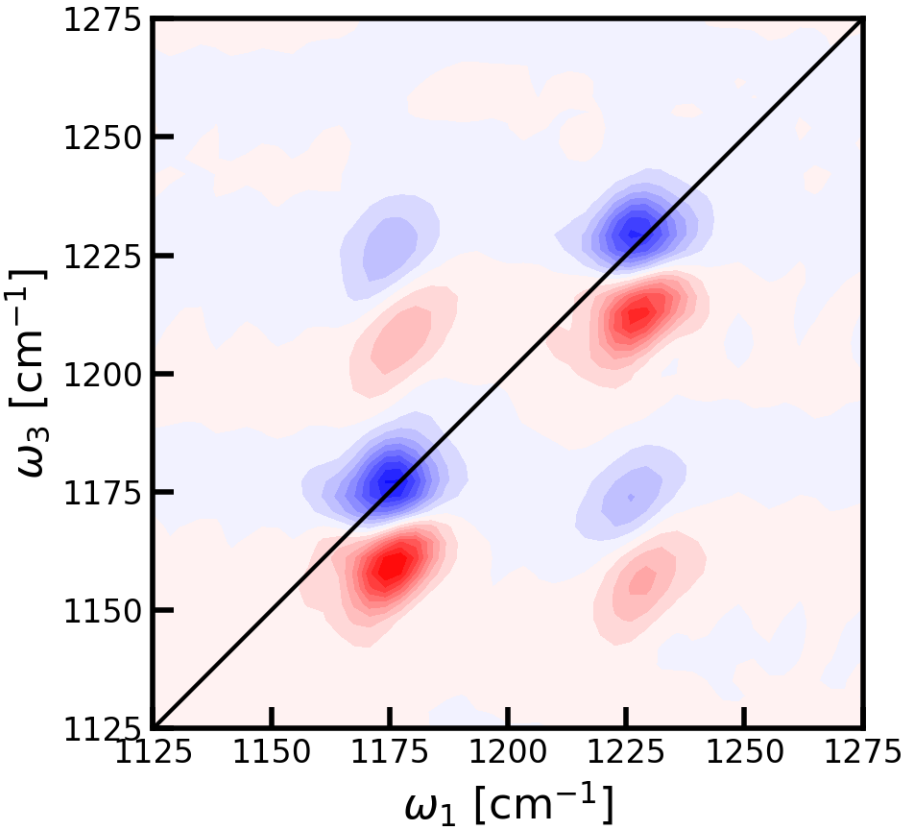
NISE Manual
Version 3.3

Thomas la Cour Jansen
University of Groningen

January 29, 2020



Front cover: © T.l.C. Jansen, 2017

Contents

Chapter 1

Introduction

The main developer of the NISE3 code is Thomas la Cour Jansen. Please, cite the appropriate references [1–5] when publishing work using this code. The code allows the calculation of the linear absorption,linear dichroism, sum-frequency generation, two-dimensional spectra (IR,UVvis, and SFG), population transfer, exciton diffusion and integrated anisotropy using the full nonadiabatic semi-classical numerical integration of the Schrödinger equation approach [2] and the sparse matrix optimization approach [4]. The code allows treating spectra of diverse systems involving intra- and intermolecular energy transfer [1, 3, 6–8], non-Gaussian dynamics [9, 10], surfaces [5], and chemical exchange [11]. This manual is not intended as an introduction to two-dimensional spectroscopy. The user is directed to the references including recent reviews [2, 12–15] and books for more information [16–18].

The code is made available as open source on github.com as link `GHlacour/NISE_2017`. The version available is intended to be friendly for developers to contribute improvements. Changes in the code are at own risk and should be reported clearly in publications. Developers wanting to contribute to the official version of the code are suggested to contact the main developer in advance to ensure that the contribution is not conflicting with that of other developers. More information for developers is given in Chapter 6. Feedback on the program and the manual are welcome via e-mail: t.l.c.jansen@rug.nl.

Current developers include: Thomas la Cour Jansen (main developer), Floris Westerman (cmake and MPI implementation).

The code use wavenumbers for frequencies and times are femtoseconds. The transition dipoles and transition polarizabilities may be given in any desired units.

Chapter 2

Installation and options

NISE has a number of dependencies:

- FFTW3 library, possibly with OpenMP/MPI support. See `http://www.fftw.org/`.
- LAPACK library, often preinstalled. See `http://www.netlib.org/lapack/`.
- CMake v3.10 or higher
- MPI v3 implementation, such as OpenMPI, MPICH (Unix) or MS-MPI (Windows)
- Modern C compiler, implementing a recent OpenMP version
- LaTeX + BibTex distribution if you want to build the documentation. Not required
- DISLIN, python, MATLAB, or gnuplot, for plotting the results using the included scripts

As some FFTW3 installations do not come with the correct CMake compatibility, it might be useful to use vcpkg as package manager for C++ libraries. This package manager is cross-platform compatible.

2.1 Building

In order to build and compile the NISE software, it is recommended to use the CMake build system. This will try to automatically link the libraries and pick the correct compiler settings.

1. Extract the source code files
2. Create a `build` directory in the main folder
3. Run `cmake ..` inside this new `build` directory.
4. If `cmake` was successful, run `make` in the same directory to start compilation.

