Documentationm for the RTA code in 3D

F. Coppens[a], M. Vincendon[a], P.-G. Reinhard[c], E. Suraud[a,b]

[a] *Université de Toulouse; UPS; Laboratoire de Physique Théorique, IRSAMC; F-31062 Toulouse Cedex, France*

[b] *Laboratoire de Physique Théorique, Université Paul Sabatier, CNRS, F-31062 Toulouse Cédex, France*

[c] *Institut für Theoretische Physik, Universität Erlangen, D-91058 Erlangen, Germany*

Abstract

This is a first draft for the CPC presenting the TDLDA+RTA code to the public. The layout of presentation has yet to be discussed. Presently it mixes theory and algorithm with code. We may also consider collecting all theory & numerics together then followed by a huge block detailing the code.

Keywords: electronic disspation, time-dependent density functional theory, ...

*Corresponding author: coppens@irsamc.ups-tlse.fr

1. Time-dependent local density approximation (TDLDA)

PGR2all: The main part of the code is very complex and much has to be explained. We need to address: LDA, pseudo-potentials, TDLDA, and MD. The question is how we arrange that in proper sectioning or whether we put generally known "basics" to appendices.
The code is still capable of coupling to a dielectric environment. We should skip that for the CPC publication.
Another question to be discussed is whether we maintain the option to run finite differences instead of FFT. If we do so, then we have to test that branch carefully.

1.1. Mean-field propagation

PGR2all: This piece is copied from our basic RTA paper and servers her just to provide formulae needed later in RTA. It has yet to be rewritten in connection with the new section on TDLDA-MD.

The starting point and dominant feature of the dynamics is the propagation at the level of the mean field. In this paper, we are dealing with the electron dynamics in metal clusters and we describe it by time-dependent density functional theory at the level of the Time-Dependent Local-Density Approximation (TDLDA) treated in the real time domain [1, 2]. It is augmented by a self-interaction correction (SIC) approximated by average-density SIC (ADSIC) [3] in order to attain correct ionization properties [4] in the course of the dynamical simulation. TDLDA is formulated within the usual Kohn-Sham picture in terms of a set of occupied single-particle (s.p.) wavefunctions $\{|\phi_\alpha\rangle, \alpha = 1...N\}$. Their dynamics is described by the time-dependent Kohn-Sham equation

$$\mathrm{i}\partial_t|\phi_\alpha\rangle = \hat{h}[\varrho]|\phi_\alpha\rangle \tag{1}$$

where $\hat{h}$ is the Kohn-Sham mean-field Hamiltonian which is a functional of the instantaneous local density $\varrho(\mathbf{r},t) = \sum_\alpha |\phi_\alpha(\mathbf{r},t)|^2$ [5, 6]. The time evolution delivered by Eq. (1) can be expressed formally by the unitary one-body time-evolution operator

$$\hat{U}(t,t') = \hat{\mathcal{T}}\mathrm{exp}\left(-i\int_t^{t'}\hat{h}(t'')dt''\right) \tag{2a}$$

where $\hat{\mathcal{T}}$ is the time-ordering operator. This yields a closed expression for the time-evolution of s.p. states

$$|\phi_\alpha(t)\rangle = \hat{U}(t,t')|\phi_\alpha(t')\rangle. \tag{2b}$$

So far, TDLDA propagates pure states. Dissipation which we will add later on leads inevitably to mixed states. This requires to generalize the description from fully occupied s.p. wavefunctions to a one-body density operator $\hat{\rho}$. Its representation in configuration space, i.e. in terms of a given set of s.p. states $|\varphi_i\rangle$, reads in general $\hat{\rho} = \sum_{ij}|\varphi_i\rangle\rho_{ij}\langle\varphi_j|$. By appropriate transformation of the s.p. basis, one can diagonalize the density matrix ρ_{ij} which defines what are called natural orbitals. The natural orbitals representation of the one-body density operator then reads

$$\hat{\rho} = \sum_{\alpha=1}^{\infty}|\phi_\alpha\rangle W_\alpha\langle\phi_\alpha| \quad . \tag{3}$$

The weights W_α represent the probability with which a state $|\phi_\alpha\rangle$ is occupied. The mean-field propagation (1) then becomes

$$\mathrm{i}\partial_t\hat{\rho} = \left[\hat{h}[\varrho], \hat{\rho}\right] \tag{4}$$

where $\hat{h}[\varrho]$ is formally the same as before and the local density is now computed as

$$\varrho(\mathbf{r}, t) = \sum_\alpha W_\alpha |\phi_\alpha(\mathbf{r}, t)|^2. \tag{5}$$

The (coherent) pure mean-field propagation (4) leaves the occupation weights W_α unchanged and propagates only the s.p. states. The mean-field propagation of an initial state (3) then reads

$$\hat{\rho}(t) \;=\; \sum_{\alpha=1}^{\infty} |\phi_\alpha(t)\rangle W_\alpha \langle\phi_\alpha(t)| = \hat{U}(t,0)\hat{\rho}(0)\hat{U}^{-1}(t,0) \tag{6}$$

where $\hat{U}$ is the mean-field evolution operator (2a).

2. The structure of the TDLDA package

2.1. The TDLDA calling tree

PGR2all: Here comes a calling tree as for RTA in section 4.2. This tree is much larger and may have to be splitted into static and dynamic part.

2.2. The TDLDA subroutines in detail

PGR2all: Here is to come a detailed description of all subroutines similar as for RTA in 4.3, but here for the LDA part. One example is given as appetizer.

`SUBROUTINE coul_mfield(rho)`

`rho(1:2*kdfull2)`	in/out
density for which Coulomb field is computed	

Computes Coulomb potential for given density by invoking Poisson solver. Th emerging coupomb potential is communicated as `chpcoul` via module `params`. In case of dielectric external media, adds pseudo-density for image charge.
PGR2all: Why is `rho` also `INTENT OUT`? Is all `1:2*kdfull2` used or only the first block `1:kdfull2`?

