Molecular-crystals-spectra

Leonardo Silvestri

April 6, 2020

Abstract

These notes describe how to use the code "Molecular-crystals-spectra" to calculate optical absorption and emission spectra of molecular crystals. A repository of the code can be found at https://github.com/leosilve/Molecular-crystals-spectra . The theory is described in a separate document.

1 Installation

The software consists of source code written in C++, a makefile and example input/output files. It requires the following additional libraries: `boost_1_57_0` and `clapack`. The makefile supplied with the software works with macOS Mojave Version 10.14.3 and Xcode version 10.2.1. The location of the files, external libraries and the compiler details need to be updated to match the user operating system. Once the external libraries have been installed and the makefile has been updated, the executables can be created with the following terminal command:

```
make -f Makefile all
```

The makefile creates 6 executables: `models.exe`, `modeld.exe`, `bands.exe`, `bandd.exe`, `spectras.exe` and `spectrad.exe`. The last letter of the file name indicates the precision used for calculations in the program: "s" for float and "d" for double.

2 Structure of input files

The software reads input parameters from text files. The value of each parameter is the first item after the colon (":"). The name of the parameter (anything before the colon) is skipped by the code. The order of the parameters is important, because the code will read the type of data it expects. Strings should be placed within double quotes. Anything after the input parameter is neglected, which allows to add comments freely anywhere, except right after the colon.

3 `model`

3.1 Description

The *model* executables calculate absorption or emission spectra of molecular crystals.

3.2 Usage

```
./models.exe model_input_file
./modeld.exe model_input_file
```

3.3 Arguments

model_input_file An input file specifying the type of calculation required.

3.4 Details

The two main files, `models.cpp` and `modeld.cpp`, initialise an `OPmodel` object with a `PhononCloud` basis set and then `solve()` it. The `OPmodel` class is defined for a generic basis set and it is initialised from an input file, such as this one:

bTPB_model.txt

```
Model name:           "bTPB_paper_abs"
Type of calculation: "absorption"
Name of the lattice: "bTPB_lattice.txt"
Basis input file: "bTPB_PC_basis.txt"
Emission input file: "bTPB_emission_input.txt"
Absorption input file: "bTPB_absorption_input.txt"

Real H matrix (1=yes): 0
N. of eigenvalues seeked (0=all): 0
Davydov components to be evaluated (leave blank if all):
```

The input file for an `OPmodel` object contains a list of the following 9 input parameters:

model_name A string with the name of the calculation. It is used to create filenames for eigenvalues, eigenvectors, dipoles, epsilon, emission data.

calc String that specifies the type of calculation. If the string starts with 'e' or 'E' the program calculates emission, if not absorption.

lattice_file String. Name of the file containing info about the crystal lattice. It is used to create a `lattice` object and a `PhononCloud` object.

Basis_file String. Name of the file containing info about the basis set. It is used, together with `lattice_file` to create a `PhononCloud` object.

emission_input_file String. Name of the file containing input to the emission calculation. It is not used if the calculation requested is absorption.

absorption_input_file String. Name of the file containing input to the absorption calculation. It is not used if the calculation requested is emission.

FLAG_REAL Integer. If `FLAG_REAL==1` the program uses solvers `solve_real_nev` or `solve_real_full`, else it uses `solve_complex_nev` or `solve_complex_full`. `solve_real_full` creates the Hamiltonian matrix using the function `Hint` of `PhononCloud` objects and then calls LAPACK's function `syev` to find eigenvalues and eigenvectors. It gives an error if any of the elements is not a real number. `solve_complex_full` creates the Hamiltonian matrix using the function `Hint` of `PhononCloud` objects and then calls LAPACK's function `heev` to find eigenvalues and eigenvectors.

NEV Integer. Number of lowest eigenvalues seeked. If >0 uses solvers `solve_real_nev` or `solve_complex_nev`, else uses solvers `solve_real_full` or `solve_complex_full`. The routines to solve for a limited number of the lowest energy eigenstates require the library ARPACK and are currently commented out because of the complexity of the installation. Please leave NEV=0, unless you install ARPACK and uncomment the corresponding sections of the code.

DCEV List of integers separated by space. List of Davydov components to include in the Hamiltonian matrix. It relies on `compute_BD_basis`. It is only used for absorption calculations.

