ELSES Code Note
(based on v.0.03.09RC [Build 20120119])

ELSES Consortium

January 23, 2012

Contents

Chapter 1

Introduction

This document is based on the code of 'ELSES v.0.03.09-RC' [Build 20120119]. The explanation will be given for the work flow in non-distributed-memory systems, except where indicated.

1.1 Update history

- (23. Jan. 2012) Initial manuscript. Based on 'ELSES v.0.03.09-RC'

1.2 Used notations

The following notations are used throughout this document.

- The word 'MD' means simulations with iterative update of atom positions, that is, (a) finite-temperature dynamics ('dynamics mode') and (b) structure opimization ('optimization mode') .
- The word 'DST' means distributed memory systems and the word 'non-DST' means shared memory systems. The DST work flow is designed for the MPI parallelism and works also without the MPI parallelism.
- The word 'node' means the parallelized unit in the MPI calculation, while the word 'thread' means the parallelized unit in the OMP calculation.
- Variables and routines are explained in the form of 'name [module name]'. For example, the expression of
  `atm_force(1:3,1:noav)` [`M_qm_domain`]
  means that the variable '`atm_force(1:3,1:noav)`' is included in the module '`M_qm_domain`'.

1.3 Notes for the DST work flow

The present code set is written so as to work fine both in the DST and non-DST systems, although the work flow for the DST system has been not yet settled particularly in its optimality.

Here several techniques for the DST work flow are explained.

1.3.1 Wrapper routines for MPI commands

A code with explicit MPI commands can not be compiled in a non-MPI compiler, since the MPI commands, such as `mpi_init`, can not be recognized.

In ELSES, MPI commands are called through wrapper routines, so that one can compile both in MPI and non-MPI environments. All the wrapper routines are prepared as (i) the true routine with MPI commands, stored in

```
src/code_for_mpi/elses-lib-mpi-wrapper.f90
```

and (ii) the dummy routine without MPI commands, stored in

```
src/code_for_mpi/elses-lib-mpi-wrapper-dummy.f90
```

The dummy routines work as in a one-node calculation in a MPI environment.

If one would like to compile in a MPI environment, one should type

```
prompt> cd src
prompt> cp code_for_mpi/elses-lib-mpi-wrapper.f90 elses-lib-mpi-wrapper-compile.f90
```

before compiling. If one would like to restore the code set for the non-MPI environment, one should type

```
prompt> cd src
prompt> cp code_for_mpi/elses-lib-mpi-wrapper-dummy.f90 elses-lib-mpi-wrapper-compile.f90
```

before compiling.

1.3.2 Suppressing redundant file output among nodes

The redundant file output should be suppressed in the MPI parallelism. The node index is stored as `myrank` [`M_lib_dst_info`] (`myrank`=0,1,2..) and the total node number is stored as `nprocs` [`M_lib_dst_info`] (See Sec. 4.2 for details). In ELSES, the nodes are classified into the three categories

- 'root' node : only one node (`myrank`=0)
- 'specific' nodes : nodes with file output. Tentatively, eight nodes with `myrank`=0,1,2 ...7, are chosen as the 'specific' nodes.
- 'other ' nodes : nodes without file output

The verbose level in the 'other' nodes is set to be (level)=-1, so as to suppress the file output, while the verbose level in the 'specific' node is set by the user input. Therefore the code

```
use M_qm_domain,  only : i_verbose
if (i_verbose >=0) write(*,*) 'Hello'
```

means the printing of 'Hello' only at the 'specific' nodes into the standard output file.

Since the output for the standard output file is not preferable, log files are defined for each node for the 'specific' nodes;

```
    log-node00000.txt
    log-node00001.txt
    log-node00002.txt
```

and so on. The log file 'log-node00001.txt', for example, is one for the node with `myrank`=1. A file output into the log files is controlled by the variable `log_unit` [`M_lib_dst_info`] (See Sec. 4.2.4). For example, the code

```
use M_lib_dst_info,  only : log_unit
if ( log_unit > 0) write(log_unit,*) 'Hello'
```

means the printing of 'Hello' only at the 'specific' nodes into the log files.

Chapter 2

Overview of work flows

2.1 Call tree from the root routine

- `elses_main`
  - `elses_process_options` [`M_options`]
    · set the verbose level
  - `elses_00_version_info` [`M_00_v_info`]
    · plot the version information
  - `set_dst_initial` [`M_md_dst`]
    · set the MPI-related variables.
    Note: The variables are set to be compatible with a one-node calculation, in a non-MPI environment.
  - `set_dst_write_log_file` [`M_io_dst_write_log`]
    · set the log file on each node
  - `show_omp_info` [`M_lib_omp`]
    · show the OMP information into the files
  - `elses_md_main_01` [`M_md_main`] or `elses_band_calculation` [`M_band_calc`]
    · call the routine for the MD calculation (See Sec. 2.2) or the band calculation (See Sec. 2.3)
  - `set_dst_final` [`M_md_dst`]
    · finalize the MPI procedure.
    Note: Nothing is done in a non-MPI environment.

2.2 'GENO' work flow for MD calculations

2.2.1 Call tree from elses_md_main_01

The call tree is summarized for the 'dynamics' and 'optimization' modes.