3. Relaxation-time approximation (RTA)

3.1. The formal background of RTA

The quantum Boltzmann equation is the quantum mechanical counterpart of the semi-classical Vlasov-Uehling-Uhlenbeck equation [7, 8]. It complements the self-consistent TDLDA propagation of the one-body density matrix $\hat{\rho}$ by dynamical correlations through a collision term. It reads in general [9, 10] $\mathrm{i}\partial_t\hat{\rho} - \left[\hat{h}, \hat{\rho}\right] = \hat{I}[\hat{\rho}]$ where the left hand side contains the mean-field propagation. the $\hat{I}$ at the right-hand side consists stands for the quantum-mechanical collision term which, however, is extremely hard to handle for finite Fermion systems. A great simplification can be achieved by the relaxation-time approximation (RTA) which was used successfully in a wide variety of homogeneus systems [11, 12]. The RTA equations for the present case of finite Fermion systems read [13]

$$\begin{aligned}
\partial_t\hat{\rho} &= -\mathrm{i}\big[\hat{h},\hat{\rho}\big] - \frac{1}{\tau_\mathrm{relax}}\left(\hat{\rho}-\hat{\rho}_\mathrm{eq}[\varrho,\mathbf{j},E_\mathrm{sp}]\right) , && \text{(7a)}\\
\varrho(\mathbf{r},t) &= \sum_\alpha |\phi_\alpha(\mathbf{r},t)|^2\, W_\alpha , && \text{(7b)}\\
\mathbf{j}(\mathbf{r},t) &= \sum_\alpha W_\alpha\phi^*_\alpha(\mathbf{r},t)\frac{\overrightarrow{\nabla}-\overleftarrow{\nabla}}{2\mathrm{i}}\phi_\alpha(\mathbf{r}) , && \text{(7c)}\\
E_\mathrm{sp} &= \sum_\alpha W_\alpha\varepsilon_\alpha , && \text{(7d)}\\
\frac{\hbar}{\tau_\mathrm{relax}} &= 0.40\frac{\sigma_{ee}}{r_s^2}\frac{E^*_\mathrm{intr}}{N} \;,\; r_s=\left(\frac{3\bar{\varrho}}{4\pi}\right)^{-1/3} \;,\; \sigma_{ee}=\sigma_{ee}(\bar{\varrho}) , && \text{(7e)}
\end{aligned}$$

where $\hat{\rho}_\mathrm{eq}$ is the density operator of the thermal equilibrium for local density $\varrho(\mathbf{r},t)$, current distribution $\mathbf{j}(\mathbf{r},t)$. The reference energy should be, in fact, the total energy $E(t)$ and computed from the actual state $\hat{\rho}(t)$. We replace that by the simpler total s.p. energy $E_\mathrm{sp}(t)$ which is legitimate because the difference to E is a functional of $\varrho(\mathbf{r})$ only, a quantity which is kept frozen by construction. The forms (7b,7c) hold for the diagonal representation. A crucial parameter is the relaxation time τ_relax which is taken over from semi-classical Fermi liquid theory, for details see [13]. Key entries are: the intrinsic (thermal) energy of the system E^*_intr (see appendix AppendixA), the actual number of particles N, the in-medium electron-electron cross section σ_{ee}, the effective Wigner-Seitz radius r_s of the electron cloud, and the average electron density $\bar{\varrho}$. Note that r_s and σ_{ee} depend on an average density $\bar{\varrho}$ because a spatially varying τ_relax would be very cumbersome to implement in a quantum mechanical expression. The average density is deduced from the r.m.s. radius r of the actual electron cloud as $\bar{\varrho}=(3N/(4\pi r^3))$[1].

This RTA equation (7) is rather involved because its entries depend in various ways on the actual state $\hat{\rho}(t)$. The most expensive piece is the instantaneous equilibrium density operator

$$\hat{\rho}_\mathrm{eq}[\varrho,\mathbf{j},E]=|\phi^{(\mathrm{eq})}_\alpha\rangle W^{(\mathrm{eq})}_\alpha\langle\phi^{(\mathrm{eq})}_\alpha| \tag{8a}$$

which minimizes LDA energy with constraint on the actual $\varrho(\mathbf{r})$, $\mathbf{j}(\mathbf{r})$ and energy E_sp. It is determined by the density-constrained mean-field (DCMF) equation

$$\begin{aligned}
\hat{h}_\mathrm{DCMF}[\varrho,\mathbf{j},E]\phi^{(\mathrm{eq})}_\alpha &= \varepsilon^{(\mathrm{eq})}_\alpha\phi^{(\mathrm{eq})}_\alpha && \text{(8b)}\\
\hat{h}_\mathrm{DCMF}[\varrho,\mathbf{j}] &= \hat{h}-\int d^3r\lambda(\mathbf{r})\hat{\varrho}(\mathbf{r})-\int d^3r\,\boldsymbol{\lambda}_{\boldsymbol{j}}(\mathbf{r})\hat{\mathbf{j}}(\mathbf{r})\\
&\quad -\mu\int d^3r(\hat{\varrho}(\mathbf{r})-\varrho(\mathbf{r},t))^2-\mu_j\int d^3r\,(\hat{\mathbf{j}}(\mathbf{r})-\mathbf{j}(\mathbf{r},t))^2 && \text{(8c)}
\end{aligned}$$

[1] Should we add that as displayed equation?

in combination with adjustment of particle number N and s.p. energy $E_{\rm sp}(t)$ through a Fermi distribution

$$W_\alpha^{\rm (eq)} = \frac{1}{1+\exp((\langle\phi_\alpha^{\rm (eq)}|\hat{h}|\phi_\alpha^{\rm (eq)}\rangle - \mu^{\rm (eq)})/T^{\rm (eq)})} \tag{8d}$$

$$\mu^{\rm (eq)} \leftrightarrow \sum_\alpha W_\alpha^{\rm (eq)} = N(t)\ ,\ T^{\rm (eq)} \leftrightarrow \sum_\alpha W_\alpha^{\rm (eq)} \langle\phi_\alpha^{\rm (eq)}|\hat{h}|\phi_\alpha^{\rm (eq)}\rangle = E_{\rm sp}(t)\ . \tag{8e}$$

Although cumbersome to evaluate, it is important to use exactly this local, instantaneous equilibrium in the relaxation term. This guarantees that the disspative step conserves local density, current, and energy as it is mandatory for a good collision term [14].

3.2. Observables specific to relaxation

Most of the observables computed with RTA are exactly the same as for TDLDA, e.g., energy, density, excitation spectra, ionization, PES, or PAD. New are observables related to the mixed character of the one-body operator which is characterized by the occupation numbers W_α. A specific quantity in that respect is the entropy which is computed in diagonal representation (3) by the standard expression [15]

$$S = -\sum_\alpha \left[W_\alpha \log W_\alpha + (1-W_\alpha)\log(1-W_\alpha)\right] \tag{9}$$

in units of Boltzmann constant. It serves as a direct indicator of thermalization and allows to read off the typical time scale of relaxation processes.

