PSML pseudopotential format

How to generate PSML pseudopotentials

and

run SIESTA and ABINIT with the same pseudo

Javier Junquera
Yann Pouillon

Matthieu Verstraete

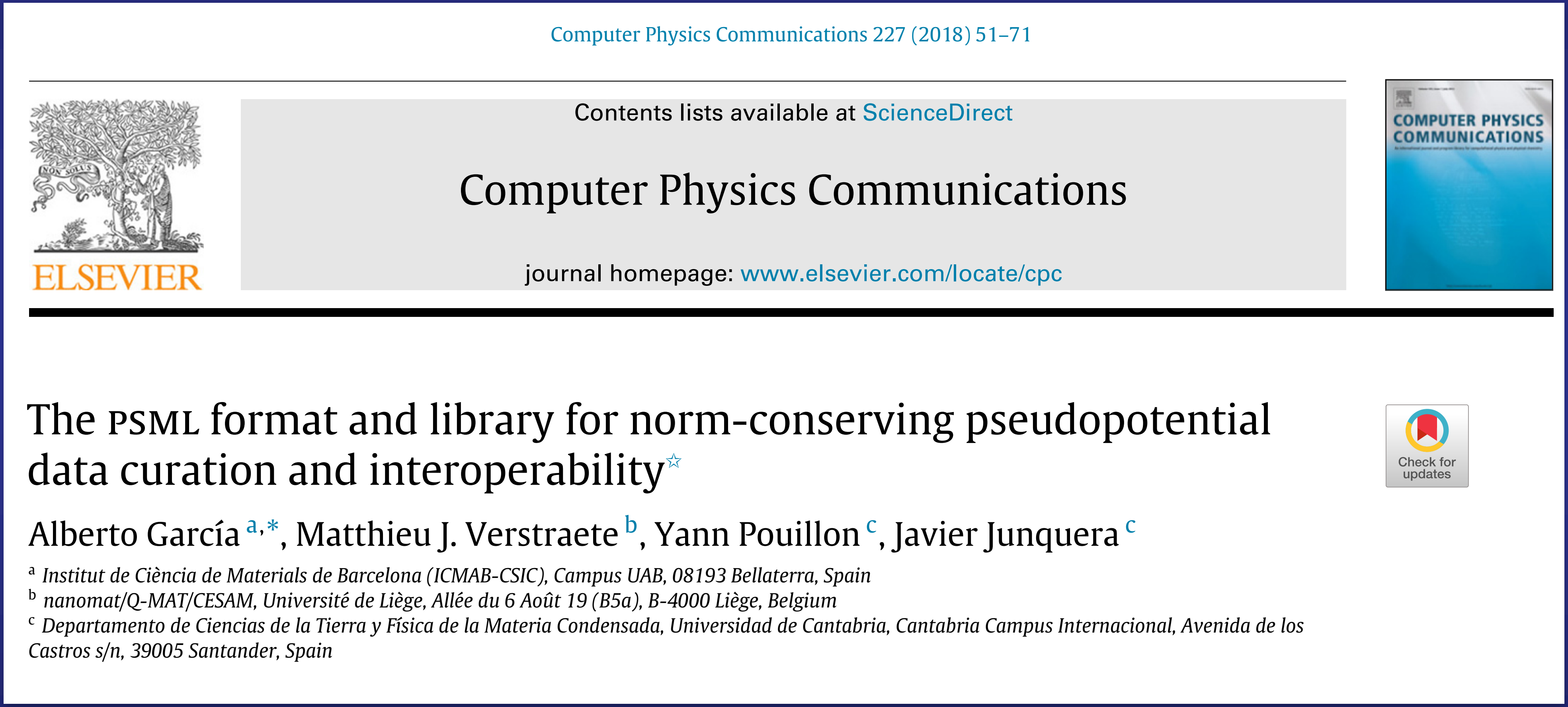
Alberto García

UC UNIVERSIDAD DE CANTABRIA

ULg UNIVERSITÉ de Liège

ICMAB INSTITUT DE CIÈNCIA DE MATERIALS DE BARCELONA
EXCELENCIA SEVERO OCHOA
CSIC CONSEJO SUPERIOR DE INVESTIGACIONES CIENTÍFICAS

Main reference



Computer Physics Communications 227 (2018) 51–71

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

ELSEVIER

The PSML format and library for norm-conserving pseudopotential data curation and interoperability[☆]

Alberto García[a,*], Matthieu J. Verstraete[b], Yann Pouillon[c], Javier Junquera[c]

[a] Institut de Ciència de Materials de Barcelona (ICMAB-CSIC), Campus UAB, 08193 Bellaterra, Spain
[b] nanomat/Q-MAT/CESAM, Université de Liège, Allée du 6 Août 19 (B5a), B-4000 Liège, Belgium
[c] Departamento de Ciencias de la Tierra y Física de la Materia Condensada, Universidad de Cantabria, Cantabria Campus Internacional, Avenida de los Castros s/n, 39005 Santander, Spain

Outline of the Tutorial

1. How to compile the different codes

Two codes to generate pseudopotentials:

ATOM
(http://www.icmab.es/siesta/Pseudopotentials)

ONCVPSP
(http://www.quantum-simulation.org/potentials/sg15_oncv/)

Two client Solid State Physic codes

SIESTA
(http://www.icmab.es/siesta)

ABINIT
(http://www.abinit.org)

2. How to generate the psml pseudopotentials with ONCVPSP and ATOM

3. How to run SIESTA and ABINIT with the same pseudopotentials

   Test of the convergence of a numerical atomic orbital basis set with respect to the asymptotic limit of a converged basis of plane waves

   Compute the equation-of-state (energy versus volume profiles) for elemental crystals, a test that has been proposed as a benchmark for the comparison of different codes

Preliminary notes in the installation of the libraries and codes

Installation processes might change from one platform to another, or be dependent on the compiler

For the sake of simplicity, we shall assume that all the required libraries will be compiled locally in a directory lib, directly beneath the $HOME directory

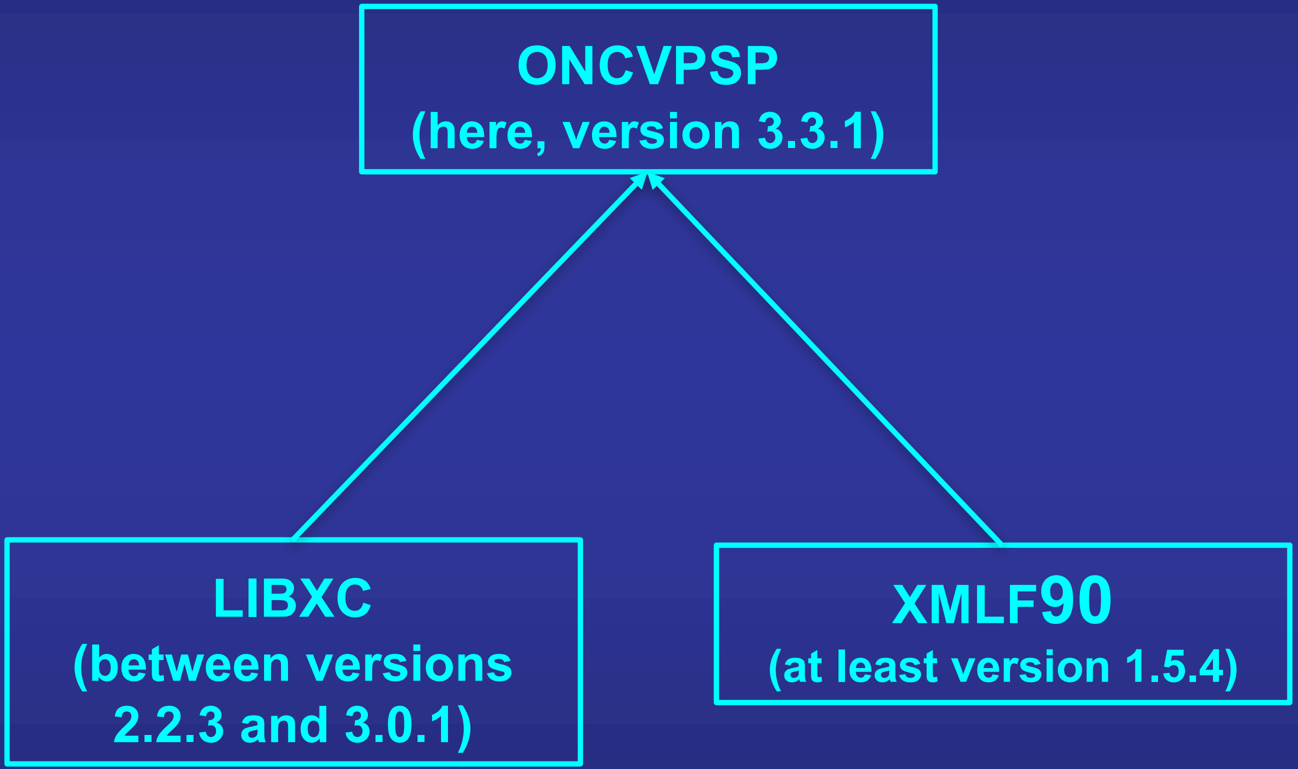
```
$cd $HOME
$mkdir lib
```

All procedures described here have been tested on a Mac with gfortran compiler Might be you have to change them slightly to accomodate to your platform.

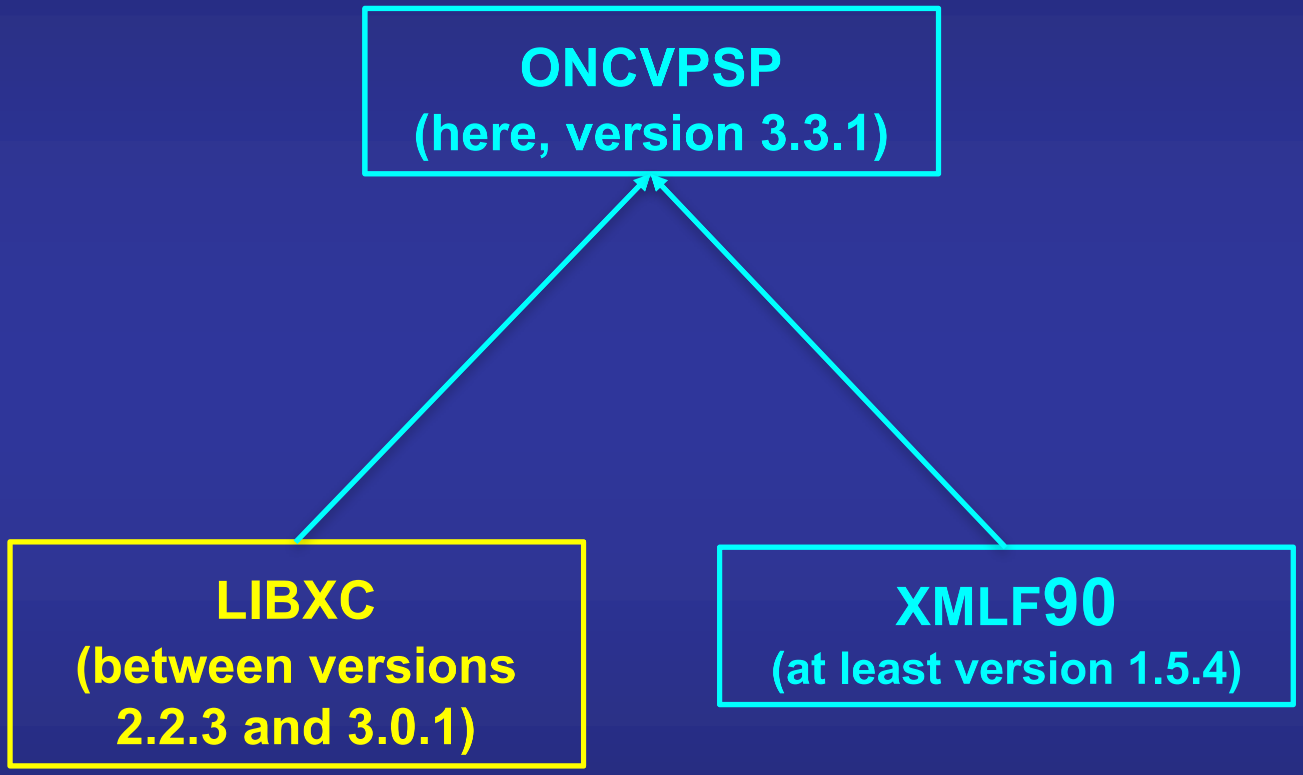
Consult your local computer administrator in case you require some extra help

Installation of ONCVPSP in order to generate PSML files

Installation of ONCVPSP in order to generate PSML files Dependence on other libraries



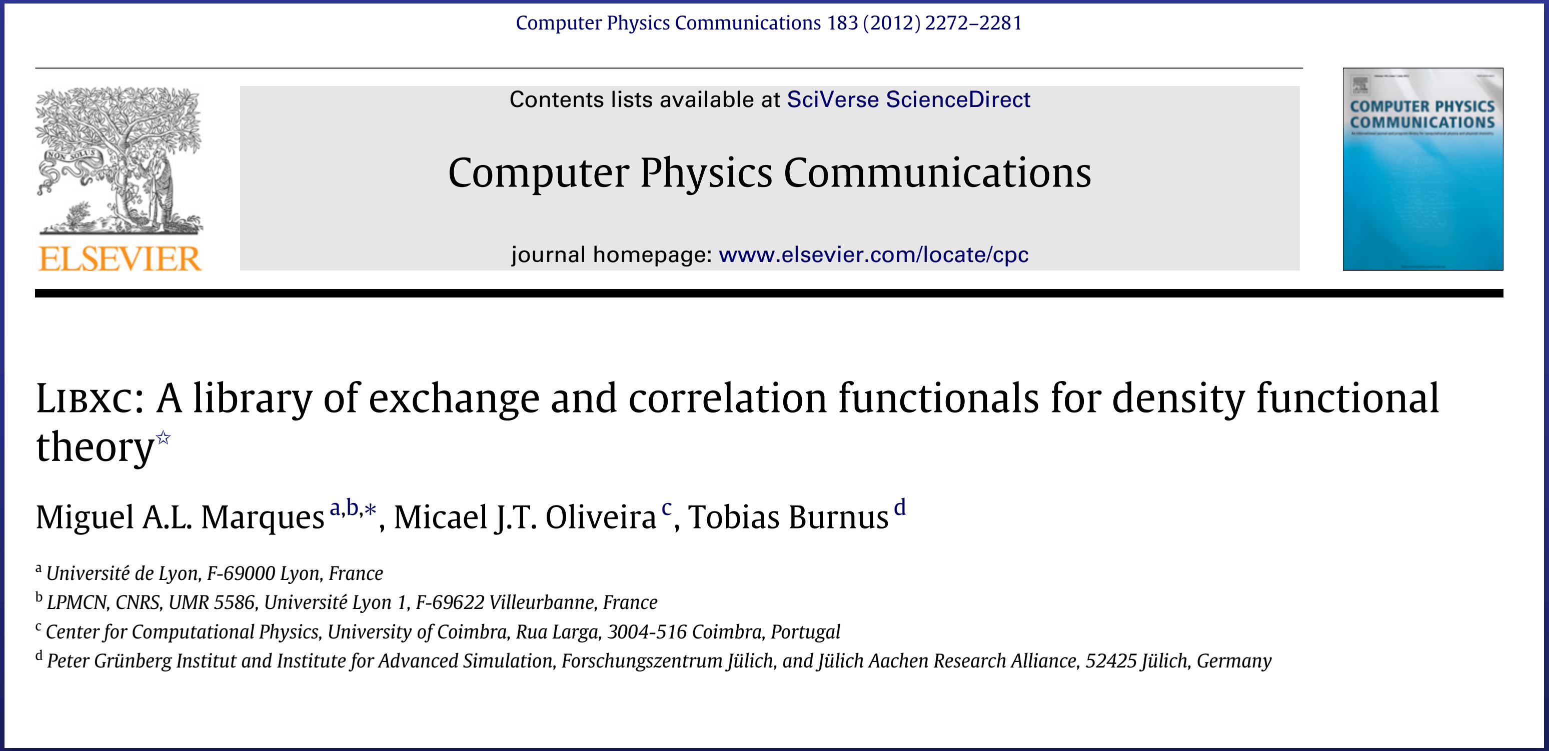
Installation of ONCVPSP in order to generate PSML files
Dependence on other libraries