The initialisation of an `OPmodel` object does:

1 read input parameters;

2 initialise the lattice;

3 initialise the basis set;

4 print the information about the model on a logfile;

5 initialises DC_eigenstates (if `DCEV.size()`$\neq$0) or `all_eigenstates` (if `DCEV.size()`=0) as new vectors with the correct size.

The `solve()` function calls `compute_absorption()` or `compute_emission()` depending on the type of calculation required.

3.4.1 `lattice_file`

The `initialize(string input_file)` function reads the input file, initialises the variables and then calculates the list of required nearest neighbours. The parameter `lattice_name` is defined as the lattice input file name (without extension). The input file for a `lattice` object contains a list of all the other input parameters. This is an example of input file:

bTPB_lattice.txt

```
------------------------------------------------
Monoclinic beta (Girlando) TPB crystal (bca* coordinates)
------------------------------------------------

Crystal axes (each row is an axis expressed in orthogonal coordinates):
8.63401 0.0 0.0
0.0 24.4801 0.0
0.0 -1.2051 9.66109

Number of molecules per cell: 4

Number of crystallographic species: 1

N. of symmetry operations: 4

Here we put the list of the symmetry operations with
1. Rotation matrix
2. Rotation axis position (in fractional coordinates)
3. Translation (in fractional coordinates)
4. Molecule mapping

Identity:

1.0 0.0 0.0
0.0 1.0 0.0
0.0 0.0 1.0

0.0 0.0 0.0

0.0 0.0 0.0

0 1 2 3

Screw:

1.0 0.0 0.0
0.0 -1.0 0.0
0.0 0.0 -1.0

0.0 0.25 0.0

0.5 0.0 0.0

1 0 3 2

Inversion:

-1.0 0.0 0.0
0.0 -1.0 0.0
0.0 0.0 -1.0

0.0 0.0 0.0
```

```
0.0 0.0 0.0

2 3 0 1

Glide:

-1.0 0.0 0.0
0.0 1.0 0.0
0.0 0.0 1.0

0.25 0.0 0.0

0.0 0.5 0.0

3 2 1 0

Position of the crystallographic molecules inside the cell
in fractional coordinates (0.25 0.0 0.0):
0.244212 0.0618166 0.944573

Dipole moments cosines (with respect to orthogonal coordinates):
0.0403848 -0.996067 0.0788692

Dipole moments norm (in Debye):
8.29084

Number of atomic sites: 28

-0.634623        4.10598               0.725854        -0.0407099
0.175224        2.97753               0.795953        0.0362062
-0.544035        -4.20763        -0.616044        0.0407332
1.11265               -3.09496        -1.92632        0.0391622
-0.546251        5.10939               1.67704               -0.000510385
1.18598               3.91576               2.788               -0.00116456
1.10277               2.90948               1.83646               -0.0391389
-0.328429        2.21105               -1.57787        0.0210857
0.0147385        1.85628               -0.169171        -0.156865
0.169816        0.589776        0.269129        0.0962887
0.150151        -0.637558        -0.485641        -0.0962745
0.872434        -3.05634        2.13289               0.0294813
0.244035        -3.07477        -0.844238        -0.0362213
0.0287345        -2.13715        1.50197               -0.0210451
1.18794               -4.19145        -2.76181        0.00115474
-0.899203        -1.47638        2.30116               0.0233777
0.143037        -1.89059        0.0439743        0.156864
-1.32551        1.54721               -2.28428        -0.0233948
-0.491201        -5.28507        -1.47876        0.000501756
0.354276        3.24132               -2.22601        -0.0295081
0.076783        3.57327               -3.5343               -0.0030522
0.810174        -3.25886        3.49506               0.00305927
0.351393        5.00758               2.71065               -0.0513954
-1.6098               1.89315               -3.59605        -0.000174416
-0.942384        -1.68555        3.67059               0.000186981
-0.905682        2.89671               -4.22384        -0.0303105
0.374271        -5.28037        -2.54575        0.0514477
-0.0873343        -2.56559        4.27093               0.0302484

MODE: "M" (Case 'F' -> Finite_sums; case 'S' -> Single_layer_sums; 'M'-> Mixed finite charge dist + Ewald; default -> Ewald_sums
MOL_MOL_INT: "T" (Type of interaction "N"= nearest neighbours; "T"=screened distributed transition charge; default=screened_dipo
NON_SCR_RADIUS: 0.0
RMAX: 50.0  (radius of the sphere used for finite sum calculations)
KLIGHT: 0.0 0.0 0.0016  (k vector of incident light, used to compute bands at the BZ center in the 3D grid mode, corresponds to

Static epsilon:
2.0 0.0 0.0
0.0 2.0 0.0
0.0 0.0 2.0

------------------------------------------------------------------------
Nearest neighbours (NN) interactions, used if mol_mol_int is set to 'N'
```

```
------------------------------------------------------------------------
NN_int N_MOL: 0    (must be equal to the number of molecules in the unit cell, if different NN interactions are set to 0)
NN_int size:  0    (number of NN interactions for each unit cell molecule)
NN_int interactions for each molecule (in eV):

(here you must supply a list of all the interaction energies of each unit cell molecule with all the relevant NN molecules,
the list must contain N_MOL*size energies)
```