3.3. Summary of the RTA procedure

The solution of the RTA equations (7) with (8) is rather involved. We briefly summarize the solution scheme for one step from $t \equiv t_0$ to $t{+}\Delta t \equiv t_1$, for more details see [13][2]. The TDLDA propagation runs at a much faster pace than relaxation. We resolve it by standard techniques [16, 5] on a time step δt which is much smaller (factor 10–100) than the RTA step Δt. We summarize this TDLDA propagation in the evolution operator $\hat{U}$ from Eq. (2a) and discuss only one RTA step:

1. We first propagate $\hat{\rho}$ by pure TDLDA. The s.p. states in diagonal representation (3) evolve as $|\phi_\alpha(t)\rangle \to |\phi_\alpha^{\rm (mf)}\rangle = \hat{U}(t_1,t_0)|\phi_\alpha(t)\rangle$, while the occupation weights $W_\alpha(t_1) = W_\alpha(t_0)$ are kept frozen (pure mean-field propagation).
2. Absorbing bounds may have removed parts from the s.p. wavefunctions and so destroyed ortho-normalization. We transform the propagated density operator to a representation in terms of natural orbitals which is diagonal representation (3) with an ortho-normal set of s.p. wavefunctions together with corresponding occupation weights $\{\phi_\alpha^{\rm (nat)}, W_\alpha^{\rm (nat)}\}$. This step can be overridden if reflecting (or periodic) boundaries are used in which case TDLDA preserves ortho-normalizaton.
3. We compute density $\varrho(\mathbf{r},t_1)$, current $\mathbf{j}(\mathbf{r},t_1)$, and total energy $E_{\rm mf}$ associated to the TDLDA-propagated density matrix $\hat{\rho}_{\rm mf}$.

[2]PGR2all: can we outsource all details to reference [13]?

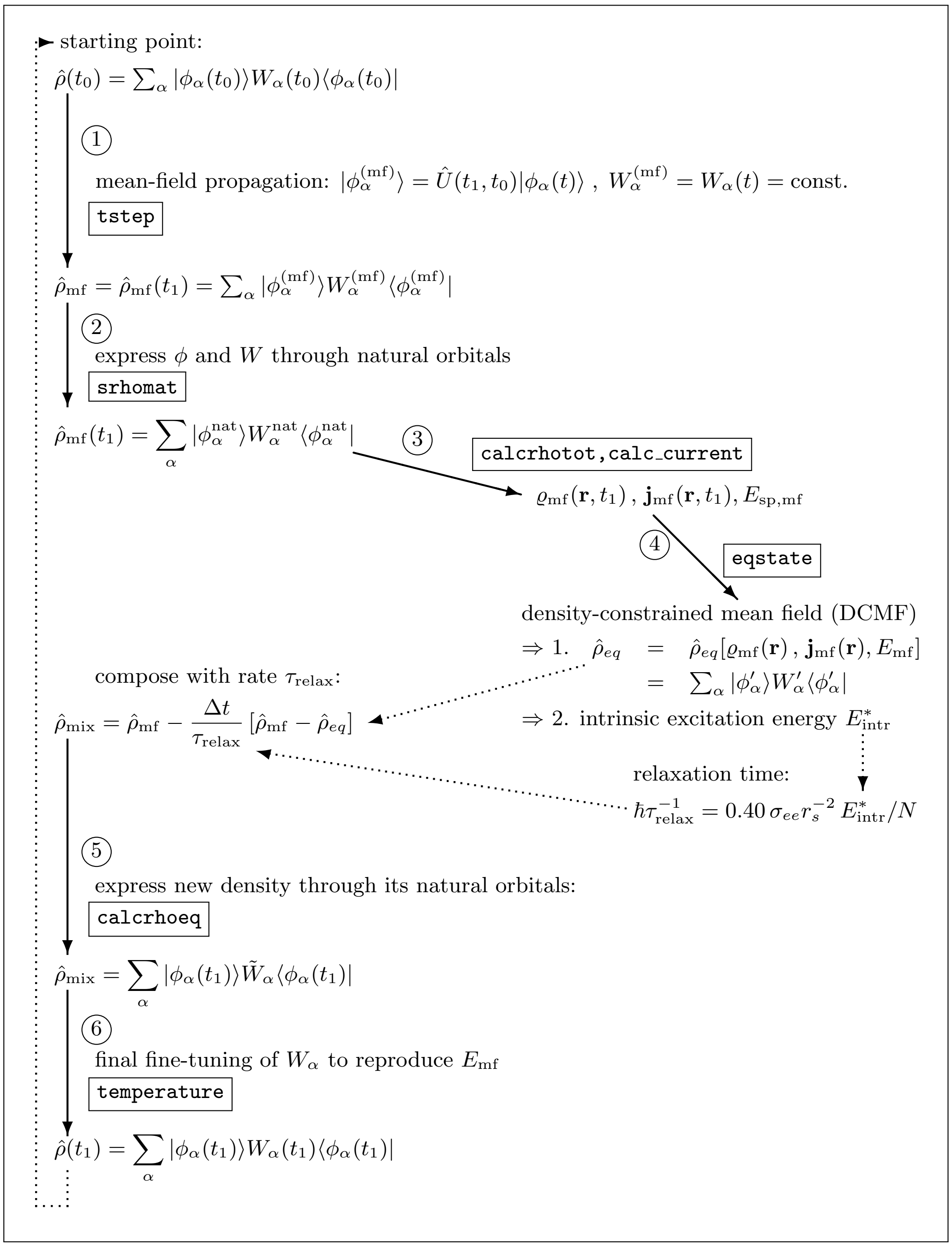
starting point:
$\hat\rho(t_0)=\sum_\alpha|\phi_\alpha(t_0)\rangle W_\alpha(t_0)\langle\phi_\alpha(t_0)|$
1
mean-field propagation: $|\phi_\alpha^{(\mathrm{mf})}\rangle=\hat U(t_1,t_0)|\phi_\alpha(t)\rangle$, $W_\alpha^{(\mathrm{mf})}=W_\alpha(t)=\mathrm{const.}$
tstep
$\hat\rho_{\mathrm{mf}}=\hat\rho_{\mathrm{mf}}(t_1)=\sum_\alpha|\phi_\alpha^{(\mathrm{mf})}\rangle W_\alpha^{(\mathrm{mf})}\langle\phi_\alpha^{(\mathrm{mf})}|$
2
express ϕ and W through natural orbitals
srhomat
$\hat\rho_{\mathrm{mf}}(t_1)=\sum_\alpha|\phi_\alpha^{\mathrm{nat}}\rangle W_\alpha^{\mathrm{nat}}\langle\phi_\alpha^{\mathrm{nat}}|$
3
calcrhotot,calc_current
$\varrho_{\mathrm{mf}}(\mathbf{r},t_1)$, $\mathbf{j}_{\mathrm{mf}}(\mathbf{r},t_1)$, $E_{\mathrm{sp,mf}}$
4
eqstate
density-constrained mean field (DCMF)
$\Rightarrow$ 1. $\hat\rho_{eq}=\hat\rho_{eq}[\varrho_{\mathrm{mf}}(\mathbf{r}),\mathbf{j}_{\mathrm{mf}}(\mathbf{r}),E_{\mathrm{mf}}]=\sum_\alpha|\phi'_\alpha\rangle W'_\alpha\langle\phi'_\alpha|$
$\Rightarrow$ 2. intrinsic excitation energy E^*_{intr}
compose with rate τ_{relax}:
$\hat\rho_{\mathrm{mix}}=\hat\rho_{\mathrm{mf}}-\frac{\Delta t}{\tau_{\mathrm{relax}}}[\hat\rho_{\mathrm{mf}}-\hat\rho_{eq}]$
relaxation time:
$\hbar\tau_{\mathrm{relax}}^{-1}=0.40\,\sigma_{ee}r_s^{-2}\,E^*_{\mathrm{intr}}/N$
5
express new density through its natural orbitals:
calcrhoeq
$\hat\rho_{\mathrm{mix}}=\sum_\alpha|\phi_\alpha(t_1)\rangle\tilde W_\alpha\langle\phi_\alpha(t_1)|$
6
final fine-tuning of W_α to reproduce E_{mf}
temperature
$\hat\rho(t_1)=\sum_\alpha|\phi_\alpha(t_1)\rangle W_\alpha(t_1)\langle\phi_\alpha(t_1)|$