Preliminaries: installation of required libraries: LIBXC

LIBXC is a library of exchange-correlation functionals for density-functional theory

The aim is to provide a portable, well tested and reliable set of exchange and correlation functionals that can be used by many electronic structure codes



Computer Physics Communications 183 (2012) 2272–2281

Contents lists available at SciVerse ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

ELSEVIER

LIBXC: A library of exchange and correlation functionals for density functional theory☆

Miguel A.L. Marques[a,b,*], Micael J.T. Oliveira[c], Tobias Burnus[d]

[a] Université de Lyon, F-69000 Lyon, France
[b] LPMCN, CNRS, UMR 5586, Université Lyon 1, F-69622 Villeurbanne, France
[c] Center for Computational Physics, University of Coimbra, Rua Larga, 3004-516 Coimbra, Portugal
[d] Peter Grünberg Institut and Institute for Advanced Simulation, Forschungszentrum Jülich, and Jülich Aachen Research Alliance, 52425 Jülich, Germany

Preliminaries: installation of required libraries:

LIBXC

For download and installation, simply visit:

http://octopus-code.org/wiki/Libxc:download

Preliminaries: installation of required libraries:

LIBXC

Installation Option 1:
From the source

```
$ tar -xvf libxc-3.0.1.tar
$ cd libxc-3.0.1
$ mkdir Gfortran
$ cd Gfortran
$ ../configure --prefix=$HOME/lib/Gfortran --enable-fortran
$ make -j4
$ make install
```

The libraries libxc.a and libxcf90.a will be automatically included in $HOME/lib/Gfortran/lib

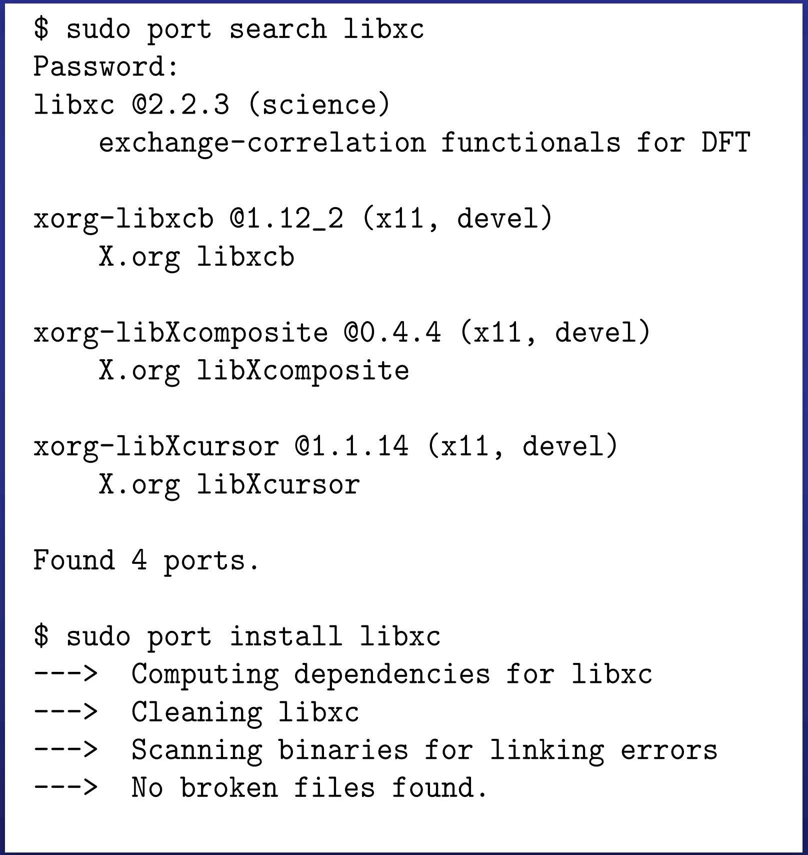
Preliminaries: installation of required libraries:

LIBXC

In Mac platforms, libxc is already included in the macports.
For installation, simply type:

```
$sudo port selfupdate
$sudo port search libxc
$sudo port install libxc
```

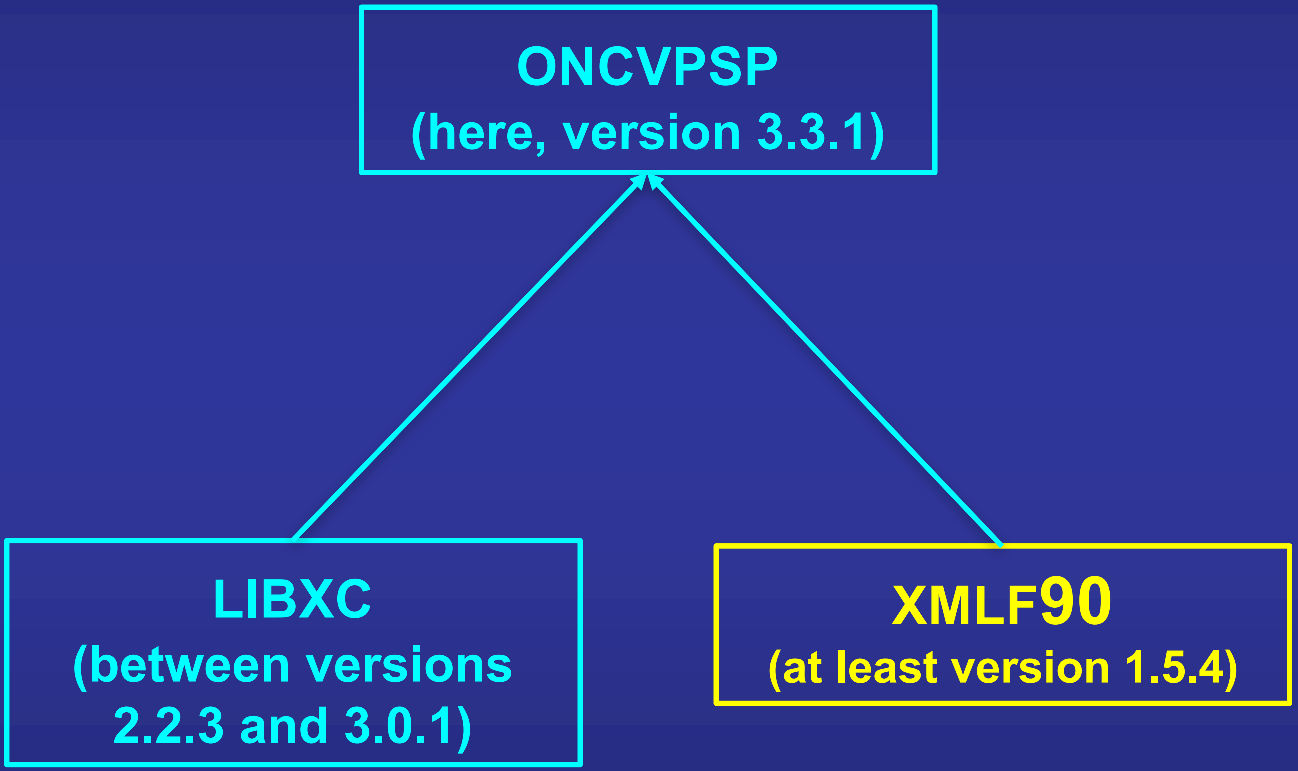
Installation Option 2:
From macports



```
$ sudo port search libxc
Password:
libxc @2.2.3 (science)
    exchange-correlation functionals for DFT

xorg-libxcb @1.12_2 (x11, devel)
    X.org libxcb

xorg-libXcomposite @0.4.4 (x11, devel)
    X.org libXcomposite

xorg-libXcursor @1.1.14 (x11, devel)
    X.org libXcursor

Found 4 ports.

$ sudo port install libxc
--->  Computing dependencies for libxc
--->  Cleaning libxc
--->  Scanning binaries for linking errors
--->  No broken files found.
```

At least in my mac, the libraries libxc.a and libxcf90.a are included in /opt/local/lib
The installation point might change from one platform to another

Installation of ONCVPSP in order to generate PSML files Dependence on other libraries



Preliminaries: installation of required libraries: XMLF90

XMLF90 is a suite of libraries to handle XML in Fortran.
It has two major components:

- A XML parsing library. The parser was designed to be a useful tool in the extraction and analysis of data in the context of scientific computing, and thus the priorities were efficiency and the ability to deal with very large XML files while maintaining a small memory footprint. The most complete programming interface is based on the very successful SAX (Simple API for XML) model, although a partial DOM interface and a very experimental XPATH interface are also present.

- A library (xmlf90-wxml) that facilitates the writing of well-formed XML, including such features as automatic start-tag completion, attribute pretty-printing, and element indentation. There are also helper routines to handle the output of numerical arrays.

Credits to Alberto García,

Download and compile the latest version of XMLF90 library, required to dump output in PSML format

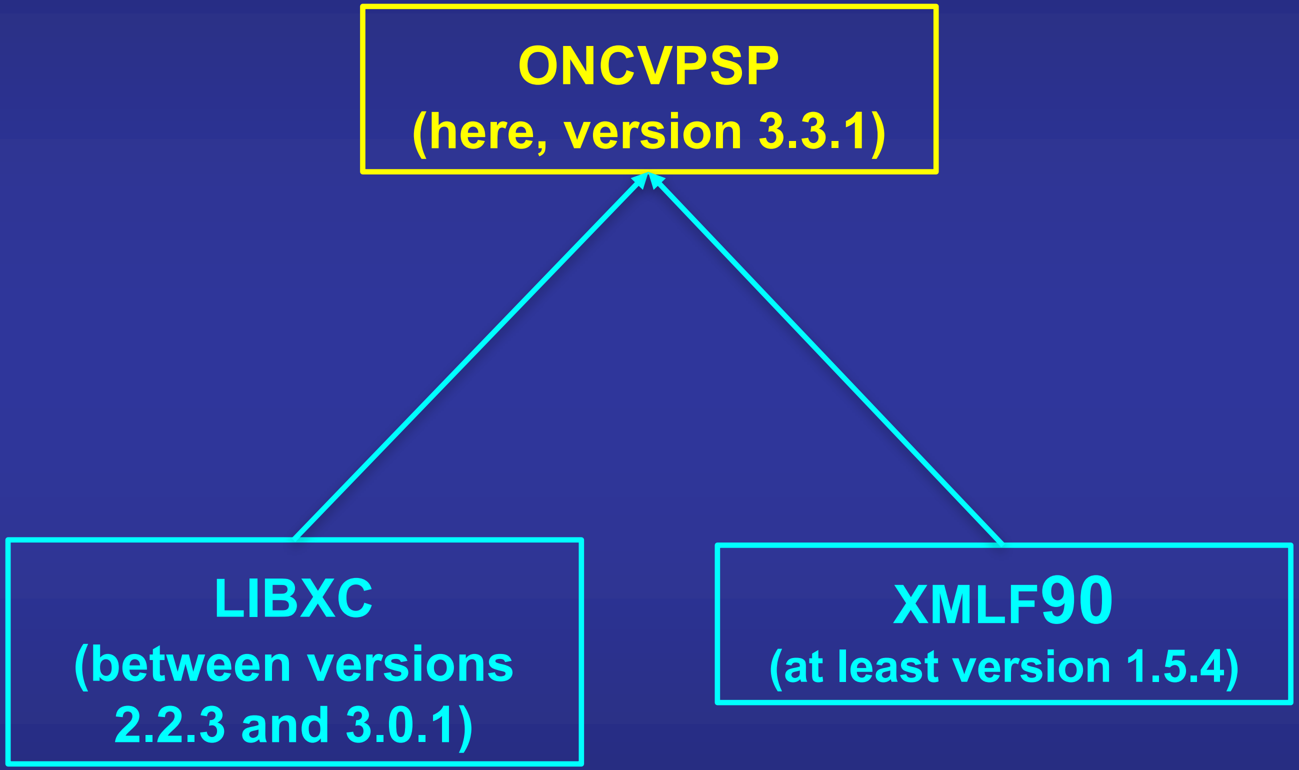
Download the latest version (minimum, version 1.5.4) from https://launchpad.net/xmlf90

Move xmlf90-1.5.4.tar (or more recent version) tar file to $HOME/lib

```
$ tar -xvf xmlf90-1.5.4.tar
$ cd xmlf90-1.5.4
$ mkdir Gfortran
$ cd Gfortran
$ ../configure --prefix=$HOME/lib/Gfortran
$ make -j4
$ make install
```

The library libxmlf90.a will bewill be automatically included in $HOME/lib/Gfortran/lib

Installation of ONCVPSP in order to generate PSML files Dependence on other libraries



ONCVPSP (here, version 3.3.1)
LIBXC (between versions 2.2.3 and 3.0.1)
XMLF90 (at least version 1.5.4)

Download the latest versions of the patcher to ONCVPSP in order to generate PSML files

