Implementation of Exchange-Correlation Energy (for meta-GGA) in Abinit within the norm-conserving approach

Aurélien Lherbier

September 27, 2020

Abstract

The aim of this report is first to explain briefly the general procedure for calculation of exchange-correlation energy in Abinit (in case of LDA, GGA) and then to discuss the way the meta-GGA case is treated. This report could be useful to any new developers in Abinit who would like to implement in the subdirectory `/56_xc`. In this report I will essentially describe the main structures of some routines such as `rhohxc.F90`, `xcden.F90`, `xcmult.F90` and `xcpot.F90`.

Chapter 1

Rule of notations

A rule of notations (see below or at the beginning of `rhohxc.F90`, at the end of the list of *Local variables*) was proposed in version 6.5.0. In one hand, the idea is to try to keep a certain consistency with the labelling of variables. Indeed the latter have been added little by little by different developers who have their own personal notations. In a second hand, the use of a good labelling of variables facilitate the understanding when reading for the first time the code. In that sence, it is often preferable to give a variable name that sticks the much as possible to the physical quantity to which it corresponds.
The following rule of notations is only a proposal which can be off course use or not, depending on you. It can also be modified or be improved. Here below is the proposition:

- `rho` (ρ) is the electronic density
- `tau` (τ) is the kinetic energy density
- `exc` (ε_{xc}) is the exchange-correlation energy density per particule
- `epsxc` (ϵ_{xc}) is the exchange-correlation energy density ($\epsilon_{xc} = \rho \times \varepsilon_{xc}$)
- `vxc` (v_{xc}) is the exchange-correlation potential
- `bigexc` (E_{xc}) is the exchange-correlation energy (for the moment it is still named "`enxc`")
- `m_norm` ($|m|$) is the norm of magnetization

- `g...` means the gradient (∇) of something (e.g. : `grho` means gradient of the electronic density)
- `g...2` means square norm of gradient ($|\nabla|^2$) of something (e.g. : `grho2` means square norm of gradient of the electronic density)
- `l...` means Laplacian ($\Delta \equiv \nabla^2$) of something (e.g. : `lrho` means Laplacian of electronic density)
- `d...d...` means first derivative of something with regards to something else ($\frac{\partial}{\partial}$).
- `d2...d...d...` means second derivative of ... with regards to ... and to ... ($\frac{\partial^2}{\partial\,\partial}$)
- etc...
- `d...` without the occurence of the second "`d`" means that this is an array which regroups several derivatives of the same quantity (e.g. : `depsxc` can contain $\frac{\partial \epsilon_{xc}}{\partial \rho}$ but also $\frac{\partial \epsilon_{xc}}{\partial |\nabla \rho|} \cdot \frac{1}{|\nabla \rho|}$)

- `..._b` means a block of the quantity ... (this is use in mpi loops which treat the data block by block)
- `..._updn` means that spin up and spin down are available in that array such as data_updn(..,1) and data_updn(..,2). (if `nspden` ≥ 2 off course, otherwise if `nspden`$= 1$ data_up(..,1) contains the total quantity).
- `..._apn` in case of positrons are concerned.

to be the closest as possible with the libxc notations we also use the following variable names:

- `vxcrho` is the first derivative of the exchange-correlation energy density with regards to the electronic density ($\frac{\partial \epsilon_{xc}}{\partial \rho} \equiv$ `depsxcdrho`).
- `vxcgrho` is the first derivative of the exchange-correlation energy density with regards to the gradient of the electronic density ($\frac{\partial \epsilon_{xc}}{|\nabla \rho|} \equiv$ `depsxcdgrho`).
- `vxclrho` is the first derivative of the exchange-correlation energy density with regards to the Laplacian of the electronic density ($\frac{\partial \epsilon_{xc}}{\partial \Delta \rho} \equiv$ `depsxcdlrho`).

- `vxctau` is the first derivative of the exchange-correlation energy density with regards to the kinetic energy density ($\frac{\partial \epsilon_{xc}}{\partial \tau} \equiv$ `depsxcdtau`).