Figure 1: Sketch of the scheme for performing one large time step $t_0\longrightarrow t_1=t_0+\Delta t$ in solving the RTA equations. The numbers in open circles indicate the steps as outlined in the text. The names in `typewriter` font refer to subroutines in the code as detailed in section 4.

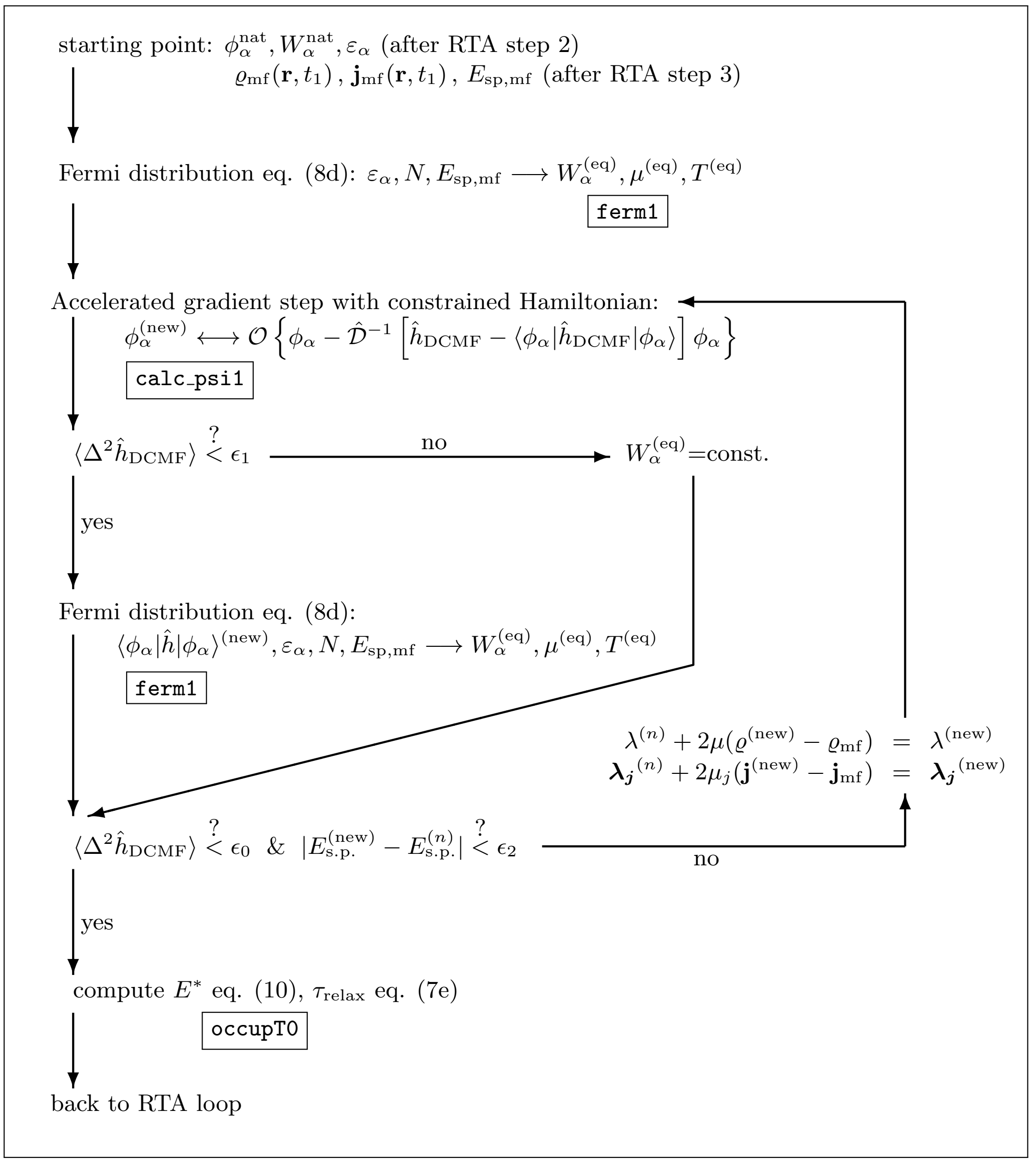
starting point: $\phi_\alpha^{\rm nat}, W_\alpha^{\rm nat}, \varepsilon_\alpha$ (after RTA step 2)
$\varrho_{\rm mf}(\mathbf{r},t_1)\,,\ \mathbf{j}_{\rm mf}(\mathbf{r},t_1)\,,\ E_{\rm sp,mf}$ (after RTA step 3)
Fermi distribution eq. (8d): $\varepsilon_\alpha, N, E_{\rm sp,mf} \longrightarrow W_\alpha^{\rm (eq)}, \mu^{\rm (eq)}, T^{\rm (eq)}$
ferm1
Accelerated gradient step with constrained Hamiltonian:
$\phi_\alpha^{\rm (new)} \longleftrightarrow \mathcal{O}\left\{\phi_\alpha - \hat{\mathcal{D}}^{-1}\left[\hat{h}_{\rm DCMF} - \langle\phi_\alpha|\hat{h}_{\rm DCMF}|\phi_\alpha\rangle\right]\phi_\alpha\right\}$
calc_psi1
$\langle\Delta^2\hat{h}_{\rm DCMF}\rangle \overset{?}{<} \epsilon_1$
no
$W_\alpha^{\rm (eq)}$=const.
yes
Fermi distribution eq. (8d):
$\langle\phi_\alpha|\hat{h}|\phi_\alpha\rangle^{\rm (new)}, \varepsilon_\alpha, N, E_{\rm sp,mf} \longrightarrow W_\alpha^{\rm (eq)}, \mu^{\rm (eq)}, T^{\rm (eq)}$
ferm1
$\lambda^{(n)} + 2\mu(\varrho^{\rm (new)} - \varrho_{\rm mf}) = \lambda^{\rm (new)}$
$\boldsymbol{\lambda_j}^{(n)} + 2\mu_j(\mathbf{j}^{\rm (new)} - \mathbf{j}_{\rm mf}) = \boldsymbol{\lambda_j}^{\rm (new)}$
$\langle\Delta^2\hat{h}_{\rm DCMF}\rangle \overset{?}{<} \epsilon_0$ & $|E_{\rm s.p.}^{\rm (new)} - E_{\rm s.p.}^{(n)}| \overset{?}{<} \epsilon_2$
no
yes
compute E^* eq. (10), $\tau_{\rm relax}$ eq. (7e)
occupT0
back to RTA loop

