Manual for the FCS/FCCS simulator software (`diffusion4`)

Dr. Jan W. Krieger <jan@jkrieger.de>

November 9, 2015

Contents

Chapter 1

Availability and Compilation

1.1 Availability

This software is available on GitHub:

`https://github.com/jkriege2/FCSSimulator`

In addition this simulator is also a part (based on the same repository) of the FCS/FCCS data evaluation software QUICKFIT 3.0 (`https://github.com/jkriege2/QuickFit3` and `http://www.dkfz.de/Macromol/quickfit/`), which also offers an integrated editor-GUI for the simulator configuration scripts. Therefore if you don't want to compile DIFFUSION4 for yourself, or modify its source-code you will only have to install QUICKFIT 3.0 and can skip the following sections 1.2 & 1.3. You can even use QUICKFIT 3.0 to edit the DIFFUSION4 configuration files and run the simulation, so also sections 1.4 & 1.5 are only of minor interest! Also QUICKFIT 3.0 can be used to easily read the FCS/FCCS-correlation curves from DIFFUSION4 and evaluate them further. However, additional results might be created by DIFFUSION4. These are usually saved in standard ASCII- or CSV-files and can be viewed and evaluated with any standard data-evaluation software. But we propose to use the open-source tools described in section 1.6, as DIFFUSION4 is optimized to work together with them!

1.2 Download and Setup of DIFFUSION4 for COmpilation

DIFFUSION4 is a C++ program, so you will need a working C++ compiler (e.g. the GCC in version $\geq$4.7, for Windows you can use MINGW-BUILDS from `http://sourceforge.net/projects/mingwbuilds/`). The program only relies on the GNU scientific library (GSL), available from `http://www.gnu.org/software/gsl/`, but also integrated in the repository (see below). In addition you will need a working BASH-shell (on Windows use MSYS available from `http://www.mingw.org/wiki/msys`) and GIT to access the repository (if you don't want to download the code manually). In general this program should compile on WINDOWS, LINUX and MACOS X.

In order to install DIFFUSION4, follow these steps:

1. check out the git repository:

```
$ git clone --recursive "https://github.com/jkriege2/FCSSimulator.git"
```

2. ensure that all submodules are checked out:

```
$ cd FCSSimulator
$ git submodule update --init --remote --recursive --force
```

3. if GSL is not available on your system, we'll have to build it now. The repository contains a BASH-script for this purpose:

```
$ cd extlibs
$ . build_dependencies.sh
$ cd ..
```

This script will build a local version of GSL. It is not installed in the system, but only in the directory `./extlibs/gsl/`. It will ask several questions: Generally you should not keep the build-directories (`n`), use as many processor, as you like (e.g. `2` for a dual-core, or `4` for a quad-core machine), you the best optimizations for your local machine, if the programm will only be used locally (twice `y`, or for safe-settings `y` and then `n`). Finally answer `y` when asked whether GSL should be compiled.

1.3 Compilation

the rest of the compilation is done using a `Makefile` in the base directory:

```
$ make
```

1.4 Running the Software DIFFUSION4

After the compilation, an command-line based executable `diffusion4` is created in the base directory. This is the simulator, which can then be started as follows:

```
$ diffusion4 [--spectra SPECTRADIRECTORY] CONFIG_SCRIPT.ini
```

Here `CONFIG_SCRIPT.ini` is a configuration file that tells the simulator what to do and the optional `-spectra SPECTRADIRECTORY` can be used to provide another directory with absorption and emission spectra. A version of this directory is available in the repository under `./spectra/`, which is also loaded automatically, if the option `--spectra` is not given. DIFFUSION4 reads absorption and emission spectra for its simulation from this directory. Note that DIFFUSION4 will not work, if a `spectra`-directory is not provided!

1.5 Running DIFFUSION4 from a Makefile

Often it is useful to run DIFFUSION4 from a Makefile, if several simulations should be done (in parallel) on a computer. here is an example Makefile for this task (see the repository directory `./example_configs/` for further examples:

```
SCRIPTS= gyuri_isat_multigfp0.ini\
         gyuri_isat_multigfp1.ini

SHELL = sh

SCRIPTS_TARGET = $(subst .ini,.target,$(SCRIPTS))

TERMINAL_COMMAND=

ifeq ($(findstring Msys,$(OS)),Msys)
EXE_SUFFIX=.exe
TERMINAL_COMMAND=
else
EXE_SUFFIX=
#TERMINAL_COMMAND=konsole -e
```

```
#TERMINAL_COMMAND=x-terminal-emulator -e
endif

all: ${SCRIPTS_TARGET}

%.target: %.ini
	@echo -e "random wait before starting $< ..."
	@bash random_sleep.sh
	@bash -c "sleep $$[ ( $$RANDOM % 10 ) + 1 ]s"
	@echo -e "starting on $< ..."
	${TERMINAL_COMMAND} ./diffusion4${EXE_SUFFIX} $< > $<.log
	@echo -e "work on $< DONE!"
```

You can run such a Makefile with the command:

```
$ make -f Makefile -j4
```

where `4` specifies the number of processors to use. This makefile also uses the BASH-script `random_sleep.sh`, which waits a random number of seconds (up to 10 or 20) before running the simulation. This ensures that the random number generator of each simulation is initialized with a different seed (the seed is taken from the system time!).