Chapter 2

Brief remind of exchange correlation equations

Let first recall the general form of the exchange-correlation energy in the case of meta-GGA. For the moment, we will consider a easier case which is a meta-GGA functional which would not depend on the kinetic energy density (τ). Nevertheless, this latter case still encompasses the LDA and GGA cases.

$$E_{xc}^{MGGA} = \int \epsilon_{xc} \left[\rho_\sigma(\mathbf{r}), \nabla\rho_\sigma(\mathbf{r}), \Delta\rho_\sigma(\mathbf{r})\right] d\mathbf{r} \tag{2.1}$$

with σ the spin index (up or down). The corresponding exchange-correlation potential is given by

$$v_{xc}^{\sigma} = \frac{\delta E_{xc}^{MGGA}}{\delta \rho_\sigma} = \frac{\partial \epsilon_{xc}}{\partial \rho_\sigma} - \left(\sum_{\alpha=x,y,z} \nabla_\alpha \left(\frac{\partial \epsilon_{xc}}{\partial \nabla_\alpha \rho_\sigma} \right) \right) + \Delta \left(\frac{\partial \epsilon_{xc}}{\partial \Delta \rho_\sigma} \right) \tag{2.2}$$

The total exchange-correlation energy (E_{xc}) is a simple scalar since it is the integral over the whole space ($\int d\mathbf{r}$) of a functional of density (plus eventually its gradient and its Laplacian) which itself depends on the space position ($\mathbf{r}$). On the contrary the potential v_{xc} is a function of the space position ($\mathbf{r}$) and then is a scalar field. In the above equation (Eq.**??**), the first term on the right hand side is the LDA contribution, the second term is the GGA contribution and the third term is the meta-GGA contribution. Note that in the code, the second term is not directly expressed in this way but with another form which is

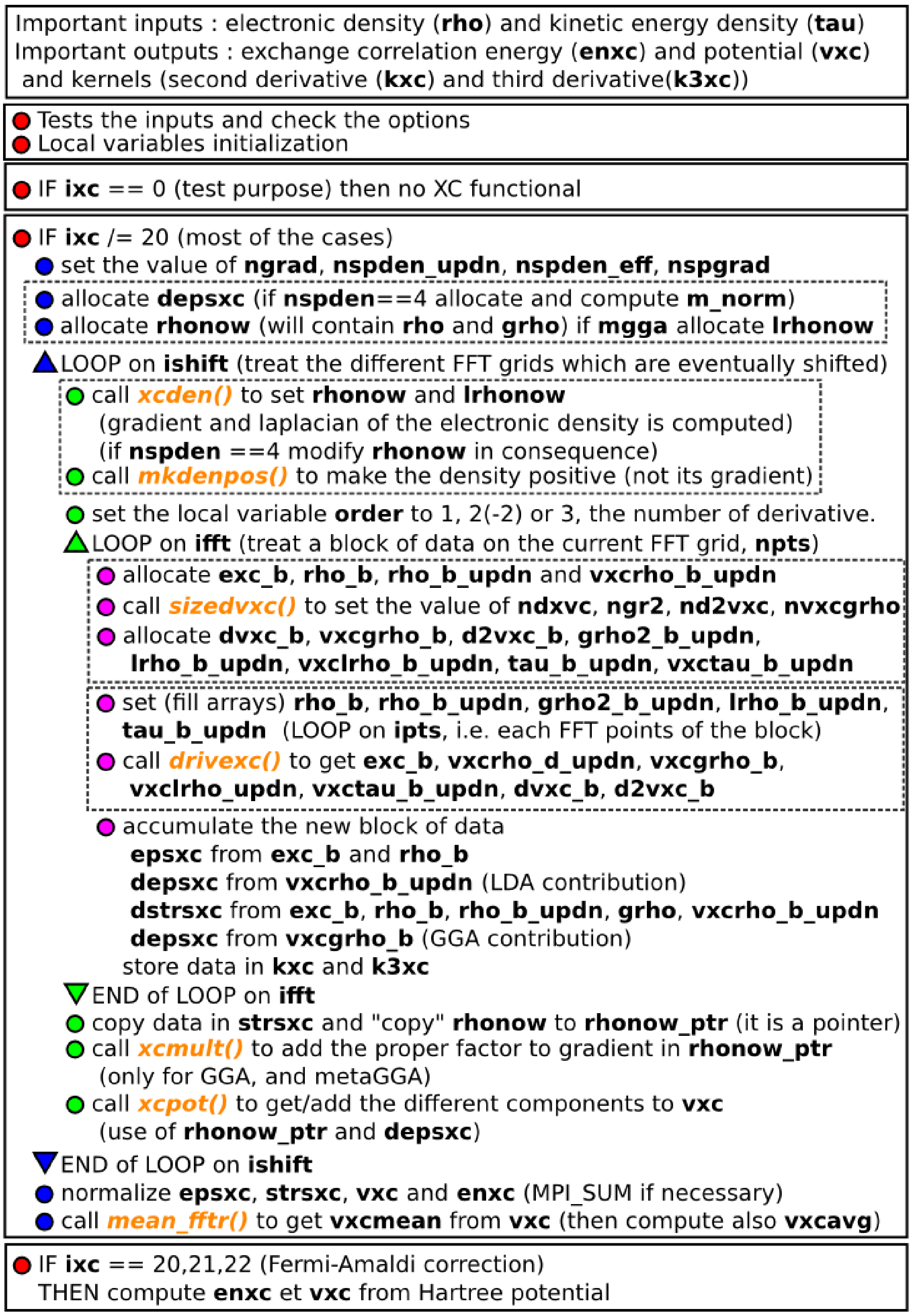
$$\sum_{\alpha=x,y,z} \nabla_\alpha \left(\frac{\partial \epsilon_{xc}}{\partial \nabla_\alpha \rho_\sigma} \right) = \sum_{\alpha=x,y,z} \nabla_\alpha \left[\nabla_\alpha \rho_\sigma \times \left(\frac{1}{|\nabla \rho_\sigma|} \cdot \frac{\partial \epsilon_{xc}}{\partial |\nabla \rho_\sigma|} \right) \right] \tag{2.3}$$

