Hilbertxx manual

Alessandro MIRONE
ESRF, BP 220, 38043 Grenoble, France

May 22, 2007

Abstract

We present here installation instruction, a hand-on guide to examples, and the theory, for the Hilbert++ code.

1 INSTALLATION

The code is provided for the linux i386 system. The distribution is composed of two archives.The first one (let's call it *installation archive*) provides all the necessary libraries that Hilbert++ needs, and a second one which is Hilbert++ itself, comprising sources, binaries, and examples.

Hilbert++ dynamically loads the libraries of the *installation archive* and does not depend on which linux-packages are installed or not installed on your linux system. There is an exception for libc and libX11. As these libraries are quite stable one should be able to run Hilbert++ on any Linux system without recompiling it.

The actual *installation archive* and Hilbert++ packages can be retrieved at these addresses : *ftp://ftp.esrf.fr/pub/scisoft/ESRF_sw/scisoft_ESRF_sw_linuxi386_03.tar.gz* and *ftp://ftp.esrf.fr/pub/scisoft/ESRF_sw/linux_i386_03/hilbertxx.tar.gz* respectively.

The installation archive must be untarred with the following commands

```
cd /
tar -xzvf yourdownloaddirectory/scisoft_ESRF_sw_linuxi386_03.tar.gz
```

which create the directory /scisoft/ESRF_sw/linux_i386_03/.

The hilbertxx directory can be untarred, from any directory, with the command :

```
cd whereever
tar -xzvf yourdownloaddirectory/hilbertxx.tar.gz
```

2 Hand on the examples

To launch the program you cd to hilbertxx/examples/exa_*class*, where *class* corresponds to an examples class. The examples classes provided so far are :

Ex. a) *class*=2p3d for L2-L3 spectroscopy

Ex. b) *class*=rixs for 1s-3p rixs spectroscopy, with a 3d resonating intermediate state, in transition metal oxides.

Then you choose an example system by cd-ing to one of the subdirectory. For example

```
cd hilbertxx/examples/exa_2p3d/Mn3+/
```

Once you are in the exa_*class*/*system* directory you start the application by this command :

```
../runthisscript
```

You get a command line python prompt:

```
>>>
```

and you can get some help

```
>>>help()
```

The help function prints some brief instruction for every hilbert++-command (hxx-command) that you can give to the program. A hxx-command is always one single line long and writing it does not require any knowledge of python. The advanced user can write python scripts. We explain for each example the available commands

2.1 The 2p3d examples class

2.1.1 Mn3+

You cd to the appropriate directory

```
cd hilbertxx/examples/exa_2p3d/Mn3+/
```

and start the script

```
../runthisscript
```

You get now a python prompt. You can give several commands. In this manual we are going to explain thoroughly all the commands. A list of these commands can be obtained, along with a brief explanation, at the python prompt using the command *help()* which produces the following output :

```
 AVAILABLE HXX-COMMANDS ( examples of )
set(system)    # or use directly python syntax to change the attributes of object  system.
               # When using the set(system) function you can also give a filename of a previously
               # saved ( and edited ) file

save(system)   #  You are asked a filename and  you have to quote it like "filename"

scriviFiles(5,prefix="datas", nhopped=0)  # To generate matrices files. This command creates the Hilbert space
               # and writes the terms composing the Hamiltonian into (sub)directory "datas" which must exist.
               # The above example generates a Hilbert space where 5 electrons are distributed in the 3d
```

```
              # shell in all possible combinations.
              # Giving nhopped=n as argument, besides the configurations with 5 electrons on 3d also the configurations
              # with 5+i electrons on 3d and 10-i on Ligands orbitals are considered ( 0<=i<=nhopped )

system.case="datas" # This chooses the system that you have expanded with scriviFiles command.
              #    You can acces this property also with the set(system)    command

res=system.GetSpectrum() #   to calculate a spectrum.  res is a list.res[0] is an array of floats : the energies.
              # res[i>=1] are the resonances ( real and imaginary part). Dipolar transitions are calculated as
              # res[i], where i=1,2,3 and Mz= -1,0,1

res=system.GetSpectrum([ fm1,f0,f1 ]) # Absorption (it is res[1].imag)  for a  definite polarisation.
              # Fm1,f0,f1 are three coefficients.
              # They can be complex. So you can define any polarisation :
              # For Z polarisation [ fm1,f0,f1 ] = [ 0, 1, 0]
              # For X polarisation [ fm1,f0,f1 ] = [-1/sqrt(2.0) , 0 ,1/sqrt(2.0)]
              # For Y polarisation [ fm1,f0,f1 ] = [ 1.0j /sqrt(2.0) , 0 ,1.0j/sqrt(2.0)]
              # ..... and so on

write(res,filename)   #  To save res on a file
              #  and the file will have several columns ( 1 for energies and two (real, imag) for each polarisation)

Plot(Curve(res[0], res[n>=1].imag, Pen(Red), "legend")) # To plot. Change n to select the polarisation that you want
              #  The plotting feature uses qt, iqt and qwt. If you run a long job in the background you must
              # desactivate this graphics feature by commenting out, in the file init.py, the lines where such packages
              # are imported

Es, S2s, L2s, SL2, occPs , Szs, Lzs= system.GetESLcounters()  #  to get statistics: a list of E, the list of 3d  L2 ,
              # the list of 3d S2, the list of 3d SL, a list of ligand expected occupation, a list of 3d Sz expectation
```

