

SIESTA: Basic Execution

Dr. Ricardo Faccio
Centro NanoMat
Facultad de Química
Universidad de la República

El Método

Linear-scaling DFT based on NAOs (Numerical Atomic Orbitals)

P. Ordejon, E. Artacho & J. M. Soler, Phys. Rev. B 53, R10441 (1996)

J. M. Soler et al, J. Phys.: Condens. Matter **14**, 2745 (2002)

- Born-Oppenheimer (relaxations, mol.dynamics)
- DFT (LDA, GGA, Hybrid, VDW)
- Pseudopotentials (norm conserving, factorised)
- Numerical atomic orbitals as basis (finite range)
- Numerical evaluation of matrix elements (3Dgrid)

Implemented in the **SIESTA method**:

D. Sanchez-Portal, P. Ordejon, E. Artacho & J. M. Soler
Int. J. Quantum Chem. 65, 453 (1997)

Siesta resources (1)

- Web page: <http://www.icmab.es/siesta>
- Lyon and Barcelona Schools
- Pseudos and basis database
- Mailing list
- Usage manual

Siesta resources (2)

- ▣ Andrei Postnikov Siesta utils page:
<http://www.home.uni-osnabrueck.de/apostnik/download.html>
- ▣ Lev Kantorovich Siesta utils page:
<http://www.cmmp.ucl.ac.uk/~lev/codes/lev00/index.html>

Siesta software package:

- Src:** Sources of the Siesta code.
- Src/Sys:** makefiles for the compilation
- Src/Tests:** A collection of tests.
- Docs:** Documentation and user conditions:
 - User's Guide (siesta.tex)
- Pseudo:** ATOM program to generate and test pseudos.
- Examples:** fdf and pseudos input files for simple systems.
- Tutorials:** Tutorials for basis and pseudo generation.
- Utils:** Programs or scripts to analyze the results.

To run Siesta you need:

- 1.- Access to the **executable file**:
- 2.- An **input file**: written in ascii (plain text) using:
Flexible Data Format (FDF)
- 3.- A **pseudopotential file** for each kind of element in the input file. Two different formats:
 - Unformatted binary (.vps)
 - Formatted ASCII (.psf) (more transportable and easy to look at)

Running siesta

Siesta has no windows interface, it runs in shell.

Main input file: “name”.fdf

- Contents:

- Physical data of the system

- Variables to control the approximations

- Format:

- Flexible Data Format (FDF) developed by A. García and J. M. Soler

FDF (I)

- Data can be given in any order

- Data can be omitted in favor of default values

- Syntax: ‘data label’ followed by its value

Character string:	SystemLabel	h2o
-------------------	-------------	-----

Integer:	NumberOfAtoms	3
----------	---------------	---

Real:	PAO.SplitNorm	0.15
-------	---------------	------

Logical:	SpinPolarized	.false.
----------	---------------	---------

Physical magnitudes	LatticeConstant	5.43 Ang
---------------------	-----------------	----------

FDF (II)

- Labels are **case insensitive** and characters **_.** are **ignored**

`LatticeConstant` is equivalent to `lattice_constant`

- Text following **#** are **comments**
- **Logical** values: `T` , `.true.` , `yes`, `F` , `.false.` , `no`

By default logicals are true: `DM.UseSaveDM`

- **Character** strings, **NOT** in apostrophes

- **Complex** data structures: **blocks**

`%block label`

`...`

`%endblock label`

FDF (III)

- **Physical magnitudes:** followed by its **units**.

Many physical units are recognized for each magnitude

(Length: `m`, `cm`, `nm`, `Ang`, `bohr`)

Automatic conversion to the ones internally required.

- You may **'include'** other FDF files or **redirect** the search to another file, so for example in the main fdf it's possible:

`AtomicCoordinatesFormat < system_xyz.fdf`

`AtomicCoordinatesAndAtomicSpecies < system_xyz.fdf`

Basic input variables

- 1.- General system descriptors
- 2.- Structural and geometrical variables
- 3.- Functional and solution method (Order-N/diagonalization)
- 4.- Convergence of the results
- 5.- Self-consistency
- 6.- Basis set generation related variables:

“How to test and generate basis sets”

General system descriptor: output

SystemName: descriptive name of the system

SystemName Si bulk, diamond structure

SystemLabel: nickname of the system to name output files

SystemLabel Si

(After a successful run, you should have files like

Si.DM : Density matrix

Si.XV: Final positions and velocities

...)

