NOSE Tutorial and Reference Manual

Tomáš Mančal

February 3, 2009

Contents

Chapter 1

Introduction

The abbreviation *NOSE* is supposed to mean *NOnlinear Spectroscopy (made) Easy* or *Not Only Spectroscopy (made) Easy*, but sometimes *NOSE* might be *NOt So Easy*. Especially, it might not be very user friendly. It is, after all, a program that serves very specific needs and there is often no time to make everything work smooth in a general case, when it already works in a special one.

The purpose of this tutorial is to introduce new user to the basics of usage of the *NOSE* program. In the next chapter we discuss the basic usage of the *NOSE* as a command line tool and the philosophy behind model spectroscopies that are used in the calculations. We briefly list the most important keywords of the two required input files – the main configuration file and the spectroscopic structure file (SSF). Because *NOSE* is a very young program the syntax of the configuration and SSF may change from version to version. So always check that you have the recent reference. In Chapter 3 two simple examples of calculation of linear optical spectra – absorption, fluorescence and circular dichroism (CD) – are presented with input file listings and plots of the results. In Chapter 4 we start to discuss elementary theory of absorption spectroscopy and go little more into the details of inner workings of *NOSE*.

All important terms and keywords used in this tutorial are listed in the Index and the end of this text.

A note is due on the history of the *NOSE* development. The *NOSE* started in 2005 as program with the main purpose of calculating non-linear optical spectra, especially, two-dimensional photon echo spectra. The development reached version 0.4.x and then it was decided to start a new clearer code from scratch. The starting version of the new *NOSE* was 0.5.0 - the version you are using is the off-spring of the rewrite effort. The reasons for this change were mostly in programming – the original *NOSE* was not very well conceived and in many ways too ambitious, and consequently hard to maintain. It could however already calculate non-linear spectra, the stage which the new *NOSE* did not reach yet. Before the new *NOSE* can completely superseed the old one, both versions will be available. There are however many differences between the two *NOSE*s, so using them both may be quite confusing – the more so that the documentation of the original, pre 0.5 *NOSE* is very limitted.

The main advantages of the new *NOSE* are its parallelization with *Message Passing Library (MPI),* availability of some more linear spectra (appart from absorption, also fluorescence and circular dichroism) and a simple and clearer syntax. And maybe, the availability of this tutorial makes it even more attractive.

Chapter 2

NOSE: An overview

From the programing perspective, *NOSE* is a *Tcl*[1] script that runs several different Fortran programs. For the user it is a command line tool that takes several input files and calculates optical spectra. It has a very simple command line syntax (if NOSE is not installed on your computer, please refer to Appendix A of this tutorial). To invoke it, just type on the command line

```
> nose
```

which is the same as typing

```
> nose --help
```

Above the character ">" represents the command line propt. In both cases you will receive the same message

```
Info  : Using the nose2 script, version 0.5.9
Info  : executable script /usr/local/bin/nose-home/0.5.9/bin/nose2

Usage: nose [options] [configfile]
Report bugs to: tomas.mancal@mff.cuni.cz

Options:
 --help ......................... prints this message
 --info ......................... prints a mode detailed message
 --tutorial [dirname] ........... generates a tutorial directory
                                  default dirname is "tutorial"
 --debugging configfile ......... runs nose in debugging mode
```

which tells you that the correct usage of the program is

```
> nose conffile
```

where *conffilename* is the name of the *NOSE* main input or configuration file. It was already said that *NOSE* is a *Tcl* script. The input file is also understood and evaluated by *NOSE* as if it were a *Tcl* script. *NOSE* provides input file keywords which are actually *Tcl* functions and evaluates everything in between as if it was a part of *Tcl* language. This enables the user to set variables, calculate values and do lots of other simple tasks inside the input file.

To run even the simplest calculation, you need two input files. The one that is specified on the command line and which contains the settings for the *NOSE* itselfs, and one that contains the

[1] For information on *Tcl* scripting language consult e.g. *http://www.tcl.tk* or Reference [2]

description of the molecular system that is simulated. The two files are described in sections 2.2 and 2.3.

The current version of *NOSE* is still very experimental and consequently many *NOSE* functions are not doing exactly what they are intended to do. An example of this is the fact that to calculate a linear absorption spectrum, we still need to specify certain values for variables that are only needed for calculation of nonlinear spectra, and that some variables described here are actually completely ignored by the current version of *NOSE*. For this my deepest apologies - but there was no time yet to polish everything.

In order to make the things little more transparent we first discuss the overall model we use to simulate molecules in *NOSE*.

2.1 Model Spectroscopies

In every theoretical description of reality we inevitably deal with models. Depending on how good these models are, our description coincides with the reality or not. The *NOSE* uses a model spectroscopy that is very useful in describing molecular systems, especially those that are composed of simpler molecular building blocks, like photosynthetic complexes, molecular aggregates etc.. This model spectroscopy might not be suitable for problems outside the molecular physics and biophysics – e.g. for description of atomic spectra or spectra of solid state materials. In the future, NOSE may include other model spectroscopies, too. Here, the basic notions of the present model will be discussed on an example of bacteriochlorophyll molecules and their aggregates.