Figure 2: Sketch of the scheme for solving the DCMF eqs. (8). This scheme expands `SUBROUTINE eqstate` in step 4 from the RTA scheme 1. The symbol $\mathcal{O}$ stands for ortho-normalization of the newset of s.p. wavefunctions and $\mathcal{D}$ for the damping operator in the accelerated gradient step (*PGR2all: Cross references to TDLDA section yet to be defined.*).

4. We determine the thermal mean-field equilibrium state $\hat{\rho}_{\rm eq}$ constrained to the given ϱ, $\mathbf{j}$, and $E_{\rm mf}$ from step 3. This is achieved by the Density-Constrained Mean eqs. (8) which is done iteratively as sketched in figure 2. The equilibrium state $\hat{\rho}_{\rm eq}$ is represented by new s.p. states $\{|\phi'_\alpha\rangle\}$ and new occupation numbers W'_α in diagonal form (3). Having these, we determine finally the excitation energy as energy relative to the temperture zero state

$$E^*_{\rm intr} = E_{\rm sp} - \sum_\alpha W_\alpha^{(T=0)} \langle\phi_\alpha^{\rm (eq)}|\hat{h}|\phi_\alpha^{\rm (eq)}\rangle \tag{10}$$

where $W_\alpha^{(T=0)}$ are the occupation numbers determined with the s.p. energies $\langle\phi_\alpha^{\rm (eq)}|\hat{h}|\phi_\alpha^{\rm (eq)}\rangle$ at temperature $T=0$. This $E^*_{\rm intr}$ thus measures the amount of thermal excitation energy in the system[3]

5. We compose the new density operator as mixture of TDLDA propagated state $\hat{\rho}_{\rm mf}$ and equilibration driving term $\hat{\rho}_{\rm mf} - \hat{\rho}_{\rm eq}$ with weight $\Delta t/\tau_{\rm relax}$ as

$$\hat{\rho}_{\rm mix} = \hat{\rho}_{\rm mf} - \frac{\Delta t}{\tau_{\rm relax}} \left[\hat{\rho}_{\rm mf} - \hat{\rho}_{eq}\right]$$

where the relaxation time $\tau_{\rm relax}$ requires the actual intrinsic excitation energy $E^*_{\rm intr}$ which is also obtained from DCMF. While evaluating the mixing, the new state is expressed in natural-orbital representation Eq. (3). This yields the final s.p. states $\{|\phi_\alpha(t_1)\rangle\}$ for this step and preliminary new occupations $\widetilde{W}_\alpha$.

6. The mixing in step 5 may have slightly changed the energy such that we remain with a small energy mismatch as compared to the goal $E_{\rm mf}$. We now apply a small iterative thermalization step to readjust the energy, as outlined in Appendix AppendixB. This then yields the final occupation weights $W_\alpha(t_1)$ which comply with energy conservation.

The above steps are sketched in Figure 1 whereby the step numbers here correspond to the ones in the Figure. The most involved part is step 4, the solution of the DCMF equations. It is expanded in details in figure 2.

As said above, the time step δt for propagation of TDLDA is very small because it is limited from above by the maximal energy on the grid representation. The stepping Δt for the relaxation term needs only to resolve the changes in the actual mean field which allows much larger values. Typically, we can do 50–200 TDLDA steps before calling one RTA step. For detailed values see the examples delivered with the code.[4]

A word is in order about the system for which the present form of RTA can be used. The relaxation time $\tau_{\rm relax}$ is allowed to depend on time which allows to accomodate changes of the dynamical state. But $\tau_{\rm relax}$ is one global number chosen according to the average electron density $\bar{\varrho}$, see eq. (7e). This requires systems which can be characterized by such an average density, i.e., systems having only small density variations in the bulk as it holds typically for metallic bonds. The RTA rate is insensitive to many details of the microscopic collision term as energy- and angle-dependent scattering cross sections [18] or a broad spectrum of relaxation rates. However, these details are usually resolved

[3] Is this definition sufficient? Then we could skip planned appendix AppendixA.
[4] PGR2all: A link to be set once we have the place for the benchmarks.

only (if at all) for fast and energetic processes which are anyway deep in the regime of semi-classical VUU. The grossly averaged treatment of RTA is acceptable for not too fast and not too energetic processes, preferably in compact systems.

4. The structure of the RTA package in `rta.F90`

4.1. Input and output related to RTA

PGR2all: This subsection needs yet to be worked out in detail.

