Computer Physics Communications 184 (2013) 841–852

Contents lists available at SciVerse ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

In silico FRET from simulated dye dynamics[☆]

Martin Hoefling, Helmut Grubmüller[*]

Max Planck Institute for Biophysical Chemistry, Am Faßberg 11, 37077 Göttingen, Germany

A R T I C L E   I N F O

Article history:
Received 5 July 2012
Received in revised form
8 October 2012
Accepted 16 October 2012
Available online 26 October 2012

Keywords:
Fluorescence resonance energy transfer
Molecular conformation
Monte Carlo
Molecular dynamics
Markov Chain
Single molecule

A B S T R A C T

Single molecule fluorescence resonance energy transfer (smFRET) experiments probe molecular distances on the nanometer scale. In such experiments, distances are recorded from FRET transfer efficiencies via the Förster formula, $E = 1/(1 + (R/R_0)^6)$.

The energy transfer however also depends on the mutual orientation of the two dyes used as distance reporter. Since this information is typically inaccessible in FRET experiments, one has to rely on approximations, which reduce the accuracy of these distance measurements. A common approximation is an isotropic and uncorrelated dye orientation distribution.

To assess the impact of such approximations, we present the algorithms and implementation of a computational toolkit for the simulation of smFRET on the basis of molecular dynamics (MD) trajectory ensembles. In this study, the dye orientation dynamics, which are used to determine dynamic FRET efficiencies, are extracted from MD simulations. In a subsequent step, photons and bursts are generated using a Monte Carlo algorithm.

The application of the developed toolkit on a poly-proline system demonstrated good agreement between smFRET simulations and experimental results and therefore confirms our computational method. Furthermore, it enabled the identification of the structural basis of measured heterogeneity.

The presented computational toolkit is written in Python, available as open-source, applicable to arbitrary systems and can easily be extended and adapted to further problems.

Program summary

Program title: md2fret
Catalogue identifier: AENV_v1_0
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AENV_v1_0.html
Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland
Licensing provisions: GPLv3, the bundled SIMD friendly Mersenne twister implementation [1] is provided under the SFMT-License.
No. of lines in distributed program, including test data, etc.: 317880
No. of bytes in distributed program, including test data, etc.: 54774217
Distribution format: tar.gz
Programming language: Python, Cython, C (ANSI C99).
Computer: Any (see memory requirements).
Operating system: Any OS with CPython distribution (e.g. Linux, MacOSX, Windows).
Has the code been vectorised or parallelized?: Yes, in Ref. [2], 4 CPU cores were used.
RAM: About 700MB per process for the simulation setup in Ref. [2].
Classification: 16.1, 16.7, 23.

External routines: Calculation of $R\kappa^2$-trajectories from GROMACS [3] MD trajectories requires the GromPy Python module described in Ref. [4] or a GROMACS 4.6 installation. The md2fret program uses a standard Python interpreter (CPython) v2.6+ and < v3.0 as well as the NumPy module. The analysis examples require the Matplotlib Python module.

Nature of problem:
Simulation and interpretation of single molecule FRET experiments.

[☆] This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

[*] Corresponding author.
E-mail address: hgrubmu@gwdg.de (H. Grubmüller).

0010-4655/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.cpc.2012.10.018

Solution method:
Combination of force-field based molecular dynamics (MD) simulating the dye dynamics and Monte Carlo sampling to obtain photon statistics of FRET kinetics.

Additional comments:
!!!!! The distribution file for this program is over 50 Mbytes and therefore is not delivered directly when download or Email is requested. Instead a html file giving details of how the program can be obtained is sent. !!!!!

Running time:
A single run in Ref. [2] takes about 10 min on a Quad Core Intel Xeon CPU W3520 2.67GHz with 6GB physical RAM

References:
[1] M. Saito, M. Matsumoto, SIMD-oriented fast Mersenne twister: a 128-bit pseudorandom number generator, in: A. Keller, S. Heinrich, H. Niederreiter (Eds.), Monte Carlo and Quasi-Monte Carlo Methods 2006, Springer; Berlin, Heidelberg, 2008, pp. 607-622.
[2] M. Hoefling, N. Lima, D. Hänni, B. Schuler, C. A. M. Seidel, H. Grubmüller, Structural heterogeneity and quantitative FRET efficiency distributions of polyprolines through a hybrid atomistic simulation and Monte Carlo approach, PLoS ONE 6 (5) (2011) e19791.
[3] D. V. D. Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark, H. J. C. Berendsen, GROMACS: fast, flexible, and free., J Comput Chem 26 (16) (2005) 1701–1718.
[4] R. Pool, A. Feenstra, M. Hoefling, R. Schulz, J. C. Smith, J. Heringa, Enabling grand-canonical Monte Carlo: Extending the flexibility of gromacs through the GromPy Python interface module, Journal of Chemical Theory and Computation 33 (12) (2012) 1207–1214.

1. Introduction

In vivo distance measurements are key to reveal mechanisms of biomolecular processes like protein folding. Monitoring distance dynamics recovers changes in the underlying molecular structure. Optical techniques such as fluorescence resonance energy transfer (FRET) are particularly suitable for this task due to their *in vivo* compatibility.

However, a number of approximations limit the accuracy at which distances R or distance distributions can be retained. To improve the accuracy, we recently proposed a technique combining molecular dynamics (MD) simulations with FRET [1].

FRET is an optical method to determine distances between two sites labeled by fluorescent dyes, which rests on the fact that the resonance energy transfer (RET) efficiency from an excited donor dye to its acceptor counterpart depends on the inter dye distance. This is described by Förster's theory [2], which results in a relation of distances R between both dyes and the transfer efficiency,

$$E_{\mathrm{RET}}(R) = \frac{1}{1 + \left(\frac{R}{R_0}\right)^6}. \tag{1}$$

The Förster radius R_0 is defined as the distance with equal probability for RET and donor fluorescence.

However, the RET efficiency also depends on the mutual orientation of the donor and acceptor. This instantaneous dye orientation during RET is inaccessible in experiments, since the molecular structure at atomic resolution is difficult to retrieve at ambient conditions. Therefore orientation distribution models and their mean values are commonly employed.

In such a model, Förster approximated the coupling between the two ground and excited state transition densities of donor and acceptor as dipole–dipole coupling. This approximation results in the $\frac{1}{R^6}$ distance dependency of the squared coupling potential, which is also included in Eq. (1). The orientation factor κ corresponds to the orientation dependency of the dipole–dipole coupling and ${R_0}^6$ in Eq. (1) is proportional to κ^2.

Förster further assumed that all dye orientations are equally likely and uncorrelated. By further assuming that dye-reorientation is fast compared to the donor lifetime, the time averaged orientation factor $\langle \kappa^2 \rangle_t = 2/3$ [3]. As a consequence, Eq. (1) is orientation independent under these assumptions.

Deviations from the isotropic mean of $\langle \kappa^2 \rangle_t = 2/3$ are present, even in systems where the dye orientation space is not strongly restricted, as was demonstrated by simulations [4,1]. Therefore, the approximations in κ^2 limit the accuracy of distances recovered from efficiencies.

To overcome the uncertainty in experimental dye orientation distributions, we evaluated a combination of dye dynamics from molecular dynamics (MD) simulations and single molecule FRET experiments [1]. To this end, we developed a computational toolkit to simulate single molecule FRET (smFRET). By comparison to the experimental efficiency distributions, this toolkit allowed us to test whether the dye dynamics are sufficiently sampled by MD simulations for distance reconstruction. Further, simulation of smFRET experiments assign the conformational heterogeneity detected by simulations to the experimentally observed heterogeneity in the efficiency distributions.

The focuses of this paper are the algorithms and the implementation of our previously described smFRET simulations [1], since they provide a versatile tool for studying FRET experiments.

2. Theory

Before we focus on the implementation and usage of the computational toolkit, the simulation ensemble setup used throughout this paper and two key aspects of the theory are covered in the following. Ref. [1] provides an in-depth discussion of the theoretical background.

2.1. Exemplary simulation ensemble setup

FRET experiments on dye labeled poly-proline laid the foundation for FRET as distance measurement method [5]. Thus, a polyproline system studied earlier by Schuler et al. with smFRET [6] serves as a model system. For this system, we were able to demonstrate that our smFRET simulation technique adequately describes the experiment [1]. The MD simulation method and particular setup is summarized in Ref. [1].

