Handling of the cluster 3D Fortran90-code

Instructions and status report

Started 23 April 2010; status 20. December 2016

Contents

1 Installation and usage

1.1 Installation

1. Choose and create a directory where to install a code.
2. Download in the chosen directory the package pw-teleman_122016.tar.gz
3. Uncompress and unpackage the file:

   ```
   gunzip pw-teleman_122016.tar.gz
   tar -xvf pw-teleman_122016.tar
   ```

   A directory 'pw-teleman' will be created. Enter this directory:

   ```
   cd pw-teleman
   ```

4. Before compilation, one should update some settings (for detailed explanations of the parameters see section **??**):
   (a) Edit `pw-teleman/code/source_f90/define.h` to choose the wanted code options.
   (b) Edit `pw-teleman/code/source_f90/params.F90` if you need to change some limiting values (rarely required).
5. To use environment modules for compiling or running a code, load a particular module. To see all available modules, run `module avail`, when you get command not found you must do for bash:

   ```
   . /etc/profile.d/modules.sh
   ```

   and for csh:

   ```
   . /etc/profile.d/modules.csh.
   ```

   Once you have done this, you can run:

   ```
   module load <module name>
   ```

   For example:

   ```
   module load intel/composer_xe_2013.4.183
   module load cuda/5.0
   module load openmpi/1.6.4-intel
   ```

6. Run `configure` with appropriate options (see alist of option in section **??**). Configure will create an appropriate Makefile for compilation. If no option is given, default compilation options will be used :

   ```
   ./configure
   ```

   or, with some options :

   ```
   ./configure --with-compiler=your_compiler --with-fft=NETLIB etc ...
   ```

7. Finally execute `make` command:

   ```
   make
   ```

The executable will be created in the `bin` directory.

Default name for the new executable is '`<prefix>_pwtelaman.<extension>`'. `prefix` prefix depends on the FFT library used, and will be '`gpu`' for Cuda cuFFT, '`fftw`' for FFTW3, '`mkl`' for FFTW3 with MKL, or '`netlib`' for NETLIB FFT pack (default). Depending on the choice made for parallelization, `<extension>` will be '`par`' (parallel), '`seq`' (sequential) or '`sim`' (simpara).

Example: an executable build using MKL and full parallelization will be named :

```
mkl_pwteleman.par
```

1.2 Options for configure script.

Options can be passed to the `configure` script in order to choose for:

- compiler;
- compilation options (static, debug);
- parallelization (MPI, OpenMp);
- FFT library (NETLIB, FFTW3, MKL, cuFFT).

If some of your libraries are not located in the system default search path, you need to add the adequate flags by setting environnement variable `LDFLAGS` when running `configure`:

```
./configure ... options ... LDFLAGS="-L/path/to/lib -L/path/to/other/lib"
```

Also, use variable CPPFLAGS in order to add nonstandard directory containing headers:

```
./configure ... options ... CPPFLAGS="-I/path/to/headers"
```

Options for `configure` script	
Usage: `./configure [--with-`*options*`]`	
`--with-compiler=<choice>`	Choice of compiler. `<choice>` can be one of : `gfortran`, `ifort`, `mpifort` (or older `mpif90`), `xlf_r` (no warranty). Default is `ifort` if installed, `gfortran` otherwise.
`--with-openmp[=dyn]`	Invoke OpenMP. `--with-openmp` : use threads for FFT. `--with-openmp=dyn` : wave function parallelization. Cannot be used together with option `--with-para`
`--with-para[=sim]`	Choice of MPI parallelization. If not specified : sequential (no parallelization) `--with-para` : use full MPI parallelization. `--with-para[=sim]`: use pseudo-parallel code (simpara, runs different inputs simultaneously). Not available with all compilers, it is advised to prefer the use of `mpifort` (or older `mpif90`). Cannot be used together with option `--with-openmp`
`--with-fft=<choice>`	Choice of the FFT library. `<choice>` can be `NETLIB`, `FFTW` (for FFTW3), `MKL` (for intel MKL) or `CUFFT` (for CUDA cuFFT). Default is `NETLIB` (furnished sources). MKL requires `fftw3xf` wrappers for FFTW3 functions. If wrappers are not intalled in standard location, user can indicate the location using option `--with-wrappers`
`--with-wrappers=<path>`	Use this option to give the path to fftw3xf wrappers for mkl, if the wrappers are not located in `$mklroot/interfaces/`. Wrappers allow MKL routines to be called intead of FFTW3 routines
`--with-mklthreads`	together with option `--with-fft=MKL`, uses MKL threads. No effect with other FFT.
`--with-static`	Use static compilation. The final executable will be able to work independantly from the external libraries. This makes a big executable.
`--with-debug`	Use compiler-specfic debug option during compilation.

