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

We present a RTA+TDLDA code on a cartesian 3D grid ....

Keywords: electron-electron collisions, electronic disspation, time-dependent density functional theory, metal cluster, plasmon, electron emission

---

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

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

1.1. 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 1.2. Subroutines coming from the TDLDA package or external sources are marked[1]

```
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
```

[1] 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.

`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 Eq. (??)[2]

`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.

[2]This equation from the theory part, yet to be written.

2. Formula from Ann.Phys. paper

2.1. Mean-field propagation

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 [? ?]. It is augmented by a self-interaction correction (SIC) approximated by average-density SIC (ADSIC) [?] in order to attain correct ionization properties [?] 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 $\varrho(\mathbf{r}, t) = \sum_\alpha |\phi_\alpha(\mathbf{r}, t)|^2$ [? ?]. The time evolution delivered by Eq. (1) can be expressed formally by the unitary one-body time-evolution operator

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

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).

2.2. RTA in quantum-mechanical framework

The generalization of the one-body phase-space distribution $f(\mathbf{r},\mathbf{p})$ to a quantum-mechanical mean-field theory is the one-body density operator $\hat{\rho}$, or one-body density matrix $\rho(\mathbf{r},\mathbf{r}')$ respectively. The equation of motion for $\hat{\rho}$ including dynamical correlations reads in general [**? ?**]

$$\mathrm{i}\partial_t\hat{\rho} - \big[\hat{h},\hat{\rho}\big] \;=\; \hat{I}[\hat{\rho}] \quad . \tag{7}$$

The left hand side embraces the mean-field propagation. It may be time-dependent Hartree-Fock or the widely used LDA version of TDDFT. The right-hand side consists of the quantum-mechanical collision term. Motivated by the successful semi-classical RTA, we import Eq. (**??**) for the quantum case as

$$\partial_t\hat{\rho} \;=\; -\mathrm{i}\big[\hat{h},\hat{\rho}\big] - \frac{1}{\tau_{\mathrm{relax}}}\left(\hat{\rho} - \hat{\rho}_{\mathrm{eq}}[\varrho,\mathbf{j},E]\right) \ , \tag{8}$$

where $\hat{\rho}_{\mathrm{eq}}$ is the density operator of the thermal equilibrium for local density $\varrho(\mathbf{r},t)$, current distribution $\mathbf{j}(\mathbf{r},t)$ and total energy $E(t)$ given at that instant of time t. These constraining conditions are, in fact, functionals of the actual state $\hat{\rho}$, i.e. $\varrho[\hat{\rho}]$, $\mathbf{j}[\hat{\rho}]$, and $E[\hat{\rho}]$. For the diagonal representation Eq.(3) of the density operator $\hat{\rho}$, they read

$$\varrho(\mathbf{r}) = \sum_\alpha |\phi_\alpha(\mathbf{r})|^2\, W_\alpha \quad , \quad \mathbf{j}(\mathbf{r}) = \sum_\alpha W_\alpha \phi_\alpha^*(\mathbf{r}) \frac{\overrightarrow{\nabla} - \overleftarrow{\nabla}}{2\mathrm{i}} \phi_\alpha(\mathbf{r}) \quad . \tag{9}$$

The energy $E(t)$ is taken as the total energy because the semi-classical concept of a local kinetic energy is ambiguous in a quantum system. This RTA equation (8) looks innocent, but is very involved because many entries depend in various ways on the actual state $\hat{\rho}(t)$. The self-consistent mean field is a functional of the actual local density, i.e. $\hat{h} = \hat{h}[\varrho]$. The instantaneous equilibrium density $\hat{\rho}_{\mathrm{eq}}$ is the solution of the stationary, thermal mean-field equations with constraint on the actual $\varrho(\mathbf{r})$, $\mathbf{j}(\mathbf{r})$ and energy E, for details see Appendix **??**.

The relaxation time τ_{relax} is estimated in semi-classical Fermi liquid theory, for details see appendix **??**. For the metal clusters serving as test examples in the following, it becomes

$$\frac{\hbar}{\tau_{\mathrm{relax}}} = 0.40 \frac{\sigma_{ee}}{r_s^2} \frac{E^*_{\mathrm{intr}}}{N} \quad , \tag{10}$$

where E^*_{intr} is the intrinsic (thermal) energy of the system (appendix **??**), N the actual number of particles, σ_{ee} the in-medium electron-electron cross section, and r_s the effective

Wigner-Seitz radius of the electron cloud. Note that r_s is tuned to the average density of the electron cloud (appendix **??**), because a spatially varying $\tau_{\rm relax}$ would be very cumbersome to implement in a quantum mechanical context. This approximation is legitimate for metallic systems where the density remains generally close to the average.

