simetuc User Manual

Pedro Villanueva-Delgado

November 3, 2017

Contents

Chapter 1

Features

- Command line interface program.

  - Run with

    ```
    simetuc config_file.txt [options]
    ```

  - See help and all options with

    ```
    simetuc -h
    ```

- The simulation is controlled by a configuration text file that the user can edit with the parameters adequate to the system of study. It includes:

  - Information about the host lattice.
  - Energy states labels.
  - Absorption and excitation (including ESA).
  - Decay (including branching ratios).
  - Energy transfer.
  - Other settings for the power and concentration dependence or optimization.

- *simetuc* works with any sensitizer and activator ion kind.

  - The examples are given for the Yb-Tm system.

- All kinds of energy transfer processes are supported:

  - Energy migration.
  - Upconversion (ETU).
  - Downconversion.
  - Cross-relaxation.
  - Cooperative processes.
  - Energy transfer from sensitizers to activators.
  - Back transfer from activators to sensitizers.

- See the example configuration file in the simetuc folder.

  - The configuration file sections are detailed and discussed in this document.
  - See also the full example in the appendix.

- Add decay experimental data as two column text data, separated by tabs or spaces.

- Different options:

  - Create the lattice.
  - Simulate the dynamics (rise and decay).
  - Optimize the energy transfer parameters.
    * Minimize the deviation between experiment and simulation.
  - Simulate the steady state.
  - Simulate the power dependence of each emission.
  - Simulate the concentration dependence of the dynamics or the steady state.

- All results are plotted and saved in the .hdf5 format. A summary of the results is saved as a text file.

- For all options `--average` uses standard average rate equations instead of microscopic ones.

Chapter 2

Installation

Python 3.6 is required. Installing Anaconda is recommended; it works with Windows (64/32 bits), Linux (64/32 bits) and Mac (64 bits).

After installing Anaconda, execute the following commands at the command prompt (*cmd.exe* for Windows, *shell* for Linux and Mac):[1]

```
conda config --add channels conda-forge
conda config --add channels pedvide
conda install simetuc
pip install settings_parser
```

(The first two commands add package repositories with up-to-date versions of all needed packages.)

or

```
pip install simetuc
```

That will download and install all necessary files.

Check that the program was installed correctly with

```
simetuc -h
```

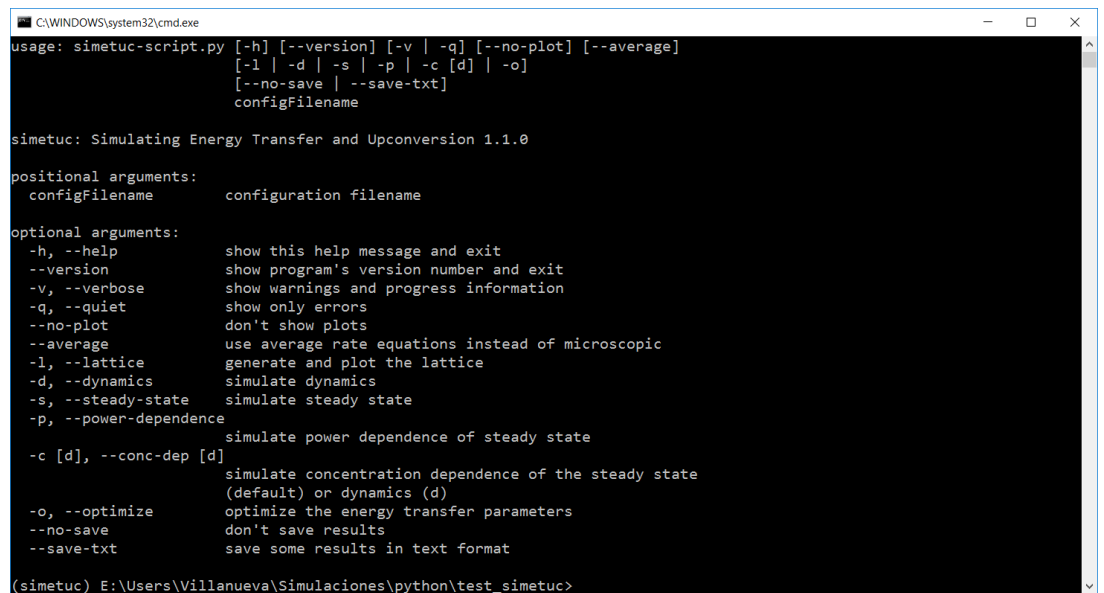which should display something similar to Figure 2.1.

2.1 Update

If you installed it using *conda*, update with:

```
conda update simetuc
```

If you installed it with *pip*, update with:

```
pip install -U simetuc
```

---

[1]Some OSX users report problems using *conda*, if after installing you can't use the program (i.e., *simetuc -h* fails because *simetuc* wasn't recognized as a command), use *pip install simetuc*.

```
C:\WINDOWS\system32\cmd.exe                                                          —    □    ×
usage: simetuc-script.py [-h] [--version] [-v | -q] [--no-plot] [--average]
                         [-l | -d | -s | -p | -c [d] | -o]
                         [--no-save | --save-txt]
                         configFilename

simetuc: Simulating Energy Transfer and Upconversion 1.1.0

positional arguments:
  configFilename        configuration filename

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  -v, --verbose         show warnings and progress information
  -q, --quiet           show only errors
  --no-plot             don't show plots
  --average             use average rate equations instead of microscopic
  -l, --lattice         generate and plot the lattice
  -d, --dynamics        simulate dynamics
  -s, --steady-state    simulate steady state
  -p, --power-dependence
                        simulate power dependence of steady state
  -c [d], --conc-dep [d]
                        simulate concentration dependence of the steady state
                        (default) or dynamics (d)
  -o, --optimize        optimize the energy transfer parameters
  --no-save             don't save results
  --save-txt            save some results in text format

(simetuc) E:\Users\Villanueva\Simulaciones\python\test_simetuc>
```

Figure 2.1: Results of `simetuc -h`.

Chapter 3

The configuration file

Some sections of the configuration file are mandatory (the starred Sections 3.1-3.5), some are optional (Sections 3.6-**??**), and some are only mandatory when performing a particular simulation type (Sections 3.9 and 3.10).

Comments can be placed anywhere in the file starting with a hash symbol (#).

The different sections of the file begin by a section name, for example lattice or states, followed by a colon (:). The different subsections and values are in a new line and indented with four spaces (not tabs!). A section can contain a single value or subsections; for example:

```
# comment1
section1: value1
section2:
    subsection1: value2 # comment2
    subsection2:
        subsubsection1: value3
```

Some of the values may be text or numbers, others, however, may be lists or a list of lists. A list is written as a comma-separated values between two square brackets:

```
section: [item1, item2, item3]
```

And for a list of lists:

```
section: [[lst1_item1, lst1_item2], [lst2_item1, lst2_item2]]
```

A full example of a configuration file, showing all options, is shown in Listing 8.1.