Additionnal flags for `configure` script	
Usage: `./configure [--with-`*options*`] [`*FLAGS*`]`	
`CPPFLAGS="<flags>"`	Used to add flags for the preprocessor at compile time. Use this when you have headers in a nonstandard directory, with `<flags>` being one or more flags of the form `-I/path/to/include`
`LDFLAGS="<flags>"`	Used to add flags for the linker at compile time. Use this if you have libraries in a nonstandard directory, with `<flags>` being one or more flags of the form `-L/path/to/library`

1.3 Basic input structure

The cluster 3D code has five entries for options:

compile time	
`define.h`	variants of the code
run time	
`for005.<name>`	general input for settings, static and dynamics
`for005ion.<name>`	ionic configuration of cluster
`for005surf.<name>`	atomic configuration of substrate (optional)
`for005`	defines the qualifier `<name>` for the other `for005...` files

The first entry have to be set before compilation. The other four are read in for an actual run and can be varied from run to run. The input structure for these files is summarized in section **??**.

1.4 Some practical advices

Important compile-time settings:

You have to chose the wanted options in 'define.h'.

Save and restart:

The parameters 'isave', 'istat', and 'irest' (in `for005.<name>`) allow to switch saving wavefunctions and restarting from them.

For `ismax>0` and `isave>1`, the static wavefunctions are saved on `rsave` after the static iterations. These can be used in two ways. Setting `istat=1` and `ismax>0` continues static iteration from `rsave`. Setting `ismax=0`, `istat=1`, `irest=0`, and, of course, `itmax>0` starts a dynamical run at time zero with the static wavefunctions

from `rsave`.

Dynamical configurations are saved on `save.<name>` after every `isave` time steps. Setting `irest=1` will continue the dymaical calculation from the stage saved in `save.<name>`.

Diagonalization amongst occupied states:

The run time option `ifhamdiag=1` activates the diagonalization of the mean-field Hamiltonian amongst the active wavefunctions in each static iteration step. This option can accelerate the convergence of the static solution significantly. *However:* At present, this method works safely only if the number of active states `nstate` equals the actual number of electrons. This has to be checked by the user. It may work in other cases, but may also induce oscillating iteration which nevers converges.

2 Input files

Compile time settings in `define.h`	
version control:	
`IVERSION`	define your own version number
grid representation of kinetic energy:	
`gridfft`	FFT
`findiff`	finite diffences 3. order (yet unsafe)
`numerov`	finite diffences 5. order (yet unsafe)
Variants of the Coulomb solver (for `gridfft`=1):	
`coufou`	FALR (standard)
`coudoub`	exact boundary conditions
parallele version:	
`parayes`	use parallelization for wavefunctions
`parano`	produce serial code
`simpara`	pseudo-parallel code, runs different inputs simultaneously
versions of SIC for electrons:	
`fullsic`	old full SIC
`symmcond`	old full SIC with double set technique
`twostsic`	new full SIC from PhD Messud (obsolete)
Compile time settings in `define.h` – part 2	
options for substrate:	
`raregas`	enables substrates