5. All executables should be available in a new `bin` directory in the main folder.

There are several options you can provide to the `cmake` command in order to customize your build:

- `-DCMAKE_BUILD_TYPE`: By default, this is set to `RelWithDebInfo`. Other options include `Debug`, `Release` and `MinSizeRel`. Refer to the CMake documentation for more information
- `-DGENERATE_DOCS`: When not set, CMake will attempt to compile this documentation only when building a Release build. You can override this by setting this variable to `true` or `false`.
- `-DACCURATE_MATHS`: When set, CMake will compile the code to use accurate mathematics implementations (as is default when using a C compiler). When not set, the compiler will use the fast-math option (`-ffast-math`, `/fp:fast` or equivalent) which may yield upto 2x speed-up at the minor cost of numerical accuracy.

After running CMake, the following build targets will have been provided for `make`:

- `all`: Same as not providing a build target, will build all source code for the program, but will skip the documentation and the examples
- `2DFFT`: Will build the 2DFFT executable, used to process results
- `translate`: Will build the translation utility, used to convert between input formats
- `NISE`: Will build the main NISE executable
- `doc`: Will build this documentation from scratch
- `examples`: Will build the code necessary for the examples, used later in this document.

2.2 Installation Trouble Shooting

If the automatic installation procedure outlined above does not work this section contains a few potential solutions. It is of course important first to verify that the libraries specified in the dependencies are available.

If the FFTW libraries are not detected by the cmake routine the follwing cmake options may be specified by hand:

```
cmake .. -DFFTW_ROOT=/cm/shared/apps/fftw/openmpi/gcc/64/3.3.8/lib
 -DFFTW_LIBRARY=/cm/shared/apps/fftw/openmpi/gcc/64/3.3.8/lib
 -DFFTW_INCLUDE_DIRS=/cm/shared/apps/fftw/openmpi/gcc/64/3.3.8/include
```

The names of the directory locations nust then be changed to the actual locations on the given system. The example abouve is for a fftw library compiled with the gcc compiler.

The program can also be installed on the Mac OSX system. However, the standard compiler does not come with OpenMP support. An OpenMP library must therefore be installed first. This can be done using the homebrew system. Details of the installation of homebrew are given on `http://brew.sh`. Then an OpenMP library as lipomp can be installed with `brew install libomp`. If the version of cmake is also not recent enough a suitable cmake version can be installed with homebrew as well. Finally, the programme can be build following the general instructions above for building, but with useing the command:

```
/usr/local/bin/cmake .. -DCMAKE_C_COMPILER="clang"
 -DOpenMP_C_LIB_NAMES="libomp" -DOpenMP_CXX_LIB_NAMES="libomp"
 -DOpenMP_libomp_LIBRARY="/usr/local/lib/libomp.dylib"
 -DOpenMP_C_FLAGS="-Xpreprocessor -fopenmp /usr/local/lib/libomp.dylib
 -I/usr/local/opt/include"
```

Here, the location of the installed cmake version and the libomp version may have to be changed to match the location when these were installed by homebrew.

Alternatively macports can be used in a very similar way to homebrew. Install libomp with `sudo port install libomp`. The location of omp.h may be different than expected by CMake, which may be fixed with `sudo ln -s /opt/local/include/libomp/omp.h /opt/local/include/omp.h`. or `sudo port install libomp +top_level`.

2.3 Parallelization

NISE is equipped with support for MPI and OpenMP, to provide a tiered parallelization solution. Currently, both the time consuming 2DIR and 2DUVvis techniques support MPI. The linear techniques do not have a parallel implementation as they are generally fast. (Techniques relying on LAPACK including Luminescence may use the MLK OpenMP support for limited speedup.)

For the two-dimensional techniques, it is recommended to understand the implemented approach for parallelization in order to achieve good performance. Each run will calculate a specified number of samples, each for 21 different polarization directions. The calculation time for each polarization direction is determined by the chosen values for `t1max`, `t2max` and `t3max`.

All polarization directions may efficiently be calculated in parallel using MPI, distributed over all registered tasks (more explanation follows later). As long as you have sufficiently many samples, this will scale very well. In general, it is recommended to have a fraction or multiple of 21 as number of tasks, in order to make sure that no cores are simply waiting around after completing their part of the calculations.

Within each polarization direction, loops over the t1 coherence times are parallelized using OpenMP. Due to communication overhead and data sharing difficulties, this does not scale as well as the MPI parallelization. If possible, it is recommended to overprovision your cores, i.e. to make the system spawn more threads than there are cores available. The larger the computation per polarization direction (so higher t2, t3, system size `singles`), the better this part will scale. It is recommended that the number of OpenMP threads is either small compared to `t1max` or that `t1max+1` is equal to an integer times the number of OpenMP threads.

For example, to run 4 tasks with each 12 threads with OpenMPI, use the following command:

```
mpirun -np 4 -x OMP_NUM_THREADS=12 --bind-to socket  /pathToNISE inputFile
```

For cluster computing refer to the manual for the cluster. Special commands as `srun` may be required for `SLURM` systems. The MPI implementation require all input files to be located at a disk available to all nodes.