3.1 Version*

At the moment, there is only one format for the configuration file; however, in the future, it is possible that an improved format is introduced that is not compatible with the current one. For this reason the version of the format has to be included, the only value allowed is 1:

```
version: 1
```

3.2   Lattice*

This section defines the name of the lattice, the unit cell parameters, the concentration of sensitizer and activator ions, and optionally, a maximum distance of interaction.

This section starts with

```
lattice:
    # values and subsections...
```

Those values and subsections are described now.

The name is an arbitrary text that describes the lattice. It will be used as a folder name under latticeData (Section 5.1.1), expData (Section 5.3.1), and results (Section 4.2). Therefore it should only contain letters, numbers, and underscore (_) characters (no spaces).

The number of unit cells is an integer greater than zero. It specifies how many unit cells along the three directions a lattice will consist of.

The concentration of sensitizer and activator, in percentages from 0 to 100%. An ion with 0% concentration will not participate in the equations and therefore will not make the simulation take longer.

```
    S_conc: 0
    A_conc: 0.3
```

The unit cell parameters consist of the three lattice distances and three angles. The distances are given in Angstrom and the angles in degrees (between 0 and 360º).

```
    # distances in Angstrom
    a: 5.9738
    b: 5.9738
    c: 3.5297
    # angles in degree
    alpha: 90
    beta: 90
    gamma: 120
```

The spacegroup determines the symmetry of the lattice. It can be given as the international short symbol (e.g., P-6) or as the International Union of Crystallography number (e.g. 174).

The sites positions and occupancies determine the positions of the doped ions (in units of the lattice distances, from 0 to 1) and their fractional occupancy (between 0 and 1). For a lattice with just one site:

```
    sites_pos: [0, 0, 0]
    sites_occ: 1
```

For a lattice with two sites, the second of which is half occupied:

```
    sites_pos: [[0, 0, 0], [2/3, 1/3, 1/2]]
    sites_occ: [1, 1/2]
```

The sites are populated randomly using the desired concentration, see Section 5.1.

A full example of the lattice section is

```
lattice: # all fields here are mandatory
    name: bNaYF4
    N_uc: 40
    S_conc: 0
    A_conc: 0.3
    # unit cell
    # distances in Angstrom
    a: 5.9738
    b: 5.9738
    c: 3.5297
    # angles in degree
    alpha: 90
    beta: 90
    gamma: 120
    spacegroup: P-6
    # info about sites.
    sites_pos: [[0, 0, 0], [2/3, 1/3, 1/2]]
    sites_occ: [1, 1/2]
```

3.2.1 Optional values

The maximum distances for the normal and cooperative interactions restrict energy transfer to ions closer than the distances. The normal maximum distance (`d_max`) is not yet implemented; however, the cooperative one is, and it's strongly recommended to use it if cooperative interactions are present. The cooperative interactions grow approximately with the cube of the number of ions, the normal interactions grow with the square. If the values are not present, they are set to infinite.

```
lattice:
    ... all other fields ...
    d_max: 100.0
    d_max_coop: 25.0
```

3.3 States*

This section determines the number of energy states per ion and their labels. Four subsections are mandatory, two per ion type:

The ion labels is any text that describes the sensitizer or activator ion. It usually is the ion's chemical symbol.

The ion states labels is a list of text. Each item in the list labels an energy state; the first is the ground state, the rest should be in order of increasing energy. Usually the term symbols of the Dieke diagram are used.

```
states:
    sensitizer_ion_label: Yb
    sensitizer_states_labels: [GS, ES]
    activator_ion_label: Tm
    activator_states_labels: [3H6, 3F4, 3H5, 3H4, 3F3, 1G4, 1D2]
```

The ion and state labels will be used to define excitation (Section 3.4), decay (Section 3.5), branching ratios (Section 3.6), and energy transfer processes (Section 3.7).

3.4   Excitation*

At the beginning of the simulation, all ions are in their ground state. An excitation source is necessary to pump them to an excited state. This source can be pulsed for dynamics or continuous for steady state simulations.

Any number of excitation subsections can be added to the file, and at least one has to be active (see below). A two-color experiment can be simulated by activating two excitations. Any number of excitations can be active.

An excitation block contains several mandatory subsections:

The label is a short unique text. It identifies the excitation and it is used in the naming of experimental data files (see Section 5.3.1), so short, informative names are encouraged.

active can be true or false and indicates whether the excitation will be used for the simulations.

The excitation power density in W cm^{-2}. For pulsed sources this has to be the instantaneous power density of the pulse. Usual values are in the order of $10^5-10^7\,\text{W cm}^{-2}$ for pulsed excitations and $10^1-10^3\,\text{W cm}^{-2}$ for continuous excitations.

The pulse width of the excitation source (in seconds) for pulsed excitations. For continuous excitations it can be omitted.

The process is the actual absorption transition. The format is `ion_label(state_label_i) -> ion_label(state_label_f)`. Where the ion and state labels were defined in Section 3.3. Both ion labels have to be present and equal; both state labels have to belong to the same ion type.

The degeneracy is the relative degeneracy of the initial and final states. The relative degeneracy of two trivalent lanthanide states with terms $^{2S_i+1}L_{iJ_i}$ and $^{2S_f+1}L_{fJ_f}$ is $g_{rel} = \frac{2J_i+1}{2J_f+1}$.

The pump rate is the absorption cross-section divided by the transition energy in units of $\text{cm}^2\,\text{J}^{-1}$, that is $R_P = \frac{\sigma(\nu)}{h\nu}$. The pump rate multiplied by the power density is equal to the absorption probability, with units of s^{-1}.

An example with two excitation blocks:

```
excitations:
    pulsed_1G4:
        active: True
        power_dens: 1e6
        t_pulse: 1e-8
        process: Tm(3H6) -> Tm(1G4)
        degeneracy: 13/9
        pump_rate: 9.3e-4
    CW_980nm:
        active: False
        power_dens: 1e2
        process: Yb(GS) -> Yb(ES)
        degeneracy: 4/3
        pump_rate: 4.4e-3
```

3.4.1 Excited state absorption

Excited state absorption (ESA) can be simulated by giving a list of transitions, degeneracies and pump rates in the corresponding fields, for example:

```
excitations:
    NIR_800:
        active: False
        power_dens: 1e5
        t_pulse: 1e-8
        process: [Tm(3H6)->Tm(3H4), Tm(3H5)->Tm(1G4)] # list
        degeneracy: [13/9, 11/9] # list
        pump_rate: [4.4e-3, 4e-3] # list
```

3.5 Decay*

This section contains the decay lifetimes in seconds. There are two sections, one for the sensitizer and one for the activator. In both cases the decay lifetimes are input with `state_label:  lifetime`:

```
sensitizer_decay:
    ES: 2.5e-3
activator_decay:
    3F4: 12e-3
    3H5: 25e-6
    3H4: 2e-3
    3F3: 2e-6
    1G4: 760e-6
    1D2: 67.5e-6
```

3.6 Branching ratios

These are two optional sections, one for the sensitizer and one for the activator.