Namelist GLOBAL	in for005.<name>
choice of system	
kxbox	nr. of grid points in x direction
kybox	nr. of grid points in y direction
kzbox	nr. of grid points in z direction
	box sizes must fulfill kxbox≥kybox≥ kzbox
numspin	number of spin components (2=full spin treatment)
	(1=spin averaged, possible problem for ADSIC)
kstate	maximum nr. of s.p. states which is possible (greater than nclust)
nclust	number of QM electrons - if set to 0 or a negative value (charge) this will be automatically calculated
	nclust $= \sum_{i=1}^{nion} Z_{ion} = charge$, where Z_{ion} is charge of each ion
nion	number of cluster ions
nspdw	number of spin down electrons
nion2	selects type of ionic background
	$0 \rightarrow$ jellium background
	$1 \rightarrow$ background from ionic pseudo-potentials
	$2 \rightarrow$ background read in from potion.dat
radjel	Wigner-Seitz radius of jellium background
surjel	surface thickness of jellium background
bbeta	quadrupole deformation of jellium background
gamma	triaxiality of jellium background
dx,dy,dz,	grid spacing (in Bohr) for the 3D numerical grid - if negative this will be set to an optimal value and a value will be suggested for KXBOX in file NX - the code stops and has to be restarted
	the grid size is defined before compilation in params.F90
	it has to be correlant with pseudopotentails
	corresponds to ecut in solid state
imob	global switch to allow ionic motion (if set to 1)
isurf	switch for Ar or MgO surface (isurf=1 activates surface
nc	number of O cores in MgO(001)
nk	number of Mg cations in MgO(001)
rotclustx,y,z	vector fo angle of initial rotation of ions
initialization of wave functions	
b2occ	deformation for initial harmonic oscillator wf's
gamocc	triaxiality for initial harmonic oscillator wf's
deocc	shift of inital Fermi energy (determines nr. of states)
shiftWFx	shift of initial wavefunctions in x direction
ishiftCMtoOrigin	switch to shift center of mass of cluster to origin
ispinsep	initialize wavefunctiosn with some spin asymmetry
init_lcao	choice of basis for wavefunction initialization
	$=0 \Longrightarrow$ harmonic oscillator functions (center can be moved by shiftWFx)
	$=1 \Longrightarrow$ atomic orbitals = WFs centered at ionic sites

Namelist `GLOBAL`	in `for005.<name>`
convergence issues	
`e0dmp`	damping parameter for static solution of Kohn-Sham equations (typically about the energy of the lowest bound state)
`epswf`	step size for static solution of Kohn-Shahm equations (of order of 0.5)
`epsoro`	required variance to terminate static iteration (order of 10^{-5})

Namelist `DYNAMIC`	in `for005.<name>`
numerical and physical parameters for statics and dynamics	
`dt1`	time step for propagating electronic wavefunctions, $\frac{\Delta t}{\Delta x^2} \leq 1$
`ismax`	maximum number of static iterations
`idyniter`	switch to s.p. energy as E0DMP for 'iter>idyniter'
`ifhamdiag`	diagonalization of m.f. Hamiltonian in static step (presently limited to fully occupied configurations)
`isitmax`	nr. of imaginary-time steps to improve static solution
`itmax`	number of time steps for electronic propagation
`ifexpevol`	exponential evolution 4. order instead of TV splitting
`iffastpropag`	accelerated time step in TV splitting (for pure electron dynamics, interplay with absorbing b.c. ??)
`irest`	switch to restart dynamics from file 'save'
`istat`	switch to read wavefunctions from file 'rsave' it continues static iteration for 'ismax>0' it starts dynamics from these wf's for 'ismax=0'
`idenfunc`	choice of density functional for LDA 1 → Perdew & Wang 1992 (default setting) 2 → Gunnarson & Lundquist 3 → only exchange in LDA
`isave`	saves results after every 'isave' steps on file 'rsave' in and after static iteration on file 'save' in dynamic propagation
`ipseudo`	switch for using pseudo-densities to represent substrate atoms
`ipsptype`	type of pseudopotentials: 0 = soft local (errf); 1 = full Goedecker; 2 = local Goedecker; 3 = read from file goed.asci (no need to specify) ; 4 = semicore read from file goed.asci
`directenergy`	`.true.` = direct computation of energy (only for `LDA`, `Slater`, `KLI`)
`ifsicp`	selects type of self-interaction correction 0 = pure LDA, 1 = SIC-GAM, 2 = ADSIC; 3 = SIC-Slater; 4 = SIC-KLI; 5 = exact exchange; 6 = inactive; 7 = localized SIC; 8 = full SIC (double set). IFSICP=7 or 8 requires switch `twostsic=1` in `define.h`. Option IFSICP=7 needs yet testing.
`icooltyp`	type of cooling (0=none, 1=pseudo-dynamics, 2=steepest descent, 3=Monte Carlo)
`ifredmas`	switch to use reduced mass for ions in dynamics
`ionmdtyp`	ionic propagation (0=none, 1=leap-frog, 2=velocity Verlet)
`ntref`	nr. time step after which absorbing bounds are deactivated
`nabsorb`	number of absorbing points on boundary (0 switches off)
`powabso`	power of absorbing boundary conditions
`ispherabso`	switch to spherical mask in absorbing bounds