Let us suppose we have a molecular aggregate in solution - it may be a self-assembled weakly bound aggregate of bacteriochlorophylls or a complex of the same molecules embeded in a protein matrix. What kind of spectroscopic properties can we expect to find in such a system? First, we will have a look on monomeric molecules, i.e. such molecules that do not cluster into bound complexes. Let us have a *bacteriochlorophyll a*, the most common type of bacteriochlorophyll in photosynthetic bacteria and let us have it dissoved in some solvent. Fig. 2.1 shows its absorption spectrum in comparison with the absorption spectrum of a complex of such bacteriochlorophylls bound into a protein matrix (together with some other molecules).

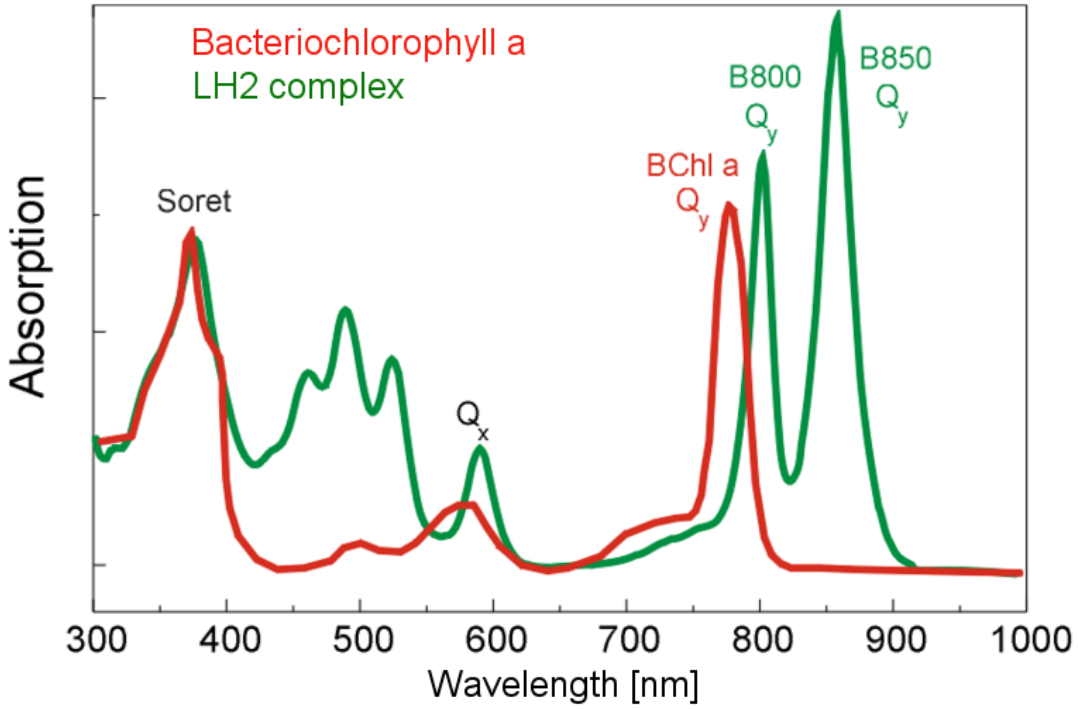

Figure 2.1: Absoption spectrum of *bacteriochrophyll a*.

The absorption spectrum has its characteristic peaks that can be assigned to transitions between certain rather well defined quantum mechanical states of the molecule. It can be found by the methods of quantum chemistry, i.e. by calculating the quantum states of the electrons in BChl molecule, that in the region of wavelength from Fig. 2.1 there are allowed transitions only between few distinct quantum states. Especially, if we concentrate into the region around 800 nm, there is only one transition called Q_Y. This particular transition is characterized by a trasition dipole moment that points in the direction of the red arrow in the left part of Fig. 2.2.

Because no other transitions are available in that spectral region we can limit our descriotion to the two levels of the BChl that have their energy difference almost resonant with the 800 nm light. Our model spectroscopy therefore understands a molecule as a two level quantum mechanical system characterized, sofar, by the transition energy ϵ_{eg} and the transition dipole moment vector $\mathbf{d}_{eg}$. If this would be all there is, we will have an absorption spectrum consisting of infinitely thin sticks. Only light with the frequency of precisely $\epsilon_{eg}/\hbar$ would be absorbed. Indeed, electronic DOF of the BChl is not all that needs to be taken into account. The interaction of the electronic states with the nuclear DOF leads to a broadening of the absorption lines and to a fast energy redistribution among the electronic levels (after optical excitation). This redistribution leads to even broader lines. To enable the descrition of the bath of nuclear DOF on the line shape, *NOSE* allows to specify a model for the so-called electron–phonon interaction. This term is well–known from the solid state physics, where phonons are quazi-particles, or elementary quanta of nuclear latice vibrations. In molecular protein chromophore aggregates we rarely have a regular latice, but the essence of the electron-nuclear interaction is similar. Therefore, the term electron–phonon interaction makes sence even our situation. The parameters that are commonly used to specify the system–bath interaction for molecular systems is so-called reorganization energy λ, and bath correlation time τ_c. Together with a stochastic Brownian oscillator model of the bath, one can derive a closed form equation for so-called bath correlation function in terms of λ and τ_c, and from the corellation function, one can derive even equations for energy relaxation rates and the broadening of the absorption lines of the agregate.