2.3.1 Efficiency considerations

Some considerations and examples to achieve higher performance:

- As OpenMP parallelization uses shared memory, it is necessary to limit each task to one node. If possible, it is recommended to limit a task even to one socket, or in case of more modern chips, 1 NUMA node. However, this might make the runtime for one polarization direction too high, so it might be worthwhile to trade some efficiency for shorter runtimes.

- It is recommended to overprovision your OpenMP threads by a factor of 2-3. So if a NUMA node has 12 cores, you could pin the threads of this task to this NUMA node and tell OpenMP to create 24-36 threads. Thread pinning differs per OS and more details can be found online. Many popular workload schedulers like SLURM, and some MPI implementations, offer this built-in (for example, `--bind-to socket` in the command above).

- The most efficient workload division also depends on the problem size, for larger problems with fewer samples, the OpenMP scaling is more efficient than for smaller problems with more samples.

- For example, on a machine with 2 12-core Xeon processors (single NUMA node per socket), it is most efficient to run 4 tasks, with 12 threads assigned to each task (overprovisioning). However, for very large problems with only 2 or 3 samples, it might be better to scale down to 2 tasks, each with 24 threads. For smaller problems with many samples, 8 tasks with 6 threads might be better. In general, it is good to do some quick performance tests beforehand.

2.4 Changelog

2.4.1 Version 3.3

Work by Thomas Jansen

- Extended MPI support to 2DIR
- Implemented true two-state behaviour for the *UVvis techniques
- Added linear dichroism
- Changed timing information for two-dimensional calculations to percentage of full calculation based
- Important change in naming of 2DIR sub techniques to contain IR at the end (GBIR, SEIR, EAIR, etc.).

2.4.2 Version 3.2

Work by Floris Westerman

- Added CMake build system and improved cross-platform compatibility
- Added MPI support to offer significantly better scaling across multiple nodes, instead of just a single one
- Improved code efficiency, around 4x speed-up of main algorithm code of *UVvis techniques

2.4.3 Version 3.1

- Included OpenMP support for two-dimensional calculations

Chapter 3

Program descriptions

This chapter contain the description of the various input parameters, file formats, and output files for the NISE code.

3.1 Units

The code use wavenumbers (cm^{-1}) for frequencies and times are in femtoseconds (fs). Users are discouraged from trying to use other units at this may lead to numerical instabilities or unit conversion errors. The transition dipoles and transition polarizabilities may be given in any desired units. The final results will scale accordingly.

3.2 Programs

Overview over programs for calculating spectra and analysis:

translate [Convert Hamiltonian and dipole trajectory between different formats.]

NISE [General code for calculating spectra]

2DFFT [Do the 2D Fourier transform]

3.3 Translate

The Hamiltonian **must** be saved in binary format (GROBIN/SKIBIN) for use in the NISE program. The translate program convert between different formats. The program also allows selecting specific sites in a Hamiltonian file and modification of the fundamental frequencies corresponding to isotope labeling. The input file format is:

InputEnergy [filename]

InputDipole [filename (only needed for GRO/SKI formats)]

InputDipoleX [filename (only needed for MIT format)]

InputDipoleY [filename (only needed for MIT format)]

InputDipoleZ [filename (only needed for MIT format)]

InputAnharm [filename (only needed for SKI format)]

InputOverto [filename (only needed for SKI format)]

InputAlpha [filename (only available for GRO format)]

OutputEnergy [filename]

OutputDipole [filename (only needed for GRO/SKI formats)]

OutputDipoleX [filename (only needed for MIT format)]

OutputDipoleY [filename (only needed for MIT format)]

OutputDipoleZ [filename (only needed for MIT format)]

OutputAnharm [filename (only needed for SKI format)]

OutputOverto [filename (only needed for SKI format)]

OutputAlpha [filename (only available for GRO format)]

Singles [number of oscillators]

Doubles [number of doubly excited states]

Skip [Doubles=Neglect doubly excited states, needed for SKIBIN]

Length [number of timesteps in the trajectory files]

Anharmonicity [A fixed anharmonicity. Only needed for generating GROBIN file from format without double excited states.]

InputFormat [GROBIN/GROASC/MITASC/SKIBIN]

OutputFormat [GROBIN/GROASC/MITASC/SKIBIN]

Modify [Leave this out if you do not wish to modify the Hamiltonian]

Select [Number of amide units to include, if selected number is less than the number of units in the original a list of the units to include should be given on the following line (i.e. 0 1 2 3 5 if 6 units are in the original and we want unit 4 excluded).]

Label [On the following line the for each selected unit the isotope labeling is given (0=native, 1=C13, 2=O18) C13 gives a 41 wavenumber frequency shift and O18 gives a 60 wavenumber shift in this implementation.]

Shift [On the following line for each selected unit a frequency shift can be provided. This can be used for modifying proline or terminal frequencies or for isotope labeling.]

3.3.1 File formats

GROBIN is the binary format needed for use with the NISE program. This contains information on the energy of both singly and doubly excited states and allows for fluctuating anharmonicities. GROASC is a text readable format. MITASC and MITTXT are text formats used by the MIT static spectrum code. SKIBIN is a binary format used by the Skinner group. It contain the singly excited states and a separate file for fluctuating anharmonicities and a file for fluctuating overtone transition dipoles. SPECTRON is a text format used by the Mukamel group SPECTRON code. The GROBIN format is the only format that allows storing the doubly excited states Hamiltonian. This will be generated automatically using harmonic rules and the anharmonicity given by the Anharmonicity keyword.

