sfacgq(ig,ias))
         ic=ias2ic(ias)
         b2=zzero
         do j=1,ngntuju(ic,ig)
           b2(igntuju(2,j,ic,ig))=b2(igntuju(2,j,ic,ig))+&
             &b1(igntuju(1,j,ic,ig))*gntuju(j,ic,ig)
         enddo
         wftmp1((ias-1)*lmmaxapw*nufrmax+1:ias*lmmaxapw*nufrmax,ig)=b2(:)
       enddo !ias
     enddo !ig
```

Second, the integral and the sum over $\mathbf{G}'$ in Eq. (13) are performed

```
wfir1=zzero
     do ig1=1,ngknr1
       ifg=igfft(igkignr1(ig1))
       wfir1(ifg)=wfsvit1(ig1,ispn1,ist1)
     enddo
     call zfftifc(3,ngrid,1,wfir1)
     do ir=1,ngrtot
       wfir1(ir)=wfir1(ir)*cfunir(ir)
     enddo
```

```
      call zfftifc(3,ngrid,-1,wfir1)
      do ig=1,ngq(iq)
        do ig2=1,ngknr2
! G1=G2-G-Gkq
          ivg1(:)=ivg(:,igkignr2(ig2))-ivg(:,igqig(ig,iq))-ivg(:,igkq)
          ifg=igfft(ivgig(ivg1(1),ivg1(2),ivg1(3)))
          wftmp1(lmmaxapw*nufrmax*natmtot+ig2,ig)=dconjg(wfir1(ifg))
        enddo
      enddo
```

The integral in Eq. (13) is done by first inverse Fourier transforming $c^*_{\mathbf{k}j,\mathbf{G}'}$ to a real space grid, using the so-called characteristic function, `cfunir` to set it to zero inside of the atomic spheres, then Fourier transforming the result back into reciprocal space. This is done so that the integral over the interstitial in Eq. (13) can be replaced by an integral over the whole unit cell so that the integrals over the exponentials becomes trivial.

Third, the sums over λ', ℓ' and m' in Eq. (11), α, and $\mathbf{G}''$ in Eq. (13) are performed using a call to `zgemm`:

```
! collect right |ket> states into matrix wftmp2
    do while ((i+n1).le.nmegqblh(ikloc))
      if (bmegqblh(1,i+n1,ikloc).ne.bmegqblh(1,i,ikloc)) exit
      ist2=bmegqblh(2,i+n1,ikloc)
      n1=n1+1
      call memcopy(wfsvmt2(1,1,1,ispn2,ist2),wftmp2(1,n1),&
        &16*lmmaxapw*nufrmax*natmtot)
      call memcopy(wfsvit2(1,ispn2,ist2),wftmp2(lmmaxapw*nufrmax*natmtot+1,n1),&
        &16*ngknr2)
    enddo !while
! update several matrix elements by doing matrix*matrix operation
!  me(ib,ig)=wftmp2(ig2,ib)^{T}*wftmp1(ig2,ig)
    call zgemm('T','N',n1,ngq(iq),wfsize,zone,wftmp2,wfsize,wftmp1,wfsize,&
      &zone,megqblh(i,1,ikloc),nstsv*nstsv)
```

However, it may be possible to further use calls to `zgemm` in order to speed up the run time, or at least make the algorithm more suitable for utilizing GPUs. Currently, the array `gntuju`, which is given by

$$G_{\lambda\ell m,\lambda'\ell'm'}(\mathbf{q}+\mathbf{G},\alpha)=\sum_{\ell''m''}4\pi(-\mathrm{i})^{\ell''}Y_{\ell''m''}(\widehat{\mathbf{q}+\mathbf{G}})\int \mathrm{d}r\, f_{\lambda\alpha\ell}(r)j_{\ell''}(|\mathbf{q}+\mathbf{G}|r)f_{\lambda'\alpha\ell'}(r) \\ \times\int \mathrm{d}\Omega\, Y^*_{\ell m}(\hat{r})Y^*_{\ell''m''}(\hat{r})Y_{\ell'm'}(\hat{r}), \tag{14}$$

uses a super-index for the first dimension that contains λ, ℓ, m, λ', ℓ' and m'. Thus, for each $\mathbf{G}$ and α (and implicitly $\mathbf{q}$), it is currently a one dimensional array. If this one-dimensional array was instead folded into a two-dimensional array, then the sums over λ, ℓ, m, λ', ℓ' and m' in Eq. (11) could be computed with two calls to `zgemm`.

V. GENU4(IQ,NWLOC) IN ADDONS/GENU4.F90

`genu4` generates the constrained screened interaction given in Eq. (5) for each $\mathbf{q}$ and ω and the results are accumulated in `u4(:,:,:,:)`. This requires the calculation of $W^{\mathrm{c}}_{\mathbf{GG'}}(\mathbf{q},\omega)$, which is given by

$$W^{\mathrm{c}}_{\mathbf{GG'}}(\mathbf{q},\omega)=v_{\mathbf{G}}(\mathbf{q})\delta_{\mathbf{GG'}}+\sum_{\mathbf{G}_1\mathbf{G}_2}v_{\mathbf{G}}(\mathbf{q})\delta_{\mathbf{GG}_1}\chi^{\mathrm{KS,\,r}}_{\mathbf{G}_1\mathbf{G}_2}(\mathbf{q},\omega)W^{\mathrm{c}}_{\mathbf{G}_2\mathbf{G'}}(\mathbf{q},\omega), \tag{15}$$

where $v_{\mathbf{G}}(\mathbf{q})$ is the Fourier transform of the Coulomb interaction and $\chi^{\mathrm{KS,\,r}}_{\mathbf{G}_1\mathbf{G}_2}(\mathbf{q},\omega)$ is Kohn-Sham response function (or polarization) calculated by excluding interband transitions in the target space. $\chi^{\mathrm{KS,\,r}}_{\mathbf{G}_1\mathbf{G}_2}(\mathbf{q},\omega)$ is calculated with the call