- `elses_md_main_01` [`M_md_main`]
  - `show_mpi_info` [`M_md_dst`]
    · show the MPI-related information into the files (even in a non-MPI environment)
  - `ini_load_geno` [`M_ini_load_geno`]
    · read the input (XML) files and allocate and set various variables
  - `elses_ini_set_velocity` [`M_md_motion`] (only in the 'dynamics' mode)
    · set the initial velocity vectors by the Maxwell-Boltzman distribution
  - do itemd2=1,itemdmx (MD loop starts)
  - `elses_qm_engine` [`M_qm_engine`]
    · calculate the total energy and force by the electronic structure theory
  - `detect_stop_signal` [`M_lib_stop_signal`]
    · detect the stop signal from the file and the time limit (*)
  - `md_motion_verlet_velocity` [`M_md_verlet`] (only in the 'dynamics' mode)
    · update the velocity data by the velocity Verlet algorithm
  - `elses_md_save_struct` [`M_md_save_struct`]
    · save the MD data, such as the atom position, into the files
  - `elses_md_output_main` [`M_md_output`]
    · plot the data, such as the energy, into the files. See Sec. 2.2.4.
  - `elses_md_motion` [`M_md_motion`]
    · update the atom positions (i) by the velocity-Verlet algorithm in the 'dynamics' mode or (ii) by the steepest-descent algorithm in the 'optimization' mode.
  - enddo (MD loop ends)

(*) In general, the simulation will be completed, when the MD iteration loop of '`do itemd2=1,itemdmx`' reaches the maximum iteration number in the 'dynamics' and 'optimization' modes or the structure optimization is well achieved in the 'optimization' mode. The simulation can be stopped, before the completion, if specified. See Sec. 3.4 for details.

2.2.2 Call tree from 'elses_qm_engine'

- `elses_qm_engine` [`M_qm_engine`]
  · calculate the total energy and force by the electronic structure theory
  output : total force (TB0+rest+CSC) : `atm_force(3,1:noav)` [`M_qm_domain`]
  output : electronic structure (TB0) energy : `etb` [`elses_mod_ene`]
  output : rest-part (repulsive) energy : `ecc` [`elses_mod_ene`]
  output : CSC-part energy : `ecsc` [`elses_mod_ene`]
  output : kinetic energy : `e_kin` [`elses_mod_md_dat`]
  - `calc_kinetic_energy` [`M_md_velocity_routines`]
    · calculate the kinetic energy
  - `qm_domain_setting` [`M_qm_domain`]
    · set the variables
  - `initial_guess_for_charge` [`M_qm_population`]
    · set the charge as the initial guess for the CSC loop
  - `set_rest_geno` [`M_qm_geno`]
    · calculate the rest (repulsive) part for energy and force.
  - `set_hamiltonian_and_overlap_geno` [`M_qm_geno`]
    · calculate the TB0 Hamiltonian and overlap matrices
  - `qm_solver_geno_main` [`M_qm_solver_geno_main`] (only in non-CSC calculation)
    · calculate the density matrix
  - `mulliken` [`M_qm_solver_geno_main`] (only in non-CSC calculation)
    · calculate the mulliken charge
  - `qm_engine_csc` [`M_qm_engine_csc`] (only in CSC calculation)
    · the CSC-loop procedure
  - `save_population_after_convergence` [`M_qm_population`]
    · store the mulliken charge
  - `set_qm_force_geno` [`M_qm_geno`]
    · calculate the electronic-structure (TB0) force
  - `set_atm_force_csc` [`M_qm_geno_CSC`] (only in CSC calculation)
    · calculate the CSC force
  - `generate_foi` [`M_qm_domain`]
    · store the total force
  - `qm_geno_output` [`M_M_qm_geno_output`]
    · calculate the energy terms.

Note : the total energy and the kinetic energy are given as

$$E_{\rm tot} \equiv E_{\rm TB0} + E_{\rm rest} + E_{\rm CSC} \tag{2.1}$$

$$E_{\rm kin} \equiv \frac{1}{2} \sum_{I}^{\rm (atom)} M_I |\boldsymbol{V}_I|^2 \tag{2.2}$$

respectively.

2.2.3 Call tree from 'qm_engine_csc'

Here, 'charge' means the population of the valence electrons and is positive.

- `qm_engine_csc` [`M_qm_engine_csc`]
  · the CSC calculation
  - 'do `i_csc_loop` =1, `n_csc_loop`' (the CSC loop starts)
  - `set_CSC_parameters_geno` [`M_qm_engine_csc`]
    · set the CSC parameters
    output : the τ parameter or the on-site Coulomb energy: : `tau_csc(1:noav)`
      Note : $\tau \equiv (16/5)\times$ (chemical hardness)
    output : the charge deviation from the 'reference' charge : `delta_e_num(1:noav)`
      Note : now the 'reference charge' is chosen as that in neutral atom.
  - `set_hamiltonian_csc` [`M_qm_geno_CSC`]
    · set the CSC Hamiltonian
  - `qm_solver_geno_main` [`M_qm_solver_geno_main`]
    · calculate the density matrix
  - `mulliken` [`M_qm_domain`]
    · calculate the Mulliken charge
      Note : formulation with I:atom index, α:orbital index, m:CSC loop index
        $q_{I\alpha}^{(m)} :=$ (Mulliken charge calculated from the density matrix)
  - '`dq  = rms_delta_q()` '
    · calculate the measure for the CSC convergence : `dq`
  - '`if( dq < ...) ... exit ...` '
    · exit the CSC loop when the convergence criteria is satisfied
  - `tune_mix_ratio` [`M_qm_engine_csc`]
    ·tune the mixing ratio, $\xi^{(m)}$, in the charge-mixing scheme for faster convergence
  - `renew_charge` [`M_qm_geno`]
    ·modify the charge in the charge-mixing scheme
      Note : formulation with I:atom index, α:orbital index, m:CSC loop index
        $q_{I\alpha}^{(m)} := q_{I\alpha}^{(m-1)} + \xi^{(m)}(q_{I\alpha}^{(m)} - q_{I\alpha}^{(m-1)})$
        with $\xi^{(1)} \equiv 1$
  - '`enddo`' (the CSC loop ends)