1.6 Usefull Additional Tools

DIFFUSION4 will store its result in a bunch of standardized output-files. Many modules in DIFFUSION4 generate data-files in the comma-separated values (CSV) format, or in other standardized data-formats (e.g. TIFF for images, or ALV-5000 ASCII-files for correlation data). Therefore you can use any standard data-evaluation software to further process the simulation results. Nevertheless, DIFFUSION4 was optimized to work well with these open-source softwares:

- GNUPLOT 4.6 is a data-plotting and evaluation tool. It is available for most platforms from

  `http://gnuplot.info/.`

  Many modules in DIFFUSION4 generate GNUPLOT-files (`*.plt`) in addition to CSV output files. These GNUPLOT-files allow to immediately generate plots from the data, output by DIFFUSION4. Usually they are generated such that they first generate a PDF of the plots and then display them on the screen.

- GRAPHVIZ is used to plot graphs, that e.g. represent the structure of a DIFFUSION4-simulation. This software is available from

  `http://www.graphviz.org/.`

  DIFFUSION4 e.g. generates `.gv`-files that can be run through the GRAPHVIZ-module `dot` and then output a graph representing all generator- and detector-objects in the DIFFUSION4-simulation and their relation to each other.

Chapter 2

Introduction to FCS

Note: This section is in part copied (and adapted) from my PhD thesis [1]!

The software DIFFUSION4 has been developed to simulate fluorescence correlation spectroscopy (FCS) and fluorescence cross-correlation spectroscopy (FCCS) measurements, based on particle trajectories.

2.1 Introduction to fluorescence correlation spectroscopy

FCS was introduced in 1974 by Magde, Elson, and Webb in Refs. [2–4]. As shown in Fig. 2.1, fluorescence is excited and detected in a tiny subvolume (a few μm^3) of the sample containing only few particles $N(t)$ at any time. The measured fluorescence intensity $F(t)$ is proportional to this particle number. Due to the diffusive motion of the particles, $N(t)$ is permanently fluctuating around its mean value $\langle N \rangle$. This is represented as:

$$N(t) = \langle N \rangle + \delta N(t) \quad \Rightarrow \quad F(t) = \langle F \rangle + \delta F(t) \quad \text{with } \langle \delta N(t) \rangle = \langle \delta F(t) \rangle = 0. \tag{2.1}$$

Here $\delta N(t)$ and $\delta F(t)$ represent the fluctuations of the particle number and the fluorescence intensity. Depending on the speed of motion (diffusion coefficient), the particle number fluctuates "faster" or "slower". This can be quantified using an autocorrelation analysis. The normalized FCS autocorrelation function is defined as

$$g(\tau) = \frac{\langle \delta F(t) \cdot \delta F(t+\tau) \rangle}{\langle F \rangle^2} = \frac{\langle F(t) \cdot F(t+\tau) \rangle}{\langle F \rangle^2} - 1, \quad \tau > 0. \tag{2.2}$$

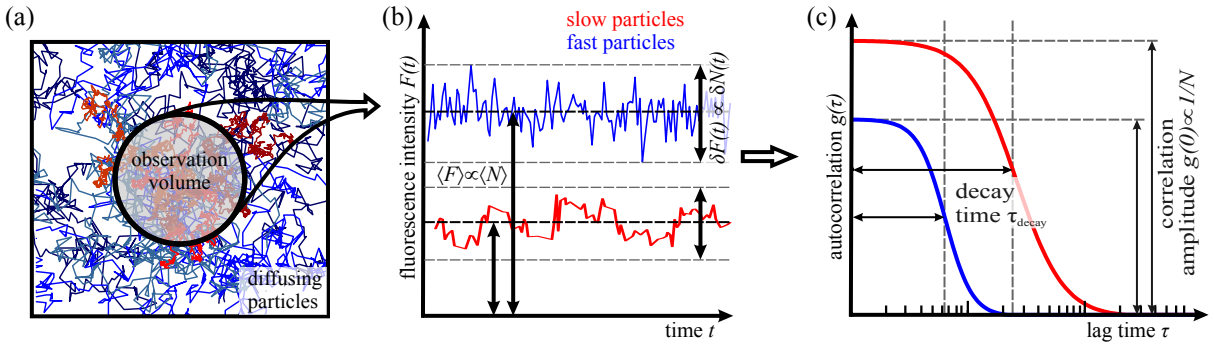

Figure 2.1: **Illustration of the principle of FCS.** (a) Sample with particles moving with different diffusion coefficients (fast: blue, slow: red). The circle indicates the observation volume. (b) Fluorescence intensity, as measured from the sample in (a), showing the fluorescence fluctuations $\delta F(t)$ around the mean intensity $\langle F \rangle$. (c) Autocorrelation functions $g(\tau)$, calculated from the intensity traces in (b). The decay times τ_{decay} are defined by $g(\tau_{\text{decay}}) = g(0)/2$.

where the averaging operation $\langle\cdot\rangle$ is defined as a a time average:

$$\langle F(t)\rangle = \lim_{T\to\infty} \frac{1}{T} \cdot \int_0^T F(t)\ \mathrm{d}t. \tag{2.3}$$

The autocorrelation function $g(\tau)$ measures the similarity of the signal $F(t)$ to its time-shifted version $F(t+\tau)$. If the fluctuations are completely random white noise, the correlation function is $g(\tau) \propto \boldsymbol{\delta}(\tau)$, which is 0 for all time lags $\tau > 0$. If $F(t)$ contains a non-random component, the correlation will be non-zero over a lag-time range, which is characteristic for the non-random process. In FCS the non-random fluctuations are caused by Brownian motion of fluorescent particles, which implies a typical dwell time of the particles in the observation volume of[1]:

$$\tau_\mathrm{D} \propto \frac{\left(\sqrt[3]{V_\mathrm{obs}}\right)^2}{D}. \tag{2.4}$$

During this time, the presence of a single particle causes a self-similarity in the fluctuations, which manifests itself as a decay of the autocorrelation function from a zero-lag amplitude $g(0) > 0$ to $g(\infty) = 0$. The half-life time τ_decay of this decay is approximately given by τ_D from Eq. (2.4).