The system is represented by the python variable *system*. Such variable is an object, its properties are the parameters. There are several ways to change such properties. One of this way is using the set(system) command. Such command prints the list of parameters :

```
>>> set(system)
---- BASE HAMILTONIAN ---
  1) base_couche1_F0 :     5.0
  2) base_couche1_F2 :     12.4156828106
  3) base_couche1_F4 :     7.81967819912
  4) base_couche0_1_F0 :     5.5
  5) base_couche0_1_F2 :     6.86721502072
  6) base_couche0_1_G1 :     5.02109490016
  7) base_couche0_1_G3 :     2.85321756768
  8) base_SO_0       :     6.568603656
  9) base_SO_1       :     0.05238772
 10) base_Sop_Zero   :     1e-05
 11) base_Sop_Minus  :     0.0
 12) base_Sop_Plus   :     0.0
 13) base_counterDL  :     -4
---- EXCITED HAMILTONIAN ---
 14) exci_couche1_F0 :     5.0
 15) exci_couche1_F2 :     13.1769757147
 16) exci_couche1_F4 :     8.299507532
 17) exci_couche0_1_F0 :     5.5
 18) exci_couche0_1_F2 :     7.6574518
 19) exci_couche0_1_G1 :     5.77390099368
 20) exci_couche0_1_G3 :     3.28715525784
 21) exci_SO_0       :     6.845918392
 22) exci_SO_1       :     0.066403136
 23) exci_Sop_Zero   :     1e-05
 24) exci_Sop_Minus  :     0.0
 25) exci_Sop_Plus   :     0.0
 26) exci_counterDL  :     -4
---- CALCULATION PARAMETERS ---
```

```
 27) case            :     ./
 28) reduc_1         :     0.8
 29) reduc_0_1       :     0.8
 30) all1            :     0.1
 31) El2l3           :     700
 32) all2            :     0.1
 33) shift           :     0
 34) npunti          :     500
 35) dxleft          :     -0.1
 36) dxright         :     0.1
 37) temp            :     0.009
 38) erange          :     0.1
 39) tolefact        :     1e-06
 40) shift_invert    :     0
 41) nsearchedeigen  :     10
 42) NstepsTridiag   :     250
 43) Vs              :     2.0
 44) Vp              :     1.0
 45) VC0             :     0.2
 46) VC1             :     0.0
 47) DREF            :     1.0
 48) ALPHAVC         :     -3.0
 49) ALPHAVSP        :     -3.0
 50) BONDS           :     [[-1.0, 0, 0], [1.0, 0, 0], [0, -1.0, 0], [0, 1.0, 0], [0, 0, -1.0], [0, 0, 1.0]]
 51) factorhopexci   :     1.0
 52) facts_hop       :     None
--------------------------------------------------------------------
 select a value to change , 0 to stop, filename to read values
```

To change a parameter one enter the corresponding number and then the new value. If, instead of a number, one enters a file-name like *"paramfile"* of a previously saved parameter file (the quotes are important), then that file will be loaded. Such file can be edited with any text-editor.

All energies are given in eV and all distances in Angstroms. The parameters can be divided in three blocs. The first two are atomic Hamiltonian parameters for the base and excited configurations respectively. The first seven atomic parameters are Slater integrals : three (F0,F2,F4) for the *d-d* interaction (couche1), and four for the interaction between the *2p* shell and the *3d* one (couche0_1). The parameter SO_n is the spin-orbit interaction for the n^{th} shell (in this example 0 is*2p* and 1 means *3d* . The one-particle energy of the orbitals is not given in input. Therefore the absolute energy scale of the spectra is arbitrary.