Go to:

https://launchpad.net/pspgenpatch

And click to download the latest version. The examples below have been produced with

patcher--oncvpsp-3.3.1--psml-3.3.1-75.tgz

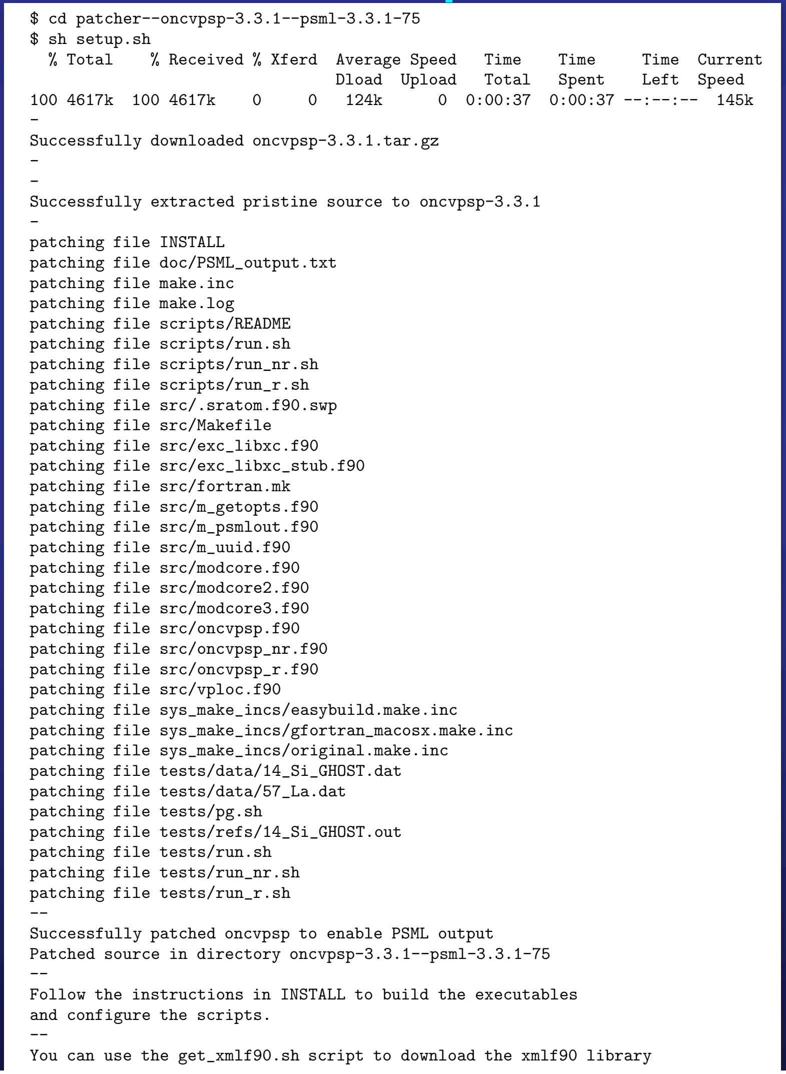
Copy the patcher to the $HOME/lib directory
Unpack the patch and enter into the directory where it is included

```
$ tar -xvf patcher--oncvpsp-3.3.1--psml-3.3.1-75.tar
patcher--oncvpsp-3.3.1--psml-3.3.1-75/
patcher--oncvpsp-3.3.1--psml-3.3.1-75/get_xmlf90.sh
patcher--oncvpsp-3.3.1--psml-3.3.1-75/oncvpsp-3.3.1--psml-3.3.1-75.patch
patcher--oncvpsp-3.3.1--psml-3.3.1-75/README
patcher--oncvpsp-3.3.1--psml-3.3.1-75/setup.sh
```

The number of the version will change in the future

Download the latest version of ONCVPSP and apply the patches

Download ONCVPSP code and apply the patches.
An automatic script will do both things for you.
Simply type:
$sh setup.sh



```
$ cd patcher--oncvpsp-3.3.1--psml-3.3.1-75
$ sh setup.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 4617k  100 4617k    0     0   124k      0  0:00:37  0:00:37 --:--:--  145k
-
Successfully downloaded oncvpsp-3.3.1.tar.gz
-
-
Successfully extracted pristine source to oncvpsp-3.3.1
-
patching file INSTALL
patching file doc/PSML_output.txt
patching file make.inc
patching file make.log
patching file scripts/README
patching file scripts/run.sh
patching file scripts/run_nr.sh
patching file scripts/run_r.sh
patching file src/.sratom.f90.swp
patching file src/Makefile
patching file src/exc_libxc.f90
patching file src/exc_libxc_stub.f90
patching file src/fortran.mk
patching file src/m_getopts.f90
patching file src/m_psmlout.f90
patching file src/m_uuid.f90
patching file src/modcore.f90
patching file src/modcore2.f90
patching file src/modcore3.f90
patching file src/oncvpsp.f90
patching file src/oncvpsp_nr.f90
patching file src/oncvpsp_r.f90
patching file src/vploc.f90
patching file sys_make_incs/easybuild.make.inc
patching file sys_make_incs/gfortran_macosx.make.inc
patching file sys_make_incs/original.make.inc
patching file tests/data/14_Si_GHOST.dat
patching file tests/data/57_La.dat
patching file tests/pg.sh
patching file tests/refs/14_Si_GHOST.out
patching file tests/run.sh
patching file tests/run_nr.sh
patching file tests/run_r.sh
--
Successfully patched oncvpsp to enable PSML output
Patched source in directory oncvpsp-3.3.1--psml-3.3.1-75
--
Follow the instructions in INSTALL to build the executables
and configure the scripts.
--
You can use the get_xmlf90.sh script to download the xmlf90 library
```

Download the latest version of ONCVPSP and apply the patches

```
$ mv  oncvpsp-3.3.1--psml-3.3.1-75 ..
$ cd ../oncvpsp-3.3.1--psml-3.3.1-75/
$ vi make.inc
```

Edit the make.inc file of ONCVPSP

```
# System-dependent makefile options for ONCVPSP
# This must be carefully edited before executing "make" in src
#
# Copyright (c) 1989-2015 by D. R. Hamann, Mat-Sim Research LLC and Rutgers
# University

##### Edit the following lines to correspond to your compilers ####

F77        = gfortran
F90        = gfortran
CC         = gcc
FCCPP      = cpp

FLINKER     = $(F90)

FCCPPFLAGS = -ansi -DLIBXC_VERSION=203  #This probably should not be changed

##### Edit the following optimization flags for your system ####

FFLAGS     = -O3 -ffast-math -funroll-loops
CFLAGS     = -O3

##### Edit the following LAPACK and BLAS library paths for your system ####

LIBS = -L/opt/local/lib -llapack -lcblas -latlas -lopenblas

#####
# The xmlf90 library (its wxml subsystem) is needed to generate XML.
# You can download xmlf90 from http://launchpad.net/xmlf90
# Put the correct path here
#
XMLF90_ROOT=$(HOME)/lib/Gfortran
LIBS += -L$(XMLF90_ROOT)/lib -lxmlf90
INC += -I$(XMLF90_ROOT)/include

##### Edit the following for to use libxc if available #####

# oncvpsp is compatible with libxc
# To build oncvpsp with libxc, uncomment 3 of the following lines and edit
# the paths to point to your libxc library and include directories
# make clean in src before rebuilding after changing this

##for libxc 2.1.0 and later use
LIBS += -L$(HOME)/lib/Gfortran/lib -lxcf90 -lxc
FFLAGS += -I$(HOME)/lib/Gfortran/include
OBJS_LIBXC =    functionals.o exc_libxc.o

##for earlier releases use
#LIBS += -L/home/drh/abinit/fallbacks/exports/lib -lxc

# Otherwise, use only the following line
#OBJS_LIBXC =   exc_libxc_stub.o
```

Select your compiler

Point to your linear algebra libraries

Point to the directory where we have installed XMLF90

Point to the directory where we have installed LIBXC

To compile ONCVPSP and run the tests

Simply type

```
$ make
```

To check the results of the tests, go to

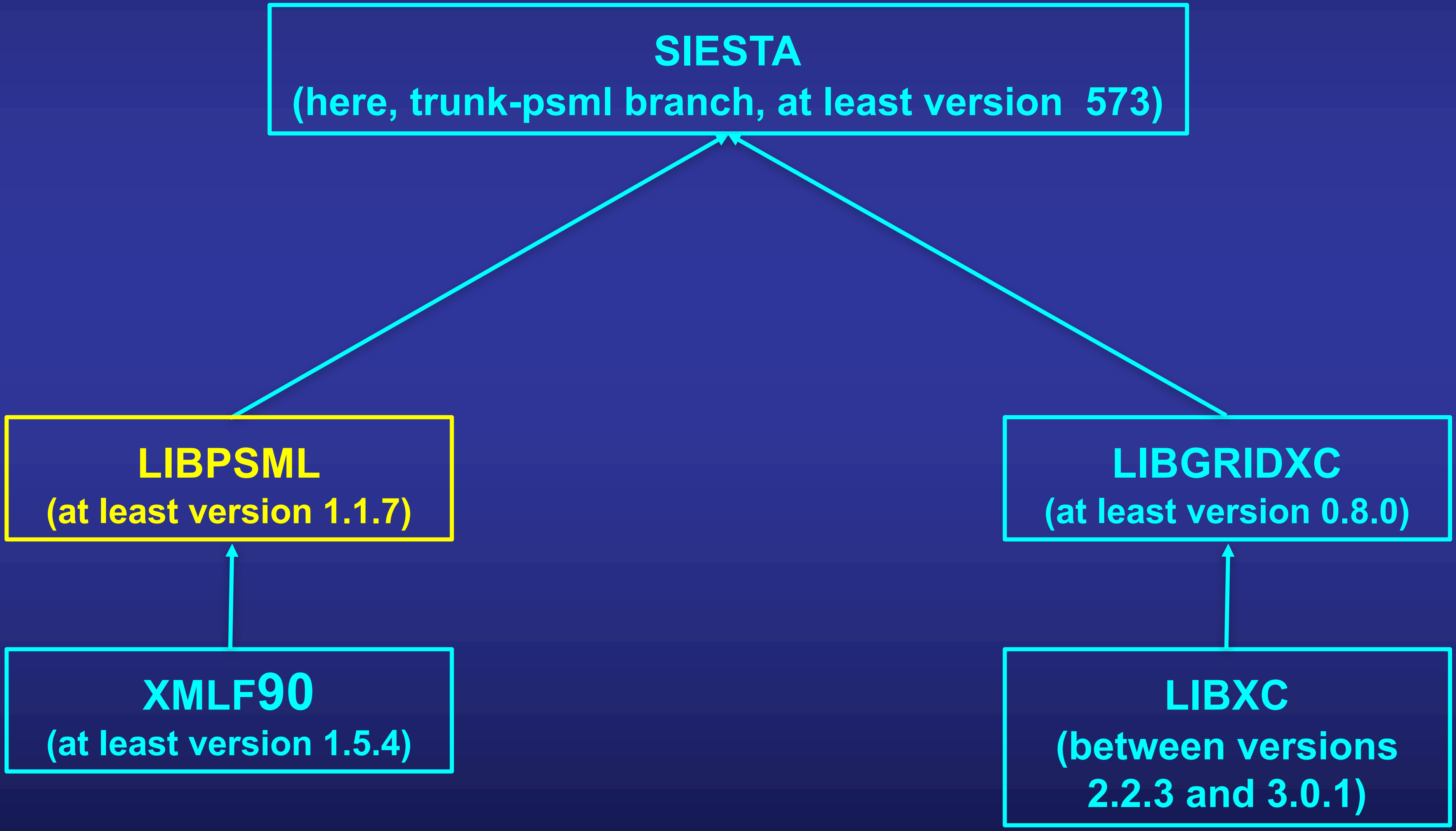
```
$ cd tests/data
```

And edit

```
$ vi TEST.reports
```

Installation of SIESTA in order to use PSML files

Installation of SIESTA in order to read PSML files
Dependence on other libraries



Preliminaries: installation of required libraries: LIBPSML

LIBPSML is a library that provides an API for parsing PSML files, including accessors for the relevant data pieces, as well as a set of routines for generating well-formed PSML.

Download the latest version of the PSML library from launchpad

Go to https://launchpad.net/libpsml
and download the tar file from the right hand side bar
Copy the .tar file that you download in the $HOME/lib directory

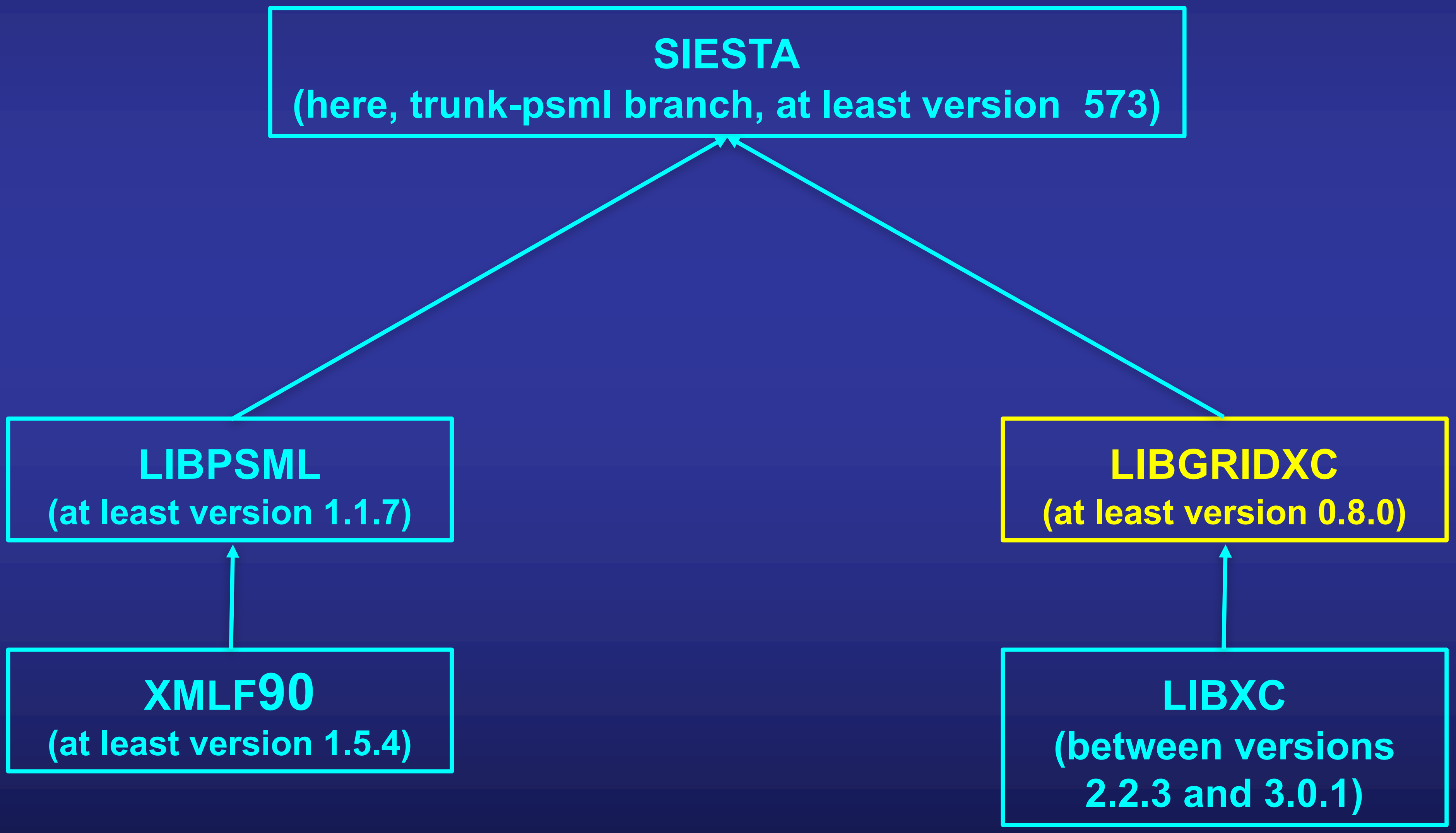
To compile the latest version of the PSML library

```
$ tar -xvf libpsml-1.1.7.tar
$ cd libpsml-1.1.7
$ mkdir Gfortran
$ cd Gfortran
$ ../configure --prefix=$HOME/lib/Gfortran --with-xmlf90=$HOME/lib/Gfortran
$ make -j4
$ make install
```

Point to the directory where the other libraries are installed

Then, the library libpsml.a should be in the directory $HOME/lib/Gfortran/lib

Installation of SIESTA in order to read PSML files
Dependence on other libraries



Preliminaries: installation of required libraries:

LIBGRIDXC

LIBGRIDXC is a library to compute the exchange and correlation energy and potential in spherical (i.e. an atom) or periodic systems.