Structural and geometrical variables

NumberOfAtoms: number of atoms in the simulation

```
NumberOfAtoms 2
```

NumberOfSpecies: number of different atomic species

```
NumberOfSpecies 1
```

ChemicalSpeciesLabel: specify the different chemical species.

```
%block ChemicalSpeciesLabel
```

```
1 14 Si
```

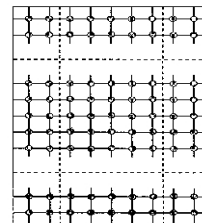
```
%endblock ChemicalSpeciesLabel
```

ALL THESE VARIABLES ARE MANDATORY

Lattice Vectors

Surfaces

Atoms in the unit cell **always** are
periodically repeated throughout space
along the lattice vectors



LatticeConstant: real length to define the scale of the lattice vectors

```
LatticeConstant 5.43 Ang
```

LatticeParameters: Crystallographic way

```
%block LatticeParameters
```

```
1.0 1.0 1.0 60. 60. 60.
```

```
%endblock LatticeParameters
```

LatticeVectors: read as a matrix, each vector on its own line

```
%block LatticeVectors
```

```
0.0 0.5 0.5
```

```
0.5 0.0 0.5
```

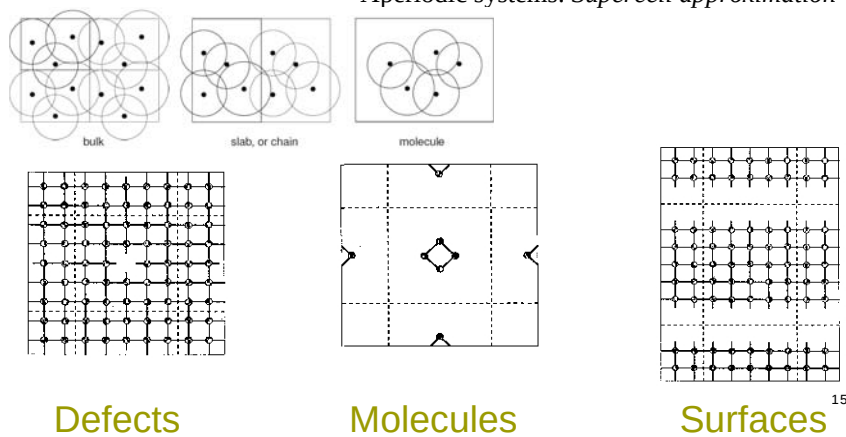
```
0.5 0.5 0.0
```

```
%endblock LatticeVectors
```

Periodic Boundary Conditions (PBC)

Atoms in the unit cell are periodically repeated throughout space along the lattice vectors

- Periodic systems and crystalline solids: $\checkmark$
- Aperiodic systems: *Supercell approximation*



Atomic Coordinates

AtomicCoordinatesFormat: format of the atomic positions in input:

Bohr: cartesian coordinates, in bohrs

Ang: cartesian coordinates, in Angstroms

ScaledCartesian: cartesian coordinates scaled to the lattice constant

Fractional: referred to the lattice vectors

`AtomicCoordinatesFormat` `Fractional`

AtomicCoordinatesAndAtomicSpecies:

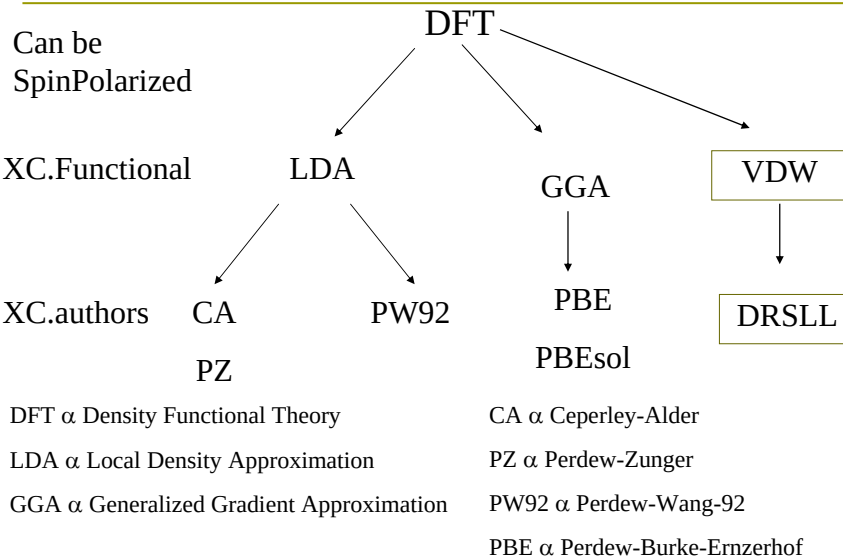
`%block AtomicCoordinatesAndAtomicSpecies`

`0.00 0.00 0.00 1`

`0.25 0.25 0.25 1`

`%endblock AtomicCoordinatesAndAtomicSpecies`

Functional



Solution method

0: Start from the atomic coordinates and the unit cell

$$\{\vec{R}\}_N \{\vec{a}\}$$

1: Compute H,S (Order N):

Hamiltonian (H), Overlap (S) matrices

$$(H - \epsilon S)C = 0$$

2: SolutionMethod

diagon

Order-N

Execution time

A general scheme: KS equations

LDA, GGA

$$E = T_o[\rho] + \int V_{ext} \rho(\vec{r}) d\vec{r} + \frac{1}{2} \int \frac{\rho(\vec{r})\rho(\vec{r}')}{|\vec{r}' - \vec{r}|} d\vec{r} d\vec{r}' + E_{xc}[\rho]$$

1-electron equations (Kohn Sham)

vary ρ

$$\left\{ -\frac{1}{2} \nabla^2 + V_{ext}(\vec{r}) + V_c(\rho(\vec{r})) + V_{xc}(\rho(\vec{r})) \right\} \Phi_i(\vec{r}) = \varepsilon_i \Phi_i(\vec{r})$$

$-\frac{Z}{r}$

$\int \frac{\rho(\vec{r}')}{|\vec{r}' - \vec{r}|} d\vec{r}'$

$\frac{\partial E_{xc}(\rho)}{\partial \rho}$

$\rho(\vec{r}) = \sum_{\varepsilon_i \leq E_F} |\Phi_i|^2$

$E_{xc}^{LDA} \propto \int \rho(r) \varepsilon_{xc}^{hom.}[\rho(r)] dr$
 $E_{xc}^{GGA} \propto \int \rho(r) F[\rho(r), \nabla \rho(r)] dr$

LDA } treats both,
exchange and correlation effects,

GGA } but approximately

k-sampling

Many magnitudes require integration of Bloch functions over Brillouin zone (BZ)

$$\rho(\vec{r}) = \sum_i \int_{BZ} d\vec{k} n(\vec{k}) |\psi_i(\vec{k})|^2$$

In practice: integral $\downarrow$ $\square$ sum over a finite uniform grid

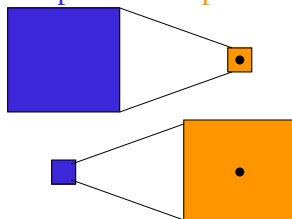
Essential for:

Small periodic systems

Metals

Magnetic systems

Real space $\square$ Reciprocal space



Good description of the Bloch states at the Fermi level

Even in same insulators:

Perovskite oxides

k-sampling

Special set of k-points: **Accurate results** for a **small #** k-points:

kgrid_cutoff (1): kgrid_cutoff 10.0 Ang

kgrid_Monkhorst_Pack (2):