This is the list of the input parameters for a `lattice` input file:

CELL Three rows of three real numbers separated by space. Crystal axes in units of Angstroms (each row is an axis expressed in orthogonal coordinates). These parameters are also used to calculate the cell volume `VCELL`.

N_MOL Integer. Number of molecules per cell

N_SP INteger. Number of crystallographic species

N_SYM_OP Integer. Number of symmetry operations

SYM A vector containing `N_SYM_OP` symmetry operations, i.e. `sym_op` objects. For each symmetry operation the code expects a colon ":" and, right after it (in this order):

1. Rotation matrix. Three rows of three real numbers separated by space.
2. Rotation axis position (in fractional coordinates). A list of three real numbers separated by space.
3. Translation (in fractional coordinates). A list of three real numbers separated by space.
4. Molecule mapping, i.e. a list of `N_MOL` integer numbers, where the order represents the starting molecule type and the number the ending molecule type.

RHO `N_SP` rows of three real numbers separated by space. Position of the crystallographic molecules inside the cell in **orthogonal** coordinates. This is calculated from the file input, which must contain the position of the crystallographic molecules inside the cell in **fractional** coordinates. Only one vector per crystallographic species is needed. The other positions are calculated using the symmetry operations.

MOL_DIPOLE Molecular dipole moments. These are calculated from 2 file inputs: (i) dipole moments cosines with respect to orthogonal coordinates (`N_SP` rows of three real numbers separated by space); and (ii) dipole moments norm in Debye (`N_SP` rows of one real number). Only one set of data per crystallographic species is needed. The other dipoles are calculated using the symmetry operations.

ATOM and CHARGE Atomic positions with respect to the CoM and atomic "transition" charges. They are calculated from a list of rows in the input file containing 4 elements: 3 coordinates and 1 atomic transition charge (a total of four real numbers per row). The integer number of atomic sites (i.e. of rows) must be provided right after the ":" and before the list itself. The positions and transition charges need to be calculated with a separate software, e.g. Gaussian. A description of the atomic "transition" charge method can be found in (Alessandrini, 2011) and (Vragovic, 2003).

MODE String. Method used to calculate the resonance-interaction matrix for free exciton bands (see Alessandrini, 2011). If =="F","S", or "M" the corresponding method is selected (see below), otherwise the default method is used. This choice is used by `compute_NN()` and by `compute_Ltilde`. The possible values are:

F Finite sums: only includes interactions between molecules within RMAX from each other in 3D;

S Single layer sums: only includes interactions between molecules on the same xy plane within RMAX from each other in 2D;

M Mixed finite charge distribution + Ewald: calculates Ewald sums (see below) and then replaces dipole interactions with transition charge distribution calculations for distances $<$ RMAX.

default Ewald sums calculated following (Philpott, 1973). It includes the long-wavelength term only if `flag_macro` $\neq$ `0`

Note that the difference between single layer (2D) and finite sums (3D) in practice is implemented by calculating the appropriate nearest neighbours.

MOL_MOL_INT String. Type of interaction between molecules. If ==“N” or “T” the corresponding method is selected (see below), otherwise the default method is used. This choice is used in the finite sums and single layer sums calculations for free exciton bands and by the `lattice::interaction` method, which is called by `PhononCloud::Hint`.