The zero-lag amplitude $g(0)$ of the autocorrelation function Eq. (2.2) yields the average particle number in the observation volume, as

$$g(0) = \frac{\langle \delta F^2(t)\rangle}{\langle F\rangle^2} \propto \frac{\langle \delta N^2(t)\rangle}{\langle N\rangle^2} = \frac{1}{\langle N\rangle}. \tag{2.5}$$

In the last step, the Poissonian nature of the randomly fluctuating particle number $N(t)$ was used, which dictates that $\langle \delta^2 N(t)\rangle \equiv \mathrm{Var}(N(t)) = \langle N\rangle$.

Today FCS is typically implemented using confocal microscopes with a very small observation volume ($V_\mathrm{obs} \approx 0.2 - 0.6\,\mu\mathrm{m}^3$), and fluorescence is detected by single-photon avalanche diodes (SPADs or simply APDs). The acquired fluorescence intensity time-trace $F(t)$ is correlated using either specialized digital electronics or a software correlator. In a final step, analytical model functions are used to determine the average particle number $\langle N\rangle$, the diffusion coefficient D (via the decay time τ_decay) and other properties of the molecular motion, such as flow speeds, anomalous diffusion exponent, kinetic reaction rates.

2.2 Modeling fluorescence in a microscope

The theoretical framework of FCS starts from a simplified model of the fluorescence microscope, which is used to acquire the data. Figure 2.2 illustrates this model. It will be explained in detail throughout this section.

The sample of volume V_sample contains $N_{\mathrm{sample},\chi}$ particles of species χ that move randomly. The motion of each particle $i = 1...N_\mathrm{sample}(\chi)$ is described by its trajectory $\vec{r}_i(t)$. The trajectories are typically not known exactly, but their statistics, such as the mean squared displacement (MSD) , is known. Finally, the local particle concentration distribution of species χ can be written as

$$c_\chi(\vec{r},t) = \frac{1}{V_\mathrm{sample}} \cdot \sum_{i=1}^{N_{\mathrm{sample},\chi}} \boldsymbol{\delta}\left[\vec{r} - \vec{r}_i(t)\right]. \tag{2.6}$$

The particles are illuminated by some kind of illumination optics (e.g. a confocal microscope or a selective plane illumination microscope (SPIM)) with an intensity distribution $I_\gamma(\vec{r})$. The index $\gamma \in \{\mathrm{g},\mathrm{r},...\}$ denotes the color channel of the microscope, e.g. $\gamma = \mathrm{g}$ for excitation at $488\,\mathrm{nm}$ and detection in the range of $[500...550]\,\mathrm{nm}$ to observe eGFP, or $\gamma = \mathrm{r}$ for excitation at $568\,\mathrm{nm}$ and detection

[1] This equation is related to the mean-squared displacement (MSD) of a particle undergoing normal diffusion, which was shown by Albert Einstein to be $\langle r^2\rangle(\tau) = 2 \cdot d \cdot D \cdot \tau$, where d is the dimensionality of the motion and D is the diffusion coefficient.

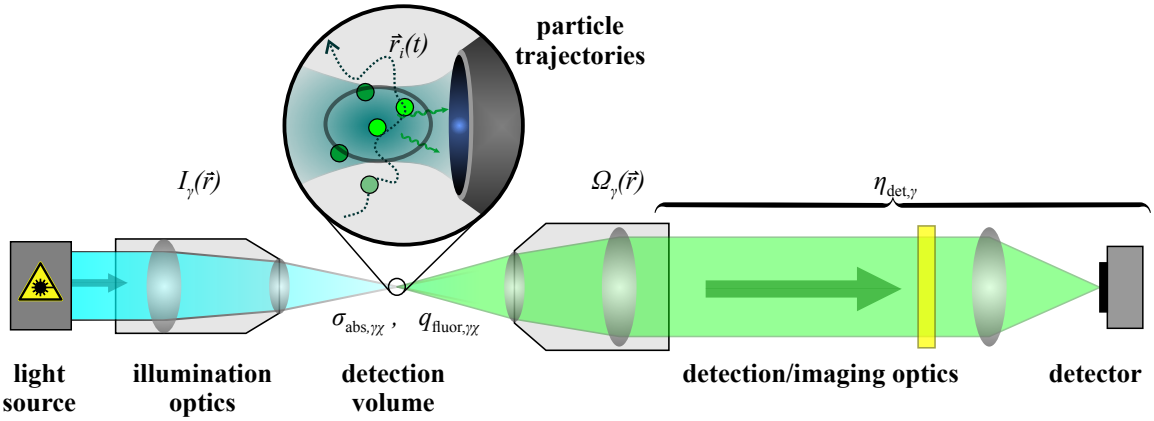

Figure 2.2: **Schematic of the optics model used for the derivation of FCS theory.** The illumination optics focuses light into an intensity distribution $I_\gamma(\vec{r})$. The detection optics is characterized by its detection probability distribution $\Omega_\gamma(\vec{r})$ and the detection efficiency $\eta_{\text{det},\gamma}$. The sample is modeled as a set of particles with trajectories $\vec{r}_i(t)$. To each species an absorption cross-section $\sigma_{\text{abs},\gamma,\chi}$ and a fluorescence quantum efficiency $q_{\text{fluor},\gamma,\chi}$ is assigned.

