Summary For SIESTA

Wanzhen He, 2019

Tsinghua University, Beijing

Instruction 1 modify to output .MDX instead of .MD

a. modify *iomd.f* file

change logical value from F to T.

"logical, save :: formtt = .true."

(output .ANI file (time-dependent trajectory) which can be read by OVITO.)

Instruction 2 *Util/siesta2cube*: usage (extended in orthorhombic lattice)

a. mkdir *name_file*

b. modify *makebox.f* file

comment "read (5,*) unitlab", then write "unitlab = 'B'"

comment "read (5,*) (obox(ii),ii=1,3)"

comment "read (5,*) (rbox(ii,1),ii=1,3)" * 3 lines

write "obox(i) = **" * 3 lines (1,2,3)

write "rbox(i,j) = **" * 9 lines (3*3) (data read from .XV file)

c. modify *rho2cube.f* file

comment "read (5,*) syslab", then write "syslab='**'"

comment "read (5,*) n1,n2,n3"

write "ni = **" * 3 lines (1,2,3) (grids)

comment "read (5,*) suffix", then write "suffix = 'RHO'"

comment "goto 103"

d. in *name_file*, mkdir *only_name_file* or *test* and copy all files (to plot density charge difference)

e. in *test*, modify *rho2cube.f* file

in *intp104* function, after two ifs, add new if

```
"if (i3.gt.**) then
   fintp = 0
end if"
```

Instruction 3 *Util/bader*: already compiled

Instruction 4 handle the TDDFT resuts: *bader, **cube, initcube, bader.sh, drho.sh, rho.sh, diffcube.py, decompxv.py* are needed

a. modify *decompxv.py* file

```
outf = open('out-cbn_h.XV','w')    ! out-**.XV
inf1 = open('cbn_h.XV')    ! **.XV
inf2 = open('cbn_h.MDX')    ! **.MDX

nxv = 2    ! modified in rho.sh
nline = 66    ! natom + 1
```

b. modify *diffcube.py* file

```
temp = open('temp.cube','w')
inf = open('100-cbn_h.cube')    ! modified in drho.sh

outf = open('diff.cube','w')
inf1 = open('pri.cube')
inf2 = open('temp.cube')
inf = open('100-cbn_h.cube')    ! modified in drho.sh
```

c. modify *rho.sh* file and *drho.sh* file

replace all label

check paths

hint: **%s#abc#def#g** / :10,50s#abc#def#g

d. prepare ***cube, initcube, drho.sh, rho.sh, diffcube.py, decompxv.py* to *work*

e. *rho.sh* in *work* and *drho.sh* in *work/rdrho*

f. prepare *bader, bader.sh* to *work/rdrho/sum*, then bash

(! should be modified by command ./bader rho.cube -ref bader.cube)

(Tips: *sumspin.py* to sum spin up/down and get new .rho to compare with .Bader)

Instruction 5 modify to IO formatted .DM and sum two .DM files

a. modify *m_iodm.F90* file

for every operations on files.

(1) change unformatted to formatted. e.g.

"open(iu, file=file, form='formatted', status='old')"

(2) add * in every read and write for formatted IO, e.g.

"write(iu,*) no_u, nspin"

b. modify *m_io_s.F90* file

similar to a. (2), but only subroutines *io_read_Sp/ io_write_Sp/ io_write_d2D/io_read_d2D* are needed to be modified. (I donnot change *io_write_d1D* and *io_read_d1D*)

c. use written sumdm.py to sum two .DM files and can be read by SIESTA

Instruction 6 modify to output TD .Bader files for TD charge analysis

a. modify *siesta_options.F90* file

add new line (could also add another ntdsavebader but I use ntdsaverho for simplicity)

"logical :: tdsavebader"

b. modify *read_options.F90* file

add new line (could also add definition of ntdsavebader)

"tdsavebader = fdf_get('TDED.Savebader', .false.)"

c. modify *files.f* file

add new line

"& tdbader = ' '," to define name

d. modify *m_iotdddft.F90* file

(1) “USE siesta_options, ONLY: **” add “tdsavebader,”

(2) add line

“PUBLIC :: write_tdbader”