N nearest neighbours: a list of interactions is supplied and stored in NN_int (see below);

T screened distributed transition charge, calculated as in (Alessandrini, 2011, eq.2)

default screened dipole interactions, calculated using molecular dipoles.

NON_SCR_RADIUS Real number. Radius (in Angstroems) beyond which distributed transition charge interactions and dipole interactions are considered screened (set it to 0 to always screen interactions and very large to disregard screening).

RMAX Real number. Radius (in Angstroems) of the sphere (for MODE=’F’) or circle (for MODE=’S’) used for finite sum calculations. Also used for MODE=’M’ to calculate corrections to Ewald sums due to finite charge distribution.

KLIGHT List of three real numbers separated by space. The k vector of incident light used to compute bands at the BZ center in the 3D grid mode. Corresponds to $\hbar\omega = 1970|\mathrm{K}LIGHT|$.

epsilon_0 Three rows of three real numbers separated by space. Static epsilon used for screening.

NN_int Nearest neighbours interactions. They are calculated from 3 inputs in the lattice input file:

N.Mol Integer. Must be equal to the number of molecules in the unit cell, if different NN interactions are set to 0. Can be probably removed as a separate input in the future, because is always equal to N_MOL.

size Integer. Number of NN interactions for each unit cell molecule

interactions A list of N.Mol*size real numbers. A list of all the interaction energies of each unit cell molecule with all the relevant NN molecules; this list must contain N.Mol*size energies. The reading loop is for (i=0; i<get_N_MOL(); i++) for (j=0; j<NN_int[i].size(); j++) infile >> NN_int[i][j].

3.4.2 Basis_file

At the moment the only type of basis that is implemented and tested is the `PhononCloud` basis set. The `initialize(string input_file, string lattice_file)` function reads the input file, initialises the variables and then calculates the basis set. The parameter `prob_name` is defined as the filename (without extension), but it is only used when printing the logfile. The input file for a `PhononCloud` object contains a list of all the other input parameters. This is an example of input file:

bTPB_PC_basis.txt

```
-----------------------------------------
Input data for each run
-----------------------------------------

Electronic molecular transition energy (in eV): 3.6
NMODES: 1
Lambda: 0.87
vib_En (in eV): 0.1735896
MAX_VIB (for each mode): 4
FLAG_2D: 0
CLOUD_RADIUS: 10
NP : 1         (only 1 to NP-particle states are included)
EXP_FAC: 0
```

```
FLAG_DW: 0  "DW_data_2.txt"
```

This is the list of the input parameters for a `PhononCloud` input file:

elec_En A real number. Electronic transition energy (in principle it could be different for each species, but we only read one value at the moment)

NMODES Integer. Number of intramolecular vibration modes (phonon modes). It can only be 1 or 2.

lambda_0 and lambda_1 List of NMODES real numbers separated by space. Huang-Rhys factor for each phonon mode. NMODES values must be supplied. Only lambda_0 is used if there is only 1 phonon mode.

vib_En_0 and vib_En_1 List of NMODES real numbers separated by space. Vibration energy for each phonon mode. NMODES values must be supplied. Only vib_En_0 is used if there is only 1 phonon mode.

MAX_VIB List of NMODES integers separated by space. A vector containing the total maximum of vibrations per exciton for each mode. NMODES values must be supplied.

FLAG_2D Integer. If ==1 the phonon cloud is 2D, i.e. nearest neighbours are calculated on the xy plane (using `lattice::compute_NN`). Else the phonon cloud is calculated in 3D. It is also used in `PhononCloud::Hint` to set `lmax=0` when `FLAG_2D==1`.

CLOUD_RADIUS Real number. Maximum radius of the phonon cloud in Angstroms. It is used when calculating nearest neighbours with `lattice::compute_NN` and in `PhononCloud::Hint` to calculate `nmax`, `mmax` and `lmax`.

NP Integer. Only states with 1 to NP-particle states are included in the basis set. If NP==1, then CLOUD_RADIUS is set to 0.

EXP_FAC Real number. Factor appearing in the calc of MAX_VIBS (vector of vectors with max n. of vibrations at each NN for multiple modes) according to `MAX_VIBS[i][j]=max(int(get_MAX_VIB(i)` $\times$ `exp(-get_EXP_FAC()norm_2(CLOUD_NN[0][j].n)/FLOAT(pow((*latticePtr).get_VCELL(),FLOAT(0.333)))))) , 1);`.