LIBGRIDXC has replaced the former SiestaXC library.
It has been included in the Electronic Structure Library.

Download the latest version of LIBGRIDXC library

Go to:

https://launchpad.net/libgridxc

And click to download the latest version. The examples below have been produced with

libgridxc-0.8.0.tgz

Copy the tarball file in the lib directory, and unpack it

```
$ tar -xvf libgridxc-0.8.0.tgz
$ cd libgridxc-0.8.0
$ mkdir Gfortran
$ cd Gfortran
$ cp ../extra/fortran.mk .
$ vi fortran.mk
```

Compile the latest version of LIBGRIDXC library

Here we shall assume that we are linking against an existing version of the LIBXC library previously installed

```
#
# Example Fortran macros: gfortran
#
# Make sure this variable is set if you intend to use libxc
LIBXC_ROOT=$(HOME)/lib/Gfortran
#
# These two instances are needed, instead of just FC
FC_SERIAL=gfortran
FC_PARALLEL=mpifort
#

FFLAGS= -O2  -fimplicit-none
FFLAGS_DEBUG= -g -O0 -fbacktrace -fcheck=all -fimplicit-none
FFLAGS_CHECKS= -g  -O0 -g -Wall -Wextra -Warray-temporaries \
              -Wconversion -fimplicit-none -fbacktrace \
              -ffree-line-length-0 -fcheck=all \
              -ffpe-trap=zero,overflow,underflow -finit-real=nan
LDFLAGS=
#
AR=ar
RANLIB=ranlib
#
DEFS_PREFIX=
INC_PREFIX= -I
MOD_PREFIX= -I
MOD_EXT=.mod
#
.F.o:
	$(FC) -c $(FFLAGS) $(INCFLAGS)  $(FPPFLAGS) $<
.f.o:
	$(FC) -c $(FFLAGS) $(INCFLAGS)   $<
.F90.o:
	$(FC) -c $(FFLAGS) $(INCFLAGS)  $(FPPFLAGS) $<
.f90.o:
	$(FC) -c $(FFLAGS) $(INCFLAGS)   $<
#
```

Point to the directory where we have installed LIBXC

Compile the latest version of LIBGRIDXC library

Here we shall assume that we are installing the serial version, and linking against an existing version of the LIBXC library previously installed

If you want to compile only a serial version of the library then

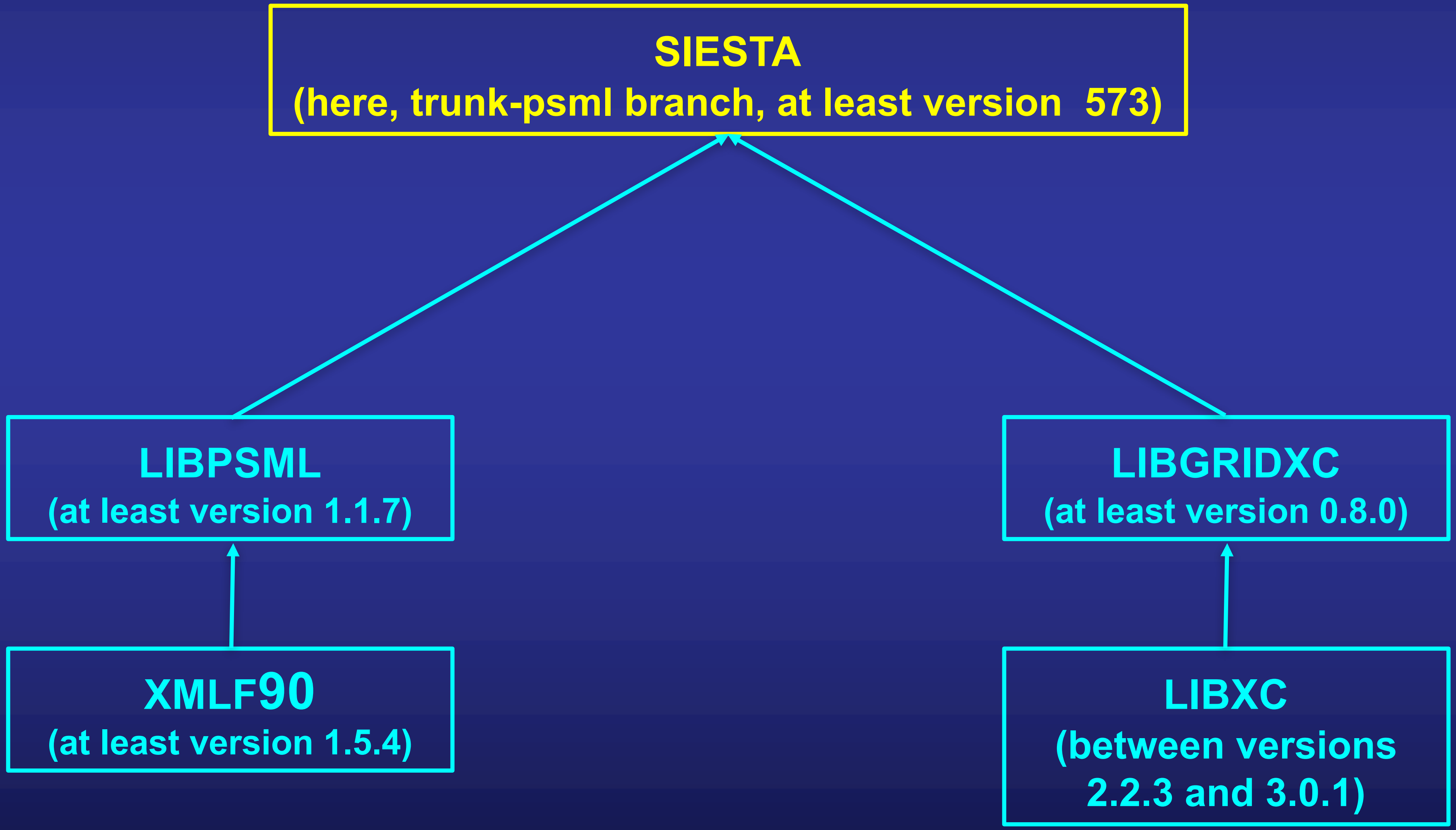
```
$ sh ../Src/config.sh
 *** Compilation setup done.
$ make clean
$ WITH_LIBXC=1 WITH_MPI= PREFIX=$HOME/lib/Gfortran sh build.sh
```

If you want to compile both serial and parallel version of the library then

```
$ sh ../Src/config.sh
 *** Compilation setup done.
$ make clean
$ WITH_LIBXC=1 WITH_MPI=1 PREFIX=$HOME/lib/Gfortran sh build.sh
```

The library libGridXC will then be installed in the directory
$HOME/lib/Gfortran/serial/lib/libGridXC.a
And/or
$HOME/lib/Gfortran/mpi/lib/libGridXC.a

Installation of SIESTA in order to read PSML files
Dependence on other libraries



Download the latest version of TRUNK-PSML branch of siesta from

Go to:

https://launchpad.net/siesta/psml-support

And download the version of siesta
siesta-psml-r0.tgz

To compile the latest version of the SIESTA compatible with PSML

Add the following lines to the usual arch.make in the Obj directory
(as a starting point, you can take the template gfortran.make included in Obj)

```
#
# --- Edit the location of your psml files
#
ROOT_GLOBAL=$(HOME)/lib/Gfortran
PSML_ROOT=$(ROOT_GLOBAL)
XMLF90_ROOT=$(ROOT_GLOBAL)
GRIDXC_ROOT=$(ROOT_GLOBAL)/serial (or mpi)
LIBXC_ROOT=$(ROOT_GLOBAL)
```

Point to the directory where PSML was installed

Point to the directory where XMLF90 was installed

Point to the directory where LIBGRIDXC was installed

Point to the directory where LIBXC was installed

Then, type

$ make

The proper location of the libraries will be done at compilation time…

```
--- reference section
PSML-related libs: /Users/javier/lib/Gfortran/lib/libpsml.a
XMLf90 libs: /Users/javier/lib/Gfortran/lib/libxmlf90.a
GRIDXC libs: /Users/javier/lib/Gfortran/serial/lib/libGridXC.a -L/Users/javier/lib/Gfortran/lib -l xcf90 -l xc
--- end of reference section
```

Installation of ATOM in order to generate PSML files

Download the lastest version of the ATOM code to generate the pseudopotentials in PSML format

Regarding the ATOM code, if you are an academic user, can be downloaded from:

https://departments.icmab.es/leem/siesta

And follow the link to Pseudopotentials.
Then, if you qualify, accept that you are an academic user
The version we shall compile here is atom-4.2.6.tgz
Copy this tar ball to the lib directory and unpack it

```
$cp atom-4.2.6.tgz $HOME/lib
```

To compile it, you need the XMLF90 and LIBGRIDXC libraries, but both of them have been previously compiled, and we have prepare the arch.make invoking them at the time of the SIESTA compilation.
So we can use the same arch.make file we used for SIESTA

```
$ tar -xvf atom-4.2.6.tgz
$ cd atom-4.2.6
$ cp <your_path_to_siesta_dir>/Obj/arch.make .
$ make
```

Installation of PSOP in order to generate fully non-local pseudopotentials from semilocal pseudopotentials in PSML

Compilation of PSOP

Finally, we need to install a standalone program (PSOP) to generate, from a pseudopotential file holding semilocal-potential information, a full non-local operator in the classic SIESTA style (special local part, plus Kleynman-Bylander projectors).
The information is produced in XML, in a form compatible with the PSML format.

This program is included in the SIESTA tree

```
$ cd <your_path_to_siesta_dir>/Pseudo/vnl-operator
$ make OBJDIR=Obj
```

Installation of ABINIT in order to read PSML files

Download abinit, at least version 8.8.1, from
http://www.abinit.org

After untar the package, type

```
$ cd abinit-8.8.1
$ mkdir Gfortran
$ cd Gfortran
$ vi psml.ac
```

This file contains instructions to configure and compile abinit with PSML.
Type the content of the box below in a file and call it psml.ac

The psml.ac file should look like something as
(we assume a serial compilation)

```
CC="gcc"
FC="gfortran"
CXX="g++"

FCFLAGS_EXTRA="-Wa,-q"
CFLAGS_EXTRA="-Wa,-q"

with_trio_flavor="psml"
with_dft_flavor="libxc"

ROOT_GLOBAL="$HOME/lib/Gfortran"
with_psml_incs="-I$ROOT_GLOBAL/include"
with_psml_libs="-L$ROOT_GLOBAL/lib -lpsml -lxmlf90"
with_libxc_incs="-I$ROOT_GLOBAL/include"
with_libxc_libs="-L$ROOT_GLOBAL/lib -lxcf90 -lxc"

with_linalg_libs="-L/usr/local/lib/ -llapack -lblas"

prefix="$ROOT_GLOBAL"
```

Point to your compilers

Add the compilation flags

Tell ABINIT that we will link against PSML and LIBXC (previously compiled)

Point to the corresponding libraries

Point to the linear algebra libraries

Define the prefix or directory where the abinit executable will be finally stored

Installation of ABINIT in order to read PSML files

To compile and run the tests, type

```
$ ../configure --with-config-file=psml.ac
$ make -j4
$ make install
$ cd tests
$ ../../tests/runtests.py -j4 psml
```

Outline of the Tutorial

1. How to compile the different codes

Two codes to generate pseudopotentials:

ATOM
(http://www.icmab.es/siesta/Pseudopotentials)

ONCVPSP
(http://www.quantum-simulation.org/potentials/sg15_oncv/)

Two client Solid State Physic codes

SIESTA
(http://www.icmab.es/siesta)

ABINIT
(http://www.abinit.org)

2. How to generate the psml pseudopotentials with ONCVPSP and ATOM

3. How to run SIESTA and ABINIT with the same pseudopotentials

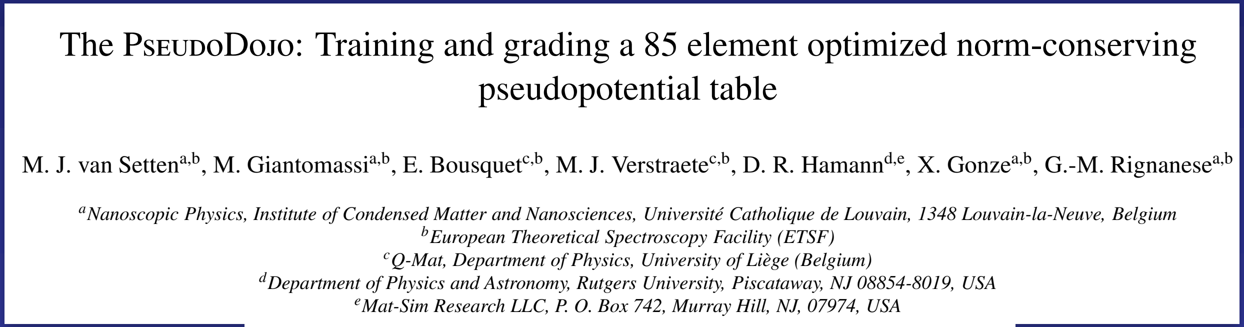
Test of the convergence of a numerical atomic orbital basis set with respect to the asymptotic limit of a converged basis of plane waves

Compute the equation-of-state (energy versus volume profiles) for elemental crystals, a test that has been proposed as a benchmark for the comparison of different codes