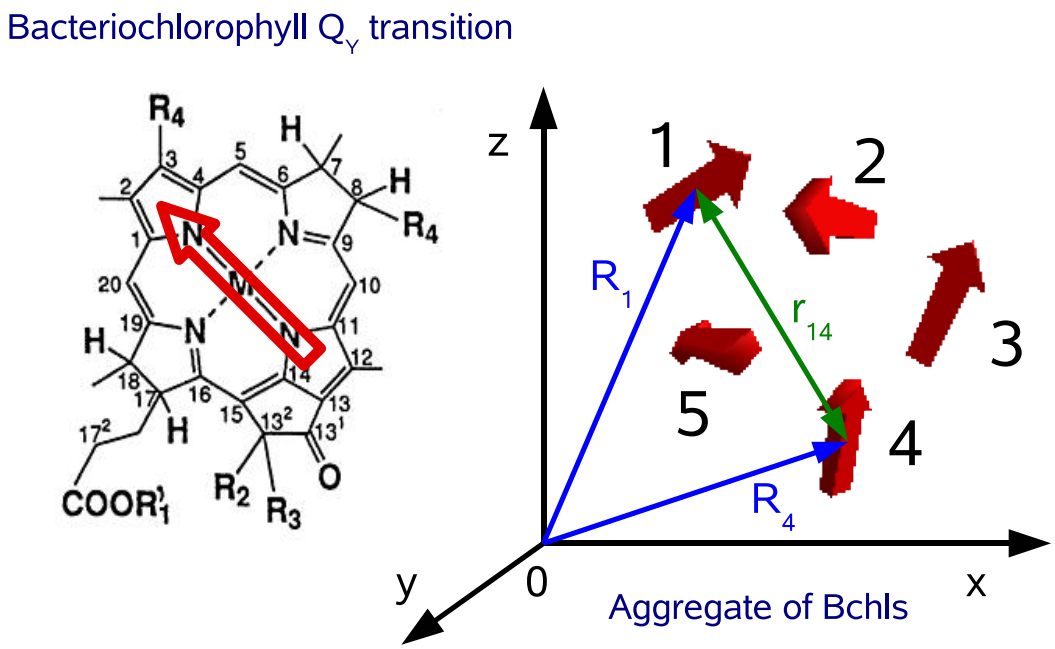

Figure 2.2: Representation of the two-level molecule by orieted dipole.

The details of the theory are beyond the scope of this chapter. However, the most important message you should take away with you from here is, that with our model spectroscopy, each molecule of the aggregate is decribed by its electronic transition energy, electronic dipole moment vector, its position in the aggregate, electronic coupling to other molecules, and depending on the model of the electron–phonon interaction, also by its environment reorganization energy and its its environment correlation time. All these quantities have to be speciefied in the input files, so that *NOSE* knows how to calculate the spectra.

In the next sections you will learn basics about how to run *NOSE* to calculate simple spectra. The details of the theory behind will be explained later is Chapter 4.

2.2 Main Input File

The main input file specifies what to calculate and how. It specifies what are the experimental conditions under which the spectrum you calculate is measured, and what information about the calculation should be written into output files. It does not specify what is the molecular system we calculate on, but it specifies the file name with the definition of that molecular system.

The recommended form of the input file name is

```
somename.conf
```

The extension *.conf* is recommended, because this file indeed describes the *NOSE* configuration for the calculation that follows. The name of the file itself should be something that describes the nature of the configuration, but it is absolutely up to you, what you want to use. The name of the input file does not even influence the name of the output files.

As we already mentioned before, all *NOSE* configuration keywords are *Tcl* commands. As any other *Tcl* commands, the keywords in the main configuration file are case sensitive. They start with lower case letter and if compond of two or more words, every new word starts with an upper case letter, with no space inbetween them. The keywords may take one or more arguments

separated by spaces. Here, we describe the most important keywords. If you write them down into a file as you go through this section, you can build a simple configuration file yourself. An example of a complete configuration file is given in the next chapter.

2.2.1 Choosing a *module* and input file

The first configuration information to specify is the *module*, i.e. the package of programs or scripts that gets used for calculation. Currently, the only working module is the one called *TDPT-3* which stands for *Time-Dependent Perturbation Theory - 3rd order*. It is meant for calculations of the non-linear spectra by perturbative methods. It contains as its subset also the first order calculations that include linear absorption spectra, circular dichroism and fluorescence (actually, the current version only contains the first order calculations, but the third order are in the pipeline). Another somewhat working module is *QME* which stands for *Quantum Master Equation*. This module can also calculate linear absorption spectrum, but its development has not yet reached a stage that allows even a toy calculation. The keyword to specify the module name is

```
moduleName name
```

The mandatory argumet *name* has to be set to *TDPT-3* or *QME* in order to get the program running correctly.

To specify what is the file with the definition of the molecular system you calculate spectra for, we use the keyword *inputFile* as in the following example

```
inputFile ./dimer.ssf
```

Here we are little more specific than before. The line above says that the input file resides in current directory and the input file name is called *dimer.ssf*. The extension *ssf* is recommended (by the author of *NOSE*) and means *Spectroscopic Structure File*. The format of the *.ssf* file will be discussed in the next section.