GROBIN and SKIBIN:

The GROBIN and SKIBIN formats are very similar. The energy file has the following format. For each snapshot in the trajectory they contain first an integer typically containing the number of the snapshot. This integer is not used in the calculation, but can be used for control purposes. Then the single excited Hamiltonian is given in floats as a tridiagonal upper matrix. For the GROBIN format the double excited Hamiltonian might be provided afterwards again in a tridiagonal upper matrix.

The dipole file has the following format. For each snapshot in the trajectory they contain first an integer typically containing the number of the snapshot. This integer is not used in the calculation, but can be used for control purposes. Then the x components of the transition dipole matrix are given in floats, followed by the y and z components. For the GROBIN format the transition dipole matrix for the single to double transitions is then given, again with the x components given first.

The anharmonicity file is only given in the SKIBIN format. For each snapshot in the trajectory they contain first an integer typically containing the number of the snapshot. This integer is not used in the calculation, but can be used for control purposes. Then the diagonal anharmonicities are then given in floats.

The overtone file is only given in the SKIBIN format. For each snapshot in the trajectory they contain first an integer typically containing the number of the snapshot. This integer is not used in the calculation, but can be used for control purposes. Then the overtone transition dipoles are then given in floats. Only the same site overtone transition dipoles are given. Transitions like $\langle 1 \mid \mu \mid 12 \rangle$ are determined by the harmonic rules.

The transition polarizability file which is not available in the SKIBIN format has the following format. For each snapshot in the trajectory they contain first an integer typically

containing the number of the snapshot. This integer is not used in the calculation, but can be used for control purposes. Then the xx components of the transition polarizability matrix are given in floats, followed by the yy and zz components.
GROASC:
The energy file contain one Hamiltonian snapshot for each line. The upper tridiagonal matrix is saved and each number is separated by a space. The transition dipoles are stored in one file. Each line contains one snapshot with the numbers separated by a space. All the x components for one snapshot are stored first followed by the y and z components. Only the single excitation Hamiltonian and the transition dipoles from the ground state to the singly excited state are saved.
MITASC:
The energy file contain one Hamiltonian snapshot for each line. The whole square matrix is saved and each number is separated by a tab. The transition dipoles are stored in one file for each cartesian component. Each line contains one snapshot with the numbers separated by a tab. Only the single excitation Hamiltonian and the transition dipoles from the ground state to the singly excited state are saved.
MITTXT:
The energy file contain one row of the Hamiltonian snapshot for each line. The whole square matrix is saved and each number is separated by a space. Snapshots are separated by an empty line. The transition dipoles are stored in one file for each cartesian component. Each line contains the cartesian coordinates of one transition dipole. The numbers separated by a space. Snapshots are separated by an empty line. Only the single excitation Hamiltonian and the transition dipoles from the ground state to the singly excited state are saved.
SPECTRON:
In the energy file each snapshot is stored after a line containing the word SNAPSHOT and the number of the snapshot (starting from 1). After this the upper tridiagonal matrix of the Hamiltonian snapshot is stored with one row on each line. The numbers are separated by a space. The dipole file also contain a line with the word SNAPSHOT and the number of that snapshot. This is followed by the transition dipoles stored with the x, y and z components for each site on one line separated by a space. Only the single excitation Hamiltonian and the transition dipoles from the ground state to the singly excited state are saved.

3.3.2 Full nonadiabatic simulation

The nonadiabatic linear and third-order response can be simulated with the 'NISE' program utilizing the numerical integration of the Schrödinger equation (NISE) approach [1,2]. The sparse and double excited state propagation is described in [3]. The coupling propagation scheme is described in [4]. The program require the Hamiltonian trajectory in binary format. The NISE program use the following input:

Hamiltonianfile [File name]

Dipolefile [File name]

Alphafile [File name] (Only needed for SFG calculations)

Anharmonicfile [File name]

Overtonedipolefile [File name]

HamiltonianType [Full/Coupling] (Full is the default, where the full Hamiltonian is given in the Hamiltonianfile, if Coupling is specified the couplings are assumed to be constant and given in the Couplingfile. The Hamiltonianfile then contain the trajectory of fluctuating diagonal elements.)

Couplingfile [File name]

PDBfile [File name]

Length [Number of snapshots in trajectory]

Samplerate [Number of snapshots between ensemble averaging]

Lifetime [The life time in fs]

Timestep [The length of each timestep in fs]

RunTimes [t1 max] [t2] [t3 max, all in timesteps]

Threshold [The threshold for the sparse matrix approximation, typical value 0.001]

Anharmonicity [0 = anharmonicities from file used, all other values result in the use of a fixed anharmonicity with that value]

Singles [Number of singly excited states]

Propagation [Sparse/Coupling default is Sparse] (Coupling recommended for fast calculations)

Couplingcut [Value in cm^{-1} below which the couplings are neglected, default 0, only used in the Coupling propagation scheme]

Trotter [The number of trotter steps in the Paarmann approximation for double excited states, 5 recommended for the Sparse propagation scheme, for the Coupling propagation scheme this is the general number of trotter steps the recommended value is then 1]