How to generate a PSML pseudopotential with ONCVPSP

How to generate a pseudopotential with ONCVPSP in PSML format

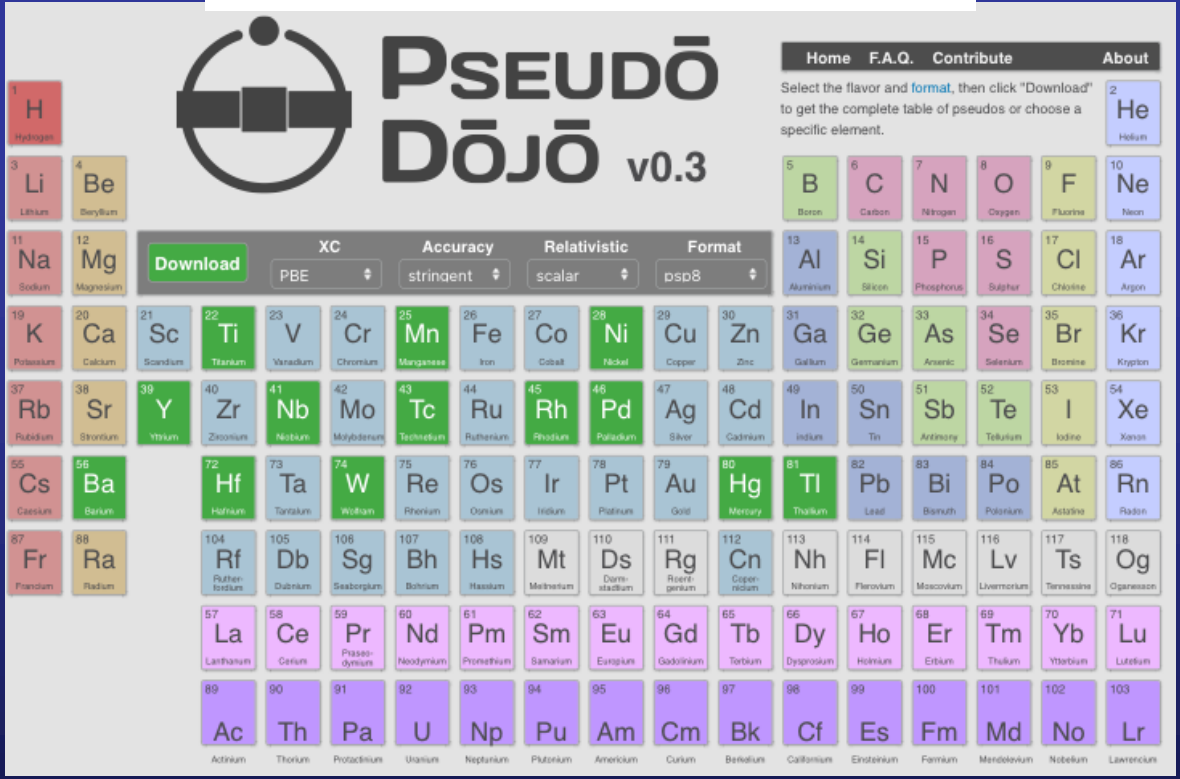
http://www.pseudo-dojo.org


The PSEUDODOJO: Training and grading a 85 element optimized norm-conserving pseudopotential table

M. J. van Setten[a,b], M. Giantomassi[a,b], E. Bousquet[c,b], M. J. Verstraete[c,b], D. R. Hamann[d,e], X. Gonze[a,b], G.-M. Rignanese[a,b]

[a]*Nanoscopic Physics, Institute of Condensed Matter and Nanosciences, Université Catholique de Louvain, 1348 Louvain-la-Neuve, Belgium*
[b]*European Theoretical Spectroscopy Facility (ETSF)*
[c]*Q-Mat, Department of Physics, University of Liège (Belgium)*
[d]*Department of Physics and Astronomy, Rutgers University, Piscataway, NJ 08854-8019, USA*
[e]*Mat-Sim Research LLC, P. O. Box 742, Murray Hill, NJ, 07974, USA*

https://arxiv.org/abs/1710.10138



Select the flavor and format, then click "Download" to get the complete table of pseudos or choose a specific element.

XC: PBE
Accuracy: stringent
Relativistic: scalar
Format: psp8

How to generate a pseudopotential with ONCVPSP in PSML format

http://www.pseudo-dojo.org

Periodic table with pseudopotentials generated with ONCVPSP code

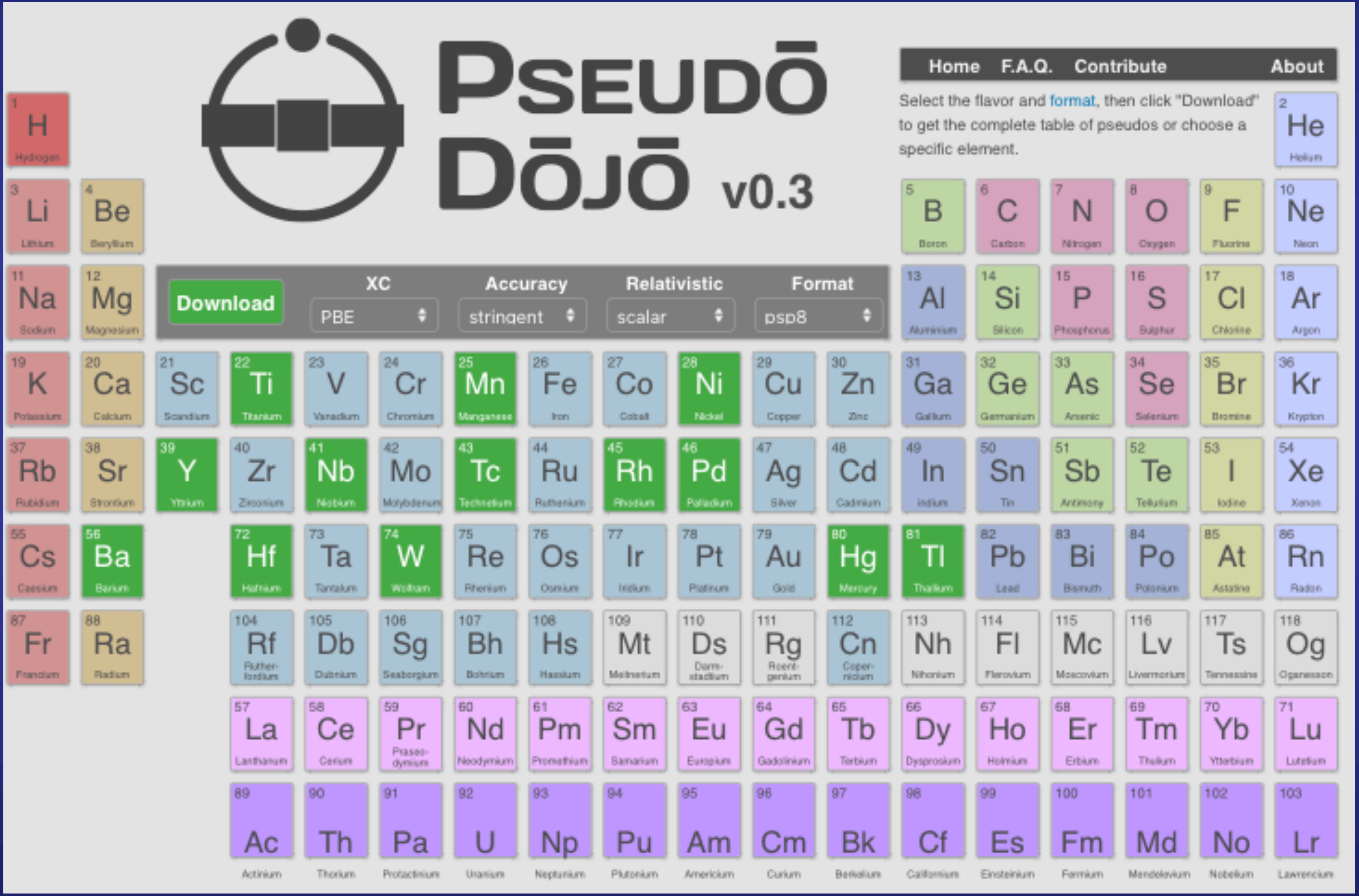
LDA, PBE, and PBE-sol flavors of exchange and correlation

They have passed the Delta-test with ABINIT
Reproducibility in density functional theory calculations of solids.
K. Lejaeghere *et. al*., Science 25 Mar 2016, Vol. 351, Issue 6280,
10.1126/science.aad3000

They are directly available in PSML format…
But , just in case you want to generate them by youself following the recipe given in the following slides

How to generate a pseudopotential with ONCVPSP in PSML format

http://www.pseudo-dojo.org



PSEUDŌ DŌJŌ v0.3

Home F.A.Q. Contribute About

Select the flavor and format, then click "Download" to get the complete table of pseudos or choose a specific element.

Download

XC: PBE
Accuracy: stringent
Relativistic: scalar
Format: psp8

1 H Hydrogen																	2 He Helium
3 Li Lithium	4 Be Beryllium											5 B Boron	6 C Carbon	7 N Nitrogen	8 O Oxygen	9 F Fluorine	10 Ne Neon
11 Na Sodium	12 Mg Magnesium											13 Al Aluminium	14 Si Silicon	15 P Phosphorus	16 S Sulphur	17 Cl Chlorine	18 Ar Argon
19 K Potassium	20 Ca Calcium	21 Sc Scandium	22 Ti Titanium	23 V Vanadium	24 Cr Chromium	25 Mn Manganese	26 Fe Iron	27 Co Cobalt	28 Ni Nickel	29 Cu Copper	30 Zn Zinc	31 Ga Gallium	32 Ge Germanium	33 As Arsenic	34 Se Selenium	35 Br Bromine	36 Kr Krypton
37 Rb Rubidium	38 Sr Strontium	39 Y Yttrium	40 Zr Zirconium	41 Nb Niobium	42 Mo Molybdenum	43 Tc Technetium	44 Ru Ruthenium	45 Rh Rhodium	46 Pd Palladium	47 Ag Silver	48 Cd Cadmium	49 In Indium	50 Sn Tin	51 Sb Antimony	52 Te Tellurium	53 I Iodine	54 Xe Xenon
55 Cs Caesium	56 Ba Barium		72 Hf Hafnium	73 Ta Tantalum	74 W Wolfram	75 Re Rhenium	76 Os Osmium	77 Ir Iridium	78 Pt Platinum	79 Au Gold	80 Hg Mercury	81 Tl Thallium	82 Pb Lead	83 Bi Bismuth	84 Po Polonium	85 At Astatine	86 Rn Radon
87 Fr Francium	88 Ra Radium		104 Rf Ruther-fordium	105 Db Dubnium	106 Sg Seaborgium	107 Bh Bohrium	108 Hs Hassium	109 Mt Meitnerium	110 Ds Darm-stadtium	111 Rg Roent-genium	112 Cn Coper-nicium	113 Nh Nihonium	114 Fl Flerovium	115 Mc Moscovium	116 Lv Livermorium	117 Ts Tennessine	118 Og Oganesson
			57 La Lanthanum	58 Ce Cerium	59 Pr Praseo-dymium	60 Nd Neodymium	61 Pm Promethium	62 Sm Samarium	63 Eu Europium	64 Gd Gadolinium	65 Tb Terbium	66 Dy Dysprosium	67 Ho Holmium	68 Er Erbium	69 Tm Thulium	70 Yb Ytterbium	71 Lu Lutetium
			89 Ac Actinium	90 Th Thorium	91 Pa Protactinium	92 U Uranium	93 Np Neptunium	94 Pu Plutonium	95 Am Americium	96 Cm Curium	97 Bk Berkelium	98 Cf Californium	99 Es Einsteinium	100 Fm Fermium	101 Md Mendelevium	102 No Nobelium	103 Lr Lawrencium

Select xc, Accuracy and relativistic type of calculation and download the pseudo in psp8 format

How to generate a pseudopotential with ONCVPSP in PSML format

```
# ATOM AND REFERENCE CONFIGURATION
# atsym  z    nc   nv     iexc    psfile
Fe 26.00    3    4       4      psp8
#
#   n    l    f        energy (Ha)
1    0    2.00    -2.5720573D+02
2    0    2.00    -3.0124388D+01
2    1    6.00    -2.5667700D+01
3    0    2.00    -3.4550911D+00
3    1    6.00    -2.2065345D+00
3    2    6.00    -2.7580112D-01
4    0    2.00    -1.9448241D-01
#
# PSEUDOPOTENTIAL AND OPTIMIZATION
# lmax
2
#
#   l,   rc,     ep,    ncon, nbas, qcut
0    1.15    -3.46    3    7     9.90
1    1.10    -2.21    3    7     9.20
2    1.30    -0.28    3    7    11.80
#
# LOCAL POTENTIAL
# lloc, lpopt,  rc(5),   dvloc0
4    5      1.10       0.00
#
# VANDERBILT-KLEINMAN-BYLANDER PROJECTORs
# l, nproj, debl
0    2    3.2606
1    2    0.8000
2    2    0.8000
#
# MODEL CORE CHARGE
# icmod, fcfact, rcfact, rcfact
3    3.00    1.70
#
# LOG DERIVATIVE ANALYSIS
# epsh1, epsh2, depsh
-24.00   12.00    0.02
#
# OUTPUT GRID
# rlmax, drl
6.00    0.01
#
#
# TEST CONFIGURATIONS
# ncnf
0
# nvcnf
#   n    l    f
```

Generate the input file for ONCVPSP

1. Edit the .psp8 file that you have downloaded from pseudo-dojo

2. Cut and paste the last lines of that file into another file. Rename this new file with an extension ".dat"

In this example, I have called it Fe.dat

3. Move this file into the tests/data directory of your ONCVPSP distribution

```
$mv Fe.dat <your_path_to_oncvpsp>/tests/data
```

4. Run

`$../run.sh Fe` (for a scalar-relativistic calculation)

or

`$../run_r.sh Fe` (for a fully relativistic calculation)

5. The pseudos are stored in Fe.psml and Fe_r.psml

How to generate a PSML pseudopotential with ATOM

How to generate and test a norm-conserving pseudopotential with ATOM in PSML format

Generate the pseudopotential using the ATOM code as usual, following the notes in the Tutorial "How to generate a norm conserving pseudopotential"

Copy the input file in the corresponding atom/Tutorial/PS_Generation directory and run

```
$ ../../Utils/pg.sh Fe.tm2.inp
==> Output data in directory Fe.tm2
==> Pseudopotential in Fe.tm2.vps, Fe.tm2.psf, and Fe.tm2.psml
```

The pseudopotentials will be on the same parent directory:

.vps (unformatted) (required to test the pseudopotential)

.psf (formatted)

.psml (in PSML format)

Remember to test the pseudopotential using the ATOM code as usual, following the notes in the Tutorial "How to test the transferability of a norm conserving pseudopotential"

How to generate and test a norm-conserving pseudopotential with ATOM in PSML format

The PSML file generated so far contains the semilocal component of the pseudopotential.