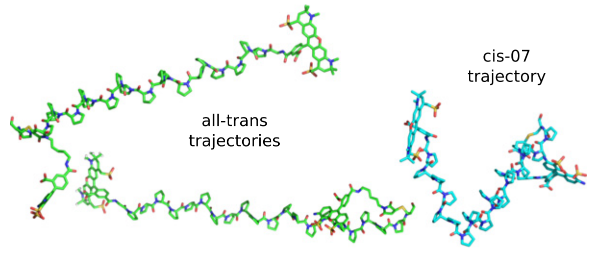

Fig. 1. Simulation trajectory ensemble. Two poly-proline isomers, all-trans (green) and cis-07 (cyan) serve as test set for the Monte Carlo FRET program. The test set contains two all-trans simulations to demonstrate the effect of dye orientation heterogeneity of the same isomeric state. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

In water, poly-proline has a non-negligible fraction of cis-isomers in the ensemble. Since the isomerization times are slow compared to simulation time, transitions are not observed during a simulation and multiple simulations at different isomeric states were performed. Moreover, multiple simulations are performed in a typical setup to achieve an improved sampling of the dye dynamics.

To demonstrate more easily the aspects of burst formation from multiple trajectories in this work, we limit the ensemble to a set of three trajectories as shown by Fig. 1. This reduced three trajectory ensemble contains two isomers of poly-proline, the all-trans as well as the cis-07 species, with the 7th peptide bond in cis-configuration. The all-trans configuration is sampled by two trajectories, one with larger and a second one with smaller dye separations. The cis-07 configuration is sampled by another single 100 ns trajectory, such that the final trajectory ensemble consists of three 100 ns trajectories in total.

2.2. FRET kinetics

The decay paths of a donor molecule in the excited state are summarized in the following kinetic scheme:

$$\begin{tikzcd} & \mathrm{D}+\mathrm{A} & & \mathrm{D}+\mathrm{A} \\ \mathrm{D}+\mathrm{A}+h\nu \arrow[r] & \mathrm{D}^*+\mathrm{A} \arrow[u, "k_{\mathrm{Di}}"'] \arrow[d, "k_{\mathrm{D}}"] \arrow[rr, "k_{\mathrm{FRET}}(t)"] & & \mathrm{D}+\mathrm{A}^* \arrow[u, "k_{\mathrm{Ai}}"'] \arrow[d, "k_{\mathrm{A}}"] \\ & \mathrm{D}+\mathrm{A}+h\nu_{\mathrm{D}} & & \mathrm{D}+\mathrm{A}+h\nu_A \end{tikzcd} \quad (2)$$

This correlation describes photo emission or thermal de-excitation events from donor D or acceptor A present in FRET experiments. After excitation, the donor is in the excited state D*. From here, thermal de-excitation k_{Di} and fluorescence k_{D} compete with the excitation transfer via FRET to the acceptor, with rate k_{FRET}. Excitation transfer to the acceptor is followed by either thermal de-excitation k_{Ai} or fluorescence decay k_{A}. The above kinetics is described by Markov chains with a Markov matrix formed by the individual transition probabilities.

Since FRET probability depends on the dye orientation and distance, the rate $k_{\mathrm{FRET}}(t)$ is time dependent as well. Due to this time dependency of the FRET rate, the transition probabilities in the Markov matrix are also time dependent. As a consequence, the Markov chain has no time independent analytical solution. Instead, Markov chains are generated by propagating the system in discrete time steps according to current transition probabilities. The transition probabilities are considered in a Monte Carlo approach by using random numbers in each step to discriminate between the possible pathways.

To obtain this simple kinetics, with $k_{\mathrm{FRET}}(t)$ as the only time dependent rate, the fluorescence quantum yields of donor and acceptor are considered independent on the dye conformation and the donor and acceptor dye decays are described by a single exponential in the absence of FRET.

2.3. Photon bursts

The second key aspect discussed here is connected to the single molecule character of experiments and to the combination of multiple MD simulations in the smFRET simulations.

MD trajectories are typically shorter than the experimental burst duration of milliseconds [6]. Therefore multiple simulations are combined to increase the sampling of dye orientations. This combination of multiple trajectories depends on the interconversion times between dye conformation populations present in the simulation trajectories.

In addition, slow dynamics of the molecular structure compared to the burst duration, such as isomerization, can appear in experiments. This may lead to conformational heterogeneity of burst efficiencies in single molecule FRET experiments [7] and has to be modeled by an appropriate combination of the trajectory ensemble as well.

To model the impact of the above mentioned heterogeneity in molecular structure and dye conformation, we suggest three different approaches to combine multiple simulation trajectories in the photon burst generation [1]. Due to the combination of multiple trajectories in the photon generation, photons in one burst may be generated from multiple trajectories as shown in Fig. 2(b) and (c):

- *All trajectories*: Photons of each burst are generated from all trajectories, according to the ensemble probability of the trajectory. This option should be used, when the entire trajectory ensemble corresponds to the dynamics within one burst. This is the case, when no slow transitions in the molecule compared to the burst duration, i.e. isomerization transitions, are present.
  As a result, each burst averages over the entire ensemble dynamics of dyes and molecule (Fig. 2(b)).
- *Trajectories of the same species*: When slow transitions of the molecule compared to the burst duration are present, a distinct species is assigned to each trajectory. This species is typically a specific isomeric state, as for poly-proline, or describes an enzyme in its substrate bound and unbound state for example. Before each burst computation, a species is randomly chosen according to the species probability and photons of the burst are generated from all species trajectories. Each burst is then constructed from photons of all trajectories of a single species.
  By this construction, each burst averages over slow and fast dye dynamics and the dynamics of the molecule within each species (Fig. 2(c)), but not over the dynamics of different species.
- *One trajectory*: Finally, only one trajectory can be used for each photon burst generation. This corresponds to systems where a single simulation samples one dye conformation and interconversion between multiple dye conformations is slow compared to the burst duration.
  As a consequence, each photon burst averages only over the fast dye dynamics and dynamics of the molecule within one trajectory (Fig. 2(d)).

The burst generation method determines the shape of the efficiency distribution [1]. This will be later demonstrated in Section 3 on the reduced ensemble that is described in Section 2.1.

3. Program usage

In the next section, the extraction of dye dynamics from simulation trajectories is discussed. This is followed by an explanation of the in- and output options of the FRET Monte Carlo `md2fret.py` program.