The branching ratios describe the fraction of emission from an initial state to a final state. The format is `initial_state -> final_state:  ratio`. The ratio is a number between zero and one. The branching ratio to the ground state does not need to be specified because it is calculated automatically from the other values.

```
sensitizer_branching_ratios: # can be empty or simply omitted
activator_branching_ratios:
    3H5->3F4: 0.4
    3H4->3F4: 0.3
    3H4->3H5: 0.1
    3F3->3H4: 0.999
    1G4->3F4: 0.15
    1G4->3H5: 0.16
    1G4->3H4: 0.04
    1G4->3F3: 0.001
    1D2->3F4: 0.43
```

3.6.1   Multiphonon relaxation and thermalization

When two states are close in energy, the upper one can decay nonradiatively to the lower one in a process known as multiphonon relaxation. This process can be described in the configuration file as a branching ratio from the upper to the lower state with the appropriate value. In the example above, the $^3F_3$ state decays into the $^3H_4$ state with a branching ratio of 0.999; this means that the MPR rate is $k_{MPR} = 0.999 \cdot k_{^3F_3}$, where $k_{^3F_3}$ is the total decay rate given in section 3.5. The radiative decay rate to the ground state is then $k_{rad} = (1 - 0.999) \cdot k_{^3F_3}$.

The lower state can also populate the upper one via phonons, and after some time a thermal equilibrium can be established. The ratio of the populations of the upper and lower states is then given by the Boltzmann distribution. This process can be described in the configuration file as a branching ratio from the lower to the upper state with the appropriate value. For example, if there is a thermal equilibrium between the $^3F_3$ and $^3H_4$ states we need to define two branching ratios in the configuration file, one for relaxation and the other for thermalization:

```
activator_branching_ratios:
...
    3F3->3H4: 0.999 # multiphonon relaxation
    3H4->3F3: 5e-5 # multiphonon thermalization
...
```

The value for the multiphonon thermalization branching ratio should be

$$k_{MPT} = k_{MPR} \exp\left(-\frac{\Delta E}{kT}\right) = 0.999 \exp(-10) \approx 5\,10^{-5}$$

where $\Delta E = 2000\,\mathrm{cm}^{-1}$ is the energy difference between the states and $kT_r = 200\,\mathrm{cm}^{-1}$ is the room temperature energy.

3.7 Energy transfer

This optional section includes one or more energy transfer processes between ions in the simulation. Each energy transfer block refers to one process and contains several fields:

The label is a short unique text. It can also be used to restrict the optimization to certain processes (Section 3.8).

The process describes the ET interaction with the format `ion_label1(init_state) + ion_label2(init_state) -> ion_label1(end_state) + ion_label2(end_state)`. Where the ion and state labels were defined in Section 3.3.

The multipolarity of the interaction is an integer number, usually 6, 8 or 10.

The strength of the interaction given in units of $\mathrm{s}^{-1}\,\text{Å}^{n}$, where n is the multipolarity. Optionally, an interaction strength for the average rate equation system can be given; this value is several orders of magnitude lower than the microscopic value, see Section 5.8. If it is not given and the average rate equations are solved, the normal strength will be used.

For example:

```
enery_transfer:
   CR50:
      process: Tm(1G4) + Tm(3H6) -> Tm(3H4) + Tm(3H5)
      multipolarity: 6
      strength: 4.3057e+09
      strength_avg: 8e+03 # optional, only for average system
   EM:
      process: Yb(ES) + Yb(GS) -> Yb(GS) + Yb(ES)
      multipolarity: 6
      strength: 4.50220614e+10
```

3.7.1 Cooperative energy transfer

Cooperative energy transfer processes involve three ions. Two ions cooperatively transfer (receive) energy to (from) a third ion. The format is very similar to the normal energy transfer processes, the only field that changes is `process`. It now must contain the ion and state labels of three initial states and three final states, for example:

```
enery_transfer:
   coop:
      process: Yb(ES) + Yb(ES) + Tm(3H4) -> Yb(GS) + Yb(GS) + Tm(1G4)
      multipolarity: 6
      strength: 1e5
```

3.8   Optimization

processes An optional list of energy transfer or branching ratio labels. If empty or not present, all ET processes will be optimized (and no branching ratios). If present and not empty, this option restricts the optimization parameters to those in the list, for example

```
processes: [CR50, 3H4->3F4]
```

will optimize only the parameters `CR50` and `3H4->3F4`, and not any other.

method This optional section defines the method used to optimize the energy transfer parameters to the experimental data. Several options are available:

- `leastsq`
- `SLSQP`
- `COBYLA`
- `L-BFGS-B`
- `basin_hopping`
- `brute_force`

simetuc uses the library *lmfit* for the optimization, see lmfit methods documentation; see also the *Notes* section in the scipy minimize documentation and scipy least_squares documentation. `leastsq` (the default), `SLSQP`, or `brute_force` are recommended, as they seem to take the least time to arrive at the minimum.

`leastsq` uses the Levenberg-Marquardt method, it is very robust and efficient with a lot of smart tricks, it's the default method.

`SLSQP` minimizes the error using a method similar to Newton-Raphson, which iteratively tries to find the parameter values that make the gradient zero.

`COBYLA` does not calculate the gradient, so it may be faster when the number of parameters is large.

`brute_force` on the other hand computes the error at a given number of points (10 points by default if not given, see *options* section below) in a given range (from 0.1 to 10 times the parameter values given in the configuration file, also user-changeable) and selects the minimum; after it, it is advised to run an `leastsq` or `SLSQP` minimization to further refine the value. If the parameter to optimize is a branching ratio, the range is restricted to [0, 1].

options Non-mandatory options for the optimization:

- `tol`: Tolerance for the optimization
- `N_points`: Number of points (per optimization parameter) for the brute_force method.
- `min_factor/max_factor`: Minimum and maximum factors that define the range of values to be simulated in the brute_force method. For example, if the parameter value is 500, a `min_factor`=0.1 and `max_factor`=10 will produce a range [50, 5000].

excitations is a optional list of excitation labels. If empty, the simulations will use the active excitation(s) in the configuration file. If not empty, the dynamics will be simulated several times, one per excitation label present, with each excitation successively set as active; the error of each simulation will be averaged and that error will be minimized. This is useful when experimental data for different excitations exist. For example:

```
excitations: [Vis_473, NIR_980]
```

3.9 Power dependence

This section is mandatory only if a power dependence simulation is performed. The starting and ending power density values and the number of steps need to be given

```
power_dependence: [1e0, 1e7, 8]
```

Note that it uses a logarithmic scale, so the above configuration will produce the power dependence at the following excitation power densities: 10^0, 10^1, 10^2, 10^3, 10^4, 10^5, 10^6, and 10^7 W cm^{-2}. See also Section 5.6.