at $[600...700]\,\text{nm}$ for mRFP. An absorption cross-section $\sigma_{\text{abs},\gamma,\chi}$ and a fluorescence quantum yield $q_{\text{fluor},\gamma,\chi}$ is assigned to each species. Then the amount of fluorescence emitted by a single fluorophore of species χ at position $\vec{r}$ into channel γ can be written as

$$I_\gamma(\vec{r}) \cdot \sigma_{\text{abs},\gamma,\chi} \cdot q_{\text{fluor},\gamma,\chi}.$$

The detection optics is described by a detection efficiency distribution $\Omega_\gamma(\vec{r})$ and a detection efficiency $\eta_{\text{det},\gamma}$. The latter summarizes any signal loss due to optical surfaces or filters in the detection beam path. The distributions $I_\gamma(\vec{r})$ and $\Omega_\gamma(\vec{r})$ are not observable independently, so they are usually combined into a single function, called molecular detection efficiency (MDE) :

$$\text{MDE}_\gamma(\vec{r}) := I_\gamma(\vec{r}) \cdot \Omega_\gamma(\vec{r}). \tag{2.7}$$

This function is proportional to the rate of fluorescence photons expected from a fluorophore at position $\vec{r}$. Its actual form for a given microscopy setup can be calculated from the PSFs of the microscope. The geometry of the detectors (e.g. square pixels of a camera) also may need to be taken into account. For confocal setups a 3-dimensional, rotationally symmetric Gaussian function with width w_γ and height z_γ is a good approximation:

$$\text{MDE}_{\text{confocal},\gamma}(\vec{r}) = I_0 \cdot \exp\left(-2 \cdot \frac{x^2+y^2}{w_\gamma^2} - 2 \cdot \frac{z^2}{z_\gamma^2}\right). \tag{2.8}$$

In Refs. [1, 5, 6] it is argued, that a properly designed and aligned SPIM has a PSF with negligible sidelobe contributions. So the PSF can also be approximated by a Gaussian function. Still the finite size of the quadratic camera pixel has to be taken into account for the final form of the MDE:

$$\text{MDE}_{\text{SPIM},\gamma}(\vec{r}) = I_0 \cdot (h_{\text{pixel}} \circledast \text{PSF}_{\text{SPIM},\gamma})(\vec{r}) = \int_{-a/2}^{a/2}\!\!\int \text{PSF}_{\text{SPIM},\gamma}(\vec{r}-\vec{r}')\ \mathrm{d}x'\ \mathrm{d}y', \tag{2.9}$$

where a is the width of the pixel in the object plane, $\circledast$ denotes convolution and $h_{\text{pixel}}(\vec{r})$ is the characteristic function, describing a camera pixel:

$$h_{\text{pixel}}(\vec{r}) = \boldsymbol{\delta}(z) \cdot \begin{cases} 1 & -\frac{a}{2} \le x \le \frac{a}{2} \quad \wedge \quad -\frac{a}{2} \le y \le \frac{a}{2} \\ 0 & \text{else} \end{cases}. \tag{2.10}$$

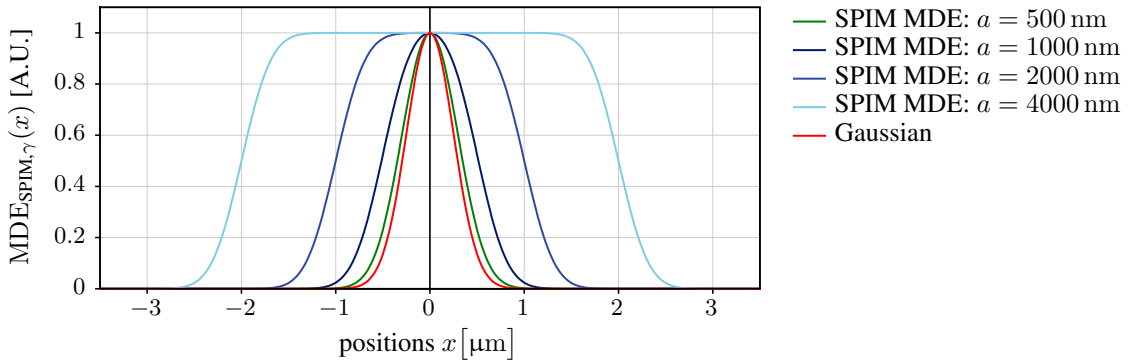

Figure 2.3: **Plots of cuts through the MDE of a SPIM in Eq.** (2.11) **along one coordinate axis.** For all plots, the PSF width was $w_\gamma = 500\,\mathrm{nm}$.