2.2.2 Specifying the "Grid" and RWA frequency

All calculations in the modules *TDPT-3* and *QME* are performed on a grid. The grid is a set of points from a three-dimensional time interval, on which all functions in the module are evaluated. In general, the third order response functions that are calculated in *TDPT-3* have three time arguments, t, T, and τ, e.g.

$$R_1(t, T, \tau).$$

In a standard notation R_1 is a well defined response function (see e.g. [1]). If we choose a time step for each of the variables, i.e. $d\tau$, dT, and dt, we get a three dimensional grid. We only evaluate functions at times that are integer multiples of the time steps. The additional constraint that is put on this grid in *NOSE* is that we are allowed to choose only one time step (one real number), say Δt and the other time steps are its integer multiples. That is,

$$d\tau = dN_1 \Delta t, \quad dT = dN_2 \Delta t, \quad dt = dN_3 \Delta t \ ,$$

where dN_1,...,dN_3 are integers. In *NOSE*, Δt is specified by the keyword

```
timeStep Dt
```

where *Dt* is a real number - the value of Δt in femtoseconds. The values dN_1, dN_2 and dN_3 are called *grid steps* and are set by the keyword

```
gridSteps  dN1 dN2 dN3
```

The three values are separated by one or more spaces. The last missing piece of information about the grid is its extent, i.e. the number of steps in each dimension. This is given by the keyword *gridExtent* which takes three arguments. To specify that we want to take 1000 steps in τ, 1000 steps in t and 5 steps in T we write

```
gridExtent 1000 5 1000
```

The reason, why the number of steps for T is so different from the others in this example will become clear later when we discuss the two-dimensional spectroscopy.To calculate linear absorption spectrum, only one time interval needs to be specified in priciple. If you specify less then two arguments, the rest will be set to zero.

One of the most important parameters that govern the numeric efficiency of the *NOSE* calculation is the rotating wave approximation (RWA) frequency. This frequency gets subtracted from all fast oscillating components of the calculation, and the calculation is performed only with the remaining slowly oscillating envelops. In order for this to work well the RWA frequency has to be close to the actual optical transition frequencies of the system we study. To set the RWA frequency, use the following syntax

```
rwa   rwaValue
```

2.2.3 Explicit Averaging over Disorder

The information about the nature and magnitude of the disorder in the molecular system under study is explicitelly specified in the *.ssf* file. Here we only set the number of realization of the disorder we want to use

```
realizations  N
```

where N is an integer number. For long calculations it may be advantageous to save the whole calculation time to time to enable restart (e.g. after a computer crash). This is set in the following way

```
restartFreq   f
```

which means that after each f calculations everything will be saved for a restart. The program continues execution and if it is stoped for any reason, it can resume calculation from the latest saved point.

The averaging over disorder is a perfect task for parallel computers. The *NOSE* is parallel computer ready and to enable parallel calculation we simply set the option

```
parallel  yes
```

If for some reason you don't want to calculate in parallel, then set

```
parallel  no
```

2.2.4 Temperature and Pulse Frequencies and Polarizations

Now we set the conditions of the experiment. The temperature in Kelvins is defined by

```
temperature  T
```

In the current version of *NOSE*, the polarization of the laser pulses incident on the sample and their frequencies are only used in the calculations of non-linear experiment and can be ignored in calculation of linear spectra (absorption, CD, fluorescence etc.). They would be set by keywords *detectionPolarization*, *pulsePolarization* and *pulseFrequency*.

2.2.5 Output Files

The output files of the calculation will be written into a directory specified by the keyword

```
outputDir dirname
```

Into this directory, the program will write files with the results of calculation. Which outputs are written is given by the value of the keyword

```
output outputId
```

where *outputId* characterizes one or more output files. The *output* keyword can appear several times and with every occurence its *outputId* is added to the list of outputs.

In this "linear" version of *NOSE* the posible outputs of the *TDPT-3* module are

polar_1 for first order polarization

polar_fl for the first order polarization that characterizes fluorescence

spect_abs for absorption spectrum

spect_fluor for fluorescence spectrum

spect_CD for circular dichroism spectrum

The *QME* module has outputs

polar_1

spect_abs

populations for populations of excited states

coherences for off-diagonal elements of the reduced density matrix

There are also keywords that specify output of some auxiliary quantities that are calculated during the execution of the program and some logging output. These are

```
saveNIS   yes/no
```

which specifies if the program saves the NIS (Nose Input Stream) message it obtained from the *Tcl* script (details are somewhat technical) and

```
saveGoft  yes/no
```

which specifies if the line broadening functions (defined in the *.ssf*) are to be saved.

That's all what a minimum *.conf* file has to include for a succesfull calculation of linear spectra. Because the

2.3 Spectroscopic Structure File Format

Here we will discuss a minimal version of the *.ssf* file that can be used with the current version of *NOSE*. As in the *.conf* file, *Tcl* comments starting with # character and *Tcl* commands are allowed in the *.ssf* file. The keywords in this file are in all upper case letters, taking arguments separated by spaces. Some keywords come in pairs with prefixes *BEGIN_* and *END_*. They enclose certain groups of keywords of similar purpose. Only the content enclosed in between the pair of keywords

```
BEGIN_SSF
...
END_SSF
```

is analyzed, everything else is ignored, i.e. the *BEGIN_SSF* marks the begining of the SSF information.