with $|\nabla\rho_\sigma| = \left(\sum_{\alpha=x,y,z} \nabla_\alpha \rho_\sigma\right)^{1/2}$.

TO BE COMPLETED FOR THE CASE WHERE KINETIC ENERGY DENSITY IS INVOLVED

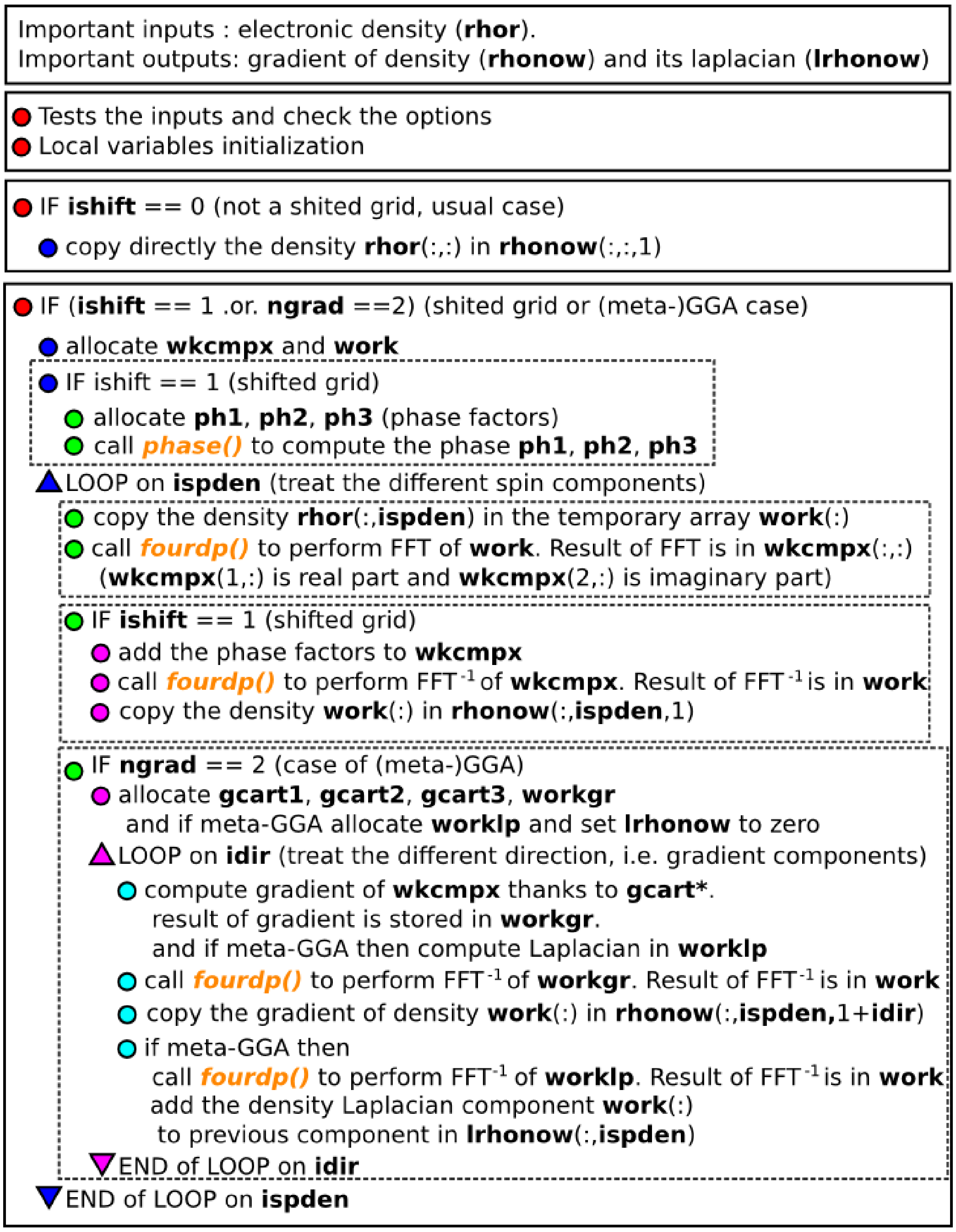
Chapter 3

The routines structures of rhohxc.F90, xcden.F90, xcmult.F90 and xcpot.F90



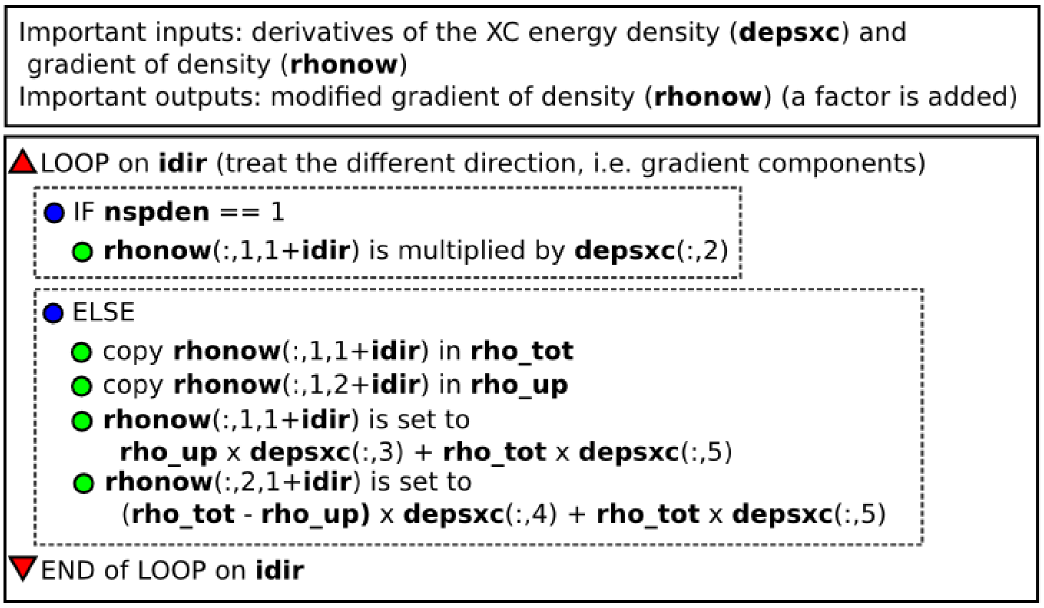
```
Important inputs : electronic density (rho) and kinetic energy density (tau)
Important outputs : exchange correlation energy (enxc) and potential (vxc)
 and kernels (second derivative (kxc) and third derivative(k3xc))

● Tests the inputs and check the options
● Local variables initialization

● IF ixc == 0 (test purpose) then no XC functional

● IF ixc /= 20 (most of the cases)
  ● set the value of ngrad, nspden_updn, nspden_eff, nspgrad
  ● allocate depsxc (if nspden==4 allocate and compute m_norm)
  ● allocate rhonow (will contain rho and grho) if mgga allocate lrhonow
  ▲LOOP on ishift (treat the different FFT grids which are eventually shifted)
    ● call xcden() to set rhonow and lrhonow
      (gradient and laplacian of the electronic density is computed)
      (if nspden ==4 modify rhonow in consequence)
    ● call mkdenpos() to make the density positive (not its gradient)
    ● set the local variable order to 1, 2(-2) or 3, the number of derivative.
    △LOOP on ifft (treat a block of data on the current FFT grid, npts)
      ● allocate exc_b, rho_b, rho_b_updn and vxcrho_b_updn
      ● call sizedvxc() to set the value of ndxvc, ngr2, nd2vxc, nvxcgrho
      ● allocate dvxc_b, vxcgrho_b, d2vxc_b, grho2_b_updn,