The convolution integral in Eq. (2.9) can be solved analytically:

$$\mathrm{MDE}_{\mathrm{SPIM},\gamma}(\vec{r}) = I_0 \cdot \frac{\left[\mathrm{erf}\left(\frac{a-2x}{\sqrt{2}\cdot w_\gamma}\right) + \mathrm{erf}\left(\frac{a+2x}{\sqrt{2}\cdot w_\gamma}\right)\right] \cdot \left[\mathrm{erf}\left(\frac{a-2y}{\sqrt{2}\cdot w_\gamma}\right) + \mathrm{erf}\left(\frac{a+2y}{\sqrt{2}\cdot w_\gamma}\right)\right]}{\left[2 \cdot \mathrm{erf}\left(\frac{a}{\sqrt{2}\cdot w_\gamma}\right)\right]^2} \cdot \exp\left(-2 \cdot \frac{z^2}{z_\gamma^2}\right) \tag{2.11}$$

As shown in Fig. 2.3, this MDE deviates significantly from a Gaussian function, if a is significantly larger than the size w_γ of the PSF.

Finally, the results of this section can be combined into the fluorescence time trace expected from a fluorophore concentration $c_\chi(\vec{r}, t)$ (see Eq. (2.6)):

$$F_\gamma(t) = \iiint_{-\infty}^{\infty} \mathrm{MDE}_\gamma(\vec{r}) \cdot \sum_{\chi \in \mathbb{S}} \eta_{\mathrm{det},\gamma} \cdot \sigma_{\mathrm{abs},\gamma,\chi} \cdot q_{\mathrm{fluor},\gamma,\chi} \cdot c_\chi(\vec{r}, t)\ \mathrm{d}V. \tag{2.12}$$

The factors $\eta_{\mathrm{det},\gamma}$, $\sigma_{\mathrm{abs},\gamma,\chi}$ and $q_{\mathrm{fluor},\gamma,\chi}$ are not distinguishable in a FCS experiment, so they are summarized into a single detection efficiency $\eta_{\gamma,\chi}$ of a fluorophore of species $\chi \in \mathbb{S}$ in channel γ:

$$\eta_{\gamma,\chi} \equiv \eta_{\mathrm{det},\gamma} \cdot \sigma_{\mathrm{abs},\gamma,\chi} \cdot q_{\mathrm{fluor},\gamma,\chi} \tag{2.13}$$

Then Eq. (2.12) can be simplified to

$$F_\gamma(t) = \iiint_{-\infty}^{\infty} \mathrm{MDE}_\gamma(\vec{r}) \cdot \sum_{\chi \in \mathbb{S}} \eta_{\gamma,\chi} \cdot c_\chi(\vec{r}, t)\ \mathrm{d}V. \tag{2.14}$$

As shown in Eq. (2.2), the autocorrelation function is written in terms of signal fluctuations $\delta F_\gamma(t)$ around a mean intensity $\langle F_\gamma \rangle$. These can be expressed in a form analogous to Eq. (2.14), if the concentration dynamics $c_\chi(\vec{r}, t)$ is also split into a fluctuation part and an average concentration:

$$c_\chi(\vec{r}, t) = \langle c_\chi \rangle + \delta c_\chi(\vec{r}, t). \tag{2.15}$$

Due to the linearity of Eq. (2.14), this finally yields:

$$\delta F_\gamma(t) = \iiint_{-\infty}^{\infty} \mathrm{MDE}_\gamma(\vec{r}) \cdot \sum_{\chi \in \mathbb{S}} \eta_{\gamma,\chi} \cdot \delta c_\chi(\vec{r}, t)\ \mathrm{d}V. \tag{2.16}$$

2.3 Theory of fluorescence correlation spectroscopy

2.3.1 The FCS autocorrelation function

For a color channel γ, the FCS autocorrelation function is defined as

$$g_\gamma(\tau) = \frac{\langle \delta F_\gamma(t) \cdot \delta F_\gamma(t+\tau) \rangle}{\langle F_\gamma \rangle^2}. \tag{2.2}$$

Using the results in Eqs. (2.14, 2.16) this can be rewritten as:

$$g_\gamma(\tau) = \frac{\sum\limits_{\chi\in\mathbb{S}} \eta_{\gamma,\chi}^2 \iiint\limits_{-\infty}^{\infty} \iiint\limits_{-\infty}^{\infty} \mathrm{MDE}_\gamma(\vec{r}) \cdot \mathrm{MDE}_\gamma(\vec{r}') \cdot \langle \delta c_\chi(\vec{r},t) \cdot \delta c_\chi(\vec{r}', t+\tau) \rangle \ \ \mathrm{d}V \ \mathrm{d}V'}{\left(\sum\limits_{\chi\in\mathbb{S}} \eta_{\gamma,\chi} \cdot \iiint\limits_{-\infty}^{\infty} \mathrm{MDE}_\gamma(\vec{r}) \cdot \langle c_\chi(\vec{r},t) \rangle \ \ \mathrm{d}V \right)^2}. \tag{2.17}$$

Here, the linearity of the integration and averaging $\langle \cdot \rangle$ was used. Furthermore, it was assumed that the concentration fluctuations from two different molecular species χ and χ' are statistically independent, i.e. $\langle \delta c_\chi(\vec{r},t) \cdot \delta c_{\chi'}(\vec{r},t) \rangle = 0$.