2.3. Summary of the procedure

The solution of the RTA equations is rather involved. We explain the necessary steps here from a practical side and unfold details in the appendices. We briefly summarize the actual scheme for one step from t to $t{+}\Delta t$. Note that mean-field propagation (actually TDLDA) runs at a much faster pace than relaxation. We resolve it by standard techniques [**? ?**] 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. Its sub-steps are sketched in Figure 1 and explained in the following whereby the label here correspond to the ones in the Figure:

1. We first propagate $\hat{\rho}$ by pure TDLDA. This means that the s.p. states in representation (3) evolve as $|\phi_\alpha(t)\rangle \to |\phi_\alpha^{(\rm mf)}\rangle = \hat{U}(t+\Delta t, t)|\phi_\alpha(t)\rangle$, while the occupation weights W_α are kept frozen (pure mean-field propagation).
2. We compute density $\varrho(\mathbf{r}, t{+}\Delta t)$, current $\mathbf{j}(\mathbf{r}, t{+}\Delta t)$, and total energy $E_{\rm mf}$ associated to the TDLDA-propagated density matrix $\hat{\rho}_{\rm mf}$.
3. We determine the thermal mean-field equilibrium state $\hat{\rho}_{\rm eq}$ constrained to the given ϱ, $\mathbf{j}$, and $E_{\rm mf}$. This is achieved by Density-Constrained Mean Field (DCMF) iterations as outlined in Appendix **??**. The actual 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).
4. We compose the new density matrix from the TDLDA propagated state $\hat{\rho}_{\rm mf}$ and the equilibration driving term $\hat{\rho}_{\rm mf} - \hat{\rho}_{\rm eq}$ with the appropriate weight $\Delta t/\tau_{\rm relax}$, as outlined in Appendix **??**. The relaxation time Eq. (10) requires the actual intrinsic excitation energy $E^*_{\rm intr}$ which is also obtained from DCMF, see appendix **??**.
5. We diagonalize the state emerging from step 4 to natural-orbital representation Eq. (3). This yields the s.p. states $\{|\phi_\alpha(t{+}\Delta t)\rangle\}$ for the next step and preliminary new occupations $\tilde{W}_\alpha$.
6. After all these steps, the initial energy $E_{\rm mf} = E_{\rm TDLDA}(t)$ may not be exactly reproduced. We may 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 **??**. This then yields the final occupation weights $W_\alpha(t{+}\Delta t)$ which comply with energy conservation.

The scheme can be used also in connection with absorbing boundary conditions [**? ?**]. The particle loss will be mapped automatically to loss of occupation weights in step 4. A word is in order about the choice of the time steps. The δt for propagation of TDLDA is limited by the maximal energy on the grid representation and thus very small (for Na clusters typically 0.005 fs). The stepping for the relaxation term needs only to resolve the changes in the actual mean field which is achieved already with $\Delta t \approx 0.5$ fs. We have tested a sequence of Δt and find the same results for all $\Delta t \leq 0.5$ fs. Changes appear slowly above that value. For reasons of efficiency, we thus use the largest safe value of $\Delta t = 0.5$ fs.

