Absorption Spectrum (NOSE Implementation White Paper 1, ver. 1.0)

Tomáš Mančal

March 19, 2009

Charles University in Prague, Faculty of Mathematics of Physics
Ke Karlovu 5, 121 16 Prague 2, Czech Republic

Abstract

This paper describes implementation details of absorption spectrum calculation in NOSE.

1 Basic terms and theoretical background

Absorption spectroscopy is the simplest and oldest spectroscopic method available. The measured absorption coefficient is related to linear response function $S^{(1)}(t)$ by

$$\alpha(\omega) \approx \frac{\omega}{n(\omega)} Re \int_0^\infty dt S^{(1)}(t) e^{i\omega t}. \tag{1}$$

The first order response function $S^{(1)}(t)$ is defined as

$$S^{(1)}(t) = Tr\{\hat{\mu}\hat{\rho}^{(1)}(t)\} = Tr\{\hat{\mu}\mathcal{U}(t)\hat{\mu}\hat{\rho}^0\} \tag{2}$$

representing all quantities in by their matrix elements on an arbitrary finite Hilbert space with a basis of vectors $|a\rangle$, e.g.

$$d_{aj} = \langle a|\mu|j\rangle, \tag{3}$$

the equation (2) leads to

$$S^{(1)}(t) = \sum_{ijab} d_{jb}\mathcal{U}_{bjai}(t) d_{ai}\rho^0_{ii}. \tag{4}$$

In this notation, we assume an existence of two distinct bands of states, one representing electronic ground- (indices i and j) and one the electronic excited state (indices a and b) of the molecule. The energy level scheme of the model is presented in Fig.

1.1 Simplified model

The simplest model of absorption spectrum is the one where so called *secular approximation* is assumed, i.e.

$$\mathcal{U}_{bjai}(t) = \delta_{ab}\delta_{ij}\mathcal{U}_{aiai}(t), \tag{5}$$

and where so-called *homogeneous limit* is valid, i.e.

$$\mathcal{U}_{aiai}(t) = e^{-\Gamma_{ai}t - i\omega_{ai}t}. \tag{6}$$

The dephasingf constant Γ_{ai} is a real number related to the width of the absorption spectrum, and ω_{ai} is the transition frequency between the state $|i\rangle$ in the ground state band and the state $|a\rangle$ in the excited state band.

2 Implementation details

The calculation of absorption spectrum is decoupled from other parts of the program. The section calculating absorption expects certain quantities to be available to it from the previous execution of the program. In the QME and TDPT-3 modules, all the quantities are available in the global variable space. If external computation by e.g. a GPU are needed, these quantities need to be sent to the computing device.

2.1 Fortran implementation

2.2 C/C++ implementation for CUDA

The main Fortran code of the NOSE calls a C function with the following interface

```
float* nosecuda_polar_1(int* N, float* rU, float* iU,
                        float* d, float* rho0)
```

The arguments are

```
/*
   N[0]  number of levels in the ground state band
   N[1]  number of levels in the excited state band
   N[2]  number of the time steps
   rU[n] real part of the coherence evolution superoperator.
         The relation between the superoperator indeces b,j,a,i and t and
         the index n is
            n = i + N[0]*(a + N[1]*(j + N[0]*(b + N[1]*t)))
   iU[n] imaginary part of the coherence evolution superoperator
   d[n]  transition dipole moment elements. The relation between
         n and the indices a, i of the states involved in the
         transition is
               n = i + N[0]*a
   rho0[a]  diagonal elements of the equilibrium ground state
            density matrix
*/
```

Based in these arguments the function returns pointer to an array of the type `float` which contains the time evolution of the response function $S^{(1)}(t)$ of Eq. (2) .

The line of code which is summed over all state indeces for each value of the index `t` can look e.g. like this

```
/* t is set before */

for (i = 0; i < N[0]; i++) {
  for (j = 0; j < N[0]; j++) {
     for (a = 0; a < N[1]; a++) {
       for (b = 0; b < N[1]; b++) {
         /* real part of the result */
         fr = d[b*N[0] + j]*d[a*N[0] + i]*rU[i +
             N[0]*(a + N[1]*(j + N[0]*(b + N[1]*t)))]*rho0[i];
         /* imaginary part of the result */
         fi = d[b*N[0] + j]*d[a*N[0] + i]*iU[i +
             N[0]*(a + N[1]*(j + N[0]*(b + N[1]*t)))]*rho0[i];
       }
     }
   }
 }
```

2.3 Fortran interface to the C code