There are some general purpose keywords that usually appear at the begining of the file. In the current version of *NOSE* you can e.g. set the used units of energy by

```
UNITS_ENERGY  units_id
```

The choice of *units_id* is *wl* for wave-length in nanometers, and *wn* for wave-number in reciprocal centimeters. The units of time are hardwired to be femtoseconds

2.3.1 Creating blocks

All information about the excited states of a molecular system that we study is concetrated into blocks. Each block is supposed to represent a set of chromophores that are strongly interacting, wheres the interaction between blocks is deemed to be weak. Even if you specify a zero coupling between transitions in a block NOSE assumes there is at least some interaction between them. Altough it really calculates absorption (and non-linear) spectra with coupling equal to zero, it e.g. assumes that fluorescence proceeds from the thermalized state of the block. Strictly speaking, if there was zero coupling, such a state would not established itself. It is necessary to keep in mind this behaviour for analysis of fluorescence (and other spectra).

There can be one or more numbered blocks in the input file, inclosed in the pair of keywords

```
BEGIN_BLOCK  block_nr
...
END_BLOCK
```

Inside the block we describe the chromophores using the *TRANSITION* keyword. Only two level chromophores are allowed in this version of *NOSE,* so the complete information consists of the chromophore *id* within the block, the position coordinates x, y, and z of the transition dipole moment, the direction coordinates dx, dy and dz of the transition dipole moment, the length of the dipole vector dd, the energy *siteEnergy* of the transition, identification number *corrfceId* of the correlation function describing the interaction of the electronic transition with the nuclear DOF, and identification number *disorderId* of the static disorder distribution. The values dx, dy and dz are used to calculate a unit vector of the dipole moment direction so that their absolute value does not matter. The correlation function and disorder will be described later. Here is the TRANSITION syntax

```
TRANSITION  id  x y z  dx dy dz dd siteEnergy corrfceId disorderId
```

To specify coupling between two chromophores we use the COUPLING keyword

```
COUPLING id1 id2 couplingEnergy
```

where *id1* and *id2* are the ids of the two chromophores in interaction and *couplingEnergy* is the coupling energy.

If more than one block is specified inside the input file one can also specify the coupling energies between chromophores from different blocks. To that end an *interblock* has to be created following the syntax below

```
BEGIN_INTER_BLOCK  bId1 bId2
...
END_INTER_BLOCK
```

where *bId1* and *bId2* are ids of the two blocks you are specifying the coupling between. To set the value of the coupling use the *COUPLING* keyword as in the normal block. Now however, the first chromophore id means a chromphore from the block *bId1* and the second id means a corresponding chromophore from the block *bId2*. Although the value of the coupling is not limitted, it is always treated as small inside the *NOSE* program. A different kind of perturbation theory is used to treat couplings inside and outside the blocks.

2.3.2 Correlation functions

Energy gap correlation functions describe the interaction of the electronic DOF with the nuclear bath of modes. The structure of the specification is

```
BEGIN_CORRF corrfId
  BEGIN_MODE  nr
  ...
  END_MODE
  ...
END_CORRF
```

There can be more than one mode. At the moment only one type of mode, an overdamped Brownian oscillator mode, is available

```
BEGIN_MODE  nr
  MODE_TYPE              BROWNIAN
  CORRELATION_TIME       tc
  REORGANIZATION_ENERGY  en
END_MODE
```

Here, *tc* is the correlation time and *en* is the reorganization energy of the mode.

2.3.3 Disorder

Slow varying local interactions of molecules with their surroundings lead to differences in the values of their site energies, inter-chromophore couplings etc. *NOSE* currently enables you to specify a general disorder distribution of any parameters using the *BEGIN_DISORDER* and *END_DISORDER* keywords

```
BEGIN_DISOREDER nr
  TYPE type
  ...
END_DISORDER
```

The keyword *TYPE* enables to specify the type of disorder. The only currently allowed value of the *TYPE* is *GAUSSIAN*. The Gaussian disorder type takes one further keyword (*WIDTH*) which gives the HWFM of the Gaussian distribution. The disorder specification is fairly general, but NOSE knows only energetic disorder in its latest version. The index of the disorder you specified can be placed as the last argument of the *TRANSITION* keyword.

Chapter 3

Running Your First Calculations

3.1 Absorption and Fluorescence of a Monomer

The simples example of a spectra that can be calculated in *NOSE* is a linear absoption and fluorescence spectrum of a two level molecule in solution or gas. The description of the molecule itself will be given by a *.ssf* file. The configuration file can look like the one called *mono.conf* with the content listed below. Notice the information you can put into the configation file using comments. It is safer to use ;# when the comment does not start at the begining of the line.

```
#
# Absorption and Fluorescence of a monomeric chromophore
#
moduleName TDPT-3
inputFile  ./mono.ssf
units wl            ;# all energies are given in wavelength
#
temperature  300
#
timeStep   1.0      ;# time step of one femtosecond
gridSteps  4    1 1 ;# only the first grid step is used
gridExtent 1000 1 1 ;# only the first extent is used
#
rwa 800             ;# rotating wave approximation
                    ;# frequency coincides with the
                    ;# monomeric transition in mono.ssf file
#
realizations 1000   ;# 1000 disorder realizations
restartFreq  500    ;# save results in the middle
#
parallel yes
#
outputDir ./out     ;# output into "out" directory
#
output  spect_abs   ;# output absorption spectrum
output  spect_fluor ;# output fluorescence spectrum
#
# end of file
#
```