MinFrequencies [minw1] [minw2] [minw3, all in reciprocal cm]

MaxFrequencies [maxw1] [maxw2] [maxw3, all in reciprocal cm]

Technique [Absorption / Luminescence / CD / 2DIR / GBIR / SEIR / EAIR / noEAIR / 2DUVvis / GBUVvis / SEUVvis / EAUVvis / noEAUVvis / Pop / Dif / Ani / Analyse, these techniques are explained in Section 5]

FFT [Number of points on each axis in 2DFFT, if bigger than max times zero padding is used]

Format [Matlab/Dislin/Gnuplot For Matlab format rephasing and non-rephasing spectra are not added]

BeginPoint [The number of the first sample calculated in this run]

EndPoint [The number of the last sample calculated in this run, if this keyword is left out all samples will be included]

Project (Should be followed by two new lines. The first with Sites [Number] identifying how many sites are included in the projection and the second line with a list of integer numbers identifying the sites included counting from zero.)

PrintLevel [0 / 1 / 2] (Default is 0, the higher number the more detailed (timing) info is given.

The minimum and maximum frequencies are used for two things. First, the average of minw1 and maxw1 are used to shift the frequencies during the simulation. By shifting all transitions by the average the oscillations of the time-evolution are reduced and the simulations can be performed with longer distance between the time points. Second, the output is reduced to frequencies within the min and the max which reduce file size.

The lifetime is currently only applied during the coherence times (t_1 and t_3) according to the formula in [4]. The waiting time dependence can trivially be accounted for by multiplying the whole spectrum with $\exp(-t_2/T_1)$.

The double excitation Hamiltonian is propagated using the Trotter formula scheme [3, 19].

During the calculation the program will create a log file called NISE.log. For linear techniques this file will be updated every time a new sample has been calculated. The update contains timing information and therefore allows for the estimation of when the complete calculation finishes. For 2D techniques the timing information will given directly in the output file tracking the progress percentage.

For exciton diffusion calculations an additional input file containing the site positions is needed. This file should be named Position.bin and have the following format. The first entry should be the size of the box in float (all sides are assumed equally long). This is followed by the x, y, and z coordinates for each site again in floats. After the coordinates of the first snapshot the coordinates for the second follows directly and the coordinates for all snapshots should be stored.

3.3.3 Program output

The linear absorption calculation provides the linear response function in time domain in the file TD_Absorption.dat and the linear dichroism response function in time domain in the file RLD.dat. The first column is the time in femto seconds. The other columns are the real and imaginary parts of the response function. It is recommended to check that these have decayed to a small value close to zero in the calculated time interval. If this is not the case the system has not lost its coherence within the time determined by t1 max and the spectrum will be too broad. The linear absorption is given in the file Absorption.dat and the linear dichroism in the file LDIR.dat. The first line is the frequency in wavenumbers, the second line is the absorption spectrum, and a third line contain zeroes. For the linear dichroism calculation the z-axis is assumed to be the unique axis.

The 2D calculations (2DIR and 2DUVvis) provides the files R(par/per/cro)(I/II).dat. These files contain the time domain third-order response functions for different polarization directions [13,20]. The first three columns are the times t1, t2, and t2 in femtoseconds. The two last columns are the real and imaginary parts of the response functions. The frequency domain response functions are found using a double Fourier transform (see section 3.4). It is recommended to check that the response function has decayed within the calculated time intervals.

The population transfer calculation provides two files. Pop.dat contain two columns. The first is the time in femtoseconds the second is the probability that if a site was initially excited it is still excited after the given time. This is averaged over all sites. The file PopF.dat contain more detailed information. The first column is the time. The following columns are the probability that if a particular site was excited initially other sites are excited later. The first N columns are the populations of the N sites following an initial excitation of the first site. Then follows the populations after excitation of the second site etc. This file might be a very large file for big systems.

For the exciton diffusion the output file is Dif.dat It has three columns. The first is the time in femto seconds. The second column is the mean square displacement of the wave function assuming that one site is initially excited. The third column is the mean square displacement of the center of the wave function assuming that one site is initially excited. The program averages over all sites as initial sites. One should be aware that periodic boundary conditions are applied in this calculation and the mean square displacements will saturate [3].

For the integrated anisotropy calculation the output file is Ani.dat. It has three columns. The first is the time in femtoseconds. The second is the integrated anisotropy and the last is the orientational correlation function.

The Hamiltonian analysis provides the delocalization length/size according to the definition of Thouless [21] directly in the program standard output.

3.4 Fourier transform

The 2DFFT program use the same input as the NISE program. Fourier transformed response is written in files named Rw(par/per/cro).(I/II).dat, where par, per, and cro denote parallel, perpendicular and cross polarized signals. I and II denote the k_I and k_{II} contributions. When Dislin format is used the 2D correlation spectrum is saved in the files 2D.(par/per/cro).dat and the 'broad pump narrow probe' pump probe signal is stored in PP.(par/per/cro).dat. The first column is ω_1, the second is ω_2, the third is the dispersive signal, and the last column is the absorptive signal. Perl scripts connected with Dislin are available for plotting or the user can use his or her favorite plotting program. The Dislin format is also used by the python plotting code included in the tutorial files. The Gunplot output is only differing from the Dislin format in an empty line following ach new value of ω_1. A Gnuplot example file is included in the tutorial. The cover picture of the manual is generated with this gnuplot file. For Matlab format only the Rw files are created. These then contain a matrix with the response. An additional file waxis.dat is created with the values of the frequencies.