```
if (screenu4) call genchi0(iq)
```

I will have more to say about the calculation of $\chi^{\mathrm{KS,\,r}}_{\mathbf{G}_1\mathbf{G}_2}(\mathbf{q},\omega)$ when I discuss `genchi0`.

Next, we must set up the arrays containing the matrix elements involved in calculating Eq. (5):

```
allocate(megqwan2(ngq(iq),megqwantran%nwt))
allocate(megqwan3(ngq(iq),megqwantran%nwt))
! compute megqwan2=<W_n|e^{+i(G+q)x}|W_n'T'> and also rearrange megqwan
do i=1,megqwantran%nwt
  n=megqwantran%iwt(1,i)
  n1=megqwantran%iwt(2,i)
  vtl(:)=megqwantran%iwt(3:5,i)
  v2=dble(vtl)
  call r3mv(avec,v2,vtc)
  zt1=exp(-zi*dot_product(vqc1,vtc))
  j=megqwantran%iwtidx(n1,n,-vtl(1),-vtl(2),-vtl(3))
  if (j.le.0) then
    write(*,'("Error(genu4) wrong index of matrix element")')
    write(*,'(" n,n1,vtl : ",5I4)')n,n1,vtl
    call pstop
  endif
  megqwan2(:,i)=dconjg(megqwan(j,:))*zt1
  megqwan3(:,i)=megqwan(i,:)
enddo
```

Setting up the matrix elements includes filling an array with the conjugate matrix elements to those in Eq. (2) and making the first index of the array that which enumerates **G**. This is because **G** is summed over in Eq. (5) and Fortran follows column major order, so it is more efficient to rearrange in this way.

Now we are ready to calculate and add a term to $W^{\mathrm{c}}_{\nu_1\nu_2\nu_3\nu_4;\,\mathbf{R}_1\mathbf{R}_2\mathbf{R}_3}(\omega = 0)$:

```
! compute 4-index U
! TODO: comments with formulas
do iwloc=1,nwloc
  iw=mpi_grid_map(lr_nw,dim_k,loc=iwloc)
! broadcast chi0
  if (screenu4) then
    call genvscrn(iq,chi0loc(1,1,iwloc),krnl,vscrn,epsilon)
  else
    vscrn=krnl
  endif
  call zgemm('T','N',megqwantran%nwt,ngq(iq),ngq(iq),zone,megqwan2,ngq(iq),&
    &vscrn,ngq(iq),zzero,zm1,megqwantran%nwt)
  call zgemm('N','N',megqwantran%nwt,megqwantran%nwt,ngq(iq),zone,zm1,&
    &megqwantran%nwt,megqwan3,ngq(iq),zzero,zm2,megqwantran%nwt)
  do it=1,megqwantran%ntr
    v2=dble(megqwantran%vtr(:,it))
    call r3mv(avec,v2,vtc)
    zt1=exp(-zi*dot_product(vqc1,vtc))/omega/nkptnr
    call zaxpy((megqwantran%nwt)**2,zt1,zm2(1,1),1,u4(1,1,it,iwloc),1)
  enddo
enddo
```

First, $W^{\mathrm{c}}_{\mathbf{G}\mathbf{G}'}(\mathbf{q},\omega)$ is calculated by solving the matrix equation in Eq. (15) with the call to `genvscrn`, which gets passed `chi0loc` the array containing $\chi^{\mathrm{KS,\,r}}_{\mathbf{G}_1\mathbf{G}_2}(\mathbf{q},\omega)$. "`loc`" indicates that the array is local to the MPI ranks along the second dimension for a particular value of the first dimension. This is because for multiple frequencies ω, the array is distributed over the first MPI dimension.

Then, the sums over $\mathbf{G}$ and $\mathbf{G}'$ in Eq. (5) are performed via matrix multiplication with the calls to `zgemm`. Next comes the loop `do it=1,megqwantran%ntr`, which is over the lattice vectors given by $\mathbf{R}_1$ in Eq. (5). The factor $\frac{\mathrm{e}^{-\mathrm{i}\mathbf{q}\cdot\mathbf{R}_1}}{\Omega}$ is multiplied by the term contained in `zm2(:,:)` for each $\mathbf{R}_1$ and accumulated in `u4(:,:,:,:)` with the call to `zaxpy`.