Namelist DYNAMIC	in for005.<name>
way of excitation	
centfx	initial boost of electronic wavefuncftions in x-direction
centfy	initial boost of electronic wavefuncftions in y-direction
centfz	initial boost of electronic wavefuncftions in z-direction
tempion	initial temperature of cluster ions
ekmat	initial kinetic energy of substrate atom (boost in x, in eV)
itft	choice of shape of laser pulse
	1 = ramp laser pulse, sine switching on/off
	2 = gaussian laser pulse
	3 = $\cos^2$ pulse
tnode	time (in fs) at which pulse computation starts
deltat	length of ramp pulse (itft = 1), in fs
tpeak	time (in fs, relative to tnode) at which peak is reached
	(for itft = 1 and 2, pulse length becomes 2*tpeak)
omega	laser frequency (in Ry)
e0	laser field strength in Ry/Bohr
e1x,e1y,e1z	orientation of pulse
e0_2	field strength of second laser pulse (only itft=3)
phase2	phase of second pulse
omega2	frequency of second pulse
tstart2	initial ime of second pulse
tpeak2	peak time of 2. pulse (pulse length is 2*tpeak2)
iexcit	modus of excitation (0=shifts, 1=rotation)
iangmo	switch to compute angular momentum
irotat	axis of rotation for excitation (x=1,y=2,z=2,xyz=4)
phirot	angle of rotation for excitation (in units of degree)
phangle	angle of "rotation" into a 1ph state
phphase	phase of "rotation" into a 1ph state
nhstate,npstate	nr. of hole and particle state for 1ph excitation
	this 1ph option can only be run from istat=1
eproj	energy of incoming projectile (= last ion in the list)
vpx,vpy,vpz	direction of the incoming projectile
taccel	time span over which the projectile is accelerated to eproj
	for taccel=0 one has to use init_lcao=1

Namelist DYNAMIC	in for005.<name>
flags for observables	
iemomsRel	calculates multipole momentes of electron density relative to origin (0) or c.m. of cluster (1)
istinf	modulus for printing information in static iteration
ifspemoms	switch to compute and print spatial s.p. moments
iftransme	switch to compute and print transition m.elements
ifrhoint_time	switch to slices of integrated densities for all times
jstinf	modulus for printing information in dynamic
jinfo	modulus for printing dynamical information on infosp.<name>
jdip	modulus for printing dipole moments on pdip.<name>
jquad	modulus for printing quadrupole moments on pquad.<name>
jesc	modulus for printing ionization pescel.<name>
jenergy	modulus for printing energy information on penergies.<name>
iflocaliz	activates computation of Becke's localization
jelf	modulus for anaylzing and printing electron localization in dynamics various files are written of the form pelf*.<name>
iflocaliz	modulus for anaylzing and printing electron localization in statics
jstinf	modulus for printing s.p. energies and variances
jpos	modulus for printing ionic positions on pposion.<name>
jvel	modulus for printing ionic velocities on pvelion.<name>
jstateoverlap	switch to compute overlap of static state with the state directly after dynamical initialization