%block kgrid_Monkhorst_Pack

4 0 0 0.5

0 4 0 0.5

0 0 4 0.5

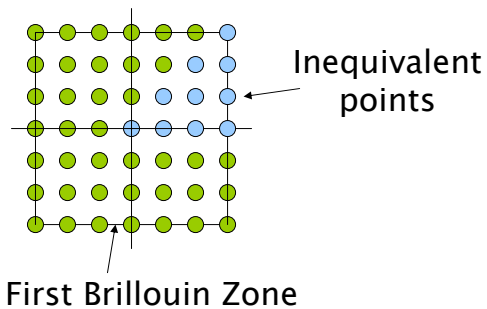
%endblock kgrid_Monkhorst_Pack

1 Moreno and Soler, PRB 45, 13891 (1992).

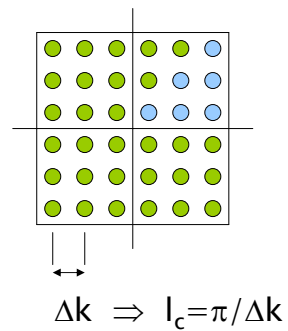
2 Monkhorst and Pack, PRB 13, 5188 (1997)

K-point sampling

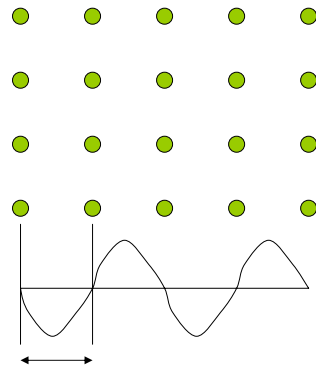
Regular k-grid



Monkhorst-Pack



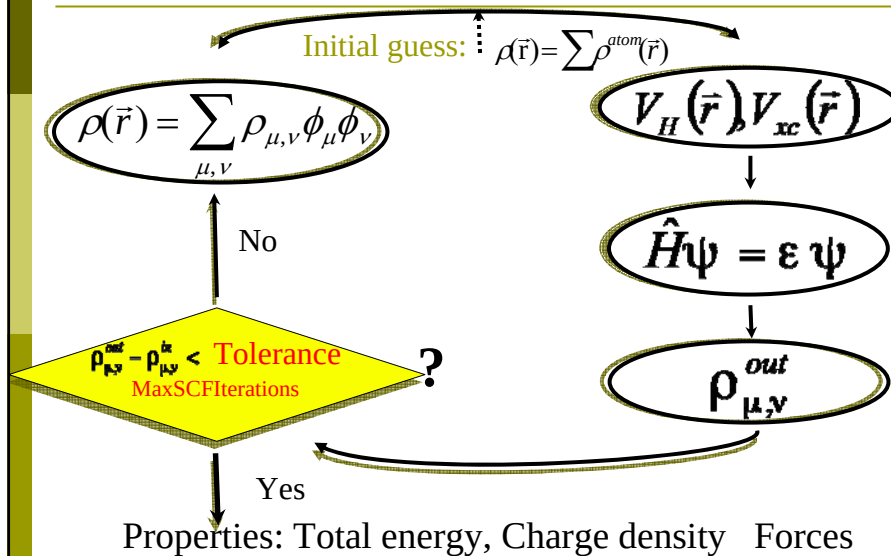
Mesh cutoff



$$\Delta x \Rightarrow k_c = \pi / \Delta x \Rightarrow E_c = \hbar^2 k_c^2 / 2m_e$$

23

Selfconsistency (SCF)



How to run Siesta

To run the serial version, from a unix/terminal:

Is siesta in your PATH?