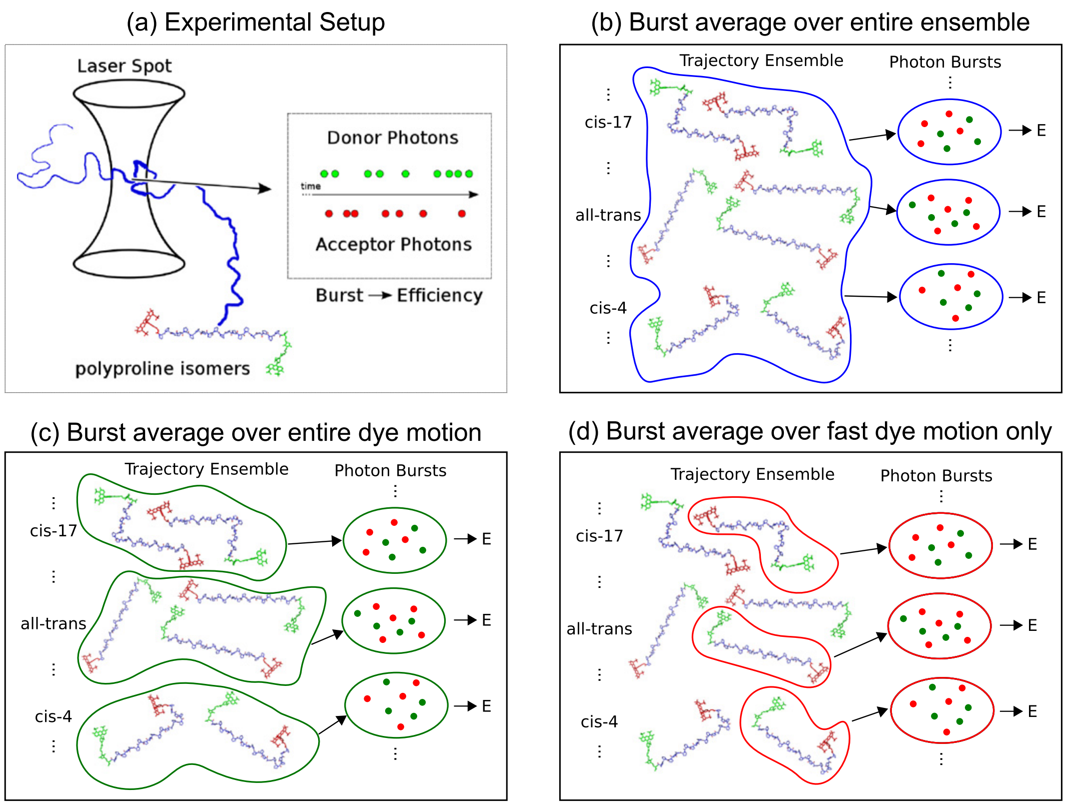

Fig. 2. Combining simulation trajectories to mimic experimental bursts. (a) Single molecule FRET experiments measure photon bursts from a single molecule, diffusing through the laser spot. (b–d) describe the bursts formation from simulations. In (b), photons of each burst originate from the entire trajectory ensemble. All trajectories of the same isomer are used for each burst in (c) whereas in (d) each burst originates from a single trajectory from the ensemble.

3.1. *Simulation trajectory preprocessing*

For smFRET simulations, time ordered samples of the inter-dye distance R and their mutual orientation of the transition dipole moments κ^2 are required ($R\kappa^2$-trajectories). This step is separated from the actual FRET Monte Carlo program to avoid dependency on trajectory processing libraries, e.g. from GROMACS.

Two tools are provided within our toolkit for this task. The first one is g_dyecoupl, also included in the GROMACS 4.6 version. As a second option, a Python program based on the GromPy module [8] is provided, which can easily be adapted to cases were the transition dipole moment definition is complex, i.e. when the dipole is out of plane, and thus the g_dyecoupl options are not suitable. Both tools process all trajectory file formats readable by GROMACS [9,10] and Visual Molecular Dynamics [11]. In the following, the conversion based on the GROMACS tool g_dyecoupl is explained.

g_dyecoupl is a GROMACS command line tool which requires an input trajectory (`-f` switch) and an index file (`-n` switch). The supplied index file contains two index groups, each defining the direction of the transition dipole moment of one of the two dyes. The dipole moments and the dye centers are determined from at least one but preferably multiple dye atom pairs as shown in Fig. 3. The corresponding index file contains:

```
[ donor ]
392 403 393 405
[ acceptor ]
8 43 9 45
```

The average vector of all pairs is used to calculate the direction of the transition dipole moment while the average atomic positions serve as dye centers for distance calculation. $R\kappa^2$-trajectories in a format readable by `md2fret.py` are written out using the `-o` switch. Further options of g_dyecoupl can be looked up using the `-h` switch.

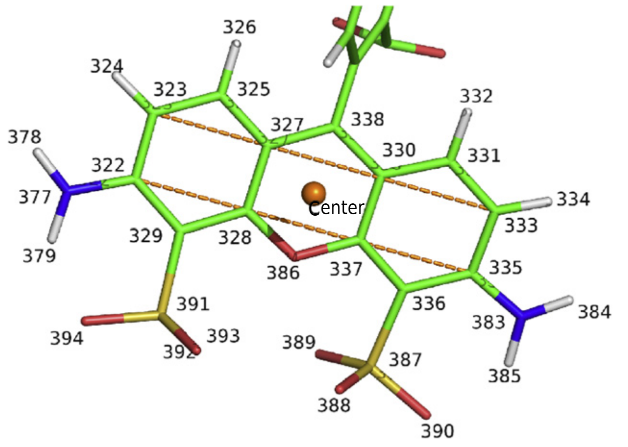

Fig. 3. Dye transition dipole moment and the dye position. The average vector of two atom pairs (323–333 and 322–335) is used to define the transition dipole moment from ground to the first excited state. The same four atoms define the center position of the dye used to determine the inter dye distance.

3.2. *Program input*

In the following, the required and optional input parameters for the FRET burst generator program `md2fret.py` are summarized and explained in detail. An overview of all input and output options is obtained when calling `md2fret.py` with the `-h` switch (see Appendix A).

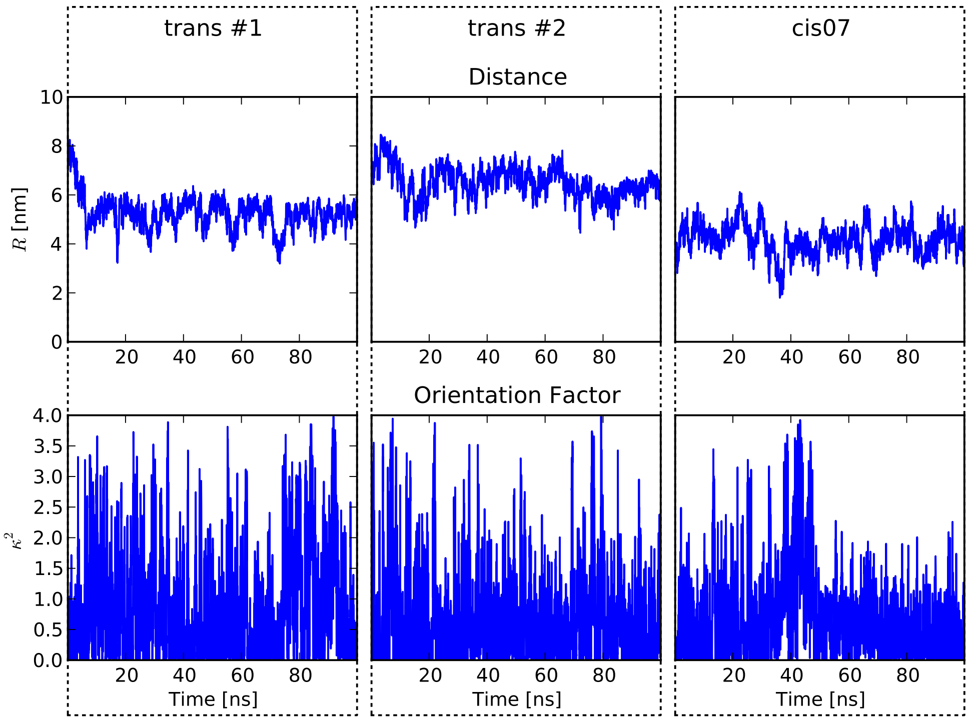

Fig. 4. Example trajectories for the dye to dye distance R [nm] and orientation factor κ^2. Each column refers to one simulation from the simulation ensemble shown in Fig. 1.

Required input:

- $R\kappa^2$-trajectories extracted from MD trajectories.
- Ensemble probabilities of trajectories.
- Configuration file.

Optional input:

- Experimental burst size distribution.
- Random number generator seed (when set to a constant integer number one obtains reproducible simulation runs).

3.2.1. $R\kappa^2$-trajectories

After pre-processing simulation trajectories with g_dyecoupl, time ordered samples of $R\kappa^2$ are obtained as shown in Fig. 4 for the three trajectories contained in the test set. A directory containing $R\kappa^2$-trajectory files is specified with the `-d` switch of `md2fret.py`. The ensemble species of each trajectory in the directory is determined from the name of the trajectory.

Each $R\kappa^2$-trajectory contains three columns:

1. the time in ps
2. the distance in nm
3. the orientation factor κ^2.

For the Monte Carlo simulation, only the distance and the orientation factor is used, the time step of the $R\kappa^2$-trajectories is read from the `deltat` option in the configuration file explained below.

The optimal time step (snapshot saving interval) depends on the fluctuation frequency of R and κ^2. In Ref. [1], we used a time step of 1 ps, which is in our system two orders of magnitudes below the fastest autocorrelation times of 150 ps in κ^2 (Table 3 in Ref. [1]).