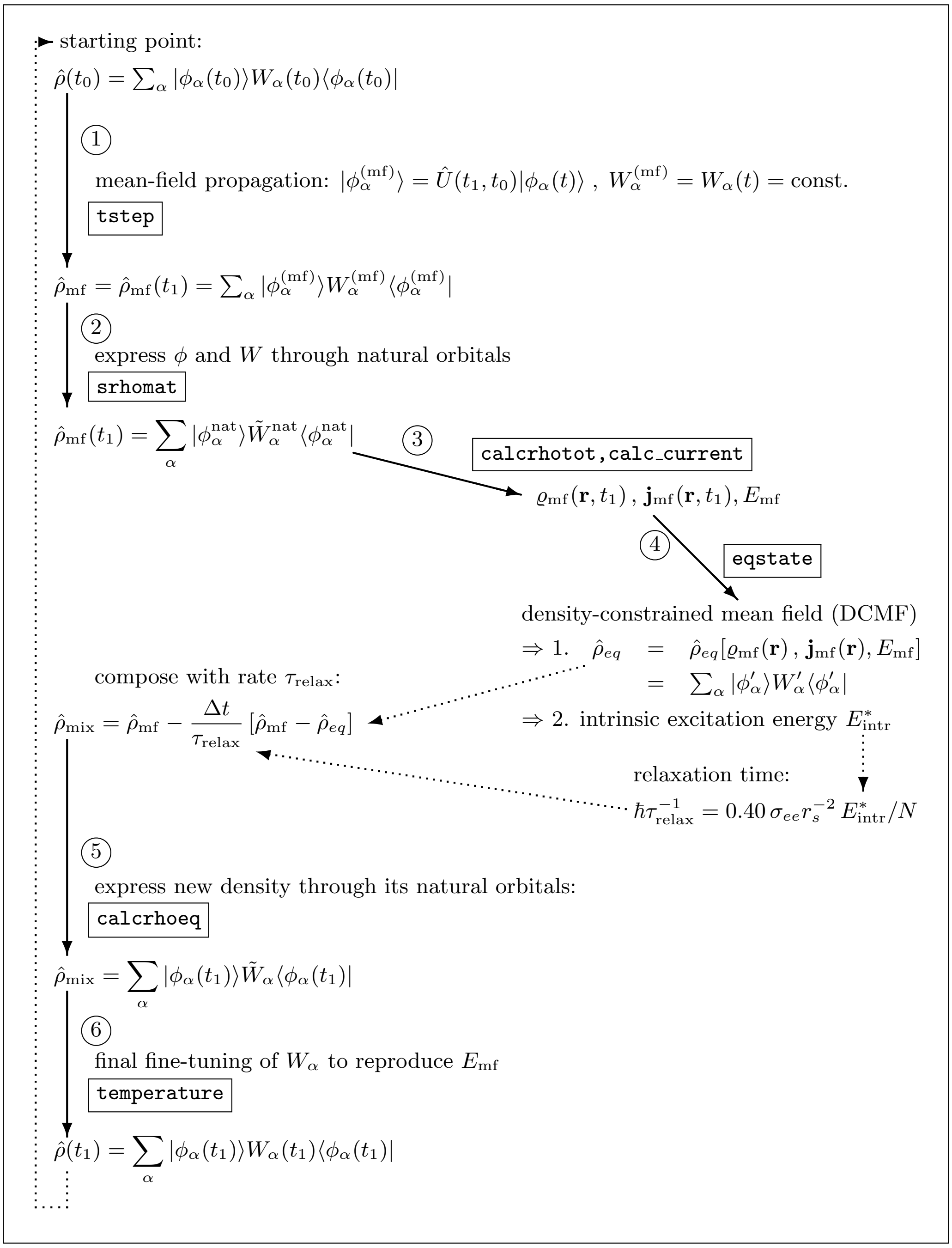

Figure 1: Sketch of the scheme for performing one large time step $t_0 \longrightarrow t_1 = t_0 + \Delta t$ in solving the RTA equations. The numbers in open circles indicate the steps as outlined in the text.

A word is in order about the range of applicability of the RTA for finite fermion systems. 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 at each instant if time one global number chosen according to the average electron density. This requires systems with only small density variations in the bulk as it holds typically for metallic bonds. The RTA is insensitive to many details of the VUU collision term as energy- and angle-dependent scattering cross sections or a broad spectrum of relaxation rates. However, these details are usually resolved 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 in compact metallic systems.

2.4. Numerical representation and computation of relevant observables

The numerical implementation of TDLDA is done in standard manner [**?** **?**]. The coupling to the ions is mediated by soft local pseudopotentials [**?**]. The Kohn-Sham potential is handled in the Cylindrically Averaged Pseudo-potential Scheme (CAPS) [**?** **?**], which has proven to be an efficient and reliable approximation for metal clusters close to axial symmetry. Wavefunctions and fields are thus represented on a 2D cylindrical grid in coordinate space [**?**]. For the typical example of the Na_{40} cluster, the numerical box extends up to 104 a_0 in radial direction and 208 a_0 along the z-axis, while the grid spacing is 0.8 a_0. To solve the (time-dependent) Kohn-Sham equations (1) we use time-splitting for time propagation [**?**] and accelerated gradient iterations for the stationary solution [**?**]. The Coulomb field is computed with successive over-relaxation [**?**]. We use absorbing boundary conditions [**?** **?**], which gently absorb all outgoing electron flow reaching the bounds of the grid and thus prevent artifacts from reflection back into the reaction zone. We take the exchange-correlation energy functional from Perdew and Wang [**?**].

A great manifold of observables can be deduced from the $\hat{\rho}(t)$ thus obtained. We will consider in the following the dipole signal, dipole spectrum, ionization, angular distribution of emitted electrons, and entropy. We focus here on the dipole moment along symmetry axis z, which is obtained from the local density as $\langle\hat{d}_z\rangle(t) = \int d^3r\, d_z(\mathrm{z})\varrho(\mathrm{r})$ where $d_z(\mathrm{z}) = z$ is the (local) dipole operator. The dipole strength distribution is computed with the methods of spectral analysis [**?**]. It is attained by an instantaneous dipole-boost excitation, collecting $\langle\hat{d}_z\rangle(t)$ during propagation, and finally Fourier transforming $\langle\hat{d}_z\rangle(t)$ into frequency domain. The angular distribution of emitted electrons is obtained from recording the absorbed electrons as in TDLDA [**?** **?**]. The angular distribution is characterized by the anisotropy parameter β_2, the leading parameter in the photo-electron angular cross section $d\sigma/d\Omega \propto (1+\beta_2 P_2(cos(\theta)+....)$ [**?** **?**] where P_2 is the second order Legendre polynomial and θ the direction with respect to laser polarization axis (here z-axis in 2D cylindrical geometry). A specific quantity to track relaxation processes is the one-body entropy which is computed in diagonal representation (3) by the standard expression [**?**]

$$S = -\sum_\alpha \left[W_\alpha \log W_\alpha + (1-W_\alpha)\log(1-W_\alpha)\right] \qquad (11)$$

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.