3.10 Concentration dependence

This section is mandatory only for concentration dependence simulations or for an optimization of all concentrations at the same time (see Section 5.4).

concentrations is a list of two lists; the first list gives the sensitizer concentrations at which the simulation will take place, the second gives the same for the activators.
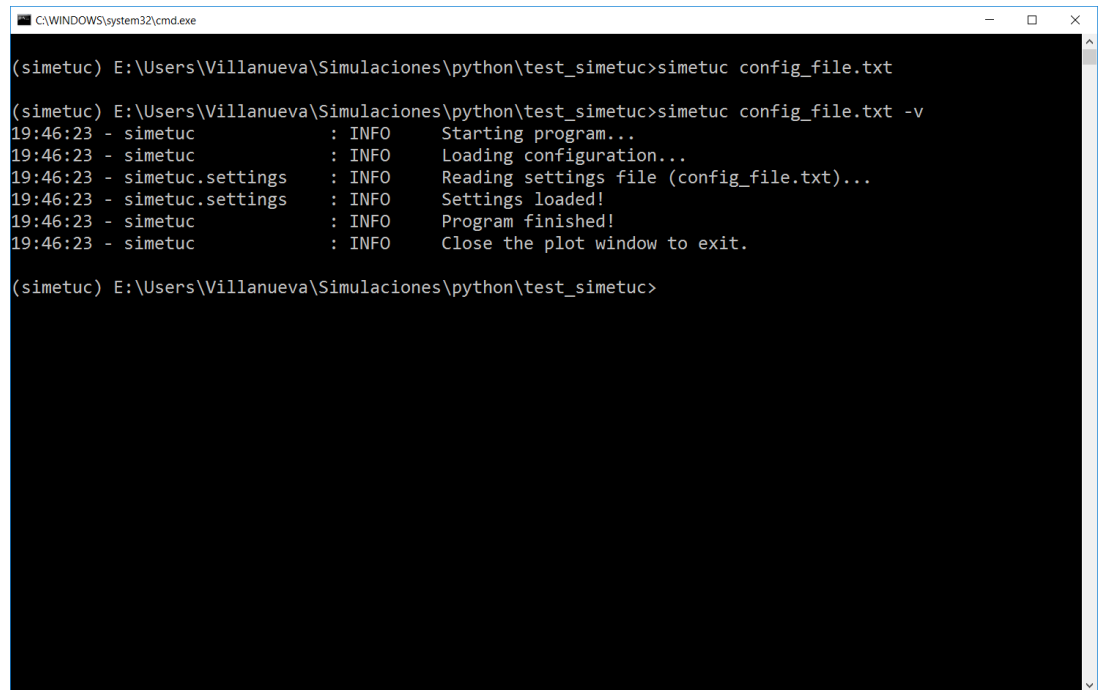
```
concentration_dependence: [[0, 0.5], [0.1, 0.2, 0.3, 0.4, 0.5]]
```

The concentration of either ion can also be fixed, so only the other ion changes:

```
concentration_dependence: [[0], [0.1, 0.2, 0.3, 0.4, 0.5]]
```

N_uc_list is a list of numbers of unit cells (see Section 3.2). The different concentrations require different `N_uc` to keep the number of simulated ions at a reasonable number. Using the same N_uc for 0.1% and 10% concentrations either wastes time for the 0.1% sample or simulates it with very few ions. This list should be as long as the total number of concentrations, which is given by the product of the length of S and A concentrations ($2 \times 5 = 10$ for the first example, $1 \times 5 = 5$ for the second).

```
N_uc_list: [75, 45, 45, 30]
```

Figure 3.1: Example of a correct configuration file check with and without the verbose option.
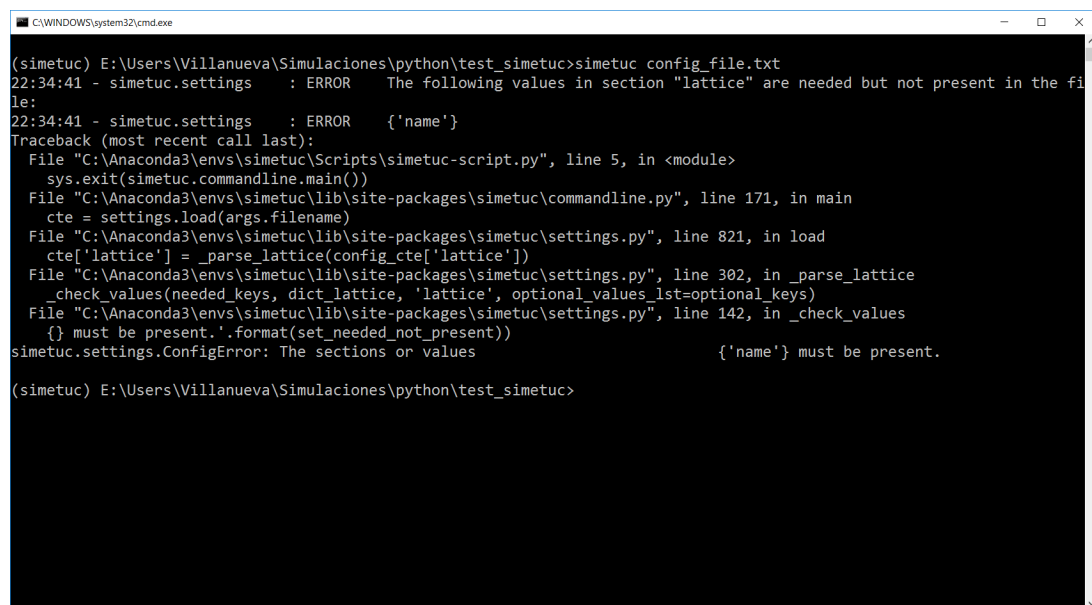
3.11 Checking the configuration file

To check that a configuration file is correct, execute:

```
simetuc config_file.cfg
```

If no warnings or errors are displayed, the file is correct. Nonetheless, it is still possible that some optional values are missing but needed for a particular simulation type or have wrong values (e.g., negative distances or concentrations above 100%). In that case, this check will not warn the user, and only trying to perform the actual simulation will show if something is missing. In any case, the program never fails silently, that is, it always warns the user if there was a problem and how to fix it.

An example of a configuration file check that succeeded is shown in Figure 3.1. An example of a check failure is shown in Figure 3.2; in this case, the name or the lattice has been omitted from the file.

Figure 3.2: Example of a wrong configuration file check. The two upper lines show the user-friendly error message; they can also be found in the error log (see Section 4.1.2). In this case the name of the lattice was omitted from the file.

Chapter 4

The logs and results folders

simetuc creates several subfolders during its usage; these subfolders are located under the current working directory (the directory from which *simetuc* is being used). Additionally, the experimental data must be saved in a folder with a specific format, see Section 5.3.1.

4.1 The log folder

Any operation of *simetuc* writes its status and current tasks on several log files, so the user can analyze what operations were performed, when, and whether they succeeded. The log files are stored in a subfolder called `logs`, which is created if it doesn't exist.

4.1.1 Normal log

This is the primary log. Here all parts of the program register their operations. For example, any operation begins with:

```
2017-01-10 19:13:51,227 - simetuc   : INFO    Starting program...
```

The log format is the following:

```
YYYY-MM-DD HH:MM:SS,mmm - simetuc[.part] : LEVEL message
```

The main program emits messages under the name *simetuc*, other parts of the program have names such as *simetuc.lattice* or *simetuc.simulations*. The log level can be `DEBUG`, `INFO`, `WARNING`, or `ERROR`.