The parameters *Sop_Zero , Sop_Minus, Sop_Plus* are the three components of an external exchange field acting on the *3d* shell. They multiply the three components of the *3d* spin operator $S_z,\ S_{-1},\ S_{+1}$. These factors can be real or complex. Finally the parameter *counterDL* is an energy which multiplies the number of electron on the oxygen orbitals.

The block called *CALCULATION PARAMETERS* contains other interesting parameters. We start discussing the last ten : *Vs, Vp, VC0, VC1, DREF, ALPHAVC , ALPHAVSP, BONDS, factorhopexci, facts_hop.* These parameters are relevant for the Hamiltonian. They describe hybridisation and crystal field.

The hybridisation of the *3d* shell with the oxygen *2p* orbitals is described by the Slater-Koster parameters Vs and Vp for $\sigma = 3\tilde{z}^2 - r^2$ and $\pi = \tilde{x}\tilde{z}, \tilde{y}\tilde{z}$ orbitals, where $\tilde{z}$ is aligned along the bond direction. The hybridisation term is summed over all the bonds given by the *BONDS* variable. The Slater-Koster

parameters are rescaled as $(R_{bond}/DREF)^{ALPHAVSP}$ where R_{bond} is the bond lenght.

The parameters *VC0* and *VC1* work in a similar way and describe the crystal field. They are the energy shift for σ and π orbitals respectively, taking $\tilde{z}$ aligned along the bond direction. The scaling factor is $(R_{bond}/DREF)^{ALPHAVC}$.

The other parameters are described here :

P) *case* : the directory where the Hilbert space has been developped by the *scriviFiles* command.

P) *reduc_1* : the usual Slater integral reduction factor for *3d* shell

P) *reduc_0_1* : reduction factor for $2p-3d$ Slater integrals.

P) *all1* : Lorentzian broadening before El2l3

P) *El2l3* : an energy beween L2 and L3.

P) *all2* : broadening after El2l3

P) *shift* : an energy shift.

P) *npunti* : number of points in the spectra.

P) *dxleft* : the spectra is calculated from dxleft....

P) *dxright* : .... to dxright

P) *temp* : when several ground-state eigenvectors are calculated they are Boltzmann-averaged with a temperature=temp.

P) *erange* : the lowest energy ground-states are calculated which are within an energy range= *erange*. The smaller the *erange*, less ground states are considered, and faster will be the calculation.

P) *tolefact* : Ground states having Boltzmanian weight lesser than tolefact are neglected

P) *shift_invert* : not used.

P) *nsearchedeigen* : The Lanczos diagonalisation of the ground state Hamiltonian searches for nsearchedeigen eigenvectors. (The smaller this number the faster the calculation)

P) *NstepsTridiag* : dimension f the tridiagonalised matrix used for the spectra.

When the program is started, the system parameters are initialised with default values. If two files, named *out36_base* and *out36_exci* , are found in the working directory, then the Slater integrals and spin-orbit-coupling are searched for in such files. These files are not mandatory. If they are not there you will have to fill the entries by hand.

The *out36* files are the output of the Cowan's rcn program. Hilbertxx comes with a gui which allows to run rcn and to generate an out36 file. To run such gui, from the working directory :

```
hilbertxx_directory/cowan/cowan in36
```

There is a short help for every rcn parameter. You can play with them and create the out36 files that you need.

The examples have already their out36 files. So you need to run rcn only for the new cases that you want to study.

To run a simulation you must first create a directory with the Hamiltonian components for the Hilbert space. Let's call it HCD (hamiltonian components directory). You obtain this by the command *scriviFiles* . When you are running *../runthisscript* , and that your working directory has subdirectory name *datas* , give the command :

```
scriviFiles(4,prefix="datas", nhopped=2)
```

the first argument is the minimum occupation number for the 3d shell. Such command creates an Hilbert for the base system and another for the excited one. The base space spans all the configuration having 4, 5 and 6 electrons on the 3d shell. The excited one spans those having 5,6 and 7 electron on 3d and 5 on the Mn 2p shell. The number of electrons on the oxygen 2p orbitals can be 10, 9, 8 and the total number of electron is conserved.

The program consider only those oxygen orbitals that are obtained projecting the 3d orbitals through the hopping operator.

When the HDC has been populated with files created by the above command, you are ready to calculate spectra. The HDC needs to be created only once, and then you can play with parameters.