Most modern electronic-structure codes do not actually use the pseudopotential in its semi-local form, but in a more efficient fully non-local form based on short-range projectors plus a “local” potential

$$\hat{V}_{ps} = \hat{V}_{\rm local} + \sum_i |\chi_i\rangle E^i_{\rm KB} \langle\chi_i|$$

The local potential “a-la-SIESTA” and the non-local projectors (Kleinman-Bylander type) can be and added to the PSML file just running the psop code

```
$ <your_path_to_siesta_dir>/Pseudo/vnl-operator/psop -K -o Fe.psml Fe.tm2.psml
```

The local parts, projectors, etc are appended, and the full PSML file is written in the file whose name is after the “–o”.
For other options, type, try psop -h
The resuting output file is the one that can be directly used by SIESTA and ABINIT

How to generate and test a norm-conserving pseudopotential with ATOM in PSML format

If you edit the last PSML file, the whole provenance is perfectly identified

```
<provenance record-number="2" creator="psop-1.1" date="2017-08-03">
<annotation source-uuid="52959b00-6d35-11e7-412e-2ea7fffd684c"
 command-line="/Users/javier/Code/Launchpad/trunk-psml/Pseudo/vnl-operator/psop -K -o Fe.psml Fe.tm2.psml"
 action="inserted-local-potential" action-cont="inserted-nonlocal-projectors" />
</provenance>
<provenance record-number="1" creator="ATM4.2.6" date="20-JUL-17">
<annotation action="semilocal-pseudopotential-generation" />
<input-file name="INP">
<![CDATA[#  PS generation with core corrections
#  GGA (Perdew-Burke-Ernzerhof) XC , relativistic
#
   pe   Fe, GGA, rcore=0.70
        tm2      3.0
 n=Fe c=pbr
       0.0       0.0       0.0       0.0       0.0       0.0
    5    4
    4    0      2.00      0.00     # 4s2
    4    1      0.00      0.00     # 4p0
    3    2      6.00      0.00     # 3d6
    4    3      0.00      0.00     # 4f0
      2.25      2.75      2.00      2.00      0.00      0.70
#                                                        |
#                                       Radius of pseudocore
]]>
</input-file>
</provenance>
```

Outline of the Tutorial

1. How to compile the different codes

Two codes to generate pseudopotentials:	Two client Solid State Physic codes
ATOM (http://www.icmab.es/siesta/Pseudopotentials)	SIESTA (http://www.icmab.es/siesta)
ONCVPSP (http://www.quantum-simulation.org/potentials/sg15_oncv/)	ABINIT (http://www.abinit.org)

2. How to generate the psml pseudopotentials with ONCVPSP and ATOM

3. How to run SIESTA and ABINIT with the same pseudopotentials

   Test of the convergence of a numerical atomic orbital basis set with respect to the asymptotic limit of a converged basis of plane waves

   Compute the equation-of-state (energy versus volume profiles) for elemental crystals, a test that has been proposed as a benchmark for the comparison of different codes

Examples to run SIESTA and ABINIT with the same pseudos

1. Visit the web page:
http://personales.unican.es/junqueraj

and follow these links:

A self-explained SIESTA tutorial
Set of self-explained SIESTA exercises
Pseudos
How to use the same pseudopotential in SIESTA and ABINIT

2. Click on Pseudos, input and Readme

3. Untar the ball file

```
$ tar –xvf  Siesta-Abinit.tar
```

This will generate a directory called Comparison-Siesta-Abinit with 4 directories:

```
$ cd Comparison-Siesta-Abinit
$ ls -ltr
$      Si              (example for a covalent semiconductor, LDA)
$      Al              (example for a sp-metal, LDA)
$      Au              (example for a noble metal, includes d-orbital, LDA)
$      Fe              (example for a transition metal, includes NLCC, GGA)
```

Examples to run SIESTA and ABINIT with the same pseudos

In every subdirectory it can be found:

```
$ cd Si
$ ls –ltr
$     Runsiesta                (files to run SIESTA)
$     Runabinit                (files to run ABINIT)
```

In the directoris for Au and Fe, you will find two extra subdirectories:
One for the pseudos generated with ATOM, and the second for ONCVPSP pseudos

Test of the convergence of a numerical atomic orbital basis set with respect to the asymptotic limit of a converged basis of plane waves

Convergence of the energy as a function of the planewave cutoff in ABINIT

Required files in: Si/Runabinit/ATOM/

1. We run the same system (same lattice vectors and internal coordinates) at the same level of approximations (same exchange and correlation functional, Monkhorst-Pack mesh etc.) at a given lattice constant.

Here it has been written for you (file Si.input.convergence)

```
#Number of atoms, chemical species and atom types
natom    2                           # Number of atoms in the unit cell
ntypat   1                           # Number of types of atoms
typat    1 1                         # Type of atoms
znucl    14.0                        # Gives nuclear charge for each type of
#Coordinates and cell variables
rprim    0.0      0.5      0.5
         0.5      0.0      0.5
         0.5      0.5      0.0
acell    5.38    5.38    5.38    Angstrom
xred -0.125 -0.125 -0.125
      0.125  0.125  0.125
#PlaneWave cutoff and k-grid mesh integration
ecutsm   0.5                         # Energy cutoff smearing (Ha)
nband    10                          # Number of bands
kptopt   1                           # Kpoints option
                                     # 0 = read directly nkpt, kpt, kptnrm and wtk
                                     # 1 = rely on ngkpt or kptrlatt, as well as
                                     #     on nshiftk and shiftk to set up
                                     #     the k points.
                                     #     Full symmetry taken into account.
                                     # 2 = 1, but only time reversal symmetry
                                     #     is taken into account.
                                     # 3 = 1, but do not take into account
                                     #     any symmetry
                                     # A negative value = rely on kptbounds,
                                     #     and ndivk to set up a band structure
                                     #     calculation along different lines
ngkpt    3 3 3                       # Number of grid points for k points
                                     #     generation
                                     # This is a 3x3x3 FCC grid,
                                     #     based on the primitive vectors
                                     #     of the reciprocal space.
                                     #     For a FCC real space lattice,
                                     #     like the present one,
                                     #     it actually corresponds to the
                                     #     so-called 6x6x6 Monkhorst-Pack grid,
                                     #     if the following shifts are used :
nshiftk 4                            #
shiftk   0.5 0.5 0.5                 # Shift for k points
         0.5 0.0 0.0
         0.0 0.5 0.0
         0.0 0.0 0.5
#Exchange-correlation
ixc                                  # Integer for exchange-correlation choice
        -1009                        # 0 = No xc
                                     # 1 = LDA or LSD, Teter Pade parametrization
                                     # 2 = LDA, Perdew-Zunger-Ceperley-Alder
                                     # 3 = LDA, old Teter rational polynomial
                                     #     parametrization, fit to Ceperley-Alder
                                     #     data (no spin-polarization: no sp)
                                     # 4 = LDA, Wigner functional (no sp)
                                     # 5 = LDA, Hedin-Lundqvist functional (no sp)
                                     # 6 = LDA, "X-alpha" functional (no sp)
                                     # 7 = LDA or LSD, Perdew-Wang 92 functional
                                     # 8 = LDA or LSD, x-only part of the PW 92
                                     # 9 = LDA or LSD, x- and RPA part of the PW92
                                     # 10= GGA, Perdew-Burke-Ernzerhof
                                      # -1009 = LDA, Perdew-Zunger in libxc
```

Diamond structure.
The lattice constant might be the experimental, theoretical or whatever other sensible choice

6 × 6 × 6 Monkhorst-Pack mesh

Ceperley-Alder (LDA) functional
In the libxc numeration, the CA functional is identified as -1009,
So here ixc might be also
ixc   -1009

Convergence of the energy as a function of the planewave cutoff in ABINIT

Required files in: Si/Runabinit/ATOM/

1. We run the same system (same lattice vectors and internal coordinates) at the same level of approximations (same exchange and correlation functional, Monkhorst-Pack mesh etc.) at a given lattice constant.

Here it has been written for you (file Si.input.convergence)

```
ndtset    21

ecut1      4.00
ecut2      5.00
ecut3      6.00
ecut4      7.00
ecut5      8.00
ecut6      9.00
ecut7     10.00
ecut8     11.00
ecut9     12.00
ecut10    13.00
ecut11    14.00
ecut12    15.00
ecut13    16.00
ecut14    17.00
ecut15    18.00
ecut16    19.00
ecut17    20.00
ecut18    25.00
ecut19    30.00
ecut20    35.00
ecut21    40.00
```

2. Change the cutoff energy for the plane waves

3. Edit the .files file and select the input file and the pseudo file (in PSML format)

```
$ more Si.files
Si.input.convergence
Si.out
Sii
Sio
t1x
Si.psml
```

4. Run the code

```
$ abinit < Si.files > Si.log &
```

Convergence of the energy as a function of the planewave cutoff in ABINIT

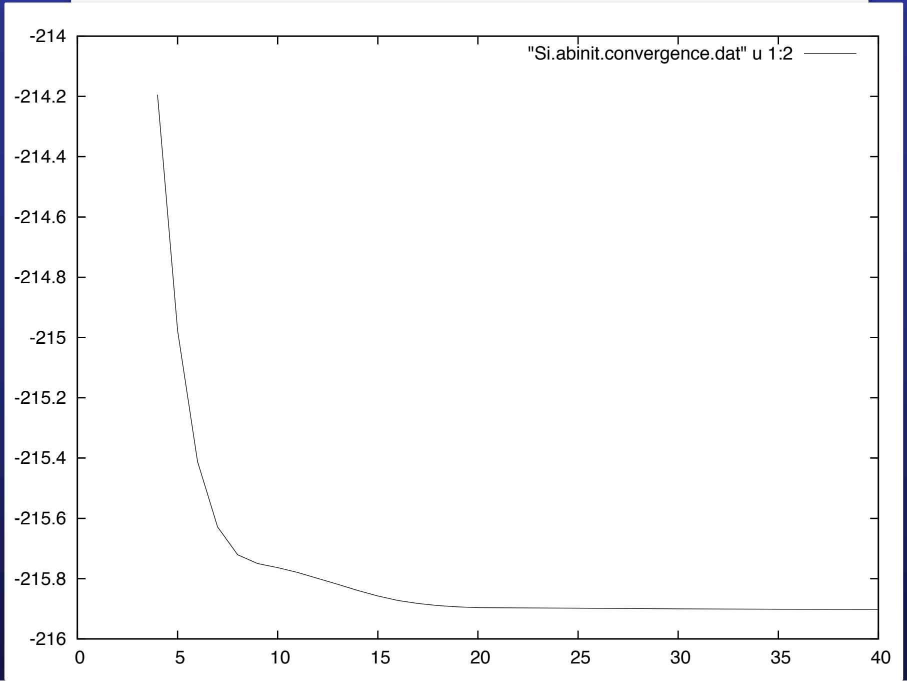
Dump the total energy as a function of the cutoff energy into a file

```
grep "Total energy(eV)=" Si.out > Si.abinit.convergence.dat
```

and edit the corresponding file that should look like this

```
# Cutoff energy (Ha)        Total energy (eV)
        4.0                -2.14194449880966E+02
        5.0                -2.14976485447252E+02
        6.0                -2.15411140367238E+02
        7.0                -2.15628685176492E+02
        8.0                -2.15720858317710E+02
        9.0                -2.15750171586496E+02
       10.0                -2.15763329519177E+02
       11.0                -2.15779812452764E+02
       12.0                -2.15799418752144E+02
       13.0                -2.15818946405721E+02
       14.0                -2.15839166115399E+02
       15.0                -2.15857714339095E+02
       16.0                -2.15872186607154E+02
       17.0                -2.15882236630660E+02
       18.0                -2.15889235709976E+02
       19.0                -2.15893739286028E+02
       20.0                -2.15896300984227E+02
       25.0                -2.15898209558594E+02
       30.0                -2.15900080538176E+02
       35.0                -2.15901735062792E+02
       40.0                -2.15902018005472E+02
```

```
$ gnuplot
gnuplot> plot "Si.abinit.convergence.dat" u 1:2 w l
gnuplot> set terminal postscript
Terminal type set to 'postscript'
Options are 'landscape noenhanced defaultplex \
   leveldefault monochrome colortext \
   dashed dashlength 1.0 linewidth 1.0 butt \
   palfuncparam 2000,0.003 \
   "Helvetica" 14 '
gnuplot> set output "Si.abinit.convergence.ps"
gnuplot> replot
```



Convergence of the energy as a function of the basis set size in SIESTA

Select:

- a given system (in this example, bulk Si)
- in a given structure (diamond structure)
- For a given lattice constant and internal coordinates
- Since we are interested in compare the performance of the basis set, it is important to converge all the rest of approximations (Mesh Cutoff, k-point grid, etc.) as much as possible
- The parameters of the simulations (lattice constant, functional, k-point sampling, etc. Must be the same as the ones used for the plane wave calculation

The input files have been prepared for you in the directory:
Si/Runsiesta/ATOM/Basis-convergence

The basis sizes, of different quality (SZ, DZ, TZ, SZP, DZP, TZP, TZDP, TZTP, TZTPF) were variationally optimized in a previous work
J. Junquera et al., Phys. Rev. B 64, 235111 (2001)

But you can try whatever basis whose convergence with respect to a plane wave calculation you want to check

Convergence of the energy as a function of the basis set size in SIESTA

Run SIESTA for all the different basis sets prepared for you

```
<your_path_to_siesta_executable>/siesta < Si.SZ.fdf    > Si.SZ.out
<your_path_to_siesta_executable>/siesta < Si.DZ.fdf    > Si.DZ.out
<your_path_to_siesta_executable>/siesta < Si.TZ.fdf    > Si.TZ.out
<your_path_to_siesta_executable>/siesta < Si.SZP.fdf   > Si.SZP.out
<your_path_to_siesta_executable>/siesta < Si.DZP.fdf   > Si.DZP.out
<your_path_to_siesta_executable>/siesta < Si.TZP.fdf   > Si.TZP.out
<your_path_to_siesta_executable>/siesta < Si.TZDP.fdf  > Si.TZDP.out
<your_path_to_siesta_executable>/siesta < Si.TZTP.fdf  > Si.TZTP.out
<your_path_to_siesta_executable>/siesta < Si.TZTPF.fdf > Si.TZTPF.out
```