The trajectory format is chosen with the `-r` switch. When stored as plain text, the file extension has to be `.dat` and the `md2fret.py` switch `-r dat` has to be used. The content of the files are three columns of floats separated by

```
...
2.6000000000000e+02 3.5592069844728e+00 1.4830565147298e-01
2.7000000000000e+02 3.5408017395565e+00 1.3730690344792e-02
2.8000000000000e+02 3.4672607815154e+00 9.0443186745303e-03
2.9000000000000e+02 3.3712963428854e+00 7.0619270783520e-02
...
```

The second option is the numpy zip format (.npz) requested with `-r npz`. The three columns have to be stored as a single numpy array 'arr_0' in the npz file, which is the default for the first stored array. The numpy zip format produces smaller files and is natively supported by the numpy Python library [12] and the GromPy based conversion script [8].

3.2.2. Trajectory probabilities in the ensemble

As explained in Section 2.3, multiple trajectories from different species can be combined for the photon generation. `md2fret.py` reads an input file specifying the probability of each species (`-p` switch). In this file, the first column is a regular expression matching all file names belonging to the species. In the second column, an arbitrary unique name has to be specified while the last column specifies the probability of the species in the ensemble. In the simplest case with only one species and thus the same weight for all trajectories, the file content looks like

```
.* all 1.0
```

For the test system containing two species, we define the probabilities of the species as follows

```
.*cis07.* 07cis 0.2
.*trans.* trans 0.8
```

Within each species, the trajectories are weighted by their sampled time.

3.2.3. Configuration Parameter File

The configuration file of the Monte Carlo FRET simulation is subdivided into multiple blocks, each with its title in rectangular brackets. The name of the configuration file is specified via the `-c` switch of `md2fret.py`. In the following, each block with configuration parameters is explained.

```
[Dye Constants]
tauD=4000
tauA=3900
QD=0.77
QA=0.85
```

In the `[Dye Constants]` block, lifetimes τ (in ps) and quantum yields Q are specified for donor D and acceptor A. The here given quantum yields also contain the detector efficiency of the instrument.

```
[FRET Constants]
R0 = 5.4
kappa = 0.66666666
```

The `[FRET Constants]` section specifies the Förster radius R_0 of the dye pair and the incorporated orientation factor κ^2 of the orientation model employed in determining R_0. Commonly, κ^2 is 2/3 when isotropic orientation distributions are used. From both specified constants the orientation independent part of the Förster radius is calculated since the orientation from the $R\kappa^2$-trajectories is used in the Monte Carlo FRET procedure.

```
[Burst Size Distribution]
#analytical burst size distribution
method = analytical
llimit = 20
ulimit = 150
lambda = -2.3

#or
#burst sizes from experimental distribution (file)
method = file

#cutoff without quantum yield correction
apply = true-photon

#or
#cutoff with quantum yield correction
apply = corrected
```

In the `[Burst Size Distribution]` block, the method and parameters for the burst size generation are specified. As a method, an analytical distribution from a power law function or burst sizes from a file can be chosen. For the power law, the exponential prefactor λ has to be specified as well as upper and lower cutoffs.

The `apply` parameter determines how the burst size cutoff was applied in the experiment. The difference is explained by the following example:

We discard all bursts that are smaller than 20 photons. Lets now assume that the donor quantum yield is 0.75 and the acceptor quantum yield is 0.5. In one burst, we got 6 donor and 12 acceptor photons. When the `true-photon` count as cutoff is applied, than the burst is discarded $(6 + 12) < 20$. However, if we use the `corrected` number of photons with perfect detector and a quantum yield of 1, the burst is accepted $(\frac{1}{0.5} \cdot 6 + \frac{1}{0.75} \cdot 12) > 20$.

We would finally emphasize two critical assumptions that need to be made both in the conventional treatment in experiments and also in our method. First, the burst size needs to be assumed dependent on constant donor and acceptor brightness only (see `QD` and `QA` parameter). In particular, the brightness of donor and acceptor has to be independent of the instantaneous FRET efficiency. Second, our burst generation method does not include any background signal from scattering or dark detector counts. Therefore, in order to facilitate direct comparison to background corrected experimental data, the background needs to be small with respect to the signal and thus only slightly affect the used photon statistics.

```
[Burst Accumulation]
#All photons in a burst are generated from...
#...a single trajectory according to the trajectory ensemble
#probability
method = trajectory

#or
#... all trajectories of a single species according to the
#species ensemble probability
method = same-species

#or
#... ALL trajectories
method = all
```

The `[Burst Accumulation]` block is used to select one of the three methods to form bursts from photons described earlier in Section 2.3.

```
[Monte Carlo]
#parameters affecting the photon generation
minstarttraj = 0
maxstarttraj = 1000
deltat = 10
photrejectdist = 1.0
rejectretry = 10

#number of bursts to generate
nbursts = 12000
```

The block `[Monte Carlo]` sets up various parameters of the photon and burst generator. The `minstart` and `maxstart` parameters define the part of the trajectory that is used for the photon generation. `minstart` is the number of samples relative to the beginning of the trajectory while `maxstart` is relative to the end of the trajectory.

The minimum starting samples have to be set to a value > 0, for example when equilibration in the trajectory is present. To avoid boundary effects the maximum starting samples should leave a part in the order of the donor decay at the end of the trajectory unused.

The time step in the trajectories (in ps) is set with `deltat`.

Example: A time step (`deltat`) of 10 ps, ignoring the first 5 ns of equilibration from each trajectory and starting a photon generation not in the last 10 ns results in parameters `minstart = 500` and `maxstart = 1000`.

To avoid the generation of photons at artificially high efficiencies where other effects like Dexter energy transfer are dominating, `photrejectdist` (in nm) and `rejectretry` can be set. Whenever a inter-dye distance below `photrejectdist` is reached during photon generation, the generation is aborted and retried up to `rejectretry` times.

`nbursts` specifies the number of bursts to generate.

```
[System]
#fastest photongenerator
photongenerator = cextension

#compiled from Python code
photongenerator = cython

#alternatively the slow Python implementation
photongenerator = Python

ncpu =  -1
```

Parameters in the `[System]` block set the photon generator and number of CPUs to use. The fastest option to generate photons is the `cextension` photon generator. This photon generator is implemented in C using the Python and numpy C API and the

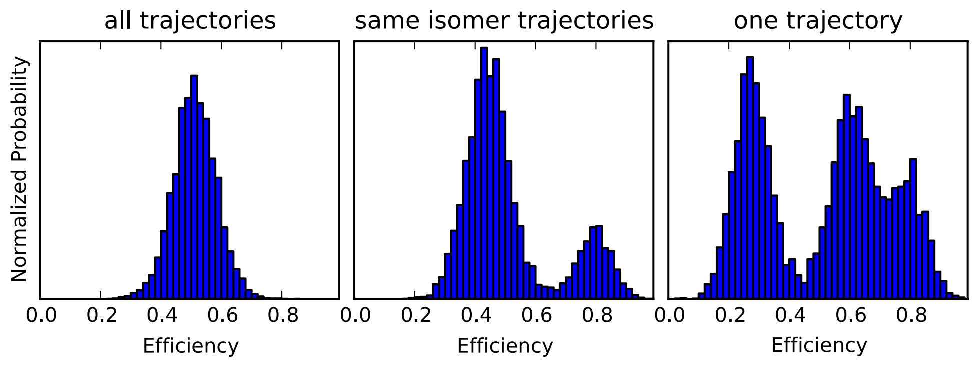

Fig. 5. Efficiencies of generated photon bursts. Left: Photons of each burst are generated from all trajectories. Middle: Photons of each burst are generated from trajectories of the same isomer only. Right: Photons of each burst are generated from a single trajectory.