4.1.2 Error log

Whenever an operation was requested, but couldn't be completed due to an user error or an internal bug, the errors are written to an error log. This is log remains empty if all operations succeeded.

4.1.3 Debug log

This log registers a lot of low level information that is usually not important, but can be relevant in the case of errors.

4.2   The results folder

All simulations produce some results that are stored in the subfolder `results/latticeName` with the hdf5 format. A summary of the results is also saved as a text file, it includes the configuration file that produced it.

4.3   Other folders

The lattice folder, Section 5.1.1.
   The experimental data folder, see Section 5.3.1.

4.4   The HDF5 format

This is a open binary format that can be open with many scientific software such as Matlab or Origin. A lightweight viewer is HDFCompass.

Chapter 5

Simulations

5.1 Creating a lattice

This is the first step in any simulation. The lattice and states section of the configuration file determine the position of the doped ions and the number of energy states.

The option for creating lattice is `-l`:

```
simetuc config_file.txt -l
```

When given with the `-v` option to show the progress information, the results look like Figure 5.1.

5.1.1 The `latticeData` folder

All created lattices are stored in a folder with the name of the lattice under the latticeData subfolder; the name of the lattice is given by the user in the configuration file, see Section 3.2. These saved files are used in subsequent simulations. The name format under which they are saved is

```
data_XXuc_y.yS_z.zA.hdf5
```

where XX is the number of unit cells, $y.y$ is the concentration of sensitizers and $z.z$ that of activators (both in percentage).

The file format is HDF5, see Section 4.4.

5.2 Simulating the dynamics

The simulation of the dynamics of a sample consists of two parts, the excitation pulse and the subsequent relaxation. The dynamics simulation includes the excitation, decay, branching ratios, and energy transfer sections of the configuration file. The output is a plot of the average decay curve of each state of the sensitizer and activator. The results are saved to a file, see Section 4.2.

The option for simulating the dynamics is `-d`:

```
simetuc config_file.txt -d
```

When given with the `-v` option to show the progress information, the results look like Figure 5.2.

(a)

(b)

Figure 5.1: Lattice creation. (a) shows the result of invoking the command
`simetuc config_file.txt -v -l`. (b) shows the positions of the doped ions,
the sensitizers as red dots and the activators as blue dots.

(a)

(b)

Figure 5.2: Dynamics simulation. (a) shows the result of invoking the command
simetuc config_file.txt -v -d. (b) shows a plot with the decay curves of
all states of the sensitizer and activator.

5.3   Adding experimental data

The comparison of the experimental to the simulated data is important if to ensure that the energy transfer processes and their strengths truly represent an accurate picture of the sample's behavior.

Decay experimental data is automatically loaded from the expData folder (see next section), normalized and compared to the simulated decay curve of the appropriate states. The deviation between the experimental data and simulation is calculated for each level, and then all state errors are added together to yield the total error.

5.3.1   The `expData` folder

This folder should contain a subfolders with the name of the lattice. Inside that subfolder several text files can be stored, each corresponding to an ion's state decay for a given concentration of sensitizers and activators. The name format is:

```
decay_ION_STATE_exc_EXCLABEL_x.xION-S_y.yION-A.txt
```

where `ION`, `STATE`, and `EXCLABEL` are labels defined in the configuration file: `ION` is an ion label, `STATE` is an state label corresponding to `ION`, and `EXCLABEL` is an excitation label defined in the excitations section of the configuration file, see Section 3.4. *x.x* and *y.y* are the sensitizer (`ION-S`) and activator (`ION-A`) concentrations.

As an example, the decay curve of the Tm^{3+} state $^1D_2$ for the β-NaYF$_4$: 0.3% Tm^{3+} sample has a name:

```
decay_Tm_1D2_exc_Vis_473_0.0Yb_0.3Tm.txt
```

and is stored in the folder `expData/bNaYF4`.

The experimental data files must have two columns separated by spaces or tabs:

1. The first one with the time in seconds.

2. The second one with the emission intensity in any units (it will be normalized).

Lines starting with `#` are ignored as comments. For example:

```
# tau_em360_NYF_03Tm_EKSPLA473
# time in seconds
0                      0.405264768901133
1.9999999999978E-7     0.42944597490052
3.99999999999956E-7    0.432506887052342
5.99999999999989E-7    0.43954698500153
7.99999999999967E-7    0.4640342822161
1E-6                   0.468931741659014
1.19999999999998E-6    0.498010407101316
1.39999999999996E-6    0.48852157943067
1.5999999999999E-6     0.51147842056933
1.79999999999997E-6    0.513927150290787

...
```
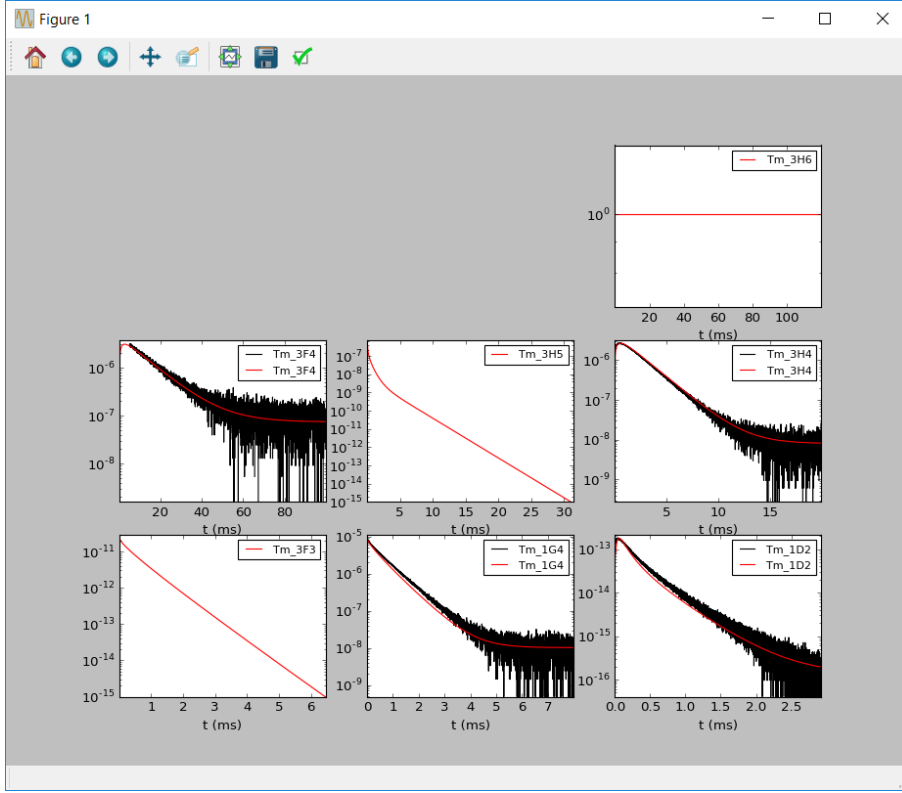
Figure 5.3: Dynamics simulation with experimental data.