Two options:

- 1) Set your PATH: `export PATH=$PATH:/wherever_siesta_is`
- 2) Include the full path of siesta: `/home/edu/siesta-2.0.1/bin/siesta`

Basic run, output in the screen:

```
edu@somewhere:>./siesta < Fe.fdf
```

Output redirected to a file:

```
edu@somewhere:>./siesta < Fe.fdf > Fe.out
```

Screen and file output:

```
edu@somewhere: ./siesta < Fe.fdf |tee Fe.out
```

Output: the header

```
SIESTA 1.2.3 -- [iorho parallel fix/O(N)] (Nov 20, 2001)
Architecture : lahey
Compiler flags: lf95 -O --warn --quiet --tpp --ntrace
SERIAL version
```

```
* Running in serial mode
```

```
>> Start of run: 3-JUL-2002 17:06:18
```

Output: dumping the input file

```
*: ***** Dump of input data file *****
SystemName      Water molecule
SystemLabel     h2o
NumberOfAtoms   3
NumberOfSpecies  2
%block ChemicalSpeciesLabel
  1  8  0      # Species index, atomic number, species label
  2  1  H
%endblock ChemicalSpeciesLabel
AtomicCoordinatesFormat Ang
%block AtomicCoordinatesAndAtomicSpecies
  0.000  0.000  0.000  1
  0.757  0.586  0.000  2
 -0.757  0.586  0.000  2
%endblock AtomicCoordinatesAndAtomicSpecies
*: ***** End of input data file *****
```

Output: processing the input

```
prinput: *****
coor: Atomic-coordinates input format = Cartesian coordinates
      coor:                          (in Angstroms)
redata: Number of spin components      = 1
redata: Long output                   = F
redata: Number of Atomic Species      = 2
redata: Charge density info will appear in .RHO file
redata: Write Mulliken Pop.           = NO
redata: Mesh Cutoff                   = 50.0000 Ry
redata: Net charge of the system      = 0.0000 |e|
redata: Max. number of SCF Iter       = 50
redata: Mixing is linear
redata: Mix DM in first SCF step ?    = F
redata: Write Pulay info on disk?     = F
redata: New DM Mixing Weight          = 0.2500
redata: No Kicks to SCF
redata: DM Mixing Weight for Kicks    = 0.5000
redata: DM Tolerance for SCF         = 0.000100
redata: Use continuation files for DM = F
redata: Neglect nonoverlap interactions = F
redata: Method of Calculation         = Diagonalization
redata: Electronic Temperature        = 0.0019 Ry
redata: Fix the spin of the system    = F
redata: Dynamics option               = Verlet MD run
redata: Initial MD time step          = 1
redata: Final MD time step            = 1
redata: Length of MD time step        = 1.0000 fs
redata: Length of MD time step        = 1.0000 fs
redata: Initial Temperature of MD run = 0.0000 K
redata: Perform a MD quench           = F
redata: *****
```

Output: coordinates and k-sampling

```
siesta: Atomic coordinates (Bohr) and species
siesta:      0.00000  0.00000  0.00000  1      1
siesta:      1.43052  1.10738  0.00000  2      2
siesta:     -1.43052  1.10738  0.00000  2      3

siesta: Automatic unit cell vectors (Ang):
siesta:      7.286412  0.000000  0.000000
siesta:      0.000000  5.746952  0.000000
siesta:      0.000000  0.000000  5.621012

siesta: System type = molecule
...

siesta: System type = bulk

siesta: k-grid: Number of k-points =   196
siesta: k-grid: Cutoff              =   14.021 Ang
siesta: k-grid: Supercell and displacements
siesta: k-grid:      7   0   0      0.000
siesta: k-grid:      0   7   0      0.000
siesta: k-grid:      0   0   7      0.000
```

Output: First MD step

```
siesta: =====
siesta:      Begin MD step =      1
siesta: =====

InitMesh: MESH =    32 x    30 x    24 =    23040
InitMesh: Mesh cutoff (required, used) =    50.000    50.384 Ry

* Maximum dynamic memory allocated =    3 MB

siesta: Program's energy decomposition (eV):
siesta: Eions =    815.854478
siesta: Ena   =    175.154399
siesta: Ekin  =    341.667405
siesta: Enl   =   -52.736793
siesta: DEna  =   -0.000001
siesta: DUScf =    0.000000
siesta: DUext =    0.000000
siesta: Exc   =  -109.951257
siesta: eta*DQ =    0.000000
siesta: Emadel =    0.000000
siesta: Eharris =  -466.430254
siesta: Etot  =  -461.720725
siesta: FreeEng =  -461.720725
```