Chapter 4

Tutorial

A tutorial example exists for two coupled chromophores. To activate this you need to run `make examples` in the `build` directory. The tutorial files are located in the folder `examples/tutorial`. The program stochastic creates the input trajectories in GROASC format and input files can be found for converting to GROBIN and for the calculation of linear and two-dimensional spectra. The stochastic code allow the user to construct a simple trajectory for two coupled three level system. It is assumed that these levels are coupled to a set of overdamped Brownian oscillators [17]. The user can vary the length of the trajectory, the duration of the time interval between snapshots, the width of the frequency distribution, the correlation time, and the degree of correlation between the two three level system fundamental frequency fluctuations. The energy difference and the average frequency for the two fundamental frequencies can be adjusted as can the coupling between them. The angle between the assumed fixed transition dipoles can be adjusted as well. An example for parameters is given in the script named run.

After generating the frequency trajectory with the stochastic program one can look at the generated trajectories in Energy.tex and Dipole.tex. The should the be converted to binary input for the NISE program. This is done with the translate program with the input given in the file inpTra. The command line entry:

```
~/NISE_2017/bin/translate inpTra
```

will accomplish this. (It is assumed that you installed NISE in your home directory. If the programme was installed somewhere else you need to run the programme from the alternative bin directory where it was installed.)

When the binary trajectory exist we can calculate the linear absorption using the input from the file input1D.

```
~/NISE_2017/bin/NISE input1D
```

One should plot the output in TD_Absorption.dat and Absorption.dat to get familiar with this. It is important that the response function has decayed almost completely within the simulated time to avoid ringing and other artifact in the spectrum. The program should finish in a about 10 seconds. The python script plot.py will plot both the response

function and the absorption spectrum. Simply type `python plot.py` assuming that python 3 is installed. Alternatively, you can plot the two first columns of each file with your own favourite plotting programme.

The linear dichroism signal can be calculated by changing the the `Technique` to `LD` which generate corresponding files with the names TD_LD.dat and LD.dat. In the turotial this is not so interesting as it is simply minus half of the absorption signal, since the dipoles are in the x, y plane and the unique angle for linear dichroism is defined to be the z-axis.

The two-dimensional infrared spectra are calculated with the input file input2D. This corresponds to the spectrum of two coupled three-level systems in this case, where the anharmonicity is 20 cm^{-1} for both three-level systems as specified in the input2D file. The programme is executed with the command:

```
~/NISE_2017/bin/NISE input2D
```

The percentage wise progress of the calculation can be followed in the output. One should expect that the calculation of one sample point takes about 10 seconds with the given settings. For a test about 100 samples are sufficient, while for nice spectra at least 1000 samples are needed. Typically more samples are needed for two-dimensional spectra than for linear absorption spectra and the more different distinct chromophore environments are present the more samples are needed. The given example should take about 15 minutes to simulate (of course depending on the computer). If you have multiple cores on your computer you can run the MPI parallel version typing:

```
mpirun -n 4 ~/NISE_2017/bin/NISE input2D
```

The number 4 tells mpi that the job should be split in 4. For this small system mpi is likely more efficient then OpenMP on typical computer architectures. The output of the NISE programme is the time domain response functions. To get the frequency domain spectra the 2DFFT program must be executed as described below.

The NISE code was originally build to consider coupled three-level systems, which is what is needed to simulate infrared spectra. If one want to simulate two-dimensional electronic spectra coupled two-level systems usually need to be considered. This can now be achieved with the 2DUVvis technique. Previously this was achieved by using an anharmonicity that moves the third-level out of the spectral window and sufficiently far away that the coupling between the third-level for each monomer and double excitations involving pairs of sites can be neglected [4, 22].

The 2D simulations can be performed in parallel using openMP. The program does this by distributing the calculations for different values of t1 on different CPUs. When running parallel it is advantageous to ensure that ([t1 max]+1) divided by the number of used CPUs is an integer number or slightly smaller than an integer number. To run the code parallel the OMP_NUM_THREADS variable has to be set equal to the number of CPUs that one want to use in the submission script (or if not running under a queueing system the variable has to be set in the used linux shell). On most clusters one should further specify to the queuing system how many CPUs are needed when submitting the job. The

user is referred to the manual of the cluster used for this kind of information as it varies from system to system. For parts of the code, where the mkl library is used for solving eigen value problems these may be run in parallel as well by setting the MKL_NUM_THREADS variable equal to the number of CPUs used for this task. The general experience is that for small problems running parallel is not efficient. The mkl library also does not perform well on large numbers of CPUs, however, the 2D simulations were found to scale well up to about 32 CPUs in recent simulations on 864 coupled OH-stretch oscillators [23]. For large problems the calculation of polarization contributions and samples can efficiently be spread over different CPUs and/or nodes for efficent massively parallel calculations.