Once the experimental data files are in the right folder and with the right names, they will be loaded automatically, like shown in Figure 5.3. The state and total errors will be shown in the commandline and recorded in the log files.

5.4 Optimizing the ET parameters

The strength of the interactions is *a priori* unknown, but, by fitting the simulations to the experimental data, it is possible to obtain realistic interaction strengths.

The option for optimizing the dynamics is `-o`:

```
simetuc config_file.txt -o
```

When given with the `-v` option to show the progress information, the results look like Figure 5.4. The total error has decreased from 0.2202 to 0.1868 when the energy transfer parameters changed from $1.3\,10^9\,\mathrm{s}^{-1}\,\mathrm{\AA}^6$ to $2.4\,10^9\,\mathrm{s}^{-1}\,\mathrm{\AA}^6$ for the parameter `CR50` and from $1.5\,10^9\,\mathrm{s}^{-1}\,\mathrm{\AA}^6$ to $4.1\,10^9\,\mathrm{s}^{-1}\,\mathrm{\AA}^6$ for the parameter `ETU53`.

The optimization procedure does not change the configuration file (*simetuc* never does), so the parameter values should be changed by the user. It is advisable to re-run a dynamics simulation with the new parameters to check that the fit is satisfactory.

Figure 5.4: Optimization of the dynamics with the command `simetuc config_file.txt -v -o`.

When optimizing a system for the first time it is recommended to first use the `brute_force` method, update the configuration file with its results, and then run the optimization again with the `leastsq` method, see Section 3.8.

If there is data for different concentrations, it is possible to optimize the parameters by calculating the error for all concentrations:

```
simetuc config_file.txt -o conc
```

This will simulate all concentrations, combine their errors and minimize it.

5.5   Simulating the steady state

If the excitation source is not pulsed, but it is kept switched on continuously, the states will reach a steady state; that is, the states will have a population that does not change anymore with time. The time it takes for the system to reach the steady state depends on the lifetimes and energy transfer processes.

The steady state populations are proportional to the emission intensity of each state under continuous excitation.

The excitation source activated in the configuration file should have a realistic excitation power density; usually pulsed sources have much higher instantaneous powers, see Section 3.4.

The option for simulating the steady state is `-s`:
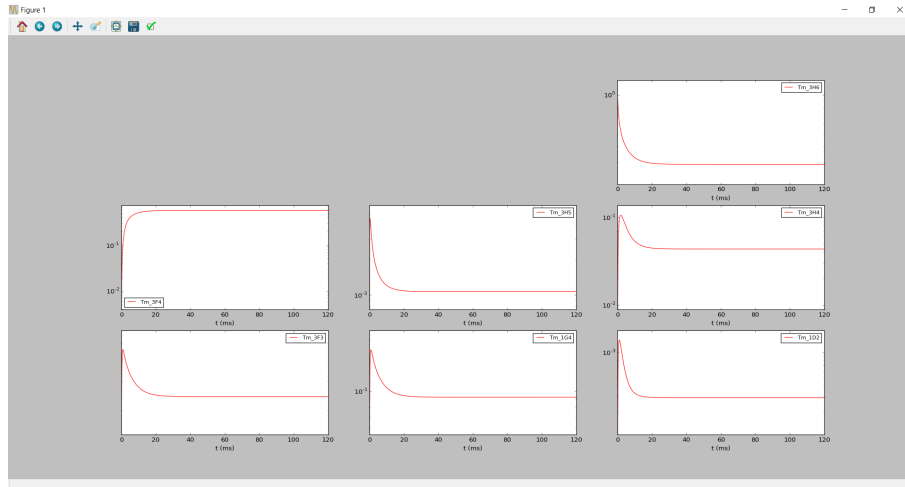
```
simetuc config_file.txt -s
```

When given with the `-v` option to show the progress information, the results look like Figure 5.5.

(a)

(b)

Figure 5.5: Steady state simulation. (a) shows the result of invoking the command `simetuc config_file.txt -v -s`. (b) shows a plot with the decay curves of all states of the sensitizer and activator. All states have reached their steady state.

5.6   Simulating the power dependence of the steady state

A standard upconversion experiment consists in measuring the emission intensity (usually the area under the emission peak, but also just the maximum) as a function of the excitation power. The power dependence of the emission I usually follows a power law: $I = P^n$, where P is the excitation power and n is the number of photons involved in the mechanism; this is an approximation that only works at low excitation powers.

This simulations are capable of producing the entire power dependence curve so it can be compared to the experimental one. *simetuc* simulates the steady state for the excitation powers defined in the configuration file and shows the steady state populations in a log-log plot. See Section 3.9 for the configuration file options.

The option for simulating the steady state is `-p`:

```
simetuc config_file.txt -p
```

When given with the `-v` option to show the progress information, the results look like Figure 5.5.

5.7   Simulating the concentration dependence of the the steady state or of the dynamics

One of the strongest points of the microscopic rate equations that *simetuc* solves is that once the lifetimes, branching ratios and energy transfer parameters are known, it is possible to change the concentrations of sensitizers and activators and simulate the decay curves or the steady state, thus predicting the behavior of a new sample.

Simulating the concentration dependence of the steady state is equivalent to simulate the emission efficiency of each state as a function of both concentrations. Simulating the concentration dependence of the dynamic is useful to compare it with experimental data, because decay curves are easier to measure than emission efficiencies.

The simulated concentration range is controlled with the configuration file, see Section 3.10. The option for simulating the concentration dependence of the steady state is `-c`:

```
simetuc config_file.txt -c
```

When given with the `-v` option to show the progress information, the results look like Figure 5.7.

The option for simulating the concentration dependence of the dynamics is `-c d`:
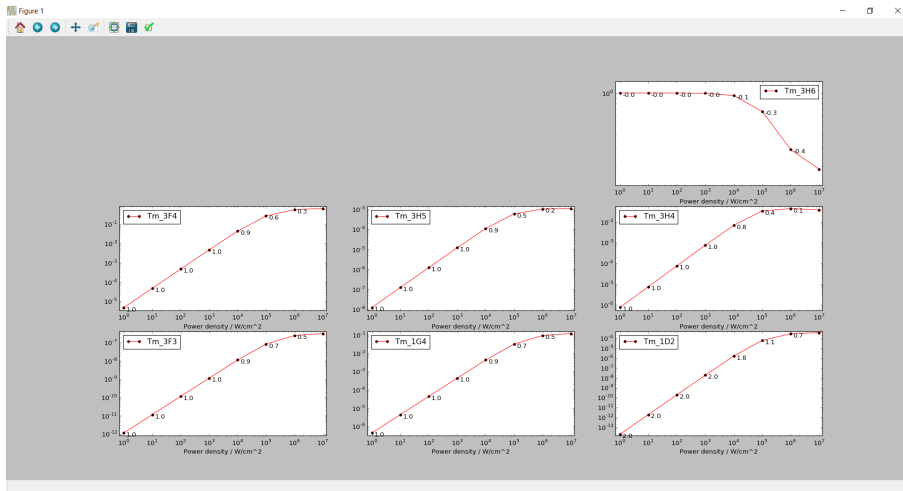
```
simetuc config_file.txt -c d
```

When given with the `-v` option to show the progress information, the results look like Figure 5.8.