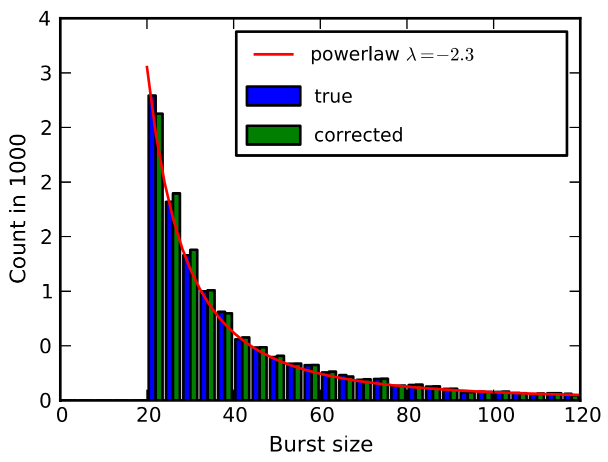

Fig. 6. Burst sizes from a FRET simulation with analytical burst size distribution. The burst sizes were generated using a power law (plotted in red). The blue bars represent the counts without corrections while for the green bars, the corrections for detector efficiency and quantum yield were applied. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

SIMD Friendly Mersenne Twister random number generator [13]. There is also much slower `Python` implementation of the photon generator, which can be used for prototyping of new or altered photon generation methods (see `tryPlainGetPhoton` and `setPhotonGenerator` functions in the `FRETUtils.Photons` Python module). When implemented in Python, the photon generator can be converted into a compiled Python extension using cython. For illustration purposes, we therefore also included a `cython` photon generator derived from the plain Python implementation in the framework.

3.2.4. *Experimental FRET bursts file (optional)*

As optional input, a burst file with experimental burst sizes can be provided with the `-k` switch of `md2fret.py`. Together with the appropriate burst generator set in the configuration file, burst sizes will be drawn randomly from the sizes experimentally obtained.

```
...
19,7,16.58802966,4.199380926
38,13,34.30130912,9.820283086
41,22,35.60566827,18.33920318
53,18,50.7423003,19.22946756
...
```

The first and second column are the measured donor and acceptor counts, while the third and fourth columns include corrections depending on the experimental setup. In the sample file, corrections for the quantum yield, the detector efficiency, cross-talk between donor and acceptor channel and background in the channels are applied as described in Ref. [14].

The used set of columns is determined by the `apply` parameter in the `[Burst Size Distribution]` block of the configuration file. Only the burst size, i.e. the sum of both channels, is used in the FRET burst generation to obtain photon statistics comparable to the experiment.

3.3. *Program output*

3.3.1. *Burst efficiencies*

An important quantity for comparison with the single molecule FRET experiments is the burst efficiency distribution. Burst efficiencies can be written out as a plain text file with one burst efficiency per line using the `-e` switch of `md2fret.py`. The output file looks like

```
...
5.0657894736842112686e-01
9.0917347865576747752e-01
5.8847736625514401886e-01
3.9450395901257567796e-01
5.5580233203775675222e-01
...
```

Fig. 5 shows three efficiency histograms demonstrating the impact of the burst formation method. When all trajectories are used for each burst generation, a single peak is obtained, whose width is limited by the shot noise of the underlying burst size distribution (left).

Using only trajectories from the same isomer for each burst generation, the all-trans and cis07 species result in separated peaks with a ratio of 80%–20% of all-trans to cis07 in our test case (middle).

If only a single trajectory is used for each burst generation, the heterogeneity between the two all-trans trajectories of our test set becomes visible. Here, the origin is in the difference in average dye distance due to different dye conformations in the all-trans trajectories in addition to the trans-cis heterogeneity (right).

The above example shows the importance of properly including longer timescale dynamics, when single simulations are not able to cover transition between conformation states.

3.3.2. *Burstsizes*

The burst sizes are returned (`-l` switch) as plain text files with two columns for uncorrected counts and corrected counts for quantum yield and detector efficiency. Both values are returned as floating point values and each line is one burst.

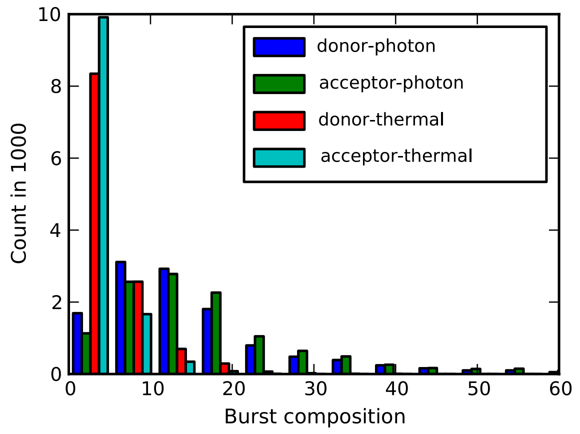

Fig. 7. Individual burst components histograms. Large donor and acceptor counts are rarely found while bursts with a small size (donor + acceptor photons) have a large probability to contain thermal decay from donor and acceptor due to the large quantum yields of the here used dyes.

```
...
2.300000000000000000e+01 2.309090909090908994e+01
7.200000000000000000e+01 7.206493506493507084e+01
8.100000000000000000e+01 8.115584415584416433e+01
2.800000000000000000e+01 2.868831168831168554e+01
...
```

Fig. 6 shows the burst size histogram from a FRET simulation using a analytical burst size distribution. For illustration purposes, the burst size distribution generating power law is plotted red in Fig. 6.

3.3.3. *Burst Composition*

For further burst analysis, the burst composition can be written out as 4 columns and one line per burst (`-b` switch).

```
...
13 3 3 0
25 65 13 14
13 29 4 3
...
```

The first and second column are the donor and acceptor photon counts and the third and fourth columns are the thermal de-excitation of donor and acceptor, respectively. Fig. 7 shows an exemplary histogram of each of the four burst components.

3.3.4. *FRET/donor-decay events*

For each burst, the trajectory point in time of donor decay (FRET, thermal or photo-emission) is written out as a series of floats in one line (`-f` switch).

```
...
74300.000000 53210.000000 26240.000000 ...
46870.000000 31260.000000 14390.000000 ...
18770.000000 41140.000000 32550.000000 ...
...
```

This analysis is particularly valuable when running FRET simulations on a single trajectory since variations with the change of FRET probability can be seen in the de-excitation counts. Moreover, it also serves to check, if the `maxstarttraj` parameter has been chosen sufficiently large, to avoid influence of the trajectory boundaries. The histogram depicted in Fig. 8 was generated with staring times on the first 90 ns of the trajectory. As shown by Fig. 8, the photon count at the trajectory boundary (100 ns) is negligible.

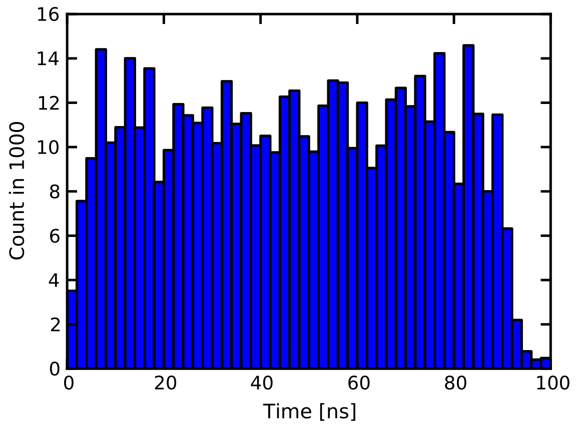

Fig. 8. Histogram of FRET and Donor Emission Events showing the dependence of Donor decay of the trajectory time. ≈ 8 ns are required for a saturation of donor emission events on the trajectories. The photon generation starting point `maxstarttraj` of 90% from the trajectory time leads to negligible boundary effects at the end of the trajectory, less than 0.4% photons are recorded beyond 95 ns.

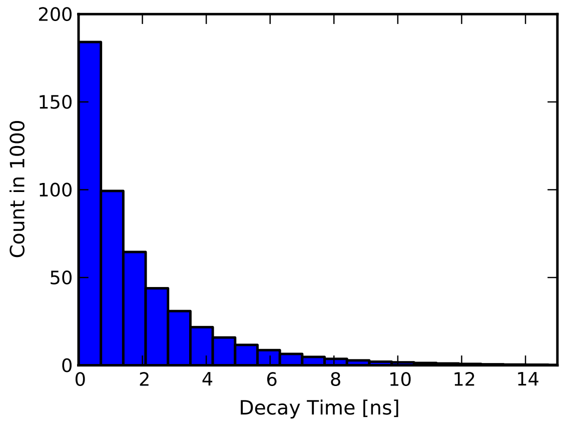

Fig. 9. Histogram of the donor decay times (lifetimes). An intrinsic lifetime (without the presence of an acceptor) of 4 ns was used for the FRET dyes resulting in this histogram.