2.3.2 Zero-lag correlation and particle numbers

The Poissonian nature of the particle number (or concentration) in the focus dictates for $\tau = 0$:

$$\langle \delta c_\chi(\vec{r},t) \cdot \delta c_\chi(\vec{r}',t) \rangle \equiv \left\langle \delta c_\chi^2(\vec{r},t) \right\rangle \cdot \boldsymbol{\delta}(\vec{r}-\vec{r}') = \langle c(\vec{r},t) \rangle \cdot \boldsymbol{\delta}(\vec{r}-\vec{r}'), \tag{2.18}$$

where the factor $\boldsymbol{\delta}(\vec{r}-\vec{r}')$ signifies, that single particles are non-interacting and therefore, also statistically independent. Therefore they are only correlated to themselves and not to other particles nearby. Using Eq. (2.18) and further assuming that the concentration does not change significantly over the observation volume (described by the MDE), i.e. $\langle c_\chi(\vec{r}) \rangle \equiv \langle c_\chi \rangle$, the zero-lag autocorrelation amplitude becomes:

$$\begin{aligned} g_\gamma(0) &= \frac{\sum\limits_{\chi\in\mathbb{S}} \eta_{\gamma,\chi}^2 \cdot \langle c_\chi \rangle \cdot \iiint\limits_{-\infty}^{\infty} \mathrm{MDE}_\gamma^2(\vec{r}) \ \ \mathrm{d}V}{\left(\sum\limits_{\chi\in\mathbb{S}} \eta_{\gamma,\chi} \cdot \langle c_\chi \rangle \cdot \iiint\limits_{-\infty}^{\infty} \mathrm{MDE}_\gamma(\vec{r}) \ \ \mathrm{d}V \right)^2} = \\ &= \frac{\sum\limits_{\chi\in\mathbb{S}} \eta_{\gamma,\chi}^2 \cdot \langle c_\chi \rangle}{\left(\sum\limits_{\chi\in\mathbb{S}} \eta_{\gamma,\chi} \cdot \langle c_\chi \rangle \right)^2} \cdot \frac{\iiint\limits_{-\infty}^{\infty} \mathrm{MDE}_\gamma^2(\vec{r}) \ \ \mathrm{d}V}{\left(\iiint\limits_{-\infty}^{\infty} \mathrm{MDE}_\gamma(\vec{r}) \ \ \mathrm{d}V \right)^2} \underset{\mathbb{S}=\{\chi\}}{=} \frac{1}{\langle c_\chi \rangle} \cdot \frac{\iiint\limits_{-\infty}^{\infty} \mathrm{MDE}_\gamma^2(\vec{r}) \ \ \mathrm{d}V}{\left(\iiint\limits_{-\infty}^{\infty} \mathrm{MDE}_\gamma(\vec{r}) \ \ \mathrm{d}V \right)^2}. \end{aligned} \tag{2.19}$$

In the last step, a single species χ was assumed. Introducing the effective volume

$$V_{\mathrm{eff},\gamma} := \frac{\left(\iiint\limits_{-\infty}^{\infty} \mathrm{MDE}_\gamma(\vec{r}) \ \mathrm{d}V \right)^2}{\iiint\limits_{-\infty}^{\infty} \mathrm{MDE}_\gamma^2(\vec{r}) \ \mathrm{d}V} \tag{2.20}$$

of the MDE, the zero-lag amplitude of the autocorrelation function can be written in terms of a particle number $\langle N_\chi \rangle = \langle c_\chi \rangle \cdot V_{\mathrm{eff},\gamma}$ within this volume:

$$g_\gamma(0) = \frac{\sum\limits_{\chi\in\mathbb{S}} \eta_{\gamma,\chi}^2 \cdot \langle c_\chi \rangle \cdot V_{\mathrm{eff},\gamma}}{\left(\sum\limits_{\chi\in\mathbb{S}} \eta_{\gamma,\chi} \cdot \langle c_\chi \rangle \cdot V_{\mathrm{eff},\gamma} \right)^2} = \frac{\sum\limits_{\chi\in\mathbb{S}} \eta_{\gamma,\chi}^2 \cdot \langle N_\chi \rangle}{\left(\sum\limits_{\chi\in\mathbb{S}} \eta_{\gamma,\chi} \cdot \langle N_\chi \rangle \right)^2} \underset{\mathbb{S}=\{\chi\}}{=} \frac{1}{\langle N_\chi \rangle}. \tag{2.21}$$

For confocal and light sheet microscopes, the MDEs were defined in section 2.2. The integrals in the effective volume in Eq. (2.20) can be calculated analytically for these specific MDEs:

$$\text{confocal:} \quad V_{\text{eff},\gamma} = \pi^{3/2} \cdot w_\gamma^2 \cdot z_\gamma, \tag{2.22}$$

$$\text{SPIM:} \quad V_{\text{eff},\gamma} = \frac{\sqrt{\pi} \cdot a^2 \cdot z_\gamma}{\left[\operatorname{erf}\left(\frac{a}{w_\gamma}\right) + \frac{w_\gamma}{\sqrt{\pi}\cdot a}\left(\mathrm{e}^{-a^2/w_\gamma^2} - 1\right)\right]^2}. \tag{2.23}$$

2.3.3 The concentration correlation factor

In the autocorrelation function Eq. (2.17) the particle dynamics is fully described by the concentration correlation factor

$$\left\langle \delta c_\chi(\vec{r}, t) \cdot \delta c_\chi(\vec{r}', t+\tau) \right\rangle =: \phi_\chi(\vec{r}, \vec{r}', \tau) \equiv \phi_\chi(\vec{r} - \vec{r}', \tau). \tag{2.24}$$

It quantifies the amount of correlation at a time-lag τ between the concentration fluctuations at two positions $\vec{r}$ and $\vec{r}'$. The last equivalence in Eq. (2.24) states that $\phi_\chi(\cdot,\cdot)$ only depends on the difference in positions, i.e. the whole system is shift-invariant in space. If the system of interest is furthermore isotropic, the self correlation function only depends on the length $\|\vec{r} - \vec{r}'\|$, i.e. $\phi_\chi(\vec{r}, \vec{r}', \tau) \equiv \phi_\chi(\|\vec{r} - \vec{r}'\|, \tau)$. Note that generally these assumptions are not necessarily true, but on the small scales of FCS measurements they usually are assumed to apply.