Namelist `SURFACE`	in `for005.<name>`
`ivdw`	handling of Van-der-Waals with substrate atoms 0 $\Longrightarrow$ no VdW 1 $\Longrightarrow$ enables full computation of VdW 2 $\Longrightarrow$ enables effective VdW through PsP parameters
`ifadiadip`	switch to adiabatic treatment of substrate dipoles
`shiftx`	global shift in x for all substrate atoms
`shifty,shiftz`	as `shiftx` for y and z direction
`mion`	mass of surface anion (16 for O in MgO(001))
`mkat`	mass of surface kation (24.3 for Mg in MgO(001))
`me`	mass of valence shell
`cspr`	spring constant for interaction between core and valence shell
`chgc0`	charge of (anion) core
`chge0`	charge of valence shell
`chgk0`	charge of cation
`sigmak`	gauss width of cation
`sigmac`	gauss width of core
`sigmav`	gauss width of valence shell
`iUseCell`	switch for reading/building lattice of substrate atoms 0 $\Longrightarrow$ lattice atoms are read in from input file 'for005surf.*' 1 $\Longrightarrow$ lattice is built from replicating unit cell and lattice parameters `rlattvec` ... are read in (see `md.F`)
`iPotFixed`	switch for Madelung summation of substrate atoms read/write electrostatic potential from particles with imob=0, so that their run-time calculation can be skipped 0 $\Longrightarrow$ do not read; calculate full potential at each iteration 1 $\Longrightarrow$ read in potFixedIon() from previously prepared file -1 $\Longrightarrow$ calculate potFixedIon() write result to a file which can be later read in by option 1, stop after that 2 $\Longrightarrow$ calculate potFixedIon() at the beginning, do not write
`ifmdshort`	includes short range interaction electron–substrate
`isrtyp(i,j)`	type of interaction between the different kinds of particles 0 $\rightarrow$ no short range interaction 1 $\rightarrow$ GSM core 2 $\rightarrow$ GSM valence shell =1 $\Longrightarrow$ Born-Mayer type 3 $\rightarrow$ GSM kation =2 $\Longrightarrow$ Argon case 4 $\rightarrow$ Na core 5 $\rightarrow$ DFT electron
`unfixCLateralRadx`	radius of cylinder with mobile cores
`unfixELateralRadx`	radius of cylinder with mobile valence electrons
`fixCBelowx`	fixes cores which lay below given x value
`iDielec`	switch to dielectic support
`xDielec`	x below which dielectric zone is activated
`epsDi`	dielectric constant in the dielectric zone

Namelist PERIO	in for005.<name>
ch	effective charge of ion
amu	mass of ion in units of hydrogen mass
dr1,dr2	radii of soft local PsP
prho1,prho2	strenghts of soft local PsP
crloc	radius for local part of Goedecker PsP
cc1,cc2	strengths for local part of Goedecker PsP
r0g,r1g,r2g	radii for non-local parts of Goedecker PsP
h0_11g,h0_22g,h0_33g	strenghts for non-local parts of Goedecker PsP
h1_11g,h1_22g,h2_11g	strenghts for non-local parts of Goedecker PsP
radiong	carrier radius for projecteor in non-local Goedecker PsP

Namelist FSIC	in for005.<name>
step	step size in iteration of localizing or symmetry condition
precis	precision in iteration of localizing or symmetry condition
SymUtBegin	nr. iteration where symmetry condition starts
	for pure localizing step set SymUtBegin¿ ismax
radmaxsym	limiting value in radius division for actucal step

Ionic structure and e^--initialization in for005ion.<name>	
	This initialization does not use NAMELIST but reads input in fixed order.
	Each line stands for one ion. Each column has a definite meaning.
Col. 1	x-coordinate
Col. 2	y-coordinate
Col. 3	z-coordinate
Col. 4	number of element in periodic system (e.g.: Na$\leftrightarrow$11)
Col. 5	only init_lcao=1: ordering of nodes in repeated initialization at this ion
Col. 6	only init_lcao=1: radius of initial Gaussian at this ion
Col. 7	only init_lcao=1: starting spin for initalization at this ion

The handling of the initialization of electronic wavefunctions is rather involved. A more detailed explanation is given in appendix **??**.

A On the initialization of the electronic wavefunctions

The basic switch is `init_lcao`. The case `init_lcao=0` is the simpler option. This initializes harmonic oscillator wavefunctions about one common center. This center is usually the origin of the coordinte-space grid. It can be moved deliberately by `shiftWFx`, `shiftWFy`, and `shiftWFz`. The initial oscillator may be deformed. Its deformation is given by the dimensionless quadrupole `b2occ` and triaxiality `gamocc` (in degree). The oscillator states are filled in order of increasing oscillator energies. A spin asymmtry can be enforced with `ispinsep=1`. This option is useful when dealing with odd electron number. The upper end of initialization is determined by `deocc`. A `deocc` $\approx$ `0` typically initializes just as many states as are occupied. If more is required, enhance `deocc`.