The *mono.conf* file requires the existence of an input file *mono.ssf*. The following is an example of simple definition of a monomeric system

```
#
# Spectroscopic structure file for a simple monomer
#
BEGIN_SSF
#
UNITS_ENERGY  wn
#
BEGIN_BLOCK 1    ;# just a sigle block with a single transition
  TRANSITION  1   0.0 0.0 0.0   1.0 0.0 0.0  1.0 12500  1  1
END_BLOCK
#
#
# Definition of correlation function 1
BEGIN_CORRF 1
  BEGIN_MODE 1
    MODE_TYPE                  BROWNIAN
    REORGANIZATION_ENERGY 100
    CORRELATION_TIME      100
  END_MODE
END_CORRF
#
#
# Definition of the disorder 1
BEGIN_DISORDER 1
  TYPE GAUSSIAN
  WIDTH 100
END_DISORDER
#
END_SSF
#
# end of file
#
```

The above listing specifies a single transition of 12500 cm^{-1} and a correlation function with reorganization energy 100 cm^{-1} and coherence time 100 fs.

To run the simulation, just type

```
> nose mono.conf
```

The output should give you information about input parameters and some details of the simulation process, including time it took to calculate different stages. In the *mono.conf* file, we have asked the program to calculate absorption and fluorescence spectra, and to save them in a directory *out* in files *spect_ abs.dat* and *spect_ fluor.dat*, respectively.

The output of the NOSE program for these two input files is shown on Fig. 3.1 .

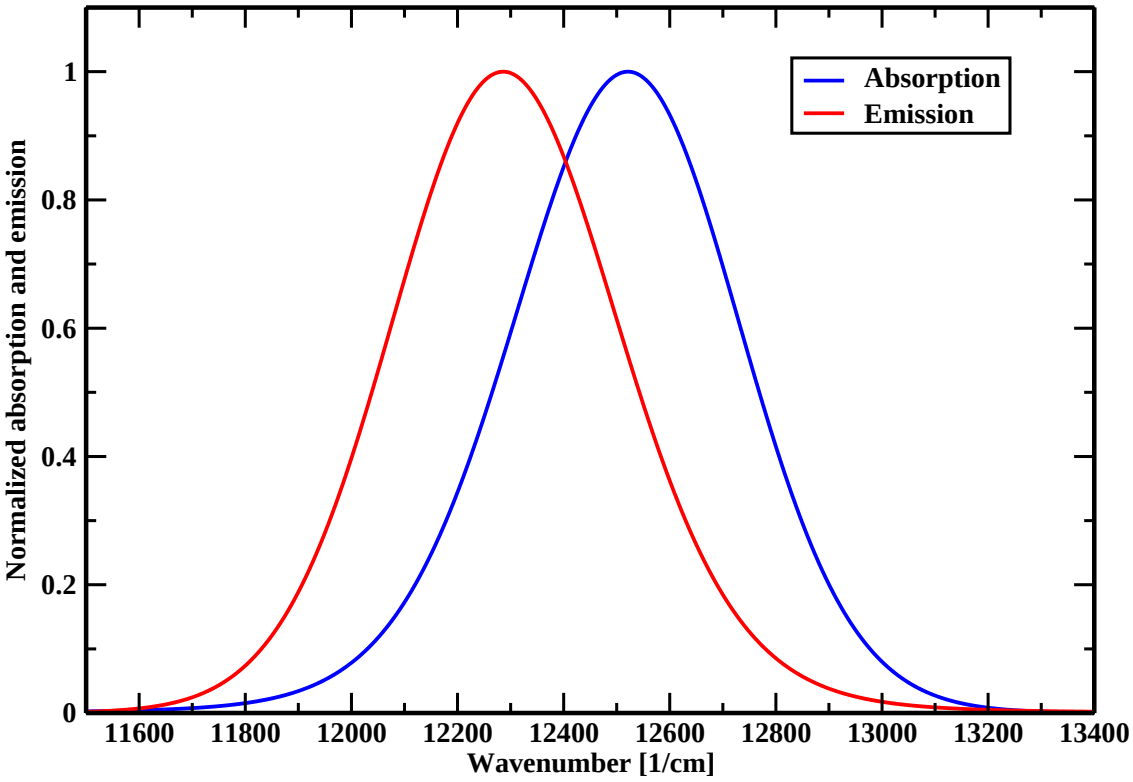

Figure 3.1: Linear absorption spectrum of a monomer

It shows two lines, both normalized to one, representing the absorption (blue) and fluorescence (red) spectrum of a monomeric two-level molecule. The interaction of the molecule's electronic transition with its surrounding (e.g. solvent) is characterized by a single overdamped Brownian mode with reorganization energy $\lambda = 100$ cm^{-1} and a correlation time $\tau_c = 100$ fs. The static disorder is Gaussian with the HWFM equal to 100 cm^{-1}. The transition was put into the origin of coordinates and its dipole moment points in the direction of the x axis.