The response functions calculated in time domain can be converted to frequency domain spectra using the double Fourier transform code with the same input as for the calculation of the response function.

```
~/NISE_2017/bin/2DFFT input2D
```

The output can be visualized with one of the plotting scripts provided with the tutorial. The recommended metod is using the `plot2D.py` python script, which is written to plot the parallel polarization spectra. The line `Data = np.loadtxt('2D.par.dat')` in the beginning of the script can be changed to load and plot one of the other data files (perpendicular spectrum `2D.per.dat` or cross polarization `2D.cro.dat'`. The plot at the cover of the manual is the parallel polarization 2DIR spectrum obtained from the tutorial.

The user is encouraged to play with the settings of the run script in the tutorial to get familiar with the code. The simple example can also be useful for getting familiar with the basics of two-dimensional spectroscopy in particular regarding the effects of coupling, correlation, relative angles between coupled chromophores.

The delocalization length [21] can be calculated with the analysis option as given in the example input file inputAnalyse. Note that for including the complete trajectory [t1 max] should be 1.

```
~/NISE_2017/bin/NISE inputAnalyse
```

Chapter 5

Details on the available techniques

In this Chapter, more details on special options and the theory behind each technique is given. The techniques marked with a star (*) are implemented with OpenMP and MPI options, while techniques implemented with OpenMP options are marked with a plus (+). All other techniques are implemented as single CPU.

5.1 Analyse

The Hamiltonian is analysed and different statistical properties are provided including the average delocalization size of Thouless [21]. This inverse participation ratio (IPR) is expressed as:

$$D_{IPR} = \left\langle \frac{1}{N} \sum_i \left(\sum_j |c_{ij}|^4 \right)^{-1} \right\rangle . \tag{5.1}$$

5.2 Pop (population transfer)

The population transfer is calculated between sites. In general, this is governed by the equation:

$$P_{fi}(t) = \langle |U_{fi}(t,0)|^2 \rangle \tag{5.2}$$

Here f and i are the final and initial sites. Generally two files are generated. In Pop.dat average of the population remaining on the initial site (P_{ii}) is calculated resulting in a value starting at 1 (all popiulation is on the initial state) and decaying to the equilibrium value 1/N (equal population on all states). In the PopF.dat file a full matrix op all combinations of initial and final states is given. The columns start start with the first initial state and all possible final states continuing to the second possible initial state and all possible final states. This file may be very large for large system sizes!

5.3 Dif (diffusion)

Not implemented yet (check NISE_2015)

5.4 Ani (anisotropy)

Not implemented yet (check NISE_2015)

5.5 Absorption

The linear absorption is calculated using the first-order response function

$$I(t) = \sum_{\alpha}^{xyz} \langle \mu_\alpha(t) U(t,0) \mu_\alpha(0) \rangle \exp(-t/T_1). \tag{5.3}$$

Both the real and imaginary parts are stored. The Fourier transform is the frequency domain absorption, which is stored in the file Absorption.dat. T_1 is the lifetime, which is often simply used as an appodization function to smoothen the spectrum.

5.6 Luminescence

The luminescence is calculated using the first-order response function

$$I(t) = \sum_{\alpha}^{xyz} \langle \frac{1}{Z} \mu_\alpha(t) U(t,0) \exp(H(0)/k_B T) \mu_\alpha(0) \rangle \exp(-t/T_1). \tag{5.4}$$

Both the real and imaginary parts are stored. The Fourier transform is the frequency domain luminescence, which is stored in the file Luminescence.dat. T_1 is the lifetime, which is often simply used as an appodization function to smoothen the spectrum. The Boltzmann term containg the Hamiltonian at time zero (H) and the temperature (to be specified in the input) ensure the emission from a termalized population of the excited state ignoring a potential Stoke's shift and effects of vibronic states. The spectrum is normalized with the partition function.

5.7 LD (linear dichroism)

The linear dichroism is calculated identically to the linear absorption except the absorption in the x and y directions are subtracted from the absorption in the z direction. This corresponds to a perfect linear dichroism setup, where the molecules are aligned along the z-axis.

5.8 CD (circular dichroism)

The circular dichroism is calculated using the first-order response function

$$I(t) = \sum_{\alpha}^{xyz} \sum_{nm} \langle r_{nm} \mu_{\alpha,n}(t) \times [U(t,0) \mu_{\alpha,m}(0)] \rangle \exp(-t/T_1). \tag{5.5}$$

Both the real and imaginary parts are stored. The Fourier transform is the frequency domain absorption, which is stored in the file Absorption.dat. T_1 is the lifetime, which is often simply used as an appodization function to smoothen the spectrum.

5.9 Raman

Not implemented yet

5.10 SFG (sum-frequency generation)

Not implemented yet (check NISE_2015)

5.11 2DIR* (two-dimensional infrared)

This calculates the two-dimensional infrared spectra assuming coupled three level systems. The techniques GBIR (ground state bleach), SEIR (stimulated emission), and EAIR (excited state absorption) provides these contributions separetely. Furthermore the sum of the ground state bleach and the stimulated emission can be calculated with the noEA technique keyword. The expressions for the response functions are given in ref. [1].