2.2.4 Call tree from 'elses_md_output_main'

- `elses_md_output_main` [M_md_output]
  - `output_energy` [M_md_output]
    · plot the energy terms into the main output file 'Output.txt'. See Quick Start for details. This routine works only at the root node.
  - `md_output_compat` [M_md_output]
    · plot the energy terms into the standard output file. This routine works only at the root node.
    Note: This routine is designed only for the compatibility to very old versions.
  - `output_force_ave_max` [M_md_output]
    · plot the maximum and average of the force into the main output file 'Output.txt'. This routine works only at the root node.
    Note: The maximum and average of the force are important, particularly in the structure optimization mode.
  - `output_atom_charge` [M_output_atom_charge]
    · plot the energy terms into the main output file 'Output.txt'. (only at the root note)
  - `qm_output_matrices` [M_qm_output_matrices]
    ·
  - `qm_output_levels` [M_qm_output_levels]
    ·
  - `output_atom_energy` [M_output_atom_energy]
    ·

2.3 Work flow in band calculation

2.3.1 Call tree from elses_band_calculation

- `elses_band_calculation` [`M_band_calc`]
  - `ini_load_geno` [`M_ini_load_geno`]
    · read the input (XML) files and allocate and set various variables
  - `qm_initialization` [`M_ini_load_geno`]
    · Initial procedure for constructing the Hamiltonian and overlap matrices.
    - `qm_domain_setting` [`M_qm_domain`]
    - `initialize_charge` [`M_qm_geno`]
  - `set_hamiltonian_and_overlap_geno` [`M_qm_geno`]
    ·
  - `write_hamiltonian_and_overlap_geno` [`M_qm_geno`]
    ·

Chapter 3

Notes for several procedures and issues

3.1 How to treat XML tags

3.1.1 Summary

This section is devoted how to treat XML tags. Hierarchical tags in the configuration XML file, such as 'config.xml', are stored in hierarchical variables. Here the following tag part is focused on;

```
<calc mode="dynamics">
   <dynamics scheme="velocity verlet">
     <delta unit="fsec"> 1.00 </delta>
     <total unit="fsec"> 10000.00 </total>
   </dynamics>
</calc>
```

These values are stored in the module `M_config` and can be shown by the code

```
  use M_config,   only : config
  write(*,*) config%calc%mode
  write(*,*) config%calc%dynamics%scheme
  write(*,*) config%calc%dynamics%delta
  write(*,*) config%calc%dynamics%total
```

3.1.2 Technical details for defining the hierarchy

In the code, the hierarchy is defined by many 'type' commands

```
type :: dynamics_type
   character(len=16)   :: scheme
   real(8)             :: delta
   real(8)             :: total
end type dynamics_type

type :: calc_type
    character(len=20)   :: mode
    type(dynamics_type) :: dynamics
end type calc_type

type :: config_type
   type(calc_type)     :: calc
end type config_type

type(config_type)      :: config
```

in the module `M_config`.

3.1.3 Technical details for work flow

The work flow for setting these valuables is summarized in the following call tree

```
- elses_main
   - elses_md_main_01
   |    - ini_load_geno
   |    |    - load_xml_config
   |    |    |    - config_load
   |    |    |    |    - calc_default
   |    |    |    |    |    - dynamics_default ! Setting the default value
   |    |    |    |    - calc_load
   |    |    |    |    |    + dynamics_load     ! Setting  from the XML file, if possible
```

3.2 Note for the atom position

3.2.1 Explanation

Several notes are given for the atom position. In ELSES, the variables for atom position $(x_i(t), y_i(t), z_i(t))$ is normalized in the unit cell lengths (`ax, ay, az`), except where indicated.

In a periodic boundary condition, the reduced and non-reduced coordinates for atom position should be distingushed. A reduced coodinate, $x_i(t)$, always satisfies $0 < x_i(t) < 1$, while the non-reduced one does not. The reduce coorinate is used, for example, when the Hamiltonian and overlap matrices are generated. The non-reduced coorinates should be used, for example, in ploting the trajectory of atom or calculating the mean square displacement.

3.2.2 Variables and work flow

Variables for atom position are defined multiply, partly because routines in different styles are connected with each other. Here is the list;

- (a) `atm_position(3, 1:noav)` [`M_qm_domain`]
  Normalized. Reduced. (mainly used in the GENO workflow)
- (b) `tx(1:noav), ty(1:noav), tz(1:noav)` [`elses_mod_tx`]
  Normalized. Reduced.
- (c) `txp(1:noav), typ(1:noav), tzp(1:noav)` [`elses_mod_tx`]
  Normalized. Non-reduced.
- (d) `config%system%structure%vatom(1:noav)%position(1:3)` [`M_config`]
  Non-normalized. Non-reduced.