According to what is known about absorption and fluorescence spectra, the difference between their maxima should correspond to so-called Stokes shift, which amount to two times the reorganization energy, i.e.

$$S = 2\lambda.$$

The spectra on Fig. (3.1) approximately fullfil this condition. In the next Chapter, we will discuss the question why S is not exaclty 2λ in this case.

3.2 Analyzing the Output

The Figure 3.1 has been created by the program called *xmgrace* that is freely available from the Internet and comes with most Linux distributions. The data for this plot can be found in the subdirectory *out* of your simulation directory, in files *spect_abs.dat* and *spect_flour.dat*. We have specified the directory name in the *mono.conf* file, and the name of the output files was chosen by *NOSE* automaticaly. If you open the output files by a text editor, you will find two columns of data looking something like this:

```
11200    0.02345
11345    0.04556
...
```

The first column represents the frequencies in cm^{-1} and the second corresponds to the values of absorption coefficient at the corresponding frequency, normalized to so that the absorption maximum is one. You might wonder how does NOSE choose the values of the frequencies at which the absorption spectrum gets evaluated.

Chapter 4

Theoretical primer: Absorption spectrum of a two-level molecule

A two level model of a molecule enables us introduce some important concepts in our model spectroscopy. The reader needs to be acquinted with quantum mechanics to be able to follow the derivations in this chapter. The theory presented here is by no means the most general theory of the two-level molecule spectroscopy. We present a model that describes our model spectroscopy, i.e. we will be concerned with molecules in solution.

4.1 Quantum Description of Molecules

Our two-level molecule is assumed to have two electronic eigenstates - a ground state $|g\rangle$ and an excited state $|e\rangle$. On top of these two electronic states, one has to take into account also a multitute of possible internal vibrational states and, to include molecule's surrounding, also possible states of the neighbouring degrees of freedom (DOF). Overall we assume, that the total Hamiltonian of the molecule can be written as

$$H_{mol} = [\epsilon_g + T_g(\{P\}) + V_g(\{Q\})]|g\rangle\langle g| + [\epsilon_e + T_g(\{P\}) + V_e(\{Q\})]|e\rangle\langle e|. \tag{4.1}$$

The corresponding energetic diagram is presented in Fig. ... The meaning of the Hamiltonian, Eq. (4.1) is the following. If the molecule is in its electronic ground state, its electronic energy is ϵ_g and its non-electronic energy (containg conributions of the intra- and solvent nuclear modes) is $T_g(\{P\}) + V_g(\{Q\})$. The later component of the energy depends on certain nuclear coordinates Q_i and impulses P_i. The number of the nuclear coordinates can be very large, and we use an abreviation $\{Q\}$ to denote the dependency of V_g on all these coordinates, and correspondingly for impulses and T_g. As you might have guessed, the operators T_g and V_g represent the kinetic and potential energies of the nuclear degrees of freedom, respectively. Similar situation is with the electronic excited state. If the molecule is in the excited state $|g\rangle$, its electronic energy is ϵ_e and the corresponding nuclear kinetic and potential energies are $T_e(\{P\})$ and $V_e(\{Q\})$. When speaking about the potential energy, we will often use the term potential energy surface (PES). A one dimensional representation of the PES of a two-level molecule is presented in Fig. ....

4.2 Semi-classical Description of Light-matter Interaction

Spectroscopy is all about interaction of light with matter. To complete our picture, we have to include Hamiltonian of the light H_T (T stands for *transversal* here) and the light-matter interaction Hamiltonian H_{int}. A general theory of light says that only co-called transveral electric (and magnetic) field corresponds to light (photons) and it also offers a general expression for both H_T and H_{int}. We will not consider details of the quantum theory of light, and we refer interested

reades to standard textbooks [Ref]. In molecular laser spectroscopy, the situation is often such, that the molecules subject to interaction with a rather strong coherent electric field. Lasers produce coherent light, ofthen in form of short pulses and the intesities of light are such that in a good approximation, the quantum opeartors of electric field intensity (of light) can be substituted by the expectation values of this intensity. This enables us to forget the quantum nature of the laser light and to treat the light part of the H_{int} Hamiltonian classically. The consequence of the replacement of the quantum description of the light by a classical one is that H_{int} now depends on a time dependent classical electric field $\mathbf{E}(t)$. We will thus write $H_{int}(t) = H_{int}(\mathbf{E}(t))$.

This is however not the last approximation to be used in our model spectroscopy. We aim to investigate small molecules and their complexes (sizes from around one to several tens of nanometers) with visible and infra-red (IR) light (wavelength of several hundereds of nanometers). As you might realize, there is at least an order of magnitude difference between the size of the molecules and the wavelength of the light.

Chapter 5

Spectra of Excitonic Complexes

5.1 Absorption, Fluorescence and Circular Dichroism of a Dimer

Here we add more detail to our model.

Appendix A: *NOSE* Installation

Here is a short guide to *NOSE* installation. The *NOSE* program is available in source code and the installation works with *GNU* compiler package *gcc*, in particular *NOSE* needs *gfortran* or *g95* to compile. So far, not much testing on other platforms than Linux was done, so sticking to Linux is the thing to do if you plan to be *NOSE* user. Most of the libraries the program requires are present in the source code, with the exception of *LAPACK* library, which is usually shipped with Linux system and the so-called *Message Passing Library (MPI)* that is also increasingly present on Linux systems. If *LAPACK* or *MPI* are not present on your system, its time to talk to the administrator or install them yourself.