Output: Self-consistency

```

siesta: iscf Eharris(eV)      E_KS(eV) FreeEng(eV)      dDmax  Ef(eV)
siesta:   1  -466.4303      -461.7207      -461.7207    1.4383 -4.2475
timer: Routine,Calls,Time,% = IterSCF           1      7.930 72.22
siesta:   2  -466.8703      -465.2425      -465.2425    0.1755 -0.1474
siesta:   3  -465.9264      -465.4655      -465.4655    0.0515 -1.5862
siesta:   4  -465.8472      -465.5656      -465.5656    0.0176 -1.9935
siesta:   5  -465.8397      -465.6346      -465.6346    0.0087 -2.1116
siesta:   6  -465.8388      -465.6857      -465.6857    0.0083 -2.1448
siesta:   7  -465.8387      -465.7240      -465.7240    0.0067 -2.1531
siesta:   8  -465.8387      -465.7527      -465.7527    0.0051 -2.1545
siesta:   9  -465.8387      -465.7742      -465.7742    0.0038 -2.1543
siesta:  10  -465.8387      -465.7903      -465.7903    0.0028 -2.1539
siesta:  11  -465.8387      -465.8024      -465.8024    0.0021 -2.1535
siesta:  12  -465.8387      -465.8115      -465.8115    0.0016 -2.1533
siesta:  13  -465.8387      -465.8183      -465.8183    0.0012 -2.1531
siesta:  14  -465.8387      -465.8234      -465.8234    0.0009 -2.1530
siesta:  15  -465.8387      -465.8272      -465.8272    0.0006 -2.1530
siesta:  16  -465.8387      -465.8301      -465.8301    0.0005 -2.1530
siesta:  17  -465.8387      -465.8322      -465.8322    0.0004 -2.1530
siesta:  18  -465.8387      -465.8338      -465.8338    0.0003 -2.1530
siesta:  19  -465.8387      -465.8351      -465.8351    0.0002 -2.1530
siesta:  20  -465.8387      -465.8360      -465.8360    0.0001 -2.1530
siesta:  21  -465.8387      -465.8367      -465.8367    0.0001 -2.1530
siesta:  22  -465.8387      -465.8372      -465.8372    0.0001 -2.1530

```

Output: Eigenvalues, forces, stress

siesta: Eigenvalues (eV):

```

ik is      eps
  1  1 -24.74 -12.70 -8.71 -6.23  1.68  4.09
      14.68 21.97 24.22 27.21 28.65 32.19
      49.89 70.65 96.18

```

siesta: Atomic forces (eV/Ang):

```

siesta:   1   0.000001  -0.504870   0.000000
siesta:   2   0.719664   0.279830   0.000000
siesta:   3  -0.719663   0.279829   0.000000
siesta: -----
siesta: Tot   0.000002   0.054788   0.000000

```

siesta: Stress tensor (eV/Ang**3):

```

siesta:  -0.012622   0.000000   0.000000
siesta:   0.000000  -0.002309   0.000000
siesta:   0.000000   0.000000   0.014000

```

Output: Total energy

siesta: Fermi energy = -2.152975 eV

siesta: Program's energy decomposition (eV):

```

siesta:-Eions = -815.854478
siesta: Ena = 175.154399
siesta: Ekin = 350.784945
siesta: Enl = -61.958840
siesta: DEna = -1.777979
siesta: DUScf = 0.727284
siesta: DUext = 0.000000
siesta: Exc = -112.912881
siesta: eta*DQ = 0.000000
siesta: Emadel = 0.000000
siesta: Ekinion = 0.000000
siesta: Eharris = -465.839084
siesta: Etot = -465.837551
siesta: FreeEng = -465.837551

```

siesta: Final energy (eV):

```

siesta: Kinetic = 350.784945
siesta: Hartree = 382.616610
siesta: Ext. field = 0.000000
siesta: Exch.-corr. = -112.912881
siesta: Ion-electron = -1072.820417
siesta: Ion-ion = -13.505807
siesta: Ekinion = 0.000000
siesta: Total = -465.837551