The work flow is summarized in the call tree.

```
- elses_main
   - elses_md_main_01
   |   - ini_load_geno
   |   |   - load_xml_config
   |   |   |   - config_load
   |   |   |   |   - system_load
   |   |   |   |   |   - structure_load    ! Allocate (d)
   |   |   |   |   |   |   + atom_load      ! Set (d) from the XML file
   |   |   - ini_load_common
   |   |   |   + load_element_alloc
   |   |   |   - load_structure_alloc
   |   |   |   |   - elses_alloc_ini_txp
   |   |   |   |   |   - elses_alloc_tx    ! Allocate (b)
   |   |   |   |   |   - elses_alloc_txp   ! Allocate (c)
   |   |   |   - allocate_velocity
   |   |   |   + set_unitcell_info
   |   |   |   - elses_md_motion_ini_set
   |   |   |   - set_structure_data       ! Set (c) from (d)
   |   |   |   |   - elses_gene_tx
   |   |   |   |       - md_update_tx     ! Set (b) from (c)
   |   |   |   + load_xml_conditions
   |   + elses_ini_set_velocity
   | ------------(MD loop starts)-------------
   |   - elses_qm_engine
   |   |   + qm_domain_setting       ! Allocate (a), if not. Set (a) from (b)
   |   |   + set_hamiltonian_and_overlap_geno  ! Use (a) for calc. H and S matrices
   |   + detect_stop_signal
   |   + md_motion_verlet_velocity
   |   - elses_md_save_struct
   |   |   - elses_xml_save_txp      ! Set (d) from (c)
   |   |   + save_restart_xml        ! Save (d) into the restart XML file
   |   |   + save_position_data      ! Save (d) into the position files
   |   + elses_md_output_main        ! Update (c). Set (b) from (c)
   |   + elses_md_motion
   | ------------(MD loop ends)-------------
  + set_dst_final
```

3.3 The booking list for the order-N formatted matrices

Two integer variables are prepared so as to define the booking list for the order-N formatted matrices;

- **njsd(jsv2,ict)** : the number of booked atoms for the **jsv2**-th atom within the **ict**-th.

- **jsv4jsd(jsd1 , ict)** : link between atom index and booking index (**jsd1**) within the **ict**-th shell.

The variables appear typically as follows;

```
ict=1
do jsv2=1,noav
  do jsd1 = 1, njsd(jsv2, ict)
      jsv1 = jsv4jsd(jsd1,jsv2)
          (operations)
  enddo
enddo
```

Here (**jsv1,jsv2**) is an atom pair in the neighbor list of the **jsv2**-th atom. Booking lists can be created with multiple shells, from (an) inner shell(s) into (an) outer shell(s), The shells are distinguished by the variable **ict**. In the present code, two shells are defined, (i) zero-sh shell or 'self' shell that contains the 'self' atom (**ict**=0), (ii) first shell or shell with the interaction radius of Hamiltonian (**ict**=1).

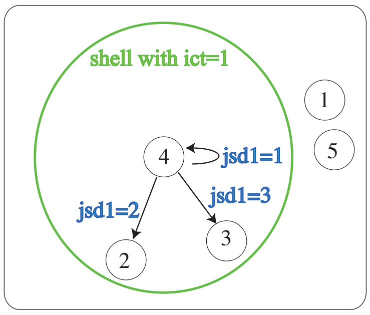

Figure 3.1: Example of booking list with five atoms (noav=5).

An example is shown in Fig. 3.1, in which five atoms are included (**noav** =5). Here we focus on the booking list of the atom labelled with **jsv2**=4 that interacts with the two atoms labelled with **jsv2**=2 and **jsv2**=3. The number of atoms in the zero-th shell is, as always, one;

```
njsd(4, 0)=1
```

and the member of the zero-th shell is, as always, only the 'self' atom (**jsv2=2**);

```
jsv4jsd(1,4)=4
```

The number of atoms in the zero-th shell is three

```
njsd(4, 1)=3
```

```
jsv4jsd(1,4)=4
jsv4jsd(2,4)=2
jsv4jsd(3,4)=3
```

3.4 Stopping the simulation before completion

3.4.1 detect_stop_signal [M_lib_stop_signal]

The simulation can be stopped before the completion, if user gives a ‘stop signal’ in the following two ways.

The ‘stop signal’ can be detected, at the routine of `detect_stop_signal`, once in every MD step (See the work flow of Sec. 2.1). When the ‘stop signal’ is detected, the data will be saved in the files and the simulation will stop.

3.4.2 Stopping the simulation by time limit

The time limit can be set in the configuration XML file. For example, the time limit of 100 minutes is realized as follows;

```
<calc>
 <limit>
   <time unit="min"> 100 </time>
 </limit>
</calc>
```

The unit can be given by (a) ‘second’, ’seconds’, ’sec’ or ’s’ for second, (b) ‘minute’, ‘minutes’, ‘min’ or ’m’ for minute, (c) ‘hour’, ‘hours’ or ‘h’ for hour, and (d) ‘day’, ‘days’, or ‘d’ for day.

The elapse time exceeds the limit at the routine of `detect_stop_signal`, the data will be saved in the file and the simulation will stop.

It is noteworthy that the above limit should be smaller than the hard limit, since the whole simulation time is always larger than the above limit, by the time for data saving.

3.4.3 Stopping the simulation by putting a file

Users can stop the simulation, when they put the file named ‘`00_stop_signal.txt`’ in the working directory and the first line of the file gives a non-zero integer. When the file is detected at the routine of `detect_stop_signal`, the data will be saved in the file and the simulation will stop.

3.5 Routines for measuring the computational time

3.5.1 get_system_clock_time [M_wall_clock_time]

Routine for get the wall clock time in second, thread save.
Example :

```
call get_system_clock_time(time_value)
```

3.5.2 get_elapse_wall_clock_time [M_wall_clock_time]

Routine for getting the wall clock time in second, *not* thread save.
Example :

```
call get_elapse_wall_clock_time(time_value)
```

3.5.3 mpi_wrapper_barrier_time [M_lib_mpi_wrapper]

3.6 Routines for numerical and mathematical procedures

3.6.1 get_determinant [M_lib_math_func]

Routine for getting the determinant of a (3×3) matrix.

3.6.2 Fermi_Dirac_Func [M_lib_math_func]

Function for obtaining the Fermi-Dirac function of $f(x) \equiv 1/(1+e^x)$.
Note: This routine was written so as to avoid an possible floating point exception in e^x with $x \to \pm\infty$.