Let us suppose that you have downloaded (from the future *NOSE* website) the installation tar ball called

```
nose-0.5.9.tar.gz
```

and have it ready in your home directory on a suitable Linux machine. The version number (0.5.2 here) can be different. In general, the higher the number, the better. We will assume here that you want to install *NOSE* just for yourself, do not have root permissions and let the source directory reside in your home directory. In that case, unzip and unpack the tar ball by typing

```
> gunzip nose-0.5.9.tar.gz
> tar xzf nose-0.5.9.tar
```

and enter the nose-0.5.6 directory that is no present in your home directory

```
> cd nose-0.5.9
```

In this directory you find welth of files and directories. They don't need to bother you, with the exeption of the *doc* directory that contains documentation, including this guide. Another usefull file is the *INSTALL* file that describes this installation procedure.

Now you will perform the three basic steps of a source code installation, the configuration, compilation and installation of the package. First we configure the package to be installed in the home directory

```
> ./configure --prefix=$HOME
```

If the configuration runs correctly, we should see a rather long listing starting with

```
checking build system type... x86_64-suse-linux
checking host system type... x86_64-suse-linux
checking for gcc... gcc
...
```

and ending with

```
...
config.status: creating src/pert/modules/tdpt3/Makefile
config.status: creating src/pert/modules/qme/Makefile
config.status: executing depfiles commands
```

Depending in your machine you may find some differences in the listing. Most probably system will not be *x86_64-suse-linux*, but something else. If no error occures you can now compile the program by typing

```
> make
```

Now you wait for compilation to finish, which may take some time. If the *make* does not give you any error messages and leaves with

```
make[1]: Leaving directory '/home/tomas/workspace/nose'
```

As the last line of the listing, you compiled the program succesfully. Testing can be done by typing

```
> make check
```

in the installation directory. This make task runs a series of simple calculations and compares their results with pre-calculated data. If you compiled *NOSE* to run in parallel, make sure that the parallel environment is ready before you make checks.

Now you only need to install it by typing

```
> make install
```

The last thing before running *NOSE* for the first time is to create a link to the *NOSE* master script. *NOSE* has a way to keep several versions ready for your choice and for that it only needs to have a single master script in your PATH. If you had *NOSE* installed previously you only need to set the newest version to be used by the master script. This you can do by typing

```
> nose --set-version 0.5.9
```

If you want to know what versions are available, use the *–list* option. You can always set any of the available versions. If this is really the first time you install *NOSE*, you might need to create the link to master script by hand. Go to the directory where *NOSE* was installed to and you should be able to find a directory *nose-home* there. Create a soft link by typing

```
> ln -s ./nose-home/nose_0.5.9 ./nose
```

This creates the desired soft link, and if the directory where the *NOSE* was installed to is in you PATH everything should work. Next time you install a new version of *NOSE* you might consider updating the link to the master script, because it can itself change. However, the master script is planned to be untouched for quite some time.

If nothing went wrong in the above steps, you are ready to run *NOSE* on your machine.

Apendix B: List of Configuration Keywords

Here you find *NOSE* configuration keywords in alphabetic order

- inputFile {*filename*}

  Specifies the name and location of the input file with the SSF definition of the simulated molecular system.

  Default value of *filename*: *nose.conf*

- moduleName {*name*}

  Specifies the *NOSE* module to be used for the simulation.

  Values of *name*: *TDPT-3* (default), *QME*

- timeStep {*dt*}

  Elementary time step of all time propagations, function evaluations and integrations.

  Value of *dt*: real number (defaut *dt* value is 1 fs)

Appendix C: Spectroscopic Structure File Keywords

General Keywords

- BEGIN_SSF

  Starts the SSF specifications. All keywords found outside the *BEGIN_SSF* - *END_SSF* keyword pair will be ignored.

- END_SSF

  Ends the SSF specification. All keywords found outside the *BEGIN_SSF* - *END_SSF* keyword pair will be ignored.

- UNITS_ENERGY *value*

  Specifies the units of energy used in the input file. The allowed values are *wl* (wavelength in nm), *wn* (wavenumber in cm^{-1}), *eV* (electronvolts) and *ifs* (inverted femtoseconds, or more precisely 2π/fs). The later are the units into which everything will be converted, because they are the internal units of the *NOSE* simulation driver. The units are valid from the point where the keyword appeares in the input file, to its next appearance. Default value is *wn*.

- UNITS_TIME *value*

  Specifies the time units used in the input file. Allowed values are *fs* (femtoseconds) and *ps* (picoseconds). The units are valid from the point where the keyword appeares in the input file, to its next appearance. Default value is *fs*.

Chromophore Block Keywords

Bibliography

[1] Shaul Mukamel. *Principles of nonlinear spectroscopy.* Oxford University Press, 1995.

[2] Brent B. Welch, Ken Jones, and Jeffrey Hobbs. *Practical Programing in Tcl and Tk.* Prentice Hall.

Index