The concentration correlation factor is (up to prefactors) equivalent to the van-Hove self correlation function of the particles [7–9], which is given in terms of single-particle trajectories $i = 1, 2, ..., N$ as [8]:

$$P_\chi(\vec{r}, \vec{r}', \tau) = \frac{1}{N} \cdot \left\langle \sum_{i=1}^{N} \boldsymbol{\delta}(\vec{r}' - \vec{r} + \vec{r}_i(0) - \vec{r} - (\vec{r}_i(\tau) - \vec{r}) \right\rangle. \tag{2.25}$$

Here, $P_\chi(\cdot,\cdot,\cdot)$ can be interpreted as the probability to find a particle at position $\vec{r}'$ at time τ, if it was initially at position $\vec{r}$. Specific forms of $P_\chi(\cdot,\cdot)$ can be calculated as the Green's function or propagator of the photon detection efficiency (PDE) , which governs the dynamics of $c_\chi(\vec{r}, t)$. A simple example for the PDE is the diffusion equation in

$$\frac{\partial c(\vec{r}, t)}{\partial t} = D \cdot \vec{\nabla}^2 c(\vec{r}, t), \tag{2.26}$$

where D is the diffusion coefficient. The Green's function is defined as the solution of the PDE for the initial condition $c_\chi(\vec{r}, 0) = \boldsymbol{\delta}(\vec{r})$ [10]. It can be used to calculate the solution $c_\chi(\vec{r}, \tau)$ of the PDE for an arbitrary initial condition $c_\chi(\vec{r}, 0)$ at any time $\tau > 0$:

$$c_\chi(\vec{r}, \tau) = c_\chi(\vec{r}, 0) \circledast P_\chi(\vec{r}, \tau) = \int \cdots \int c_\chi(\vec{r}', 0) \cdot P_\chi(\vec{r}, \vec{r}', \tau)\ \mathrm{d}^d r'. \tag{2.27}$$

Here $\circledast$ denotes a convolution and d is the dimension of the space, in which the motion takes place. Finally, the concentration correlation factor is given by

$$\phi_\chi(\vec{r}, \vec{r}', \tau) = \left\langle \delta c_\chi(\vec{r}, t) \cdot \delta c_\chi(\vec{r}', t+\tau) \right\rangle = \left\langle c_\chi \right\rangle \cdot P_\chi(\vec{r}, \vec{r}', \tau). \tag{2.28}$$

With these definitions, Eq. (2.17) can be slightly rewritten:

$$g_\gamma(\tau) = \frac{\sum\limits_{\chi\in\mathbb{S}} \eta_{\gamma,\chi}^2 \langle c_\chi \rangle \iiint_{-\infty}^{\infty} \iiint_{-\infty}^{\infty} \mathrm{MDE}_\gamma(\vec{r}) \cdot \mathrm{MDE}_\gamma(\vec{r}') \cdot \phi_\chi(\vec{r}, \vec{r}', \tau)\ \mathrm{d}V\ \mathrm{d}V'}{\left(\sum\limits_{\chi\in\mathbb{S}} \eta_{\gamma,\chi} \langle c_\chi \rangle\right)^2 \cdot \left(\iiint_{-\infty}^{\infty} \mathrm{MDE}_\gamma(\vec{r})\ \mathrm{d}V\right)^2} =$$

$$= \frac{\sum\limits_{\chi\in\mathbb{S}} \eta_{\gamma,\chi}^2 G_\gamma^\chi(\tau)}{\left(\sum\limits_{\chi\in\mathbb{S}} \eta_{\gamma,\chi} \langle c_\chi \rangle\right)^2}. \tag{2.29}$$

In this form a non-normalized correlation function

$$G_\gamma^\chi(\tau) := \frac{\left\langle \delta F_\gamma^\chi(t) \cdot \delta F_\gamma^\chi(t+\tau) \right\rangle}{\eta_{\gamma,\chi}^2 \cdot \left(\iiint_{-\infty}^{\infty} \mathrm{MDE}_\gamma(\vec{r}) \ \mathrm{d}V \right)^2} = \\ = \langle c_\chi \rangle \cdot \frac{\iiint_{-\infty}^{\infty} \iiint_{-\infty}^{\infty} \mathrm{MDE}_\gamma(\vec{r}) \cdot \mathrm{MDE}_\gamma(\vec{r}') \cdot \phi_\chi(\vec{r}, \vec{r}', \tau) \ \mathrm{d}V \ \mathrm{d}V'}{\left(\iiint_{-\infty}^{\infty} \mathrm{MDE}_\gamma(\vec{r}) \ \mathrm{d}V \right)^2} \quad (2.30)$$

of the fluctuations $\delta F_\gamma^\chi(t)$ caused by a single species χ in a channel γ is introduced.