3.6.3 dhsort [M_lib_sort]

Routine for heap sort program

3.6.4 rndini, rndu [M_lib_random_num]

Routine for generating random numbers.
Note: This routine is used for generating the initial velocity in the Maxwell-Boltzmann distribution.

Chapter 4

Important variables

Variables are written in the form of 'variable name [module name]', such as
`atm_force(3,noav)` [`M_qm_domain`].

4.1 Variables for controlling the verbose level and the work flow

4.1.1 i_verbose [M_qm_domain]

integer; (unchanged throughout the simulation) (different value among the nodes)
Level for verbose mode
i_verbose ≥ 1 (for verbose mode) / $= 0$ (for quiet mode).
Default : **i_verbose** $= 0$
Note: The value of **i_verbose** $= 1$ will be set, when the elses runs with the option of '-verbose';

```
prompt> elses -verbose ./config.xml
```

Note: In the DST work flow, this variable is different among nodes so as to suppress redundant file output among nodes. See Sec. 1.3.2.

4.1.2 c_system [M_qm_domain]

character(len=32)
Name of selected system or Hamiltonian type.
Ex. **c_system='geno'**.
Note: The variable is set, in the configuration XML file, by the 'model' tag (**model="geno"**).

4.1.3 config%calc%distributed%set [M_config]

logical
Switch for the DST work flow.

4.1.4 config%calc%solver%scheme [M_config]

character(len=16)
Solver scheme, such as 'eigen' for the eigen state solver.

4.2 Variables for DST calculation

4.2.1 myrank [M_lib_dst_info]

integer; (unchanged throughout the simulation), (different value among the nodes)
The node index in the DST calculation (`myrank`=0,1,2..)
Note: `myrank` = 0 in the non-DST work flow
Note: set in the routine '`set_dst_initial`' [`M_md_dst`]

4.2.2 nprocs [M_lib_dst_info]

integer; (unchanged throughout the simulation)
The number of nodes in the DST calculation.
Note : `nprocs` =1 in the non-DST work flow Note: set in the routine '`set_dst_initial`' [`M_md_dst`]

4.2.3 mpi_is_active [M_lib_dst_info]

logical; (unchanged throughout the simulation)
Status whether the MPI parallelism is active or not.
Note: set in the routine '`set_dst_initial`' [`M_md_dst`]

4.2.4 log_unit [M_io_dst_write_log]

integer,
An log file named 'log-node00000.txt' appear and contains technical data, such as consumed time of specific routines. The file unit number of for log file at each node. The value is meaningful (0 < `log_unit` < 99) only among the specific nodes and is not (`log_unit` =-1) among the other nodes. See Sec. 1.3.2.
Note : set in the routine '`set_dst_write_log_file`' [`M_io_dst_write_log`]

4.3 Physical variables for atom species

4.3.1 nos [M_qm_domain]

integer
Number of elements (atom species).
Ex. NOS=2 for binary system such as GaAs.

4.3.2 nval(1:nos) [M_qm_domain]

integer
Number of valence electron orbitals for each atom species.
Ex. nval(1)=4, nval(2)=9 : the first atom species has four (s, p_x, p_y, p_z) orbitals and the second one has nine orbitals.

4.3.3 element_name(1:nos) [M_qm_domain]

character(len=8)
Element name, such as 'Si' for silicon.

4.3.4 awt(1:nos) [elses_mod_mass]

real(8)
Atomic weight.
Note : set in the routine '`SetAtomParameters`' [`M_qm_geno_Huckel_atom_params`]
Note : used only for the calculation of `amm(noav)` (See Sec. 4.5.5) in the routine '`elses_set_mass2`' (in the file 'elses-xml-02.f').

4.4 Physical variables for the unit cell

4.4.1 ax,ay,az [M_qm_domain]

real(8); atomic unit
Unit cell length, essential even in non-periodic system
Note: Now only orthogonal cell vectors are supported.

4.4.2 i_pbc_x, i_pbc_y, i_pbc_z [M_qm_domain]

integer
Boundary condition for x, y, z directions.
Ex. i_pbc_x=1(periodic)、=0(non periodic)

4.5 Physical variables for atoms

4.5.1 noav [M_qm_domain]

integer
The number of atoms in the quantum mechanical calculation.

4.5.2 atm_element(1:noav) [M_qm_domain]

integer,
The element index of the atom. For example, the value is one or two in a binary system (**nos**=2).

4.5.3 atm_position(1:3,1:noav) [M_qm_domain]

real(8), normalized unit ($0 < x_i, y_i, z_i < 1$)
The atom position (x_i, y_i, z_i) normalized by the unit cell lengths (ax,ay,az);

4.5.4 atm_force(1:3,1:noav) [M_qm_domain]

real(8); atomic unit
The atom force

4.5.5 amm(1:noav) [M_qm_domain]

real(8), atomic unit
(mass of atom)$^{-1}$
Note : set in the routine '`elses_set_mass2`' (in the file 'elses-xml-02.f')

4.5.6 velx(1:noav),vely(1:noav),velz(1:noav) [elses_mod_vel]

real(8);
The variables related for velocity, *not* the velocity vectors ($\boldsymbol{V}_I$).
Note : This variable is $h\boldsymbol{V}_I$ with the MD time step h. The dimension is length and the unit is normalized by the unit cell lengths.
Note : used in the routines, such as
    `allocate_velocity` [`M_md_velocity_routines`] : allocation
    `calc_kinetic_energy` [`M_md_velocity_routines`] : calculation of the kinetic energy

4.6 Scalar and vector variables for electronic states

4.6.1 total_electron_number [M_qm_domain]

real(8);
The total electron number in the system.