         lrho_b_updn, vxclrho_b_updn, tau_b_updn, vxctau_b_updn
      ● set (fill arrays) rho_b, rho_b_updn, grho2_b_updn, lrho_b_updn,
        tau_b_updn  (LOOP on ipts, i.e. each FFT points of the block)
      ● call drivexc() to get exc_b, vxcrho_d_updn, vxcgrho_b,
         vxclrho_updn, vxctau_b_updn, dvxc_b, d2vxc_b
      ● accumulate the new block of data
         epsxc from exc_b and rho_b
         depsxc from vxcrho_b_updn (LDA contribution)
         dstrsxc from exc_b, rho_b, rho_b_updn, grho, vxcrho_b_updn
         depsxc from vxcgrho_b (GGA contribution)
        store data in kxc and k3xc
    ▽END of LOOP on ifft
    ● copy data in strsxc and "copy" rhonow to rhonow_ptr (it is a pointer)
    ● call xcmult() to add the proper factor to gradient in rhonow_ptr
      (only for GGA, and metaGGA)
    ● call xcpot() to get/add the different components to vxc
      (use of rhonow_ptr and depsxc)
  ▼END of LOOP on ishift
  ● normalize epsxc, strsxc, vxc and enxc (MPI_SUM if necessary)
  ● call mean_fftr() to get vxcmean from vxc (then compute also vxcavg)

● IF ixc == 20,21,22 (Fermi-Amaldi correction)
  THEN compute enxc et vxc from Hartree potential
```

Figure 3.1: Scheme of the routine rhohxc.F90.



```
Important inputs : electronic density (rhor).
Important outputs: gradient of density (rhonow) and its laplacian (lrhonow)

● Tests the inputs and check the options
● Local variables initialization

● IF ishift == 0 (not a shited grid, usual case)
    ● copy directly the density rhor(:,:) in rhonow(:,:,1)

● IF (ishift == 1 .or. ngrad ==2) (shited grid or (meta-)GGA case)
    ● allocate wkcmpx and work