(3) add subroutine write_tdbader

“SUBROUTINE write_tdbader (filesOut)

```
        TYPE(filesOut_t), INTENT(INOUT)          :: filesOut

        IF (tdsavebader) THEN
          IF (mod(istp,ntdsaverho) .eq. 0) THEN
            write(filesOut%tdbader,"(i0,a)") istp, '.TDBader'
          ELSE
            filesOut%tdbader = ' '
          END IF
        END IF

        END SUBROUTINE write_tdbader”
```

e. modify *dhscf.F* file

(1) “use m_iotddft, only: write_tdrho, write_tdbader”

(2) for primary savebader, add arguments to distinguish tdbader.

“if (filesOut%toch .ne. ' ' .and. savebader) then

```
    call save_bader_charge(trim(slabel)// ".BADER")
  endif”
```

(3) add blocks for output tdbder

“call write_tdbader(filesOut)

```
  if (filesOut%tdbader .ne. ' ') then
    call save_bader_charge(filesOut%tdbader)
  endif”
```

(4) modify subroutine save_bader_charge

“subroutine save_bader_charge(baderfile)” add arguments

"CHARACTER(LEN=70) :: baderfile" add this new line

comment "call write_rho(trim(slabel)// ".BADER", cell,", add "call write_rho(baderfile, cell,"

comment "call write_rho(trim(slabel)// ".BADER", cell,", add "call write_rho(baderfile, cell," (twice)

Instruction 7 modify to output TD .DM for TD DOS analysis

a.—d. similar with I**nstruction** 6 a.—d.

e. modify *dhscf.F* file

(1) add use

"use m_iotddft, only: write_tdrho, write_tdbader, write_tddm"

"use siesta_geom, only: nsc

use sparse_matrices, only: DM_2D

use m_iodm, only: write_dm"

(2) add blocks for output

"call write_tddm(filesOut)

if (filesOut%tddm .ne. ' ') then

call write_dm(filesOut%tddm, nsc, DM_2D)

endif"

Instruction 8 use LUA, compile LUA/flook

a. flook: use the *siesta/Docs/install_flook.bash*, just copy one version of flook to this dir is ok. Pay attention that the print on the screen is the only way to modify arch.make (commands in gitlab.com do not include fdict things.)

b. a. is enough to compile siesta. But to use *.lua file, lua is needed. I directly copied the dir in the new cluster (since cnmm could not connect the internet), and I'm not sure if this also works if git commands done in mac?

c. some previous hints but nothing to do with essential steps a. and b.: (1) modify */home/cnmm/bin/packages/flook/aotus/external/lua-5.3.5/src/Makefile*: $(MAKE) $(ALL) SYSCFLAGS="-DLUA_USE_LINUX" SYSLIBS="-Wl,-E -ldl -lreadline -lncurses" (after linux:, add -lncurses). (2) another error: Catastrophic error: could not set locale "", try: export

LANG=en_US.utf8; export LC_ALL=en_US.utf8

d. Some mistakes in the *siesta/Tests/lua*(or flos*)*, some works and others report "rdiag: Error in Cholesky factorisation". Solution: add kpoints setting in the input, especially try 221 when 111 fails. (Ref: siesta_launchpat web)

e. The MD.VariableCell command fails in the TDDFT version and reports "forrtl: severe (151): allocatable array is already allocated". modify the */Src/state_init.F* that comment

ALLOCATE(eo(no_u,spin%spinor,kpoint_scf%N))

ALLOCATE(qo(no_u,spin%spinor,kpoint_scf%N))

also, I added "auxchanged = .false.", but I'm lazy to check whether this matters.

Instruction 9 change cell and scale atoms in TDED

a. change *siesta_move.F*, the label is idyn = 1 (verlet) for TDED, but could add td_elec_dyn section to change cells.

(1) "use m_dynamics, only: nose, verlet2, npr, anneal, pr" add "ctded"

(2) in case(1), distinguish TDED and verlet

b. modify *dynamics.f* file

(1) DO NOT USE QUENCH OR RESTART!