The case `init_lcao=1` initializes wavefunctions which are localized at the ions. This option is richer and a bit hard to handle. In a first step, the total number of wavefunctions is estimated and it is computed how many wavefunctions have to be initialized then for each ion. At one given ion, initialization starts with the $1s$ oscillator state. The first choice of spin is taken from column 7 of `for005ion.<name>` which initializes the entry of the actual ion in the the `ipol` array. The value `ipol=+1` sets a spin up (i.e. `ispin=1` in the code) as the first choice, while `ipol=-1` sets a spin down (i.e. `ispin=2`). If more than one state is to be occupied, the next is then the $1s$ state with opposite spin. Next comes the $1p_i$ state with first spin where i is the direction given as first entry in column 5. For example if column 5 selects `'yzx'`, the $1p_y$ comes here. Occupation continues in order given by column 5 and 7 until the wanted number of orbitals at this ion site is reached. Column 6 sets the oscillator radius for the initialization at this ion (which allows to deal efficiently with systems consisting of very different ions). Column 7 becomes important for ions associated with an odd number of electrons as, e.g., hydrogen. One ought to distribute an equal collection of spins up and down over the whole system to avoid unnaturally polarized molecules.

B Open ends and to-be-dones

Status of Fortran90 code development:

- All `common` blocks have been replaced by modules and corresponding `USE` command. The then appearing dependences are mapped in the `makefile`.
- All code is now genuinly double precision and can be complied without the `autodouble` option. Only exception if the FFT package `fftpack.F90` in connection with `NETLIB` which still requires the `autodouble`, as handled explicitely in the `Makefile`.
  Note that the precision is set at the header of `params.F90` and used as a `KIND` parameter in typical Fortran90 fashion. The name is set to `DP`.
- The somewhat dangerous practice of reusing workspace has been abandoned. Workspace is now associated dynamically with the `ALLOCATE/DEALLOCATE` mechanisms.
- The compiled code works now for all box sizes and number of s.p. states as long as memory allows. The box size and maximum number of states is now entered in `for005.<name>` in namelist `GLOBAL`.

Next in Fortran90 code development:

- Remove numbered labels and `GOTO` in favour of `CYCLE` or `EXIT` switches.
- Exploit compact vector operations to simplify long (and nested) `DO` loops.
- The access `USE kinetic` has been given too generously. Confine that to routines which really need it.
- The module `params.F90` collects practically all global variables. It should be disentangled to more specific modules with restricted access.
- There are still problems with running substrates. For example the leap-frog switch does not propagate the substrate electrons. This case has to be tested.
  In future, it may be that the whole subtrate part is trated in a separate program connected to the electronic part by a master routine written in ttt python.
- Full SIC has yet to be implemented.
- The code should be successively moved to `IMPLICIT NONE`.

Open problems of general nature:

- The implementation of GSlat and full SIC needs to be checked and updated if necessary.
- Check PES and PAD for the option `parayes`.
- Option `iaddCluster` is presently questionable. It may be extended to allow for initialization of cluster collisions.
- The computation of pseudo-potentials from the substrates valence electrons should be separated from the slower atomic (ionic) parts. This concerns routine `calcpseudo`.
- The setting for the valence-electron mass in 'vstep' may be wrong for the case of MgO.
- Check proper setting of 'time' in outputs.
- Exponential propagation should yet be certified to cooperate with ionic motion.
- Subgrids for Gaussian pseudo-densities have fixed grid size of ± 7 points. This should be made more flexible to accommodate mesh size in relation for PsP radius.
- Although not necessary for performance, one may replace DO loops by the Fortran 95 SUM construct. This will make the code more transparent. This also holds for other compact Fortran 95 constructs.
- Present parallele version still needs to specify the number of nodes at compile time. This should be changed to allow dynamical adjustment of number of nodes.
- Spin-averaged code (`numspin`=1) does not reproduce the results from full spin calculation in case of ADSIC. Check.