The `NAMELIST dynamic` contains the following variables used in the RTA procedure:[5]

`jrtaint:`

`rtamu:`

`rtamuj:`

`rtasumvar2max:`

`rtaeps:`

`rtae0dmp:`

`rtasigee:`

`rtars:`

`rtatempinit:`

`rtaforcetemperature:`

Most observables were already defined at TDLDA stage and thus are returned in the standard output files as explained in the TDLDA section[6]. New output files specific to observables from RTA are:[7]

`prta:`

`peqstat:`

`pspeed:`

`prhov:`

[5]We should define a new `NAMELIST` and shift the RTA variables to there.
[6]Cross reference yet to be defined.
[7]These files seem to carry only protocol for numerics. Where do we print energy balance ($E^*_{\rm intr}$ etc)?

4.2. The calling tree

Here is an oversight over the tree structure of the RTA routines. Those subroutines contained in `rta.F90` are explained in detail in section 4.3. Subroutines coming from the TDLDA package or external sources are marked[8]

```
RTA
 ├──── srhomat
 │      ├──── scalar
 │      └──── cdiagmat
 │             └──── HEigensystem                    ↔ library routine
 ├──── eqstate
 │      ├──── calcrhotot
 │      ├──── calc_current
 │      ├──── calc_Eref
 │      ├──── fermi1
 │      │      └──── occT1
 │      ├──── calc_psi1
 │      │      ├──── calc_hamiltonien
 │      │      │     calc_var
 │      │      │      └──── cproject                 ↔ obsolete?
 │      │      ├──── calcrhotot
 │      │      ├──── calc_current
 │      │      ├──── cschmidt                        ↔ TDLDA package
 │      │      ├──── calc_ekin                       ↔ TDLDA package
 │      │      ├──── nonlocalc                       ↔ TDLDA package
 │      │      ├──── fftf                            ↔ FFTW3 package
 │      │      └──── fftback                         ↔ FFTW3 package
 │      ├──── coul_mfield                            ↔ TDLDA package
 │      ├──── dyn_mfield                             ↔ TDLDA package
 │      └──── info                                   ↔ TDLDA package
 ├──── occupT0
 │      └──── indexx
 ├──── calcrhoeq
 │      ├──── cdiagmat
 │      │      └──── HEigensystem                    ↔ library routine
 │      └──── indexx
 ├──── calc_Eref
 ├──── CorrectEnergy2
 ├──── OccupPerSpin
 ├──── temperature
 │      └──── lmdif1
 ├──── dyn_mfield                                    ↔ TDLDA package
 ├──── info                                          ↔ TDLDA package
 └──── analyze_elect                                 ↔ TDLDA package
```

[8]The `HEigensystem` seems copied from some library. This could cause copyright problems if we publish the code. Is it from BLAS/LINPACK? Then we could replace the Fortran source by a library call.

4.3. The subroutines in detail

`SUBROUTINE rta(psi,aloc,rho,iterat)`

`iterat`	in	external iteration number (TDLDA time step)
`psi(1:kdfull2,1:kstate)`	in/out	set of s.p. wavefunctions
`rho(1:2*kdfull2)`	in/out	local densities for spin up and down
`aloc(1:2*kdfull2)`	in/out	local potentials for spin up and down

Basic RTA routine performing density constrained mean-field (DCMF) iterations, energy adjustment, admixing of local equilibrium states by calls to subroutines (see calling tree).

`SUBROUTINE calcrhoeq(psiorthloc,psieqloc,psiloc,occuporthloc,occuploc,nstateloc)`

`nstateloc`	in	number of s.p. states in spin block
`psiorthloc(1:kdfull2,1:nstateloc),`	in	set of TDLDA wavefunctions (natural orbitals)
`occuporthloc(1:nstateloc)`	in	occupations of TDLDA states
`psieqloc(1:kdfull2,1:nstateloc)`	in	set of local-equilibrium wavefunctions
`psiloc(1:kdfull2,1:nstateloc)`	out	set of final mixed wavefunctions
`occuploc(1:nstateloc)`	in/out	

Encapsulated in `SUBROUTINE rta`. Performs the mixing of TDLDA states with local-equilibrium state according to relaxation rate for one spin block. The mixed densty matrix is expanded in a representation by both sets of s.p. states.

`SUBROUTINE calc_Eref(occup,ispin,Ei,Eref)`

`occup(1:nstate)`	in	occupation number for s.p. states.
`ispin(1:nstate)`	in	spin assignement for s.p. states.
`Ei(1:nstate)`	in	spin assignement for s.p. states.
`Eref(1:2)`	out	sum of s.p. energies per spin.

Computes the weighted sum of s.p. energies as reference energy for DCMF. The sum is accumulated for each spin separately.

`SUBROUTINE fermi1(ekmod,eref,occup,ispinact,T0i,T1i,T2,mu)`

`ekmod(1:kstate)`	in	given s.p. energies, spin up block first, then spin down
`eref`	in	reference energy = wanted sum of s.p. energies
`ispinact`	in	spin for which routine is run
`T0i, T1i`	in	lower and upper temperature for search
`occup(1:kstate)`	in/out	occupation numbers, spin block-wise
`T2`	out	final temperature for which Fermi distribution matches `eref`
`mu`	out	final chemical potential

Determines thermal Fermi occupation such that given sum of s.p. energies `eref` and particle number is matched. Is done for each spin separately. Solution scheme is bracketing. Refers to `SUBROUTINE OccT1` while iterating temperatur `T2`.
PGR2all: Nr. of spin-up/spin-down states comes through `m_params`*. We should protocol all such entries. First step is to augment each* `USE` *by* `ONLY` *such that the explicitely communicated variables becomes visible. Important variables may then be listed explicitely.*
PGR2all: Routine requires that arrays are sorted in continuous blocks of spin. Do we have an initial check for that? And we need to address that in the general part which explains the layout of arrays.

`SUBROUTINE OccT1(occrefloc,enerloc,Etotloc,muloc,occtotloc,n,T,occuploc)`

`enerloc(1:n)`	in	s.p. energies for actual spin
`n`	in	number of s.p. states treated here
`T`	in	temperature
`occrefloc`	in	wanted total number of particles
`occuploc(1:n)`	out	thermal occupation numbers for given `T` and s.p. energies
`muloc`	out	chemical potential (Fermi energy)
`occtotloc`	out	final total number of particles
`Etotloc`	out	sum of s.p. energies

Determines by bracketing chemical potential `muloc` for given array of s.p. energies, temperature `T`, and wanted number of particles `occrefloc` with precision `1D-12`. Delivers with it thermal occupation numbers and corresponding total particle number and sum of s.p. energies.
PGR2all: This routine is specific to /tt SUBROUTINE ferm1. Could we encapsulate it by a /tt CONTAINS?