In the example directory there is already a file with parameter which correspond to a case with strong hybridisation. Such file is named *paramn* and can be loaded using the *set(system)* command.

now you can calculate a spectra :

```
res=system.GetSpectrum()
```

and you can plot the isotropic absorption

```
Plot(Curve(res[0], (res[1]+res[2]+res[3]).imag, Pen(Red), "legend"))
```

Several expectation values can be calculated with the following commands

```
Es, S2s, L2s, SL2, occPs , Szs, Lzs= system.GetESLcounters()
```

Eachs one of the variable Es, S2s, L2s, SL2, occPs , Szs, Lzs is a list of expectation values of a given operator. There is an expetation values for each calculated ground eigenvectors. The operators are the energy, $S.S$, $L.L$, $2S.L$, the occupation of the ligand orbitals, S_z, L_z. The angular operators are restrained to the 3d shell.

2.2 The rixs examples class

The directory hilbertxx/examples/exa_rixs/ contains the code to calculate the 1s-3p rixs.

2.2.1 Mn3+

As in the previous example, you run the command

```
../runthisscript
```

from the example working directory hilbertxx/examples/exa_rixs/Mn3+. You get a resume of the available commands by *help()*. You can calculate 1s-3d absorption spectra, as in the previous example, and RIXS. The absorption can be calculated either for the quadrupolar or for the dipolar interaction. The dipolar absorption can be calculated for centrosymmetric systems. To do this the program calculate a dipolar transition to a virtual 4p state which is projected through an hybridisation hoperator onto the oxygen orbitals. The command to get the dipolar absorption is the same as for the previous example :

```
res=system.GetSpectrum()
```

The quadrupolar absorption can be calculated for a defined polarisation. You define the polarisation giving its 5 components on a L=2 basis.

```
res=system.GetSpectrum([fm2,  fm1,f0,f1, fp2 ])
```

where fm2, fm1,f0,f1, fp2 are five coefficients of the polarisation in the angular momentum representation. For example for xy polarisation [fm2, fm1,f0,f1, fp2] = [1,0,0,0,-1] while to calculate x2y2 [fm2, fm1,f0,f1, fp2] = [1,0,0,0,1].

The *scriviFiles* command has one more argument *spinfixed* . It defaults to zero (no spin constrain). If it is set to one, the generation of the Hilbert space is done under the constraint of fixed total spin S_z. This has in general little effect on the absorption spectra because the spin-orbit coupling is small in the 3d shell. The calculation is faster.

The use of the *set(system)* command is the same as in the previous example. The diffence is that there is some more parameter for the final state (for rixs) and some more for the 4p-oxygen hybridisation that is used to calculate the dipolar 1s-3d transition. These latter parameters are *Dips , Dipp*, which are the Slater-Koster σ and π parameters for 4p-2p hybridisation, and *ALPHADIPO* , which is the analogous of *ALPHAVSP*.

We have provided a parameter file for the study case of Mn : *"paramnH"* The calculation must be runned in the Hilbert space created with the following command :

```
scriviFiles(4, prefix="datas", nhopped=2, spinfixed=1  )
```

The studied ion is in a non-centrosymmetric position and we can calculate both the quadrupolar and dipolar absorption. The two following commands

```
resx2y2 = system.GetSpectrum(  [ 1.0 , 0.0 , 0 , 0 ,  1.0 ] )
resyz   = system.GetSpectrum(  [ 0.0 , 1.0 , 0 , 1.0 , 0 ] )
resdip=system.GetSpectrum()
```

compute the absorption spectra for $x^2 - y^2$, yz and dipolar polarisation respectively.

You can compare the polarisation dependance of the quadrupolar absorption :

```
Plot(Curve(resx2y2[0], resx2y2[1].imag, Pen(Red), "x2y2"),
   Curve(resyz[0], resyz[1].imag, Pen(), "yz"))
```

and you can plot the dipolar isotropic spectra

```
Plot(Curve(resdip[0], (resdip[1]+resdip[2]+resdip[3]).imag, Pen(Red), "dipole"))
```

Now we can calculate a rixs spectra

```
res= system.GetRIXS( polarisationIn=[0.0, 1.0,0.0, 1.0,0.0] ,polarisationOut=[1.0,0.0], ein=26.43,
   eout1=630, eout2=790, dout=0.1, gammain=0.2, gammaout=  [ 0.5 ,  20 , 1.0  ])
```

where we have tuned he energy is tuned to the first absorption peak. The calculation of rixs is done considering only one ground state, the lowest one regardless of the temperature and *erange* parameter.