(a)

(b)

Figure 5.6: Power dependence simulation. (a) shows the result of invoking the command `simetuc config_file.txt -v -p`. (b) shows a plot with the emission power dependence of all states of the sensitizer and activator. The simulated excitation powers are taken from the configuration file. In this case, unrealistically high powers have been simulated to show the saturation regime.

5.8 Microscopic or average rate equations

It is useful to compare the results of the microscopic rate equation model to the standard average rate equation model. All simulations can be performed with the average model using the `--average` flag in a command, for example

```
simetuc config_file.txt -d --average
```

will simulate the dynamics using the average model and

```
simetuc config_file.txt -p --average
```

will simulate the power dependence of the average model.

In the average model, all sensitizer ions are represented as one average ion (same for the activators), therefore all energy migration processes are ignored (technically speaking, the average model has infinite energy migration between all states).

(a)

(b)

(c)

Figure 5.7: Concentration dependence of the steady state. (a) shows the result of invoking the command `simetuc config_file.txt -v -c`. (b) and (c) show plots with the concentration dependence of the emission. The simulated concentrations are taken from the configuration file. In (b) the sensitizer concentration is fixed at 0%, only the activator concentration changes. In (c) both concentrations change.

(a)

(b)

Figure 5.8: Concentration dependence of the dynamics. (a) shows the result of invoking the command `simetuc config_file.txt -v -c d`. (b) shows a plot with the decay curves at different concentrations. The simulated concentrations are taken from the configuration file.

The energy transfer parameters in the average model are not the microscopic parameters between two ions, but an average of all interactions between the ions in the sample, which are at different distances. This means that knowledge of the average ET parameters for a sample with a certain concentration does not give much information about the ET parameters at other concentrations. That is, fitting decay data with the average model gives limited information (average ET strengths) about that sample and nothing about different samples.

Chapter 6

Publications and acknowledgments

This software has been described and used in these publications:

- Villanueva-Delgado, P.; Krämer, K. W. & Valiente, R. Simulating Energy Transfer and Upconversion in β-NaYF$_4$: Yb^{3+}, Tm^{3+}, *J. Phys. Chem. C*, **2015**, *119* (41), pp 23648–23657. DOI: 10.1021/acs.jpcc.5b06770.

- Villanueva-Delgado, P.; Krämer, K. W.; Valiente, R.; de Jong, M. & Meijerink, A. Modeling Blue to UV Upconversion in β-NaYF$_4$: Tm^{3+}, *Phys. Chem. Chem. Phys.*, **2016**, *18*, pp 27396-27404, DOI: 10.1039/C6CP04347J.

- Villanueva-Delgado, P. Upconversion in β-NaYF$_4$: Yb^{3+}, Tm^{3+}: Synthesis, Experiments, and Models. Ph.D. Dissertation, University of Bern, **2016**.

If you use this software in a scientific publication, please cite the appropriate articles above.