4.6.2 temp_for_electron [M_qm_domain]

real(8); atomic unit
The 'electronic temperature' or the level-broadening parameter in the Fermi-Dirac distribution.

4.6.3 chemical_potential [M_qm_domain]

real(8); atomic unit
The chemical potential used in the Fermi-Dirac distribution.

4.6.4 e_num_on_basis(nvl, noav) [M_qm_domain]

real(8);
Charge (valence electron population) on basis.
Note : `nvl` $\equiv$ max (nval(1:nos)) ; the maximum number of the valence orbitals per atom

4.6.5 e_num_on_atom(1:noav) [M_qm_domain]

real(8);
Charge on atom.

4.6.6 previous_e_num_on_atom(1:noav) [M_qm_domain]

real(8);
Charge on atom at the previous CSC loop.
Note; This quantity is needed only in the CSC loop.

4.6.7 delta_e_num(1:noav) [M_qm_domain]

real(8);
The charge deviation on atom from the 'reference' value.
Note: the reference value is set to be that in neutral atom now.

4.7 Order-N formatted matrices for electronic states

Note : `nvl` $\equiv$ max (nval(1:nos)) ; the maximum number of the valence orbitals per atom
Note : `noao` : maximum number of the atoms in the booking list; local variable

4.7.1 dhij(1:nvl, 1:nvl, 1:noao, 1:noav) [M_qm_domain]

real(8); atomic unit
Total Hamiltonian ($H \equiv H_{TB0} + H_{\mathrm{CSC}}$) in an order-N formatted matrix.

4.7.2 dsij(1:nvl, 1:nvl, 1:noao, 1:noav) [M_qm_domain]

real(8); atomic unit
Overlap matrices in the order-N formatted matrix.

4.7.3 dbij(1:nvl, 1:nvl, 1:noao, 1:noav) [M_qm_domain]

real(8); atomic unit
Density matrices in the order-N formatted matrix.

4.7.4 dpij(1:nvl, 1:nvl, 1:noao, 1:noav) [M_qm_domain]

real(8); atomic unit
Energ density matrices in the order-N formatted matrix.

4.7.5 ham_tb0(1:nvl, 1:nvl, 1:noao, 1:noav) [M_qm_domain]

real(8); atomic unit
TB0 Hamiltonian (H_{TB0}) in the order-N formatted matrix.

4.7.6 ham_csc(1:nvl, 1:nvl, 1:noao, 1:noav) [M_qm_domain]

real(8); atomic unit
CSC Hamiltonian (H_{CSC}) in the order-N formatted matrix.

4.7.7 ddsij(1:nvl, 1:nvl, 1:noao, 1: noav, 1:3) [M_qm_domain]

real(8); atomic unit
Derivative of the overlap matrix with respect to the atomic coordinates

4.7.8 dham_tb0(1:nvl, 1:nvl, 1:noao, 1:noav, 1:3) [M_qm_domain]

real(8); atomic unit
Derivative of the TB0 Hamiltonian matrix with respect to the atomic coordinates

4.7.9 ddhij(1:nvl, 1:nvl, 1:noao, 1:noav, 1:3) [M_qm_domain]

real(8); atomic unit

4.8 The member list for the order-N formatted matrices

4.8.1 jsv4jsd(1:noao,1:noav), njsd(1:noav,0:1) [M_qm_domain]

integer
njsd(jsv2,ict) : the number of booked atoms for the **jsv2**-th atom within the **ict**-th.
jsv4jsd(jsd1 , ict) : link between atom index and booking index (**jsd1**) within the **ict**-th shell.
See Sec. 3.3.

4.9 Variables used in the CSC formulation

4.9.1 gamma_csc(1:noav, 1:noav) [M_qm_domain]

real(8);
$\gamma_{I,J}$ for atom pair (I, J)

4.9.2 dgamma_csc(1:3, 1:noav, 1:noav) [M_qm_domain]

real(8);

4.9.3 fgamma_csc(1:noav, 1:noav) [M_qm_domain]

real(8);

4.10 Variables for temperature and heat bath

4.10.1 config%system%temperature [elses_mod_thermo]

real(8); atomic unit
Heat bath temperature specified by user.

4.10.2 amq [elses_mod_thermo]

real(8); atomic unit
$(\text{mass})^{-1}$

4.10.3 thb, thbold [elses_mod_thermo]

real(8); atomic unit
The coordinate of the heat bath freedom at the present and previous time steps, respectively.

4.10.4 vhb, vhbold [elses_mod_thermo]

real(8); atomic unit
The quantity of (velocity) $\times$ (time step) for the heat bath freedom at the present and previous time steps, respectively.

4.11 Physical and mathematical constants

4.11.1 pi [M_lib_phys_const]

real(8), parameter
pi = .314159265358979323D1

4.11.2 ev4au [M_lib_phys_const]

real(8), parameter
For unit conversion in energy : 1 au = (**ev4au**) eV .
ev4au = 2.0d0*13.6058d0

4.11.3 angst [M_lib_phys_const]

real(8), parameter
For unit conversion in length : 1 au = (**angst**) Å
angst = 0.529177d0

4.11.4 au_fsec [M_lib_phys_const]

real(8), parameter
For unit conversion in time : 1 au = (**au_fsec**) Å
au_fsec = 124.06948d0/3.0d0

4.11.5 au_mass [M_lib_phys_const]

real(8), parameter
For unit conversion in weight
au_mass = 1.6605655d-24/9.109534d-28