    ● IF ishift == 1 (shifted grid)
        ● allocate ph1, ph2, ph3 (phase factors)
        ● call phase() to compute the phase ph1, ph2, ph3
    ▲ LOOP on ispden (treat the different spin components)
        ● copy the density rhor(:,ispden) in the temporary array work(:)
        ● call fourdp() to perform FFT of work. Result of FFT is in wkcmpx(:,:)
          (wkcmpx(1,:) is real part and wkcmpx(2,:) is imaginary part)
        ● IF ishift == 1 (shifted grid)
            ● add the phase factors to wkcmpx
            ● call fourdp() to perform FFT^-1 of wkcmpx. Result of FFT^-1 is in work
            ● copy the density work(:) in rhonow(:,ispden,1)
        ● IF ngrad == 2 (case of (meta-)GGA)
            ● allocate gcart1, gcart2, gcart3, workgr
              and if meta-GGA allocate worklp and set lrhonow to zero
            ▲ LOOP on idir (treat the different direction, i.e. gradient components)
                ● compute gradient of wkcmpx thanks to gcart*.
                  result of gradient is stored in workgr.
                  and if meta-GGA then compute Laplacian in worklp
                ● call fourdp() to perform FFT^-1 of workgr. Result of FFT^-1 is in work
                ● copy the gradient of density work(:) in rhonow(:,ispden,1+idir)
                ● if meta-GGA then
                  call fourdp() to perform FFT^-1 of worklp. Result of FFT^-1 is in work
                  add the density Laplacian component work(:)
                   to previous component in lrhonow(:,ispden)
            ▼ END of LOOP on idir
    ▼ END of LOOP on ispden
```