```

Output: timer (real and cpu times)

timer: CPU execution times:

timer:	Routine	Calls	Time/call	Tot.time	%
timer:	siesta	1	13.660	13.660	100.00
timer:	Setup	1	0.850	0.850	6.22
timer:	bands	1	0.000	0.000	0.00
timer:	KSV_init	1	0.000	0.000	0.00
timer:	IterMD	1	12.800	12.800	93.70
timer:	hsparse	2	0.005	0.010	0.07
timer:	overfsm	2	1.095	2.190	16.03
timer:	IterSCF	23	0.461	10.600	77.60
timer:	kinefsm	2	1.010	2.020	14.79
timer:	nlefsm	2	2.780	5.560	40.70
timer:	DHSCF	23	0.128	2.950	21.60
timer:	DHSCF1	1	0.060	0.060	0.44
timer:	DHSCF2	1	0.190	0.190	1.39
timer:	REORD	186	0.001	0.130	0.95
timer:	POISON	24	0.020	0.480	3.51
timer:	DHSCF3	23	0.110	2.520	18.45
timer:	rhoofd	23	0.030	0.690	5.05
timer:	CELLXC	23	0.027	0.610	4.47
timer:	vmat	23	0.018	0.410	3.00
timer:	diagon	22	0.002	0.050	0.37
timer:	rdiag	22	0.002	0.040	0.29
timer:	DHSCF4	1	0.180	0.180	1.32
timer:	dfscf	1	0.150	0.150	1.10

>> End of run: 3-JUL-2002 17:06:32

Saving and reading information (I)

Some information is stored by Siesta to restart simulations from:

- Density matrix: DM.UseSaveDM
- Localized wave functions (Order-N): ON.UseSaveLWF
- Atomic positions and velocities: MD.UseSaveXV
- Conjugent gradient history (minimizations): MD.UseSaveCG

All of them are logical variables

EXTREMELY USEFUL TO SAVE LOT OF TIME!

Saving and reading information (II)

Information needed as input for various post-processing programs, for example, to visualize:

- Total charge density: SaveRho
- Deformation charge density: SaveDeltaRho
- Electrostatic potential: SaveElectrostaticPotential
- Total potential: SaveTotalPotential
- Local density of states: LocalDensityOfStates
- Charge density contours: WriteDenchar
- Atomic coordinates: WriteCoorXmol and WriteCoorCeri

All of them are logical variables

Analyzing the electronic structure (I)

•**Band structure** along the high symmetry lines of the BZ

BandLineScale: scale of the k vectors in BandLines

BandLineScale pi/a

BandLines: lines where band energies are calculated.

%block BandLines

1 1.000 1.000 1.000 L

20 0.000 0.000 0.000 \Gamma

25 2.000 0.000 0.000 X

30 2.000 2.000 2.000 \Gamma

%endblock BandLines

Analyzing the electronic structure (II)

•**Density of states**: total and projected on the atomic orbitals

- Compare with experimental spect

- Bond formation

- Defined as:

$$g(\epsilon) = \sum_i \sum_{\mathbf{k}} \delta(\epsilon - \epsilon_i(\mathbf{k}))$$

$$\simeq \sum_i \sum_{\mathbf{k}} \frac{1}{\sigma\sqrt{\pi}} \exp\left(-\frac{(\epsilon - \epsilon_i(\mathbf{k}))^2}{\sigma^2}\right)$$

ProjectedDensityOfStates:

%block ProjectedDensityOfStates

-20.00 10.00 0.200 500 eV

%endblock ProjectedDensityOfStates

Analyzing the electronic structure (III)

•Population analysis: Mulliken prescription

- Amounts of charge on an atom or in an orbital inside the atom
- Bond formation
- Be careful, very dependent on the basis functions

WriteMullikenPop

WriteMullikenPop 0 = None

1 = Atomic and orbitals charges

2 = 1 + atomic overlap pop.

3 = 2 + orbital overlap pop.

References

- Siesta Caminando by Andrei Postnikov
- Eduardo Anglada's Lectures