FLAG_DW and DW_datafile If FLAG_DW==1 considers a symmetric well vibrational potential for the ground state molecule and reads its parameters from `DW_datafile`. The double well part of the code has not been tested.

The initialisation function also computes the basis set with `compute_BASIS` and stores it in `BASIS`, which is a vector of `ME` objects. `PhononCloud` objects also have the function `compute_BD_BASIS`, which is called by the absorption calculation if DCEV is specified. Note that BD_BASIS is a vector of vectors of `MPSTATE` objects.

3.4.3 emission_input_file

Emission spectrum is calculated for a thin slab of the material, assumed to be laying on the xy plane, with z being the normal to its surface. Emission is calculated in the weak electronic coupling approximation using the theory described in the theory notes. Emission calculations are only implemented for the following cases: for one phonon mode, the basis set can only contain up to two-particle states; for two phonon modes the basis set can only contain up to one-particle states. An error message is returned if the basis set does not satisfy these criteria. Emission is calculated by the function `OPmodel::compute_emission()`, which relies on the function `PhononCloud::compute_emission(emitting_state, em_dir)`. Input files are similar to the one here:

bTPB_emission_input.txt

```
-----------------------------------------
Input data for emission computation
-----------------------------------------
Each data must be placed right after the corresponding "double dot" sign
The z axis is normal to the sample surface

hw_min: 2.5
hw_max: 3.7
hw_step: 0.002
width: 0.02
T: 0 50 100 200 300
em_dir: 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0
kgrid: 4 4 4
```

This is the list of the input parameters for an `emission_input_file`:

hw_min Real number. Minimum energy of the calculated spectrum in eV.

hw_max Real number. Maximum energy of the calculated spectrum in eV.

hw_step Real number. Resolution of the calculated spectrum in eV.

width Real number.

T_vec List of real numbers separated by space. List of temperatures (in Kelvin) for which emission spectrum is calculated.

em_dir_vec List of 3*n real numbers separated by space. List of n directions along which emission is calculated. Each direction is determined by the three components of a vector (along x,y,z).

n0, n1, n2 List of 3 integers. Number of points along k_x, k_y, k_z used for calulcations in the wave vector space.

`compute_emission()` does `read_emission_input_data`, calls `solve(flag_macro=1)`, writes emission intensities to file, computes emission spectra and writes spectra to file.

3.4.4 absorption_input_file

Absorption spectrum is calculated for a thin slab of the material, assumed to be laying on the xy plane, with z being the normal to its surface. Absorption is computed starting from the dielectric tensor calculated from the dipoles and eigenvalues produced by the `compute_absorption` function. The experimental spectrum is calculated using a transfer matrix approach and the input parameters specified in the input file, as in this example:

bTPB_absorption_input.txt

```
-----------------------------------------
Input data for spectrum computation
-----------------------------------------
Each data must be placed right after the corresponding "double dot" sign
The z axis is normal to the sample surface

------------ Experimental configuration

Sample thickness (micron): 0.001
Incident angles (degrees): 0.0
Angle of the incidence plane with xz plane (degrees): 0.0
Refractive index of first medium (the one from which light is coming): 1.0
Refractive index of third medium (the one to which light is exiting): 1.0

hw_step: 0.01 (in eV)
epsilon_inf: 2.1
gamma_coeff: 0.1
```

This is the list of the input parameters for an `absorption_input_file`:

d_slab Real number. Thickness of the sample in micron.

inc_angle Real number. Incident angle in degrees.

gamma_angle Real number. Angle of the incident plane to the xz plane in degrees.

na Real number. Refractive index of first medium (the one from which light is coming).

nf Real number. Refractive index of third medium (the one to which light is exiting):

hw_step Real number. Resolution of the calculated absorption spectrum in eV.

epsilon_inf Real number. High frequency value of the dielectric constant.

gamma_coeff Real number. Damping factor corresponding to the imaginary part of the dielectric constant. It is added to each diagonal component of the dielectric tensor to reproduce absorption linewidth.

`compute_absorption` calls `solve(flag_macro=0)`, prints eigenvalues, eigenvectors and dipoles, From here the functions are in the `TransferMatrix.h` `read_spectrum_input_data`, `write_epsilon` and `write_absorption_spectrum`.

3.5 Output

3.6 Example