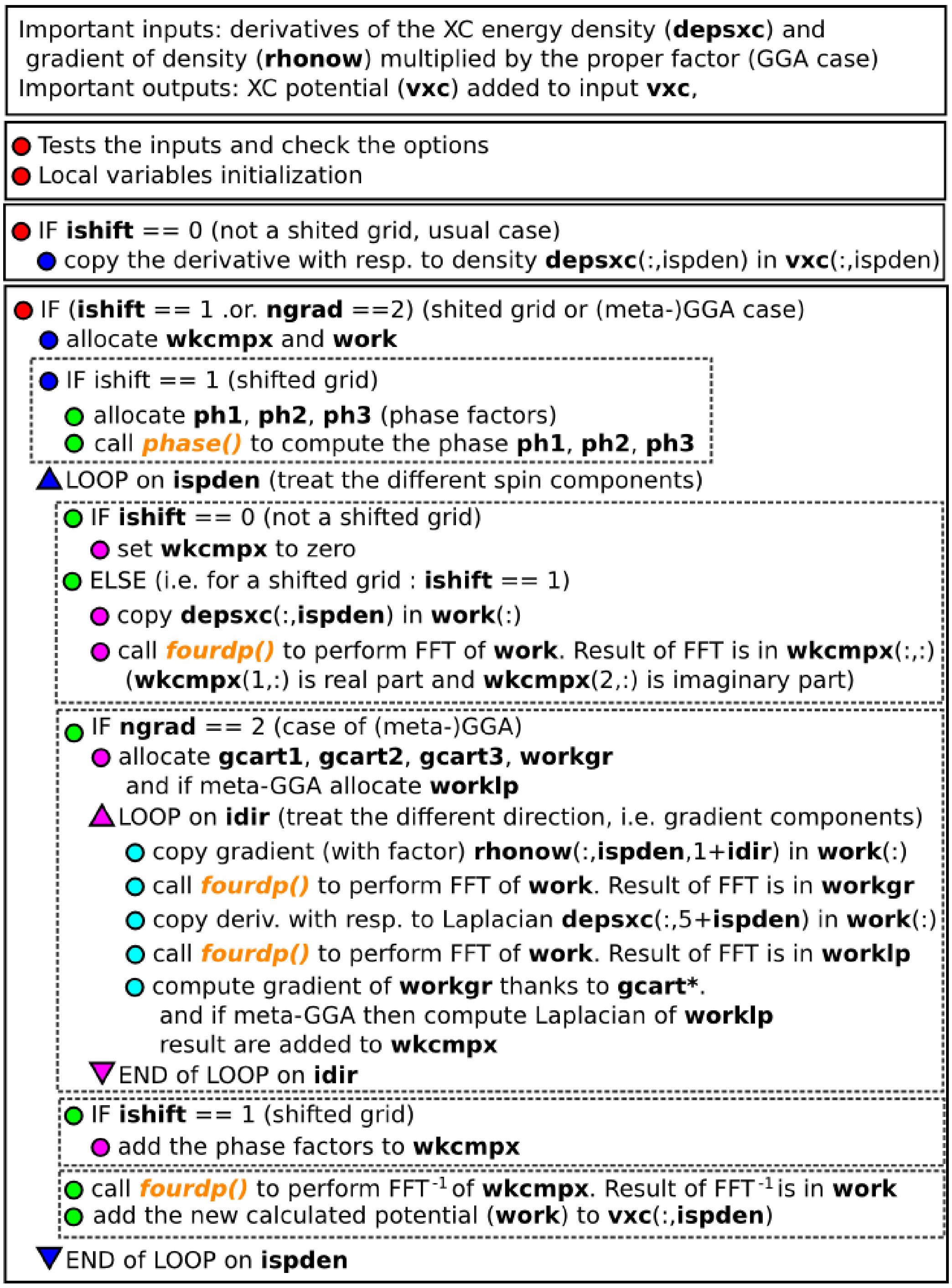
Figure 3.2: Scheme of the routine `xcden.F90`.