5.12 2DSFG (two-dimensional sum-frequency generation)

Not implemented yet (check NISE_2015)

5.13 2DUVvis* (two-dimensional electronic spectroscopy)

This calculates the two-dimensional infrared spectra assuming coupled two level systems. The techniques GBUVvis (ground state bleach), SEUVvis (stimulated emission), and EAUVvis (excited state absorption) provides these contributions separetely. Furthermore the sum of the ground state bleach and the stimulated emission can be calculated with the noEAUVvis technique keyword.

5.14 2DFD (fluorescence detected two-dimensional spectroscopy)

The 2DFD spectrum can be calculated in the approximation that all exciton pairs annihilate to produce a single exciton long before fluorescence occur with the noEAUVvis technique. The response functions governing this technique are provided in ref. [24].

Chapter 6

Information for developers

Contributions from developers are welcome. The code is available on github: (https://github.com/GHlacour/NISE_2017). It is advisable to contact the mail developer before planning contributions to avoid competing comtributions. All new techniques should be added in new separate files which are colled from the main
tt NISE.c code. Implementations should as far as possible make use of existing modules for reading input and data. New propagation schemes or Hamiltonains should be added through the NISE_subs.c modules.

Chapter 7

Acknowledgement

The author wish to thank Foppe de Haan for helping with the code and Arend Dijkstra for helpful discussions in particular regarding the polarization directions and the double excited state propagation. Alexander Paarmann is gratefully thanked for discussions regarding the double excited state propagation. Carsten Olbrich is thanked for providing a gnuplot file that was adjusted to give the example provided with the tutorial. The Skinner group and Carlos Baiz are gratefully acknowledged for helping making the code more user friendly and with correcting minor bugs. Chungwen Liang is acknowledged for corrections of the Manual and implementation of the linear SFG code. The NWO is gratefully acknowledged for financial support making is possible to write the initial versions of this code.

Bibliography

[1] T. L. C. Jansen and J. Knoester. *J. Phys. Chem. B*, **110**:22910–22916, (2006).

[2] T. L. C. Jansen and J. Knoester. *Acc. Chem. Res.*, **42**(9):1405–1411, (2009).

[3] T. L. C. Jansen, B. M. Auer, M. Yang and J. L. Skinner. *J. Chem. Phys.*, **132**:224503, (2010).

[4] C. Liang and T. L. C. Jansen. *J. Chem. Theory Comput.*, **8**:1706–1713, (2012).

[5] C. Liang, M. Louhivuori, S. J. Marrink, T. L. C. Jansen and J. Knoester. *J. Phys. Chem. Lett.*, **4**:448–452, (2013).

[6] D. Cringus, T. L. C. Jansen, M. S. Pshenichnikov and D. A. Wiersma. *J. Chem. Phys.*, **127**:084507, (2007).

[7] T. L. C. Jansen and J. Knoester. *Biophys. J.*, **94**:1818–1825, (2008).

[8] A. G. Dijkstra, T. l. C. Jansen and J. Knoester. *J. Phys. Chem. A*, **114**:7315–7320, (2010).

[9] T. L. C. Jansen, D. Cringus and M. S. Pshenichnikov. *J. Phys. Chem. A*, **113**:6260, (2009).

[10] S. Roy, M. S. Pshenichnikov and T. L. C. Jansen. *J. Phys. Chem. B*, **115**:5431–5440, (2011).

[11] T. L. C. Jansen and J. Knoester. *J. Chem. Phys.*, **127**:234502, (2007).

[12] P. Hamm, M. H. Lim and R. M. Hochstrasser. *J. Phys. Chem. B*, **102**:6123–6138, (1998).

[13] R. M. Hochstrasser. *Chem. Phys.*, **266**(2-3):273–284, (2001).

[14] M. Cho. *Chem. Rev.*, **108**:1331, (2008).

[15] S. Mukamel. *Annu. Rev. Phys. Chem.*, **51**:691, (2000).

[16] M. Cho. *Two-dimensional Optical Spectroscopy*. CRC Press, Boca Raton, 2009.

[17] S. Mukamel. *Principles of Nonlinear Optical Spectroscopy*. Oxford University Press, New York, 1995.

[18] P. Hamm and M. T. Zanni. *Concepts and Methods of 2D Infrared Spectroscopy*. Cambridge University Press, Cambridge, 2011.

[19] A. Paarmann, T. Hayashi, S. Mukamel and R. J. D. Miller. *J. Chem. Phys.*, **128**:191103, (2008).

[20] M. T. Zanni, N.-H. Ge, Y. S. Kim and R. M. Hochstrasser. *P. Natl. Acad. Sci. USA*, **98**:11265, (2001).

[21] D. J. Thouless. *Phys. Rep.*, **13**:93, (1974).

[22] C. Olbrich, T. L. C. Jansen, J. Liebers, M. Aghtar, J. Strümpfer, K. Schulten, J. Knoester and U. Kleinekathöfer. *J. Phys. Chem. B*, **115**:8609–8621, (2011).

[23] L. Shi, J. L. Skinner and T. L. C. Jansen. *Phys. Chem. Chem. Phys.*, **18**:3772–3779, (2016).

[24] Tenzin Kunsel, Vivek Tiwari, Yassel Acosta Matutes, Alastair T. Gardiner, Richard J. Cogdell, Jennifer P. Ogilvie and Thomas L. C. Jansen. *J. Phys. Chem. B*, **123**(2):394–406, (2019).