3.3.5. *Decay times*

The decay times or lifetime of the donor can be written out as a series of floats per photon with one burst per line (`-t` switch).

```
...
1000.000000 60.000000 5590.000000 ...
590.000000 1420.000000 650.000000 ...
3480.000000 3120.000000 2460.000000 ...
...
```

As shown by Fig. 9, the presence of FRET as competing decay channel reduces the intrinsic donor lifetime of 4 ns.

3.3.6. *Binary dump for custom analysis*

The binary output requested with the `-z` switch pickles[1] the Python burst objects into a file. The burst objects can then be unpickled in custom Python analysis code.

4. Program structure

In the following, the algorithms to generate FRET bursts from simulation are summarized. The main routine is subdivided into three parts:

[1] Python object serialization to obtain data persistence.

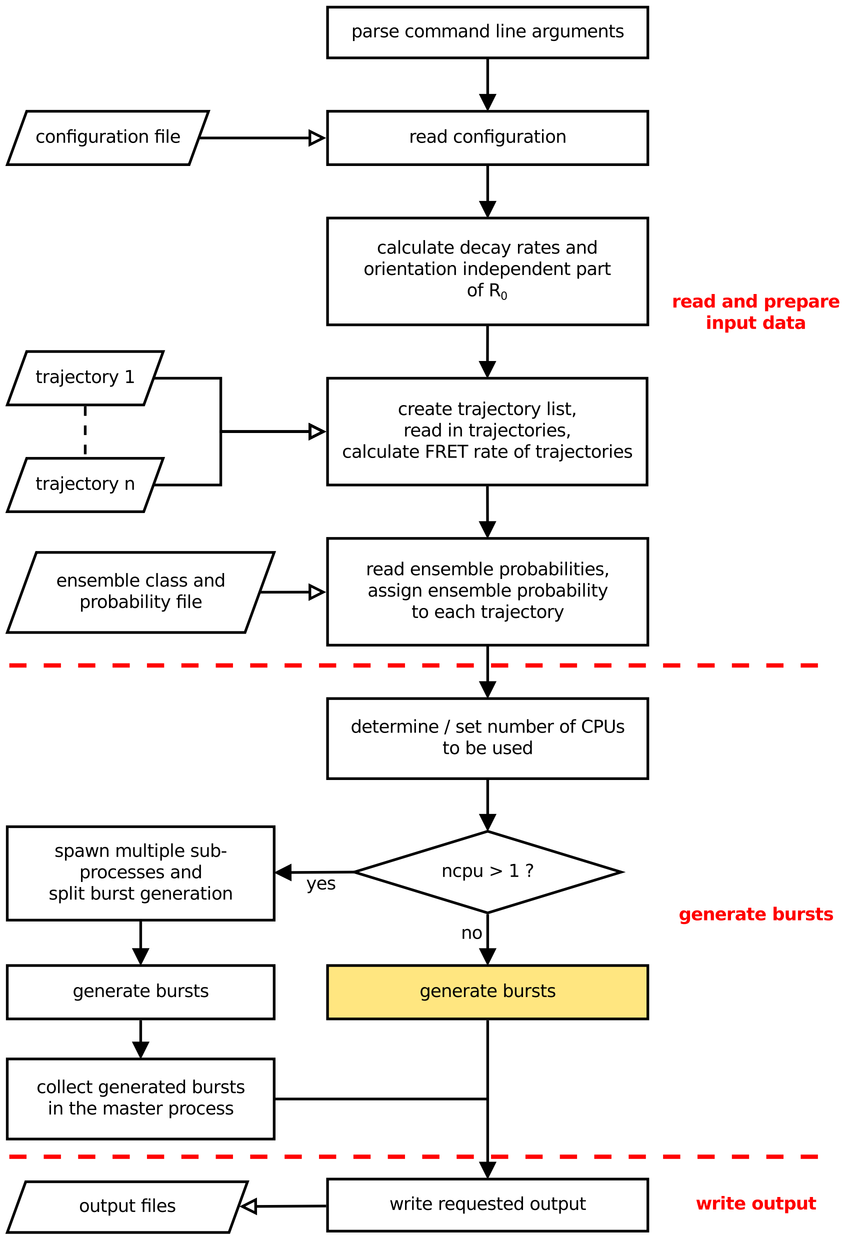
parse command line arguments
configuration file
read configuration
calculate decay rates and orientation independent part of R0
read and prepare input data
trajectory 1
trajectory n
create trajectory list, read in trajectories, calculate FRET rate of trajectories
ensemble class and probability file
read ensemble probabilities, assign ensemble probability to each trajectory
determine / set number of CPUs to be used
spawn multiple sub-processes and split burst generation
ncpu > 1 ?
yes
no
generate bursts
generate bursts
generate bursts
collect generated bursts in the master process
output files
write requested output
write output

Fig. 10. Flowchart of the main program. The program is subdivided into three parts: Input preparation, burst generation and writing the output. The yellow generate burst routine is shown in Fig. 11. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

1. reading input and preparation
2. photon burst generation
3. writing output (not discussed).

4.1. *Reading input and preparation*

As shown in Fig. 10, the first step is parsing the command line arguments and reading in the configuration file. After this, the static decay rates (Eqs. 6 and 7 in Ref. [1]) are calculated from the fluorescence lifetimes and quantum yields. Also the orientation independent part (Eq. 9 in Ref. [1]) of the Förster radius R_0 is determined here. Next, all $R\kappa^2$-trajectories in the specified trajectory directory are read in and converted into FRET efficiency time series using Eq. (1). As the final step of the preparation, the ensemble species probabilities are read in and an ensemble species is assigned to each trajectory.

4.2. *Photon burst generation*

The next step is the generation of the requested number of bursts for which multiple sub processes can be launched, depending on the requested number of CPU cores. Fig. 11 shows the subdivision of the burst generation into the generation of the burst sizes and the actual generation of each burst.

To determine burst sizes, either an analytical function or an experimental burst size distribution is used. Currently, the only analytical function implemented is the power law. From this power law, the burst sizes are generated based on acceptance/rejection depending on the function value in the specified range. Any function defined on the given burst range (`llimit` and `ulimit`) can serve as a probability density function using this method.

When experimental burst sizes are used, first the burst list is read in and the burst sizes are calculated. For the burst size generation, a random entry from the experimental list is drawn.