`SUBROUTINE Calc_psi1(psi1,aloc,rhotot0,rhototloc,curr0,curr1,j,lambda,mu,lambdaj,muj,sumvar2,e`
combined with encapsulated `SUBROUTINE calc_hamiltonien`.

`j`	in	number of DCMF iteration, used here for print
`lambda(1:kdfull2,1:2)`	in	Lagrange parameter for density for spin up&down
`lambdaj(1:kdfull2,1:3)`	in	Lagrange parameter for current
`mu, muj`	in	driving parameter for augmented Lagrangian
`aloc(1:2*kdfull2)`	in	local potentials for spin up and down
`rhoto0(1:kdfull2,1:2)`	in	initial density *PGR2all: not used ??*
`curr0(1:kdfull2,1:3)`	in	wanted current
`psi1(1:kdfull2,1:kstate)`	in/out	set of s.p. wavefunctions iterated
`rhototloc(1:kdfull2,1:2)`	out	actual density according to `psi1`
`curr1(1:kdfull2,1:3)`	out	actual current from `psi1`
`ekmod(1:nstate)`	out	final s.p. energies
`eal`	out	final sum of s.p. energies
`sumvar2`	out	variance of s.p. energies

Performs one damped gradient step of with density & current constrained Hamiltonian.
PGR2all: The density array distinguishes spin up/down while the current array does not. Reason?
PGRcommThe IN & OUT assignments in this subroutine have to be updated.

`SUBROUTINE eqstate(psi,aloc,rho,psi1,occuporth,iterat)`

`iterat`	in	actual iteration number (for printing)
`psi(1:kdfull2,1:kstate)`	in	initial set of s.p. wavefunctions
`psi1(1:kdfull2,1:kstate)`	out	final set of s.p. wavefunctions
`aloc(1:2*kdfull2)`	in/out	local part of potential, spin up/down stacked in blocks
`rho(1:2*kdfull2)`	in	initial density, spin up/down stacked in blocks
`occuporth(1:kstate)`	in	occupation numbers for `psi` and still the same for `psi1`.

DCMF iterations by reapeatedly calling `Calc_psi1`, updating Lagrangian parameters for density & current constraints, and occassionally tuning temperature to achieve correct energy. The latter is done by calling `fermi1`. The local potential is kept constant during

DCMF iteration and updated only at the very end.
PGR2all: Fetches nr. of spin up/down from `m_params`*.*
PGR2all: Lagrange parameters are started from scratch. May it be faster to recycle the previous Lagrange parameters?
PGR2all: Density `rho` *is entered via list and still recomputed as* `rhotot0`*. Unnecessary doubling?*

`SUBROUTINE OccupT0(occloc,esploc,Estar)`

`esploc(1:nstate)`	in	given s.p. energies
`occloc(1:nstate)`	in	given occupation numbers
`Estar`	out	excitation energy relative to T=0 distribution

Computes thermal excitation energy as difference of actual energy to the energy obtained by Fermi distribution for $T = 0$. The latter distributions is computed for the given s.p. energies which are the same as used for the thermal state.

`SUBROUTINE calcrhotot(rho,q0)`

`q0(1:kdfull2,1:kstate)`	in	set of s.p. wavefunctions for which density is accumulated
`rho(kdfull2,2)`	out	resulting density

Computes local density for set of wavefunctions `q0`. Note that two crucial information is communicated via module `params`, namely `occup`, the array of occupation numbers, `ispin` the array assigning spin top each s.p. state, and `nstate`, the number of s.p. states.
PGR2all: Exploiting the sorting of spin in blocks of s.p. states, we could rewrite the code with to `SUM` *statements.*

`SUBROUTINE calc_var(hpsi,psi1,sumvar2)`

`psi1(kdfull2,kstate)`	in	set of s.p. states for which variance of s.p. energies of calculated
`hpsi(kdfull2,kstate)`	in/out	array $H \rightarrow \psi_\alpha$, on input in k-space, on output in r-space
`sumvar2`	out	summed variance of s.p. energies

Computes the sum of variances of the s.p. energies, $\langle\hat{\Delta h}^2|rangle$.
PGRcommThe routine projects from each $hath\psi_\alpha$ all s.p. states ψ_β from the pool of states. That is too much. The s.p. variance should be $\sum_\alpha\langle|\psi_\alpha|(\hat{h}-\varepsilon_\alpha)^2|\psi_\alpha\rangle$ where $\varepsilon_\alpha = \langle|\psi_\alpha|\hat{h}|\psi_\alpha\rangle$.

`SUBROUTINE forceTemp(amoy,occup,n,temp,mu)`

`amoy(1:n)`	in	given s.p. energies
`occup(1:n)`	in	given thermal occupation
`n`	in	number of s.p. states
`temp`	in	temperature
`mu`	out	emerging chemical potential

Determines chemical potential for given s.p. energies and temperature by call to `OccT1`.
PGR2all: Obsolete and never used.

`SUBROUTINE fermi_init(ekmod,T,occup,ispinact)`

`ekmod(1:nstate)`	in	given s.p. energies
`T`	in	given temperature
`ispinact`	in	actual spin
`occup(1:kstate)`	in/out	initial occupation and resulting Fermi distribution for `T`.

Determines Fermi distribution for given s.p. energies and temperature. Searches appropriate chemical potential `mu` by bracketing. Use for repreated calls to `FUNCTION occ`.
PGR2all: This routine `fermi_init` and the related `FUNCTION occ` are never used, thus obsolete. May be removed.

`SUBROUTINE srhomat(psi,aloc,psiorth,occuporth)`

`psi(1:kdfull2,1:kstate)`	in	set of s.p. wavefunctions, not orth-normalized
`psiorth(1:kdfull2,1:kstate)`	out	ortho-normalized natural orbitals
`aloc(1:2*kdfull2)`	in	actual local potential
`occuporth(1:kstate)`	out	occupation numbers for ortho-normalized states

Computes the density matrix of initial state goiven by set of wavefunctions `psi` together with their occupations `occup`, the latter communicated through module `params`. Then diagonalizes the density matrix and computes on `psiorth` the new wavefunctions associated with diagonal representation of the density matrix.
Finally updates running transformation matrix `psitophi` which is communicated and stored through module `params`.
PGR2all: Usage and propagation of `psitophi` is somewhat hidden because it is handled through a module. Needs to be explained somwhere.