Search the Total Energy as a function of the basis size and store it in a file

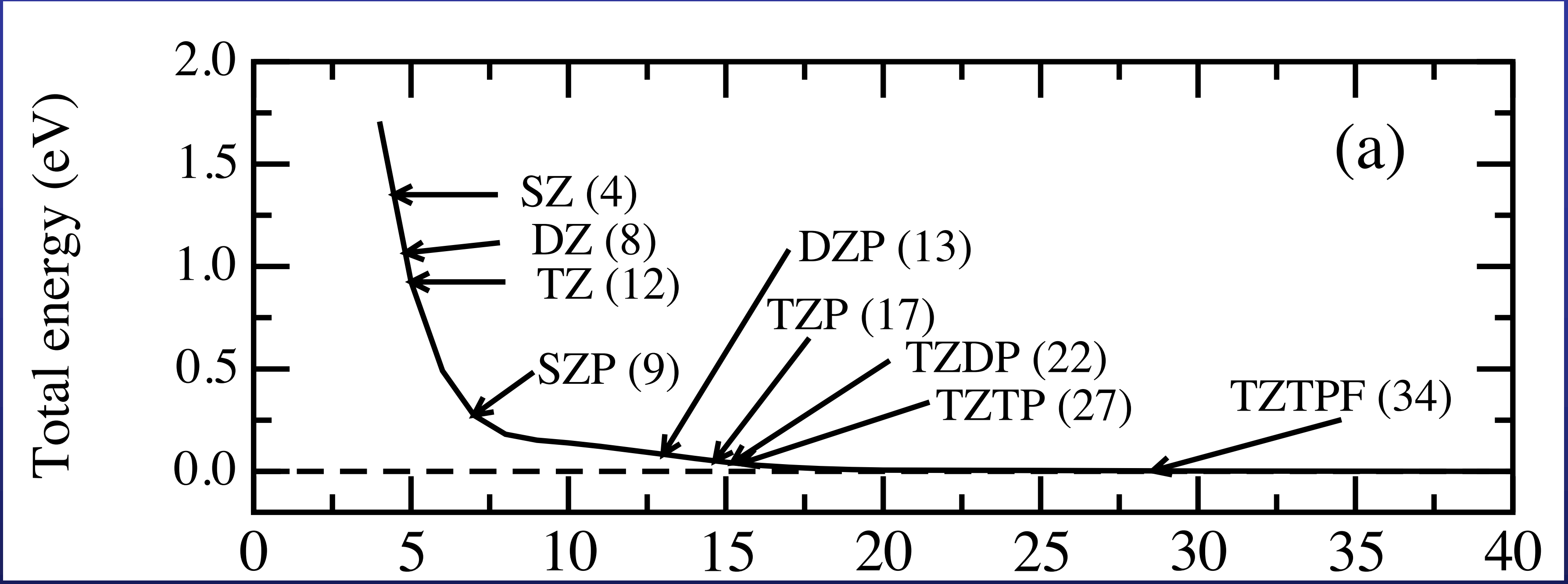
```
$ grep FreeEng *out > Si.siesta.basis.dat
```

Edit the file (Si.siesta.basis.dat) and search for the equivalent plane wave cutoff in ABINIT in the file Si.abinit.convergence.dat previously prepared

```
# Basis size   Number of orbitals per Si      Total energy (eV)   Equiv. PW(Ha)
#   SZ                      4                    -214.551542         4.4572457
#   DZ                      8                    -214.835617         4.8208821
#   SZP                     9                    -215.627384         6.9954995
#   TZ                     12                    -214.977739         5.0009001
#   DZP                    13                    -215.819105        13.0045
#   TZP                    17                    -215.851210        14.649865
#   TZDP                   22                    -215.860688        15.20432
#   TZTP                   27                    -215.864488        15.467147
#   TZTPF                  34                    -215.899542        28.522052
```

Test of the convergence of a numerical atomic orbital basis set with respect to the asymptotic limit of a converged basis of plane waves

You can plot again the energy versus plane wave cutoff, taking the converged energy as a reference, and put on top of this figure the equivalence with the NAO basis set



Comparing the pseudopotential in SIESTA and ABINIT

To be totally sure that we have run SIESTA and ABINIT with the same peudopotential operator, i.e. with the same decomposition in local part and Kleinman-Bylander projectors:

1. Edit one of the output files in SIESTA and search for the following lines:

```
Reading KB projs from Si psml data

PSML: Kleinman-Bylander projectors:
   l= 0   rc=  1.936440   Ekb=  4.661340
   l= 1   rc=  1.936440   Ekb=  1.494238
   l= 2   rc=  1.936440   Ekb= -2.809035
   l= 3   rc=  1.936440   Ekb= -0.959387
```

2. Edit the log or the output file in ABINIT and search for the following lines:

```
- psxml2ab: ps_Number_of_Projectors not relativistic     4
- psxml2ab: ps_Number_of_Projectors scalar relativistic     0
- psxml2ab: ps_Projector_L      0
- psxml2ab: ps_Projector_Ekb     0.2330670221E+01
- psxml2ab: ps_Projector_L      1
- psxml2ab: ps_Projector_Ekb     0.7471191667E+00
- psxml2ab: ps_Projector_L      2
- psxml2ab: ps_Projector_Ekb    -0.1404517392E+01
- psxml2ab: ps_Projector_L      3
- psxml2ab: ps_Projector_Ekb    -0.4796933758E+00
- psxml2ab: ps_Number_of_Projectors SOC     0
```

The Kleinman-Bylander energies should be the same upto numerical roundoff errors

Note: In SIESTA they are written in Ry and in ABINIT they are in Ha.

Compute the equation-of-state (energy versus volume profiles) for elemental crystals: a test that has been proposed as a benchmark for the comparison of different codes (related with the delta-test)

Running the energy versus lattice constant curve in ABINIT

1. Same input as before but…

```
#input for bulk Si in the diamond structure.

ecut     13.005                       # Energy cutoff (Ha)

ndtset    10

acell1    5.30    5.30    5.30    Angstrom
acell2    5.32    5.32    5.32    Angstrom
acell3    5.34    5.34    5.34    Angstrom
acell4    5.36    5.36    5.36    Angstrom
acell5    5.38    5.38    5.38    Angstrom
acell6    5.40    5.40    5.40    Angstrom
acell7    5.42    5.42    5.42    Angstrom
acell8    5.44    5.44    5.44    Angstrom
acell9    5.46    5.46    5.46    Angstrom
acell10   5.48    5.48    5.48    Angstrom
```

… setting the plane wave cutoff to the equivalent one to a DZP basis set (we want to compare the results at this level of basis set quality)

… and changing the lattice constant embracing the minimum

2. Change in the .files the name of the input file

```
Si.input.latcon
Si.out
Sii
Sio
t1x
Si.psml
```

3. Run the code

```
$ abinit < Si.files > Si.log &
```

Running the energy versus lattice constant curve in ABINIT

Dump the total energy as a function of the lattice constant in a file

```
grep Total Si.abinit.latcon.out > Si.abinit.latcon.dat
```

and edit the corresponding file that should look like this

```
#  Lattice constant (Ang)    Total energy (eV)
         5.30               -2.15791847642832E+02
         5.32               -2.15803559782380E+02
         5.34               -2.15811827098470E+02
         5.36               -2.15816907060262E+02
         5.38               -2.15819046038271E+02
         5.40               -2.15818275434177E+02
         5.42               -2.15814668717931E+02
         5.44               -2.15808352394512E+02
         5.46               -2.15799487882416E+02
         5.48               -2.15788013554804E+02
```

Running the energy versus lattice constant curve in SIESTA

Run siesta for the equivalent quality of the basis set (DZP in this case) and for the same lattice constants.
The input files have been prepared for you in the directory
Si/Runsiesta/ATOM/Latcon

Dump the total energy as a function of the lattice constant in a file

```
$ grep FreeEng *out > Si.siesta.latcon.dat
```

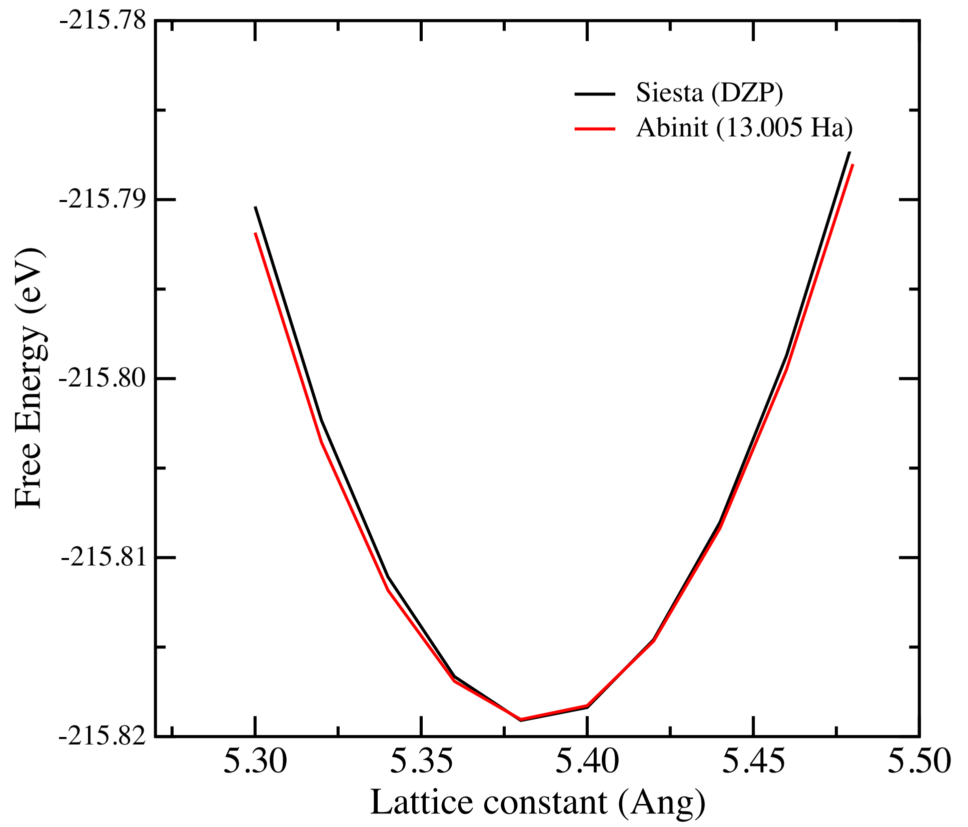
and edit the corresponding file that should look like this

```
#
# Lattice Constant (Ang)        Free Energy (eV)
#                                 SIESTA (DZP)
        5.30                     -215.790374
        5.32                     -215.802367
        5.34                     -215.811092
        5.36                     -215.816652
        5.38                     -215.819105
        5.40                     -215.818383
        5.42                     -215.814594
        5.44                     -215.808031
        5.46                     -215.798719
        5.48                     -215.786737
```

These data have been obtained with a double-zeta plus polarization basis set, optimized at the theoretical lattice constant with a pressure of 0.05 GPa

Comparing the energy versus lattice constant in SIESTA and ABINIT: bulk Si (covalent semiconductor)

```
gnuplot> plot "Si.abinit.latcon.dat" u 1:2 w l,
"../../Runsiesta/ATOM/Latcon/Si.siesta.latcon.dat" u 1 :2 w l
gnuplot> set terminal postscript color

Terminal type is now 'postscript'
Options are 'landscape enhanced defaultplex \
   leveldefault color colortext \
   dashlength 1.0 linewidth 1.0 pointscale 1.0 butt noclip \
   nobackground \
   palfuncparam 2000,0.003 \
   "Helvetica" 14  fontscale 1.0 '
gnuplot> set output "Si.compar.latcon.ps"
gnuplot> replot
```



Comparing the pseudopotential in SIESTA and ABINIT in a metallic system: bulk Al (*sp* metal)

To be totally sure that we have run SIESTA and ABINIT with the same peudopotential operator, i.e. with the same decomposition in local part and Kleinman-Bylander projectors:

In ABINIT:
Search for the Kleinman-Bylander energies in the log or out file

```
- psxml2ab: ps_Number_of_Projectors not relativistic     4
- psxml2ab: ps_Number_of_Projectors scalar relativistic     0
- psxml2ab: ps_Projector_L     0
- psxml2ab: ps_Projector_Ekb     0.1556932920E+01
- psxml2ab: ps_Projector_L     1
- psxml2ab: ps_Projector_Ekb     0.4394082275E+00
- psxml2ab: ps_Projector_L     2
- psxml2ab: ps_Projector_Ekb    -0.8914633661E+00
- psxml2ab: ps_Projector_L     3
- psxml2ab: ps_Projector_Ekb    -0.3082875005E+00
```

In SIESTA:
Search for the Kleinman-Bylander energies in the output file

```
Reading KB projs from Al psml data

PSML: Kleinman-Bylander projectors:
   l= 0   rc=  2.333733   Ekb=  3.113866
   l= 1   rc=  2.333733   Ekb=  0.878816
   l= 2   rc=  2.333733   Ekb= -1.782927
   l= 3   rc=  2.333733   Ekb= -0.616575
```

The Ekb should be exactly the same.
Remember than in ABINIT, they are writen in Ha, while in SIESTA are dumped in Ry

Comparing the pseudopotential in SIESTA and ABINIT in a metallic system: bulk Al (*sp* metal)

For the case of metallic system, besides the k-point sampling we have to pay particular attention to the occupation option

SIESTA

Default: Fermi-Dirac

```
ElectronicTemperature  0.02 Ry
```

ABINIT

```
occopt   3          # Occupation Option
                    # Fermi-Dirac smearing (finite-temperature metal)
tsmear   0.01       # Temperature of smearing (in Ha)
```

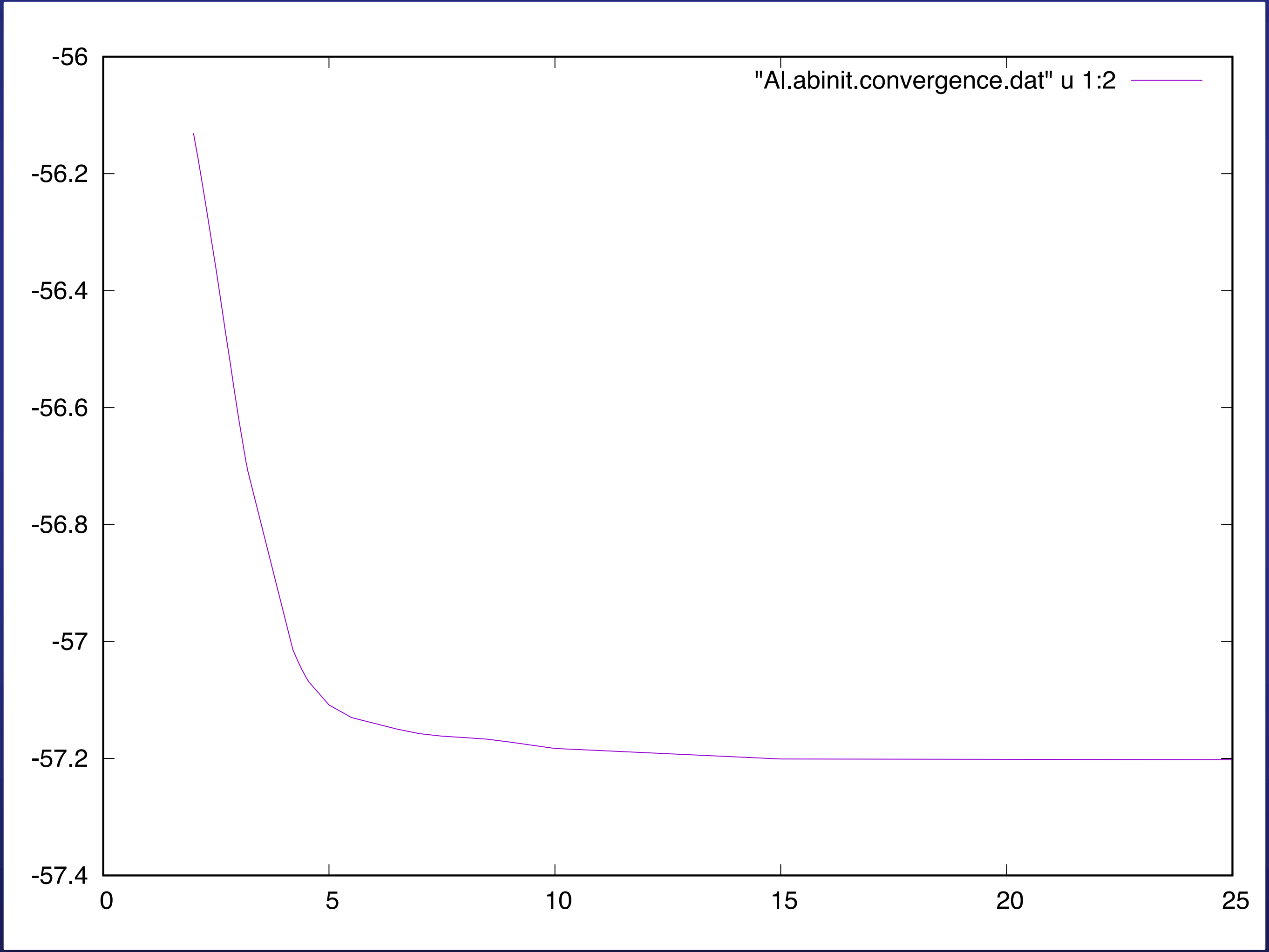
Also, as explained in the Tutorial
"Convergence of electronic and structural properties of a metal with respect to the k-point sampling: bulk Al"
we should look at the Free Energy and not to the Kohn-Sham energy

```
grep FreeEng *.out > Al.siesta.latcon.dat
```

```
grep Total Al.out > Al.abinit.latcon.dat
```