6.1   Acknowledgments

Chapter 7

License

Chapter 8

Appendix: Example configuration file

Listing 8.1: Exampe configuration file

```
1   # settings file
    # USE SPACES AND NOT TABS

    version: 1 # mandatory, only 1 is supported at the moment
5
    # mandatory section
    lattice:
    # all fields here are mandatory
        name: bNaYF4
10      N_uc: 50

        # concentration
        S_conc: 0.1
        A_conc: 0.1
15
        # unit cell
        # distances in Angstrom
        a: 5.9738
        b: 5.9738
20      c: 3.5297
        # angles in degree
        alpha: 90
        beta: 90
        gamma: 120
25
        # the number is also ok for the spacegroup
        spacegroup: P-6

        # info about sites.
30      # If there's only one site, use:
        # sites_pos: [0, 0, 0]
```

```
        # sites_occ: 1
        sites_pos: [[0, 0, 0], [2/3, 1/3, 1/2]]
        sites_occ: [1, 1/2]
35
        # optional
        # maximum distance of interaction for normal ET and for cooperative
        # if not present, both default to infinite
        d_max: 100.0
40      # it's strongly advised to keep this number low,
        # the number of coop interactions is very large (~num_atoms^3)
        d_max_coop: 50


    # mandatory section
45  states:
    # all fields here are mandatory,
    # add any label if necessary
    # i.e.: just "sensitizer_ion_label: label" on a line
    # but don't delete them.
50  # If you set the sensitizer concentration to zero,
    # they will be ignored in all calculations
        sensitizer_ion_label: Yb
        sensitizer_states_labels: [GS, ES]
        activator_ion_label: Tm
55      activator_states_labels: [3H6, 3F4, 3H5, 3H4, 3F3, 1G4, 1D2, 1I6, 3P0]



    # mandatory section
    excitations:
60  # the excitation label can be any text
    # the t_pulse value is only mandatory for the dynamics
    # at least one excitation must be present and active
        Vis_473_1:
            active: False
65          power_dens: 1e7 # power density W/cm^2
            t_pulse: 5e-9 # pulse width, seconds
            process: Tm(3H6) -> Tm(1G4) # both ion labels are required
            degeneracy: 13/9 # initial_state_g/final_state_g
            pump_rate: 9.3e-3 # absorption cross-section/energy in cm2/J
70      Vis_473:
            active: True
            power_dens: 1e7
            t_pulse: 5e-9
            #process: [Tm(3H6) -> Tm(1G4), Tm(3F3) -> Tm(3P0)]
75          process: [Tm(3H6) -> Tm(1G4), Tm(3H4) -> Tm(1I6)]
            degeneracy: [13/9, 7/1]
            pump_rate: [9.3e-3, 4e-2]
        Vis_473_3:
            active: False
80          power_dens: 1e7
            t_pulse: 5e-9
```

```
              process: [Tm(3H6) -> Tm(1G4), Tm(3F4) -> Tm(1D2), Tm(3F3) -> Tm(3P0)
                  ]
              degeneracy: [13/9, 9/9, 7/1]
              pump_rate: [9.3e-3, 5e-4, 4e-2]
85
      # mandatory section
      sensitizer_decay:
      # lifetimes in s
          ES: 2.5e-3
90
      # mandatory section
      activator_decay:
      # lifetimes in s
          3F4: 12e-3
95        3H5: 25e-6
          3H4: 2e-3
          3F3: 2e-6
          1G4: 775e-6
          1D2: 67.5e-6
100       1I6: 101.8e-6
          3P0: 5e-6


      # optional section
      sensitizer_branching_ratios:
105
      # optional section
      activator_branching_ratios:
          3H5->3F4: 0.4
          3H4->3F4: 0.3
110       3H4->3H5: 0.1
          3F3->3H4: 0.999
          1G4->3F4: 0.15
          1G4->3H5: 0.16
          1G4->3H4: 0.04
115       1G4->3F3: 0.02
          1D2->3F4: 0.43
          1I6->3F4: 0.6
          1I6->3H4: 0.16
          1I6->1G4: 0.14
120       1I6->1D2: 0.0
          3P0->1I6: 1.0


      # optional section
      energy_transfer:
125   # error given for samples 0.1, 0.3, 0.5, 1.0% Tm
          # 1G4 and 3H4
          # no ESA
          # with CR50_2(2.980e+09) and CR40(1.003e+07), e=0.46471
          # with CR50(2.587e+09, 11.2 A) and CR30(1.602e+08, 8.3 A), e=0.22872
130
```

```
        # 1G4 CR
        CR50:
            process: Tm(1G4) + Tm(3H6) -> Tm(3H4) + Tm(3H5) # good overlap
            multipolarity: 6
135         strength: 3.035e+09 # works well from 0.1 to 5%
        CR50_2:
            process: Tm(1G4) + Tm(3H6) -> Tm(3F3) + Tm(3F4) # good overlap
            multipolarity: 6
            strength: 0 # 3e+09
140
        # 3H4 CR
        CR30:
            process: Tm(3H4) + Tm(3H6) -> Tm(3F4) + Tm(3F4) # 1000 cm-1 mismatch
            multipolarity: 6
145         strength: 1.318e+08 # works well from 0.1 to 1%
        CR30_2:
            process: Tm(3H4) + Tm(3H6) -> Tm(3H5) + Tm(3F4) # 1000 cm-1 mismatch
            multipolarity: 6
            strength: 0 # 1.325e+10 # works well from 0.1 to 1%
150     CR40:
            process: Tm(3F3) + Tm(3H6) -> Tm(3F4) + Tm(3H5) # 300 cm-1 mismatch
            multipolarity: 6
            strength: 0 # 5e+8 # works badly from 0.1 to 1%

155     # 1D2
        # no ESA
        # with ETU53(4.621e+05, 2.7 A), e=0.50993
        # with ETU55(5.427e+00), e=3.1665

160     # UC to 1D2
        ETU53:
            process: Tm(1G4) + Tm(3H4) -> Tm(1D2) + Tm(3F4) # Best
            multipolarity: 6
            strength: 1.821e+11
165     ETU55:
            process: Tm(1G4) + Tm(1G4) -> Tm(1D2) + Tm(3F3)
            multipolarity: 6
            strength: 0 # 4.621e+9

170     # CR from 1D2
        CR60:
            process: Tm(1D2) + Tm(3H6) -> Tm(3H4) + Tm(3F3)
            multipolarity: 6
            strength: 0
175
        # 1I6
        # with ESA
        # no ET, e=4.6543 (all 1I6 curves are equal)
        # ET:
180     # ETU55_1I6(1.375e+05, 2.2 A) and CR70(5.465e+10, 13.3 A) error: 0.79155
```

```
        # 1D2 to 1I6 ETU, wrong power dependence
        ETU63:
            process: Tm(1D2) + Tm(3H4) -> Tm(1I6) + Tm(3F4)
185         multipolarity: 6
            strength: 0
        ETU62:
            process: Tm(1D2) + Tm(3H5) -> Tm(3P0) + Tm(3H6)
            multipolarity: 6
190         strength: 0


        # UC to 1I6
        ETU53_1I6:
195         process: Tm(1G4) + Tm(3H4) -> Tm(3H6) + Tm(1I6) # -700 cm-1 mismatch
            multipolarity: 6
            strength: 0 # 1e5
        ETU53_1I6_2:
            process: Tm(1G4) + Tm(3H4) -> Tm(1I6) + Tm(3H6) # 700 cm-1 mismatch
200         multipolarity: 6
            strength: 0 # 1e9
        ETU55_1I6:
            process: Tm(1G4) + Tm(1G4) -> Tm(3H5) + Tm(1I6) # -400 cm-1 mismatch
                , best
            multipolarity: 6
205         strength: 1.603e+06
        ETU54:
            process: Tm(1G4) + Tm(3F3) -> Tm(3P0) + Tm(3H6) # 700 cm-1 mismatch
            multipolarity: 6
            strength: 0 #8.180e+09
210     ETU54_2:
            process: Tm(1G4) + Tm(3F3) -> Tm(1I6) + Tm(3H6) # 700 cm-1 mismatch
            multipolarity: 6
            strength: 0 # 1e9


215     # CR from 1I6
        CR70_2:
            process: Tm(1I6) + Tm(3H6) -> Tm(3H4) + Tm(1G4) # 700 cm-1 mismatch
            multipolarity: 6
            strength: 0 # 2.445e+11 # 3.472e+11 # error: 2.1543
220     CR70:
            process: Tm(1I6) + Tm(3H6) -> Tm(1G4) + Tm(3H4) # 700 cm-1 mismatch
            multipolarity: 6
            strength: 0 # 5.465e+10
        CR27:
225         process: Tm(1I6) + Tm(3H5) -> Tm(1G4) + Tm(1G4) # 400 cm-1 mismatch
            multipolarity: 6
            strength: 0 # 6.518e+9 # 6.518e+10 # with ETU55_1I6: 2.779e+04,
                error: 2.3277
        CR71:
```

```
                process: Tm(1I6) + Tm(3F4) -> Tm(1D2) + Tm(3H4) # -400 cm-1 mismatch
230             multipolarity: 6
                strength: 0 # 5e9 # 5.084e+10 # with ETU55_1I6: 2.979e+04, error:
                    xxxx
            CR74:
                process: Tm(1I6) + Tm(3H4) -> Tm(1G4) + Tm(1D2)
                multipolarity: 6
235             strength: 0
            CR70_3:
                process: Tm(1I6) + Tm(3H6) -> Tm(1D2) + Tm(3H5)
                multipolarity: 6
                strength: 0 # 5e9
240


    # optional settings for optimization
    optimization:
        # optional: a list of energy_transfer or branching ratio labels to
            optimize.
245     # the fewer the number of parameters, the faster the optimization
        processes: [CR30, ETU55_1I6, CR50, ETU53]

        # optional: method for optimization of ET parameters. It can be:
        # leastsq, SLSQP, COBYLA, L-BFGS-B, or brute_force.
250     # leastsq, SLSQP or brute_force are recommended.
        method: COBYLA

        # various options for the optimization methods
        options:
255         tol: 1e-3
            N_points: 10
            min_factor: 1e-1
            max_factor: 1e1


260     # optional: optimize using these excitations
        #excitations: [Vis_473, NIR_980]



    # minimum and maximum excitation powers and the number of points
265 # to calculate the power dependence
    # note: a logarithmic scale is used
    # it's only mandatory if you want to calculate the power dependence
    power_dependence: [1e1, 1e7, 7]


270 # concentrations is a list of two lists:
    # [[sensitizer concentrations], [activator concentrations]]
    # to simulate for the concentration dependence
    # it's only mandatory if you want to calculate the concentration dependence
    # N_uc_list is an optional list of number of unit cells, one per
        concentration
275 concentration_dependence:
```

```
concentrations: [[0], [0.1, 0.3, 0.5, 1.0]]
N_uc_list: [75, 45, 45, 30]
```