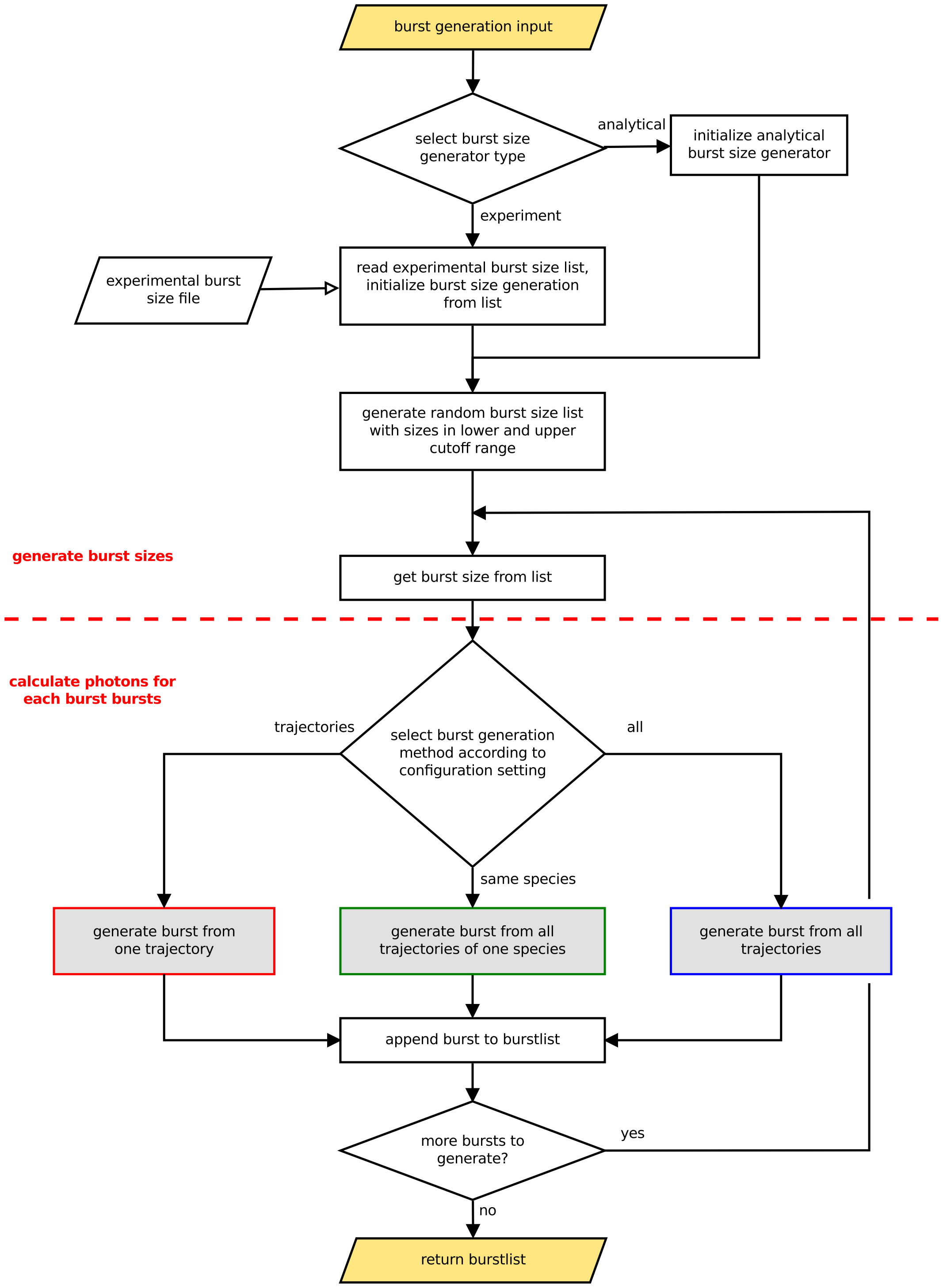
burst generation input
select burst size generator type
analytical
initialize analytical burst size generator
experiment
experimental burst size file
read experimental burst size list, initialize burst size generation from list
generate random burst size list with sizes in lower and upper cutoff range
generate burst sizes
get burst size from list
calculate photons for each burst bursts
trajectories
select burst generation method according to configuration setting
all
same species
generate burst from one trajectory
generate burst from all trajectories of one species
generate burst from all trajectories
append burst to burstlist
more bursts to generate?
yes
no
return burstlist

Fig. 11. Flow chart of the burst size generation and burst generation method selection. The burst generation is subdivided in two parts. First the burst sizes are picked either from experimental burst sizes or are generated according to an analytical distribution. Second, for each burst size, the actual burst generation following one of the three burst protocols (all, same species, trajectories) is executed. The box colors of each method box correspond to the paths in Fig. 12.

Once a list with the burst sizes is generated with one of the two methods, the photon composition is determined for each burst. For this calculation three different methods are implemented as shown in Fig. 12. In each of the methods, a random species from the ensemble is selected (1) followed by the random selection of a trajectory from the species (2). Then a photon is generated from the selected trajectory. Depending on the method, step (1) and (2) (represented in blue color in Fig. 12), just step (2) (green in Fig. 12) or none of the two steps (red in Fig. 12) is repeated for each subsequent photon in the burst.

Fig. 13 shows the generation of a photon, once a trajectory has been selected. First a random starting time is picked. Then, a random number is generated to discriminate between the possible pathways summarized in Scheme 2. When neither donor fluorescence nor FRET transfer to the acceptor occurs, the time is incremented and a new random number is drawn. Important here

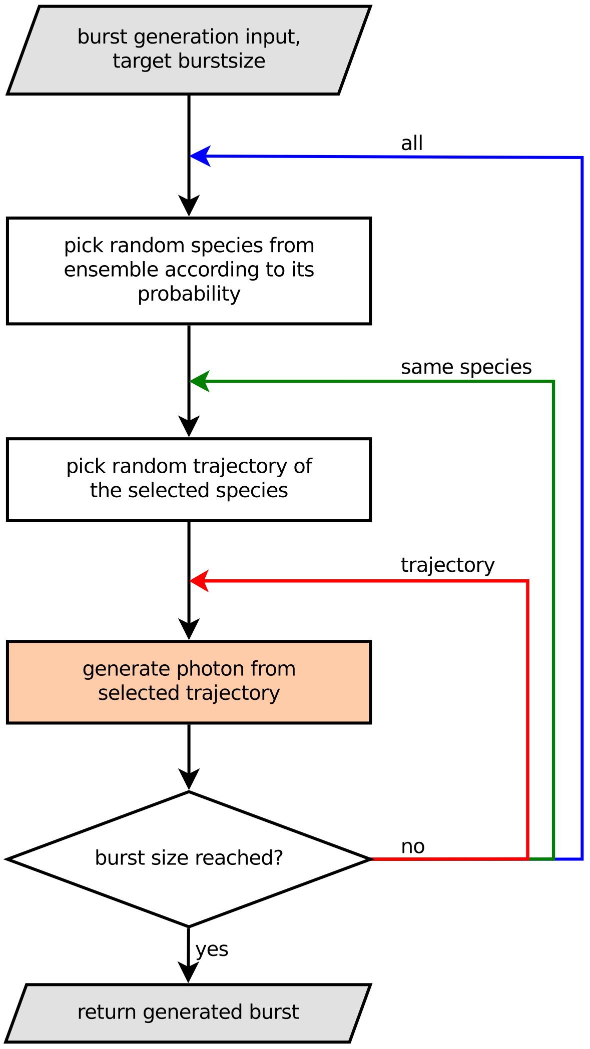
burst generation input, target burstsize
all
pick random species from ensemble according to its probability
same species
pick random trajectory of the selected species
trajectory
generate photon from selected trajectory
burst size reached?
no
yes
return generated burst

Fig. 12. Flowchart of a single burst calculation. Three steps are involved: 1st, picking a random species. 2nd, picking a random trajectory. 3rd, generating a photon from the trajectory. Depending on the burst calculation method selected (see Fig. 11), steps one (blue) and two (green, blue) or just three (red, green, blue) are repeated. The actual photon generation from the trajectory (light red filled box) is shown in Fig. 13. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

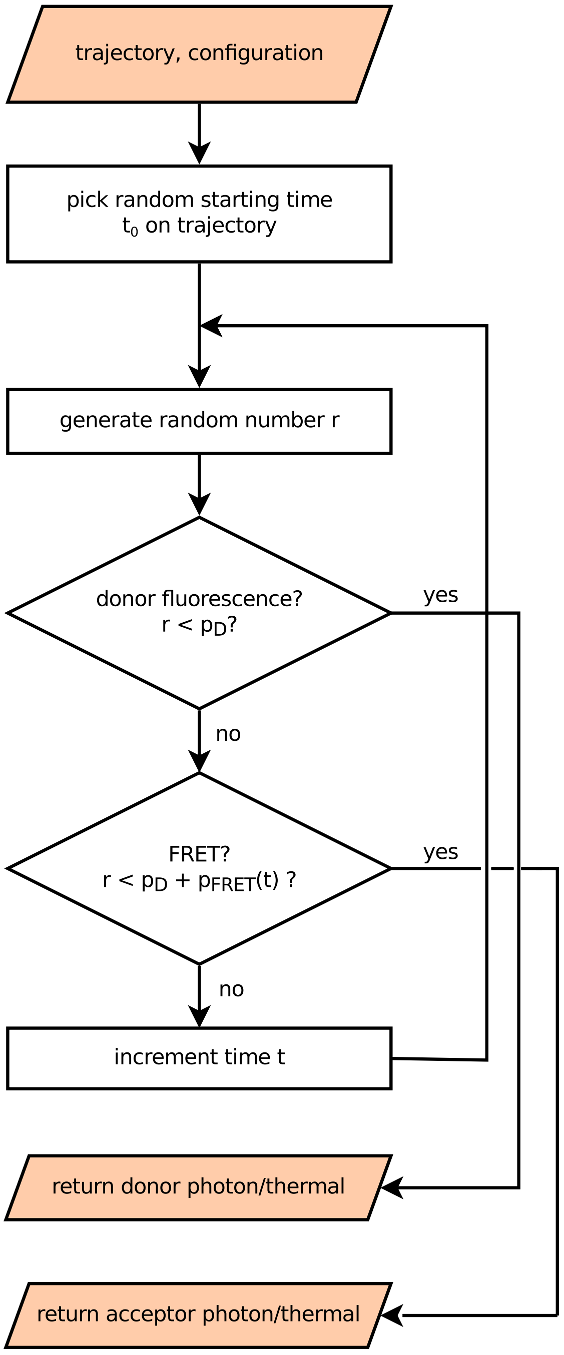
trajectory, configuration
pick random starting time t0 on trajectory
generate random number r
donor fluorescence? r < pD?
yes
no
FRET? r < pD + pFRET(t) ?
yes
no
increment time t
return donor photon/thermal
return acceptor photon/thermal

Fig. 13. Photon generation flowchart. First a random starting time t_0 is picked. Then a random number r is generated to discriminate between donor fluorescence and thermal de-excitation with probability p_D versus FRET to the acceptor dye with probability $p_{FRET}(t)$. If none of both events occurred, the time t is incremented and the process is repeated. The probabilities are normalized such that the sum of p_D, p_{FRET} and the probability for no event occurring is 1, and the random number r is drawn from the interval $0 < r < 1$.

is that the absolute FRET probability (Eqs. 8–10 in Ref. [1]) does change with time and thus with each increment while the absolute donor fluorescence probability stays constant.