Running the energy versus cutoff energy in ABINIT: bulk Al (a *sp* metal)

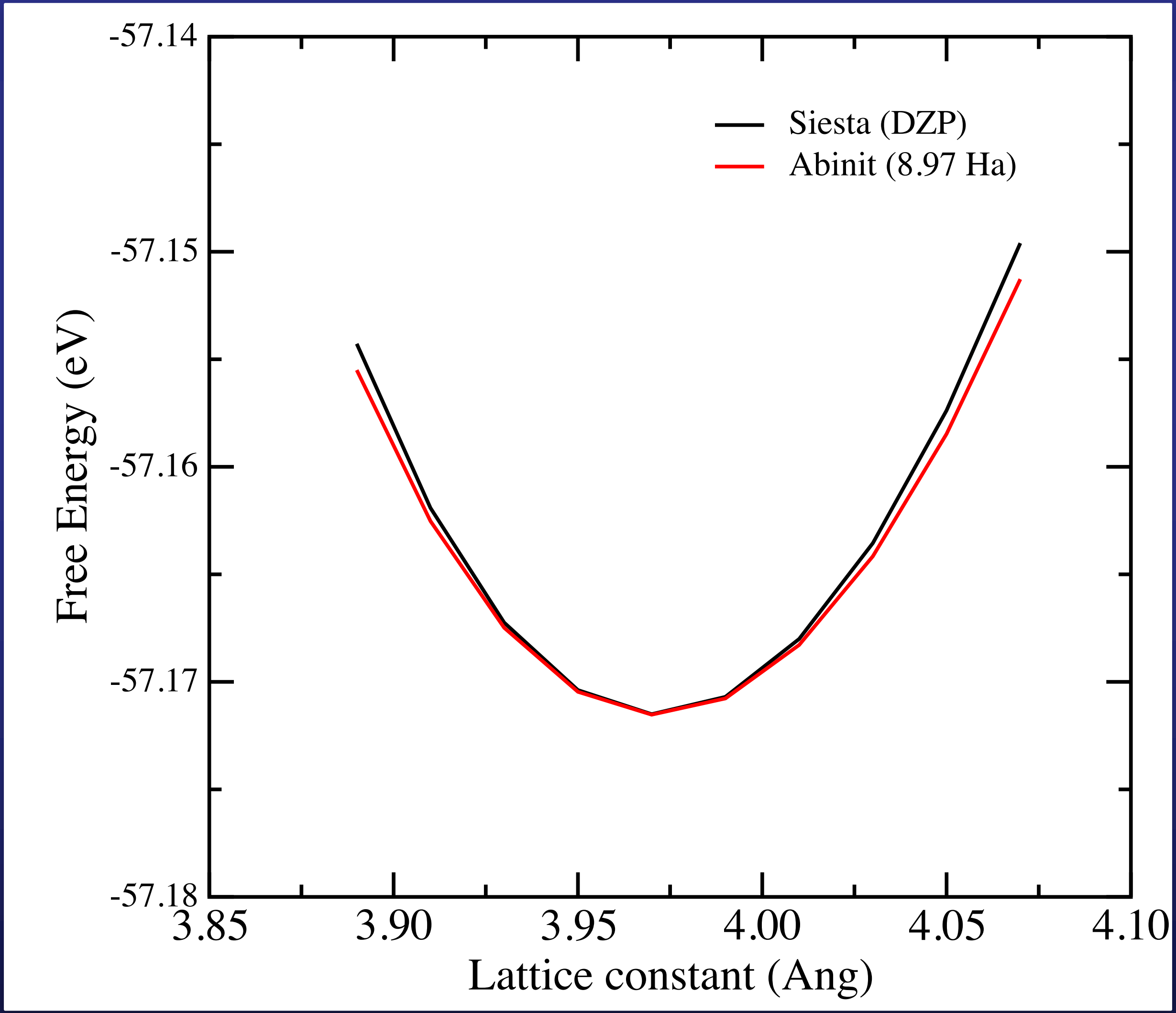
Lattice constant 3.97 Å



Comparing the energy versus lattice constant in SIESTA and ABINIT: bulk Al (sp metal)

Basis set of Siesta: DZP optimized with a pressure of 0.001 GPa at the theoretical lattice constant of 3.97 Å)

Plane wave cutoff in Abinit: 8.97 Ha



Comparing the pseudopotential in SIESTA and ABINIT: bulk Fe (a magnetic transition metal)

For the case of metallic system, besides the k-point sampling we have to pay particular attention to the occupation option.

Now, besides:

- The system is spin polarized
- We use a GGA functional
- We include non-linear partial core corrections in the pseudo

SIESTA

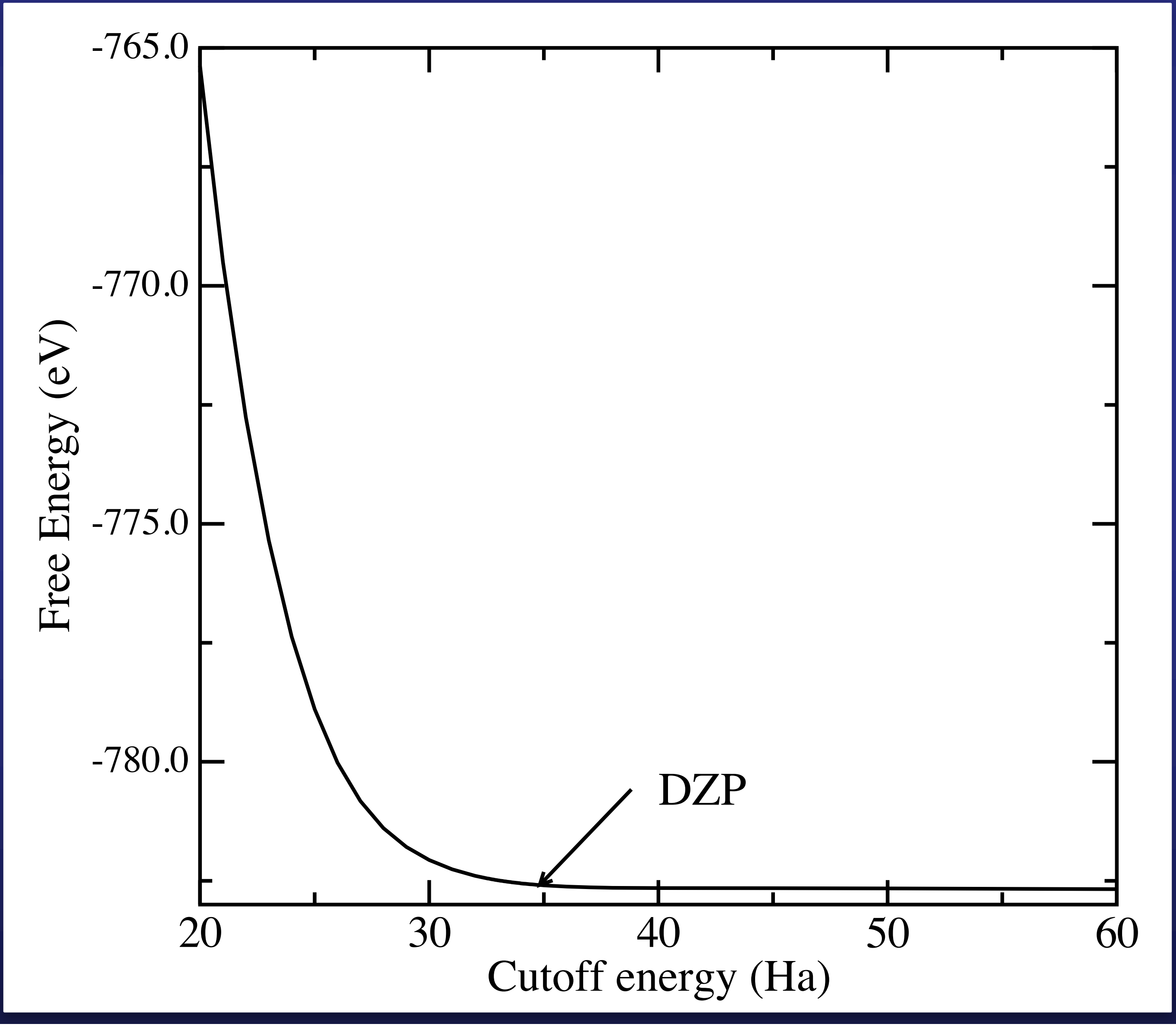
```
XC.functional          GGA
XC.authors             PBE
SpinPolarized          .true.
```

ABINIT

```
nsppol   2                          # Number of spin polarizations
spinat   0.0 0.0 1.0                # Spin for atoms
ixc      11                         # Integer for exchange-correlation choice
```

Running the energy versus cutoff energy in ABINIT: bulk Fe (a magnetic transition metal)

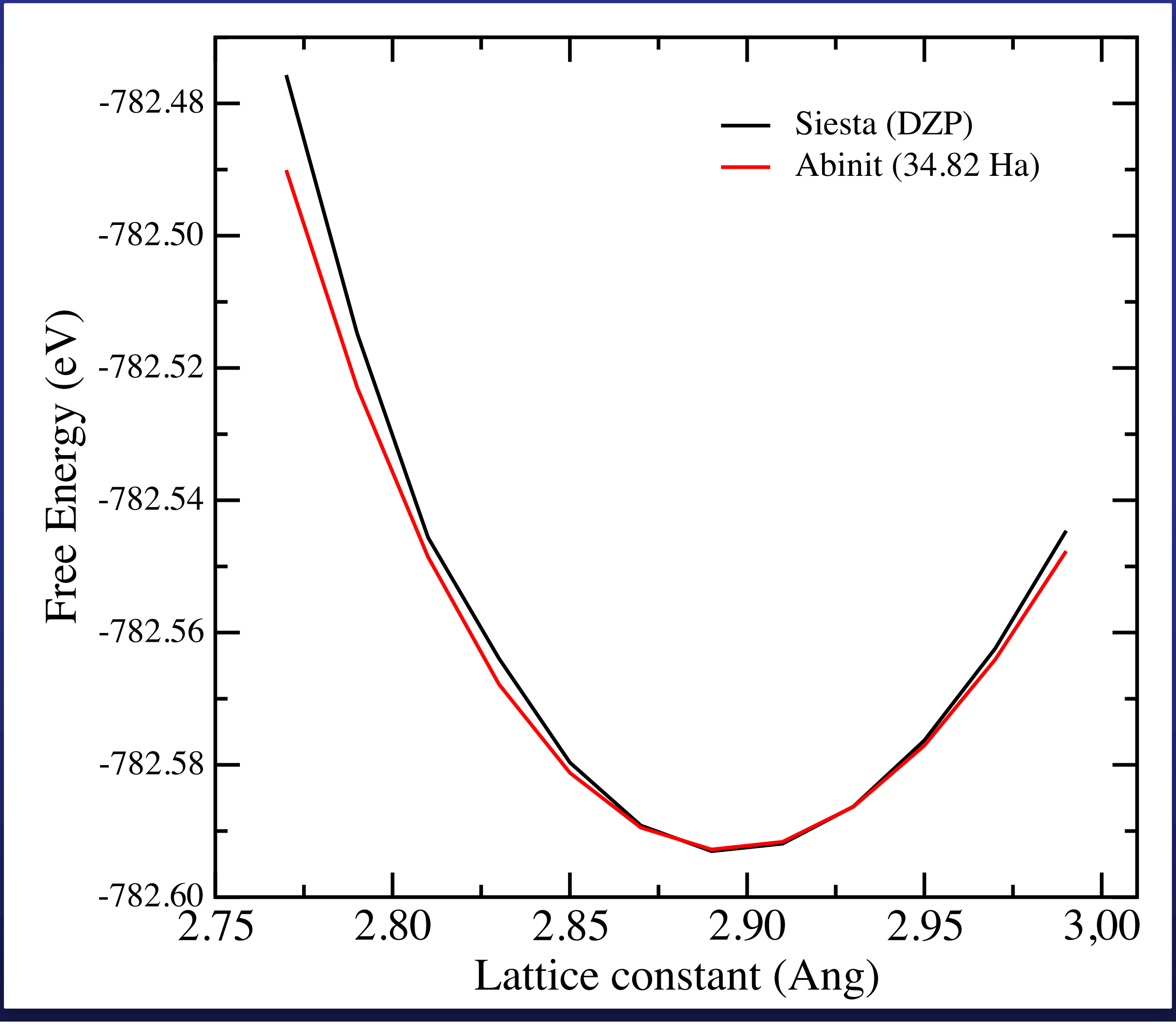
Lattice constant 2.87 Å



Comparing the energy versus lattice constant in SIESTA and ABINIT: bulk Fe(magnetic transition metal)

Basis set of Siesta: DZP optimized without pressure at the experimental lattice constant of 2.87 Å

Plane wave cutoff in Abinit: 34.82 Ha



Funding

SPANISH INITIATIVE FOR ELECTRONIC SIMULATIONS WITH THOUSANDS OF ATOMS: CÓDIGO ABIERTO CON GARANTÍA Y SOPORTE PROFESIONAL: SIESTA-PRO
Proyecto financiado por el Ministerio de Economía, Industria y Competitividad,
y cofinanciado con Fondos Estructurales de la Unión Europea
Referencia: RTC-2016-5681-7
Objetivo Temático del Programa Operativo:
"Promover el desarrollo tecnológico, la innovación y una investigación de calidad"

UNION EUROPEA
FONDO EUROPEO DE
DESARROLLO REGIONAL
"Una manera de hacer Europa"

GOBIERNO
DE ESPAÑA

MINISTERIO
DE ECONOMÍA, INDUSTRIA
Y COMPETITIVIDAD

AGENCIA
ESTATAL DE
INVESTIGACIÓN