(2) "public :: npr, nose, verlet2, pr, anneal" add "ctded"

(3) add subroutine ctded

c. modify *read_options.F90* file

(1) add lines

```
"xsr = fdf_get('MD.XStrainRate',0.0_dp)
 ysr = fdf_get('MD.YStrainRate',0.0_dp)
if (leqi(dyntyp,'TDED')) then
        if (ionode) then
            write(6,6) 'redata: strain rate of cell in x direcation',xsr
            write(6,6) 'redata: strain rate of cell in y direcation',ysr
        endif
   endif"
```

(2) change varcel

".or. (idyn==1 .and. td_elec_dyn == .true.) &

.and. (idyn/=2) &"

comment ".and. (idyn/=1) .and. (idyn/=2) &"

(3) add lines

"vcel_tded = fdf_get('TDED.Varcell', .false.)"

d. modify siesta_*options.F90* file

(1) add lines

"real(dp) :: xsr ! strain rate along x

 real(dp) :: ysr ! strain rate along y"

(2) add lines

"logical :: vcel_tded ! whether to variable cell using xsr/ysr in TDDFT"

Instruction 10 couple NVT (nose thermostat) with TDED

a. change codes based on codes after **instruction 9**.

b. modify siesta_*options.F90* file

add lines

"logical :: nose_tded ! whether NVT (nose) with TDDFT"

c. modify *read_options.F90* file

add lines

"nose_tded = fdf_get('TDED.nose', .false.)"

d. modify *siesta_move.F* file

~~(1) "use m_dynamics, only: nose, verlet2, npr, anneal, pr, ctded" add "ntded"~~

(2) in case(1), distinguish TDED (ctded) and verlet and TDED(nose)

```
"       if ( td_elec_dyn ) then
            if ( nose_tded ) then
                call nose( istp, iunit, na_u, cfa, tt, dt, amass, mn,
     .              ntcon, va, xa, Ekinion, kn, vn, tempion )
            elseif (vcel_tded) then
                call ctded(istp, iunit, iquench, na_u, cfa, dt,
     .              amass, ntcon, va, xa, Ekinion, tempion,
     .              xsr, ysr, ucell)
            else
                call verlet2(istp, iunit, iquench, na_u, cfa, dt,
```

```
     .                    amass, ntcon, va, xa, Ekinion, tempion)
                      endif
                  else
                      call verlet2(istp, iunit, iquench, na_u, cfa, dt,
     .                 amass, ntcon, va, xa, Ekinion, tempion)
                  endif"
```

~~e. modify *dynamics.f* file~~

~~(1) "public :: npr, nose, verlet2, pr, anneal, ctded" add "ntded"~~

~~(2) add subroutine ntded~~

Instruction 11 phonon calculation in SIESTA

Ref: youtube video by Pritam Kumar Panda

a. prepared files

(1) *psf/phonon.fdf/siesta.fdf

(2) fcbuild and vibra (**/Util/Vibra*)

(3) gnubands (**/Util/Bands*)

b. create supercell

(Ref: do not define Kpoints due to Gamma phonon calculations only)

(1) ./fcbuild <phonon.fdf, output FC.fdf

(could copy different Bandlines)

(2) siesta <siesta.fdf | tee phonon.out, output label.FC (force matrix)

(MeshCutoff and DM.Tolerance should be accurate?)

(3) ./vibra <phonon.fdf, output label.vectors (calculate eigenvectors) and label.bands

(4) ./gnubands -F label.bands > bands.dat

(5) plot bands.dat

e.g. xmgrace bands.dat (first open XQuartz, then DISPLAY=:0.0 xmgrace)

Instruction 12 add constant velocity for stopping power at high *K*

Ref: Manifold curvature and Ehrenfest forces with a moving basis (arxiv.org:2107.05092)

a. modify *dynamics.f* file

modify verlet2 section

(1) Quench section is not touched at all, be careful

(2) add nfixv parameter

(3) only modify the va/vold part, the force is not zero to calculate the new coord.

b. modify siesta_*options.F90* file

add lines

“integer :: naofconv ! Number of constant-velocity atoms using verlet2: from last”

c. modify *read_options.F90* file

add lines

“naofconv = fdf_get('Verlet2.nconv', 0)”

d. modify siesta_*move.F90* file

for verlet2 function, add naofconv parameters