2.3 The df examples class

The directory hilbertxx/examples/exa_df/ contains the code to calculte dipolar d to f transitions. This example considers a crystal field on the f shell but not hybridisation.

2.3.1 Ho

This example is very similar to the pd case. The only difference is that there are three parameters for the crystal field and no hybridisation.

3 Description of the code

The low level objects are coded in C++ and wrapped in python. The description of the system and the manipulation of basic objects is done in python.

There are two levels of use of Hilbert++ : the developper level and the user level.

At the user level one uses a command line interface which allows to manipulate the parameters of an already defined system.

At the developper level one can create new systems by doing some scriptic programmation in python.

The Hilbert space is spanned by a set of second quantisation base vectors:

$$e = c_i^+ c_k^+ \ . \ . \ . c_l^+ |0\rangle$$

Thee code represents a base vector by a C++ object *eo*. Such object contains two chains of bits : *eo.val* and *eo.signs*. The i^{th} bit of *eo.val* is 1 or zero and is equal to the occupancy of i^{th} one-particle state. The *eo.sign* chain of bits is

used to keep track of fermionic statistics. it is the integral modulus 2 of $eo.val$. If, for example, $c.val = 00010010....$ then $c.signs = 00001110....$

The creation and annihilation operators are represented under the form of one particle base vectors : they are represented by a base vector co whose $co.val$ has only one bit set to one: the bit corresponding to the operator one particle state. The operation of a co creation operator on a base state is implemented in the following way : first the AND bitwise operation is applied to $co.val$ and $eo.val$. When the operator one particle state is already occupied in the vector eo the AND operator gives non-null bit string and the result is set to zero. Otherwise the result is a new vector $reso$ times a multiplicative ± 1 factor. The vector $reso$ is composed by a $reso.val$ which is obtained by the OR bitwisse operation on $co.val$ and $eo.val$. The signs strings $reso.signs$ is the result of the XOR operation on $co.signs$ and $eo.signs$. The multiplicative factor is $+1$ if $(co.val\ AND\ eo.signs) = 0$ otherwise it is -1

Starting from these basic objects, base vectors and creation/annihilation operators, the generic operators and vectors are constructed. An operator is in general a linear combination of creation/annihilation operators or of products of them. A vector is a linear combination of base vectors.

The generation of a Hilbert space is done starting from a seed basis, which is a basis formed by one or few base vectors, and a wanderer, which is an operator which, operating on the basis, creates new basis vectors. The wanderer is applied several times on the basis, making it grow until no new base vectors are found.

The ground state space spans all the configuration having 4, 5 and 6 electrons on the 3d shell. The excited state one spans those having 5,6 and 7 electron on 3d and 5 on the Mn 2p shell. The number of electrons on the oxygen 2p orbitals varies accordingly to conserve the total number of electrons.

The number of oxygen orbitals is in principle $6N_b$ where N_b is the number of bonds. As in the above model Hamiltonian the oxygen orbitals are degenerated we can arbitrarily choses a rotated basis for the oxygen one-particle wavefunctions. The basis that we choose is such that the first 10 basis vector are obtained projecting the $3d$ space on the oxygen orbitals through the hopping operator and applying a Grahm-Schmidt orthonormalization. The remaining vectors span a space that is not connected to our problem and are neglected.

For an absorption study two Hilbert spaces are expanded, one for the ground configurations, and another for the excited ones. The different components of the Hamiltonian are calculated separately and wrote on files , in the form of sparse matrices.

This is done writing a Hamiltonian component (kinetic energy, electron-electron interactions, spin-orbit coupling, etcetera....) as a generic operator. Then such operator is applied to every basis vector i of the considered Hilbert space. The result is a linear combination of a small number of basis vectors to which the vector i is connected through the operator.

The sparse matrix, representing the operator, is a sequence of triples (j,i,c_{ji}) where i and j are two connected basis vector indices and c_{ji} is the matrix element. The total Hamiltonian are recomposed as linear combination of its components. The dipolar and quadrupolar operators connecting the Hilbert

spaces between them are represented as space matrices as well.