Important inputs: derivatives of the XC energy density (**depsxc**) and gradient of density (**rhonow**)
Important outputs: modified gradient of density (**rhonow**) (a factor is added)

```
▲ LOOP on idir (treat the different direction, i.e. gradient components)
  ● IF nspden == 1
    ● rhonow(:,1,1+idir) is multiplied by depsxc(:,2)
  ● ELSE
    ● copy rhonow(:,1,1+idir) in rho_tot
    ● copy rhonow(:,1,2+idir) in rho_up
    ● rhonow(:,1,1+idir) is set to
        rho_up x depsxc(:,3) + rho_tot x depsxc(:,5)
    ● rhonow(:,2,1+idir) is set to
        (rho_tot - rho_up) x depsxc(:,4) + rho_tot x depsxc(:,5)
▼ END of LOOP on idir
```

Figure 3.3: Scheme of the routine `xcmult.F90`.



```
Important inputs: derivatives of the XC energy density (depsxc) and
 gradient of density (rhonow) multiplied by the proper factor (GGA case)
Important outputs: XC potential (vxc) added to input vxc,

● Tests the inputs and check the options
● Local variables initialization

● IF ishift == 0 (not a shited grid, usual case)
   ● copy the derivative with resp. to density depsxc(:,ispden) in vxc(:,ispden)

● IF (ishift == 1 .or. ngrad ==2) (shited grid or (meta-)GGA case)
   ● allocate wkcmpx and work
   ● IF ishift == 1 (shifted grid)
      ● allocate ph1, ph2, ph3 (phase factors)
      ● call phase() to compute the phase ph1, ph2, ph3
   ▲ LOOP on ispden (treat the different spin components)
      ● IF ishift == 0 (not a shifted grid)
         ● set wkcmpx to zero
      ● ELSE (i.e. for a shifted grid : ishift == 1)
         ● copy depsxc(:,ispden) in work(:)
         ● call fourdp() to perform FFT of work. Result of FFT is in wkcmpx(:,:)
           (wkcmpx(1,:) is real part and wkcmpx(2,:) is imaginary part)
      ● IF ngrad == 2 (case of (meta-)GGA)
         ● allocate gcart1, gcart2, gcart3, workgr
           and if meta-GGA allocate worklp
         ▲ LOOP on idir (treat the different direction, i.e. gradient components)
            ● copy gradient (with factor) rhonow(:,ispden,1+idir) in work(:)
            ● call fourdp() to perform FFT of work. Result of FFT is in workgr
            ● copy deriv. with resp. to Laplacian depsxc(:,5+ispden) in work(:)
            ● call fourdp() to perform FFT of work. Result of FFT is in worklp
            ● compute gradient of workgr thanks to gcart*.
              and if meta-GGA then compute Laplacian of worklp
              result are added to wkcmpx
         ▼ END of LOOP on idir
      ● IF ishift == 1 (shifted grid)
         ● add the phase factors to wkcmpx
      ● call fourdp() to perform FFT^-1 of wkcmpx. Result of FFT^-1 is in work
      ● add the new calculated potential (work) to vxc(:,ispden)
   ▼ END of LOOP on ispden
```

Figure 3.4: Scheme of the routine `xcpot.F90`.