`SUBROUTINE scalar(tab1,tab2,scal,ispin, mess)`

`tab1(1:kdfull2,1:kstate)`	in	1. set of s.p. wavefunctions
`tab2(1:kdfull2,1:kstate)`	in	2. set of s.p. wavefunctions
`ispin(1:nstate)`	in	spin of s.p. states
`mess`	in	message for print inside routine
`scal(nstate,nstate)`	out	matrix of wavefunction overlaps

`SUBROUTINE cdiagspin(mat, eigen, vect, N)`

`mat(N,N)`	in	complex Hermitean matrix to be diagonalized
`N`	in	dimension of matrix
`eigen(N)`	out	resulting eigenbvalues
`Vect(N,N)`	out	resulting eigenstates

Driver routine for diagonalization of a complex Hermitean matrix of dimension `N` which consists in a two blocks for separate spin. Refers for each single block to routine `cdiag` and subsequent library routines contained therein.

`SUBROUTINE indexx (n,arrin,indx)`

`n`	in	length of array
`arrin(1:n)`	in	array to be sorted
`indx(1:n)`	out	pointer array

Evaluates sorting of an array in ascending order.

`SUBROUTINE occupPerSpin(mess,Occ)`

`mess`	in	character variable with comment printed inside routine
`Occ(1:2)`	out	total number of particles in each spin

Computes number of particles in each sin block. Uses `nstate` and occupations `occup` from module `params`.

`CorrectEnergy2(Wref,Eref,w,E,Wout,nloc)`

`W(1:nloc)`	in	initial occupations numbers
`E(1:nloc)`	in	given s.p. energies
`Wref`	in	reference particle number to be reached
`Eref`	in	reference sum of s.p. energies to be reached
`nloc`	in	actual number of states
`Wout(nloc)`	out	readjusted occupation numbers

Final energy correction by one step along Fermi distribution (using Taylor expansion about actual distribution), see appendix AppendixB).

`SUBROUTINE ordo_per_spin(psi)`

`psi(1:kdfull2,1:kstate)`	in/out	s.p. wavefunctions before and after reordering

Reorder states in two blocks of spin up and down. Applies that reshuffling to all relevant field of states, s.p. wavefunctions `psi`, spin per state `ispin`, and occupations `occup`.
PGR2all: Routine has been rendered obsolete by new initialization of states which produces immediately the correct sorting. But routine should be kept for possible later use (e.g., mixing states from different sources.

`SUBROUTINE temperature(mu,T)`

`mu`	out	resulting chemical potential
`T`	out	resulting temperature

Takes s.p. energies `amoy` and occupations `occup` from module `params` and fits a Fermi distribution to it. Temperature and chemical potentials of the fitted distribution are returned via list. Calls a fitting routine `lmdif1` using subroutine `ff` as argument.

`SUBROUTINE ff(m,n,X,FVEC,IFLAG)`

`X(1:n)`	in	array handling chemical potential and temperature
`Fvec(1:m)`	out	array of mismatches of distributions
`n`	in	number of parameters of model, actually 2
`m`	in	number of entries in array
`iflag`	in	flaf possibly written (actually not used)

Mismatch of `occup` (via modules `params`) from Fermi distribution to given chemical potential and temperature. To be used in fitting routine `lmdef1`.

`SUBROUTINE cproject(qin,qout,ispact,q0)`

`qin(1:kdfull2)`	in	s.p. wvaefunction to be projected
`q0(1:kdfull2,1:kstate)`	in	set of s.p. wavefunctions which is projected out from `qin`
`ispact`	in	spin associated with `qin`
`qout(1:kdfull2)`	out	projected s.p. wavefunction

Projects away from `qin` all contributions of the set `q0`.
PGR2all: This routine may become obsolete if we recode the the variance in routine `calc_var` *to meet the standard definition.*

AppendixA. The intrinsic excitation energy

PGR2all: Intrinsic excitation energy yet to be explained. May also become part of DCMF explanation.

AppendixB. Iterative correction of total energy

PGR2all: Yet to be imported.

[1] E. K. U. Gross, W. Kohn, Adv. Quant. Chem. 21 (1990) 255.
[2] E. K. U. Gross, J. F. Dobson, M. Petersilka, Top. Curr. Chem. 181 (1996) 81.
[3] C. Legrand, E. Suraud, P.-G. Reinhard, J. Phys. B 35 (2002) 1115.
[4] P. Klüpfel, P. M. Dinh, P.-G. Reinhard, E. Suraud, Phys. Rev. A 88 (2013) 052501.
[5] P.-G. Reinhard, E. Suraud, Introduction to Cluster Dynamics, Wiley, New York, 2004.
[6] R. M. Dreizler, E. K. U. Gross, Density Functional Theory: An Approach to the Quantum Many-Body Problem, Springer-Verlag, Berlin, 1990.
[7] G. F. Bertsch, S. Das Gupta, Phys. Rep. 160 (1988) 190.
[8] Y. Abe, S. Ayik, P.-G. Reinhard, E. Suraud, Phys. Rep. 275 (1996) 49.
[9] H. Reinhardt, P.-G. Reinhard, K. Goeke, Phys. Lett. B 151 (1985) 177.
[10] K. Goeke, P.-G. Reinhard, H. Reinhardt, Ann. Phys. (N.Y.) 166 (1986) 257.
[11] D. Pines, P. Nozières, The Theory of Quantum Liquids, W A Benjamin, New York, 1966.
[12] N. W. Ashcroft, N. D. Mermin, Solid State Physics, Saunders College, Philadelphia, 1976.
[13] P.-G. Reinhard, E. Suraud, A quantum relaxation-time approximation for finite fermion systems, Ann. Phys. (N.Y.) 354 (2015) 183.
URL `http://dx.doi.org/10.1016/j.aop.2014.12.01`
[14] K. Gütter, P.-G. Reinhard, C. Toepffer, Phys. Rev. A 38 (1988) 1641.
[15] L. E. Reichl, A Modern Course in Statistical Physics, Wiley, New York, 1998.
[16] F. Calvayrac, P.-G. Reinhard, E. Suraud, C. A. Ullrich, Phys. Rep. 337 (2000) 493.
[17] P.-G. Reinhard, P. D. Stevenson, D. Almehed, J. A. Maruhn, M. R. Strayer, Phys. Rev. E 73 (2006) 036709.
[18] E. Giglio, P.-G. Reinhard, E. Suraud, Phys. Rev. A 67 (2003) 043202.