To calculate absorption and RIXS spectra we need to compute Eigenvectors, eigenvalues :

$$HX_n = \lambda_n X_n$$

spectra :

$$\langle X| \frac{1}{\omega - H + i\gamma} |X\rangle$$

inversion

$$\frac{1}{\omega - H + i\gamma} |X\rangle$$

To do this we need just we apply the Lanczos Algorithm and the conjugate-gradient method. which are based on the simple multiplication

$$y = Hx$$

We have implemented the Lanczos algorithm with thick restart from Kesheng Wu & Horst Simon [**?**]. The absorption is obtained finding first the ground eigenvectors X_n by the Lanczos method. Then the vector DX_n, where D is the dipolar (or quadrupolar) operator is calculated in the excited Hilbert space. This vector is taken as initial vector at the step zero of the Lanczos tridiagonalisation procedure. A large enough number N of steps is performed to generate the diagonal α_i $(0 \leq i \leq N)$ and off diagonal β_j $(1 \leq i \leq N)$ elements. Te resonance amplitude is then given by the following continued fraction :

$$\cfrac{1}{\omega + i\gamma - \alpha_0 - \cfrac{\beta_1^2}{\omega + i\gamma - \alpha_1 - \cfrac{\beta_2^2}{\cdots}}}$$

The imaginary part of the above expression is proportional to the absorption.

The RIXS signal is calculated in a similar way and one takes as initial step for the tridiagonalisation procedure, the vector

$$D_{out} \frac{1}{\omega_i n - H + i\gamma} |D_{in} X_n\rangle$$

where D_{in} and D_{out} are the dipolar (or quadrupolar) operators for incoming and outgoing photons respectively.

4 Details on the hybridisation components of the Hamiltonian

We consider a metal ion surrounded by oxygens atoms. The total hamiltonian is :

$$H = H_{ato} +$$
$$\sum_b (t_{\sigma,b} d^+_{3\tilde{z}^2-r^2} p_{\tilde{z}} + t_{\pi,b}(d^+_{\tilde{x}\tilde{z}} p_{\tilde{x}} + d^+_{\tilde{y}\tilde{z}} p_{\tilde{y}}) + C.C.) +$$
$$\epsilon_p \sum_b (p^+_{\tilde{x}} p_{\tilde{x}} + p^+_{\tilde{y}} p_{\tilde{y}} + p^+_{\tilde{z}} p_{\tilde{z}}) +$$
$$\sum_b (V_{\sigma,b} d^+_{3\tilde{z}^2-r^2} d_{3\tilde{z}^2-r^2} + V_{\pi,b}(d^+_{\tilde{x}\tilde{z}} d_{\tilde{x}\tilde{z}} + d^+_{\tilde{y}\tilde{z}} d_{\tilde{y}\tilde{z}})) \quad (1)$$

In this expression H_{ato} is the atomic Hamiltonian. The sum $\sum_b$ runs over the metal-oxygen bonds b and the $\tilde{z}$ axis is aligned with the bond direction. For the d metals The hopping is parametrised by the Slater-Koster parameters t_σ, t_π which are rescaled according to the bond lenght. Still for the d metals the electrostatic crystal field is parametrised by the parameters V_σ which is the energy shift of the $3\tilde{z}^2 - r^2$ orbital, and by V_π which is the energy shift of the $\tilde{x}\tilde{z}$ and $\tilde{y}\tilde{z}$ orbitals. The crystal field parameter are rescaled as well. The sum over the spin is omitted in the formulae for conciseness. For the df case there is one more parameter for the crystal field : in total there are three parameter VC0, for z^3, VC1 for $(5z^2 - r^2)x$ and $(5z^2 - r^2)y$, and finally VC2 for zxy and $z(x^2 - y^2)$.

5 Acknoweledgement

The author has benefitted of preliminary discussions with Peter Krueger, now at the Universite' de Dijon, about Lanczos tridiagonalisation. The code, originally pure C++, has been ported to python with the help of Michael Profeta.

6 final comment

The code is provided not only in binary, ready to use form, but also with the sources. This because, analogously to what happens with research, the only interesting code is the one that has not yet been written. Therefore having the source will allow you to better adapt it to your research. The provided examples can nonetheless be very useful and many things can be learned working with the parameters. As everybody does, the author does not know what he will be doing in the future, if expanding the code in some direction or doing something completely different. However if you have some interesting subject we can eventually discuss it together. The distribution does not contain yet example treating neighbouring metal ions like in [2]. Future versions will.

References

[1] Kesheng Wu and Horst Simon, Thick-restart Lanczos method for large symmetric eigenvalue problems. SIAM Journal on Matrix Analysis and Applications Vol. 22, No. 2, pp. 602-616, 2001.

[2] Alessandro Mirone, S.S. Dhesi, G. van der Laan, *Spectroscopy of La0.5Sr1.5MnO4 orbital ordering: a cluster many-body calculation* EPJB 2006, VOL 53; NUMBER 1, pages 23-28