4.11.6 ev2kel [M_lib_phys_const]

real(8), parameter
For unit conversion in energy : 1 eV = (**ev2kel**) kelvin
ev2kel = 1.60217733d0/1.380658d0*10000.0d0

4.11.7 ene_j4kel [M_lib_phys_const]

real(8), parameter
For unit conversion in energy : 1 eV = (**ene_j4kel**) J
ene_j4kel = 1.60219d-19

4.12 Miscellaneous parameters

4.12.1 DOUBLE_PRECISION [M_qm_domain]

integer, parameter
An integer parameter defined by

```
   integer, parameter   :: DOUBLE_PRECISION=kind(1d0)
```

so as to specify double-precision variables, as follows;

```
   use M_qm_domain, only : DOUBLE_PRECISION
   real(DOUBLE_PRECISION) :: A
```

Chapter 5

Theory note

5.1 Velocity-Verlet algorithm in MD simulaion

The velocity-Verlet algorithm is used for the numerical integration of the Newton equation in MD simulaions. This section is devoted to a brief description for the micro-canonical and canonical ensembles.

5.1.1 Basics

The second order Taylor expansion for position and velocity of a particle is given as

$$\begin{aligned} \dot{\boldsymbol{R}}(t) &= \dot{\boldsymbol{R}}(t-h) + h\boldsymbol{A}(t-h) + \frac{h^2}{2}\dot{\boldsymbol{A}}(t-h) + O(h^3) & (5.1) \\ \boldsymbol{R}(t+h) &= \boldsymbol{R}(t) + h\dot{\boldsymbol{R}}(t) + \frac{h^2}{2}\boldsymbol{A}(t) + O(h^3) & (5.2) \end{aligned}$$

where h is the time interval and $\boldsymbol{R}$, $\dot{\boldsymbol{R}}$ and $\boldsymbol{A} \equiv \ddot{\boldsymbol{R}}$ are position, velocity and acceleration of the particle, respectively. Here the Taylor expansion of acceleration

$$\boldsymbol{A}(t) = \boldsymbol{A}(t-h) + h\dot{\boldsymbol{A}}(t-h) + O(h^2). \tag{5.3}$$

gives the relation of

$$\frac{h^2}{2}\dot{\boldsymbol{A}}(t-h) = \frac{h}{2}\left(\boldsymbol{A}(t) - \boldsymbol{A}(t-h)\right) + O(h^3). \tag{5.4}$$

When Eq. (5.4) is introduced into Eq. (5.1), one obtains the formulation of

$$\dot{\boldsymbol{R}}(t) = \dot{\boldsymbol{R}}(t-h) + \frac{h}{2}\left(\boldsymbol{A}(t) - \boldsymbol{A}(t-h)\right) + O(h^3), \tag{5.5}$$

a formulation without $\dot{\boldsymbol{A}}$.

5.1.2 Algorithm for micro-canonical ensemble

The energy and the Newton equation for a system with N_A particles are given by

$$E = \sum_I \frac{M_I}{2} \dot{\boldsymbol{R}}_{\boldsymbol{I}}^2 + U(\{\boldsymbol{R}_{\boldsymbol{I}}\}) \tag{5.6}$$

$$\ddot{\boldsymbol{R}}_I = \boldsymbol{A}_I \equiv -\frac{1}{M_I}\frac{\partial U}{\partial \boldsymbol{R}_{\boldsymbol{I}}}, \tag{5.7}$$

respectively. The corresponding velocity-Verlet algorithm is as follows

$$\dot{\boldsymbol{R}}_I(t) = \dot{\boldsymbol{R}}_I(t-h) + \frac{h}{2}\{\boldsymbol{A}_I(t) + \boldsymbol{A}_I(t-h)\} + O(h^3) \tag{5.8}$$

$$\boldsymbol{R}_I(t+h) = \boldsymbol{R}_I(t) + h\dot{\boldsymbol{R}}_I(t) + \frac{h^2}{2}\boldsymbol{A}_I(t) + O(h^3) \tag{5.9}$$

The flow chart of calculations are given by

$$\Rightarrow \boldsymbol{R}_I(t) \Rightarrow \boldsymbol{A}_I(t) \Rightarrow \dot{\boldsymbol{R}}_I(t) \Rightarrow \boldsymbol{R}_I(t+h) \Rightarrow \tag{5.10}$$

The error of the total-energy conservation at the n-th MD time step ($t = nh$) is estimated as

$$\delta E_n \equiv E(nh) - E((n-1)h) \propto h^3 \tag{5.11}$$

for one MD step. For a finite time evolution with a finite time-interval τ, the number of MD steps is $\nu \equiv (\tau/h)$. So the errors at all MD steps are summed up to be

$$E(\tau = \nu h) - E(0) = \sum_n^{\nu} \delta E_n \propto \nu h^3 = \left(\frac{\tau}{h}\right) h^3 = \tau h^2 \tag{5.12}$$

5.1.3 Algorithm for canonical ensemble

Now we turn to explain the finite-temperature dynamics within the Nosé 'thermostat' method. We do not derive the formulation and just show the resultant Newton equation with the temperature T;

$$\begin{aligned}\ddot{\boldsymbol{R}}_I &= \boldsymbol{A}_I - \dot{\eta}\dot{\boldsymbol{R}}_{\boldsymbol{I}} && (5.13)\\ \ddot{\eta} &= \frac{1}{Q}\left[\sum_I M_I \dot{\boldsymbol{R}}_{\boldsymbol{I}}^{\,2} - 3N_A k_B T\right]. && (5.14)\end{aligned}$$