5. Concluding remarks

In summary, the toolkit presented here enables accurate simulation of smFRET experiments from MD trajectories. The smFRET simulation method tested on poly-prolines successfully detected the heterogeneity of FRET efficiencies and in addition assigned structures from the simulation ensemble to this heterogeneity.

Furthermore, such simulations provide a systematic study of how the common model assumptions affect the accuracy of reconstructed distances [1]. The toolkit also allows analysis of burst properties, FRET efficiencies and their relationship to experimental setup parameters.

Future directions will include an improved treatment of the dye coupling beyond dipole–dipole coupling, i.e. multipole coupling and direct methods [15,16]. Also, the experimentally observed quenching effects of specific dye conformations [17] and due to the environment [18] can be modeled and included in the framework, e.g. as time dependent quantum yields in the future. The flexibility of the toolkit further enables an easy adaption to related experimental FRET techniques.

Finally, the program is distributed under an open-source license and the clean Python implementation allows adaption and extension of the toolkit to custom problems.

Acknowledgments

We thank Petra Kellers and Christian Blau for carefully reading the manuscript, and Stephanus Fengler, Timo Graen, as well as Andrea Vaiana for helpful discussions.

This work has been supported by the German Israeli Foundation grant (Agreement No. 1000.89.9/2008) and the SFB 755 (Nanoscale Photonic Imaging). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Appendix. FRET burst generator input and output overview

```
Usage: md2fret.py [options]

Options:
  -h, --help            show this help message and exit

  Required input options:
    -d RKDIR, --directory=RKDIR
                        directory with R-Kappa trajectories
    -c fret.conf, --configfile=fret.conf
                        configuration filename
    -p probabilities.dat, --probabilites=probabilities.dat
                        definition and probability of trajectory classes

  Optional input:
    -k exp.dat, --expbursts=exp.dat
                        experimental bursts size distribution file
    -s python_default, --seed=python_default
                        random number generator seed
    -r TRAJFORMAT, --trajformat=TRAJFORMAT
                        trajectory format: npz (numpy), dat (plaintext)

  Output options:
    -z binary-output.dump, --binary-output=binary-output.dump
                        binary output file
    -e efficiencies.dat, --output-efficiencies=efficiencies.dat
                        efficiency output file
    -l burstsizes.dat, --output-burstlenghts=burstsizes.dat
                        burstsize output file
    -b burstcomps.dat, --output-burstcomp=burstcomps.dat
                        burstcomposition output file
    -f endtimes.dat, --output-endtimes=endtimes.dat
                        endtime output file
    -t decaytimes.dat, --output-decaytimes=decaytimes.dat
                        decaytime output file
```

References

[1] M. Hoefling, N. Lima, D. Hänni, B. Schuler, C.A.M. Seidel, H. Grubmüller, Structural heterogeneity and quantitative FRET efficiency distributions of polyprolines through a hybrid atomistic simulation and Monte Carlo approach, PLoS One 6 (5) (2011) e19791.

[2] T. Förster, Zwischenmolekulare energiewanderung und fluoreszens, Ann. Phys. 2 (1–2) (1948) 55–75.

[3] R.E. Dale, J. Eisinger, W.E. Blumberg, The orientational freedom of molecular probes. The orientation factor in intramolecular energy transfer, Biophys. J. 26 (2) (1979) 161–193. http://dx.doi.org/10.1016/S0006-3495(79)85243-1.

[4] D.B. VanBeek, M.C. Zwier, J.M. Shorb, B.P. Krueger, Fretting about FRET: correlation between kappa and *R*, Biophys. J. 92 (12) (2007) 4168–4178.

[5] L. Stryer, R.P. Haugland, Energy transfer: a spectroscopic ruler, Proc. Natl. Acad. Sci. USA 58 (2) (1967) 719–726.

[6] B. Schuler, E.A. Lipman, P.J. Steinbach, M. Kumke, W.A. Eaton, Polyproline and the "spectroscopic ruler" revisited with single-molecule fluorescence, Proc. Natl. Acad. Sci. USA 102 (8) (2005) 2754–2759.

[7] R.B. Best, K.A. Merchant, I.V. Gopich, B. Schuler, A. Bax, W.A. Eaton, Effect of flexibility and cis residues in single-molecule FRET studies of polyproline, Proc. Natl. Acad. Sci. USA 104 (48) (2007) 18964–18969.

[8] R. Pool, A. Feenstra, M. Hoefling, R. Schulz, J.C. Smith, J. Heringa, Enabling grand-canonical Monte Carlo: extending the flexibility of GROMACS through the GromPy python interface module, J. Chem. Theory Comput. 33 (2012) 1207–1214.

[9] D.V.D. Spoel, E. Lindahl, B. Hess, G. Groenhof, A.E. Mark, H.J.C. Berendsen, GROMACS: fast, flexible, and free, J. Comput. Chem. 26 (16) (2005) 1701–1718.

[10] B. Hess, C. Kutzner, D. van der Spoel, E. Lindahl, GROMACS 4: algorithms for highly efficient, load-balanced, and scalable molecular simulation, J. Chem. Theory Comput. 4 (3) (2008) 435–447.

[11] W. Humphrey, A. Dalke, K. Schulten, VMD—visual molecular dynamics, J. Mol. Graphics 14 (1996) 33–38.

[12] H.P. Langtangen, Python Scripting for Computational Science, in: Texts in Computational Science and Engineering, Springer, 2010.

[13] M. Saito, M. Matsumoto, SIMD-Oriented fast Mersenne twister: a 128-bit pseudorandom number generator, in: A. Keller, S. Heinrich, H. Niederreiter (Eds.), Monte Carlo and Quasi-Monte Carlo Methods 2006, Springer, Berlin, Heidelberg, 2008, pp. 607–622.

[14] B. Schuler, Application of single molecule förster resonance energy transfer to protein folding, Methods Mol. Biol. 350 (2007) 115–138.

[15] A. Munoz-Losa, C. Curutchet, B.P. Krueger, L.R. Hartsell, B. Mennucci, Fretting about FRET: failure of the ideal dipole approximation, Biophys. J. 96 (12) (2009) 4779–4788.

[16] B.P. Krueger, G.D. Scholes, G.R. Fleming, Calculation of couplings and energy-transfer pathways between the pigments of LH2 by the ab initio transition density cube method, J. Phys. Chem. B 102 (27) (1998) 5378–5386.

[17] A.I. Sulatskaya, A.A. Maskevich, I.M. Kuznetsova, V.N. Uversky, K.K. Turoverov, Fluorescence quantum yield of thioflavin *t* in rigid isotropic solution and incorporated into the amyloid fibrils, PLoS One 5 (10) (2010) e15385.

[18] G.P. Acuna, M. Bucher, I.H. Stein, C. Steinhauer, A. Kuzyk, P. Holzmeister, R. Schreiber, A. Moroz, F.D. Stefani, T. Liedl, F.C. Simmel, P. Tinnefeld, Distance dependence of single-fluorophore quenching by gold nanoparticles studied on dna origami, ACS Nano 6 (4) (2012) 3189–3195.