The conserved energetic quantity is given by

$$H^* = \sum_I \frac{M_I}{2}\dot{\boldsymbol{R}}_{\boldsymbol{I}}^{\,2} + U(\{\boldsymbol{R}_{\boldsymbol{I}}\}) + \frac{Q}{2}\dot{\eta}^2 + 3N_A k_B T\eta, \tag{5.15}$$

where the third and fourth terms can be interpreted as the kinetic and potential energies of the 'thermostat' freedom η. The parameter Q corresponds to the mass of the 'thermostat'. One can prove that the time average of the present dynamics gives the ensemble average in the canonical ensemble. As a practical viewpoint, if we choose a proper value for Q, the 'thermostat' works well and the kinetic energy is expected to be almost constant:

$$\sum_I \frac{M_I}{2}\dot{\boldsymbol{R}}_{\boldsymbol{I}}^{\,2} \approx \frac{3}{2}N_A k_B T. \tag{5.16}$$

The corresponding velocity-Verlet algorithm is as follows;

$$\begin{aligned}\dot{\eta}(t) &= \dot{\eta}(t-h)\\ &\quad + \frac{h}{2Q}\left[\sum_I M_I \dot{\boldsymbol{R}}_I(t)^2 + \sum_I M_I \dot{\boldsymbol{R}}_I(t-h)^2 - 6N_A k_B T\right] && (5.17)\\ \dot{\boldsymbol{R}}_I(t) &= \dot{\boldsymbol{R}}_I(t-h)\\ &\quad + \frac{h}{2}\left[\boldsymbol{A}_I(t) + \boldsymbol{A}_I(t-h) - \dot{\eta}(t)\dot{\boldsymbol{R}}_I(t) - \dot{\eta}(t-h)\dot{\boldsymbol{R}}_I(t-h)\right] && (5.18)\\ \eta(t+h) &= \eta(t) + h\dot{\eta}(t) + \frac{h^2}{2Q}\left[\sum_I M_I \dot{\boldsymbol{R}}_I(t)^2 - 3N_A k_B T\right] && (5.19)\\ \boldsymbol{R}_I(t+h) &= \boldsymbol{R}_I(t) + h\dot{\boldsymbol{R}}_I(t) + \frac{h^2}{2}\left[\boldsymbol{A}_I(t) - \dot{\eta}(t)\dot{\boldsymbol{R}}_I(t)\right]. && (5.20)\end{aligned}$$

Equations (5.17) and (5.18) are implicit formula and are rewritten as

$$\begin{aligned}0 &= \dot{\eta}(t) - \dot{\eta}(t-h) - \frac{h}{2Q}\left[K + \frac{K^*}{(1+\frac{h}{2}\dot{\eta}(t))^2} - 6N_A k_B T\right] && (5.21)\\ \dot{\boldsymbol{R}}_I(t) &= \frac{1}{1+\frac{h}{2}\dot{\eta}(t)}\boldsymbol{W}_I(t), && (5.22)\end{aligned}$$

respectively, where

$$\begin{aligned}K &\equiv \sum_I M_I \dot{\boldsymbol{R}}_I^{\,2}(t-h) && (5.23)\\ K^* &\equiv \sum_I M_I \boldsymbol{W}_I^{\,2}(t) && (5.24)\\ \boldsymbol{W}_I(t) &\equiv \left\{1 - \frac{h}{2}\dot{\eta}(t-h)\right\}\dot{\boldsymbol{R}}_I(t-h) + \frac{h}{2}\left\{\boldsymbol{A}_I(t) + \boldsymbol{A}_I(t-h)\right\}. && (5.25)\end{aligned}$$

Equation (5.21) is the equation for $x \equiv \dot{\eta}(t)$ and is solved iteratively using the Newton-Raphson method;

$$x^{(j+1)} = x^{(j)} - \frac{f(x^{(j)})}{f'(x^{(j)})} \tag{5.26}$$

where j is the number of the iterations and

$$f(x) \equiv x - \dot{\eta}(t-h) - \frac{h}{2Q}\left[K + \frac{K^*}{(1+\frac{h}{2}x)^2} - 6N_A k_B T\right] \tag{5.27}$$

$$f'(x) = 1 + \frac{h^2}{2Q}\frac{K^*}{(1+\frac{h}{2}x)^3}. \tag{5.28}$$

This iterative algorithm needs a proper initial value $x^{(0)}$ and we choose the value to be

$$x^{(0)} = \dot{\eta}(t-h) + \frac{h}{Q}\left[\sum_I M_I \dot{\boldsymbol{R}}_I^2(t-h) - 3N_A k_B T\right], \tag{5.29}$$

which is given by Eq.(5.17) under the assumption of

$$\sum_I M_I \dot{\boldsymbol{R}}_I^2(t-h) \approx \sum_I M_I \dot{\boldsymbol{R}}_I^2(t). \tag{5.30}$$

The algorithm is summarized as follows, where the variables $\{\boldsymbol{R}_I(t), \dot{\boldsymbol{R}}_I(t-h), \boldsymbol{R}_I(t-h), \boldsymbol{A}_I(t), \boldsymbol{A}_I(t-h), \dot{\eta}(t-h), \eta(t-h)\}$ are already obtained:

1. Calculate $\dot{\eta}(t)$ using the Newton-Raphson method with Eq. (5.26) and the initial value of Eq. (5.29).
2. Calculate $\{\dot{\boldsymbol{R}}_I(t)\}$ using Eq. (5.22).
3. Determine $\eta(t+h)$ and $\{\boldsymbol{R}_I(t+h)\}$ using Eqs. (5.19) and (5.20) .

Note that if we set the values of η and $\dot{\eta}$ to be zero at the all time steps, Eqs.(5.18) and (5.20) are reduced to Eqs.(5.8) and (5.9).