The next subsections will give the exact mathematical form of $\phi_\chi(\vec{r}, \vec{r}', \tau)$ and the FCS autocorrelation functions $G_\gamma^\chi(\tau)$ and $g(\tau)$ for different situations frequently encountered in FCS measurements.

2.3.4 Normal diffusion

The most common dynamics in FCS is normal diffusion. Here, the particle concentration dynamics $c_\chi(\vec{r}, t) = \langle c_\chi \rangle + \delta c_\chi(\vec{r}, t)$ is governed by the diffusion equation Eq. (2.26):

$$\frac{\partial \left(\langle c_\chi \rangle + \delta c_\chi(\vec{r}, t) \right)}{\partial t} = D_\chi \cdot \vec{\nabla}^2 \left(\langle c_\chi \rangle + \delta c_\chi(\vec{r}, t) \right) \quad \Rightarrow \quad \frac{\partial \delta c_\chi(\vec{r}, t)}{\partial t} = D_\chi \cdot \vec{\nabla}^2 \delta c_\chi(\vec{r}, t). \quad (2.31)$$

The Green's function of the this PDE Eq. (2.31) is

$$P_\chi(\vec{r}, \vec{r}', \tau) = \frac{1}{\left(4\pi D_\chi \tau\right)^{3/2}} \cdot \exp\left(-\frac{(\vec{r}' - \vec{r})^2}{4 D_\chi \tau} \right). \quad (2.32)$$

Note that the diffusion equation Eq. (2.26) (and likewise Eq. (2.31)) can be interpreted statistically and then also a single-particle mean squared displacement (MSD) can be calculated. For normal diffusion in d dimensions, Albert Einstein derived:

$$\mathrm{MSD}(\tau) \equiv \langle r^2 \rangle(\tau) = 2d \cdot D \cdot \tau. \quad (2.33)$$

With this result and the MDEs in Eqs. (2.22, 2.23), the FCS autocorrelation function for normal diffusion can be calculated. The MDEs as well as Eq. (2.32) can be separated into three factors that depend solely on a single direction x, y or z. Therefore, also the autocorrelation function separates into three directional components:

$$G_\gamma^\chi(\tau) = \langle c_\chi \rangle \cdot G_{\gamma,x}^\chi(\tau) \cdot G_{\gamma,y}^\chi(\tau) \cdot G_{\gamma,z}^\chi(\tau). \quad (2.34)$$

Each directional factor is defined by

$$G_{\gamma,x}^\chi(\tau) = \frac{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathrm{MDE}_{\gamma,x}(\xi) \cdot \mathrm{MDE}_{\gamma,x}(\xi') \cdot \phi_{\chi,x}(\xi, \xi', \tau) \mathrm{d}\xi \mathrm{d}\xi'}{\left(\int_{-\infty}^{\infty} \mathrm{MDE}_x(\xi) \mathrm{d}\xi \right)^2}. \quad (2.35)$$

Here the directional components of $\mathrm{MDE}_\gamma(\vec{r})$ and of $\phi_\chi(\vec{r}, \vec{r}', \tau)$ are denoted by an additional index x.

For a 3-dimensional Gaussian MDE (Eq. 2.8), the directional factor in Eq. (2.35) is given for the x and y direction by

$$G_{\gamma,x}^\chi(\tau) = \frac{1}{\sqrt{\pi} \cdot w_\gamma} \cdot \left(1 + \frac{4 D_\chi \tau}{w_\gamma^2} \right)^{-1/2}. \quad (2.36)$$

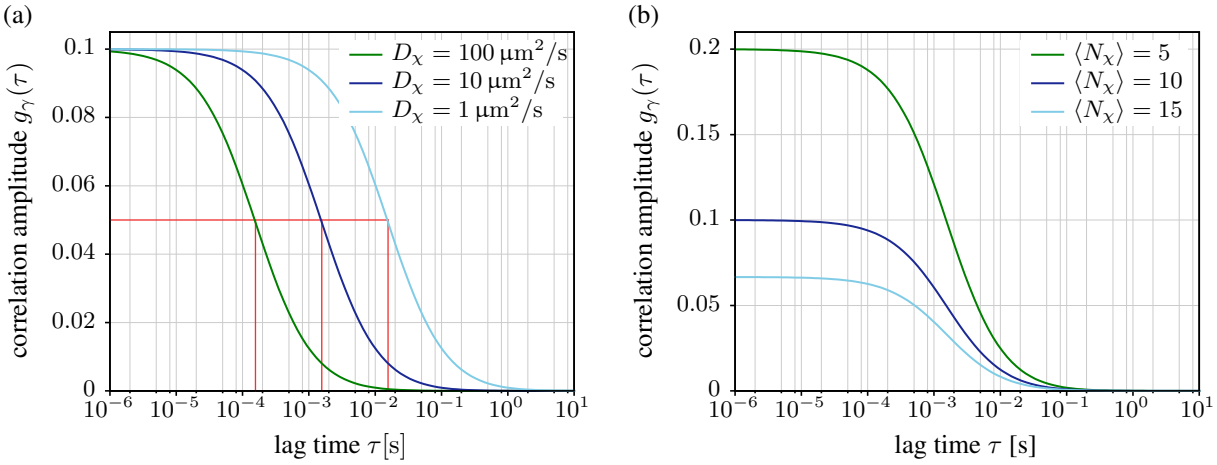

Figure 2.4: **Plots of the autocorrelation function for a confocal microscope and 3-dimensional normal diffusion, as defined in Eq.** (2.38)**.** In (a) the diffusion coefficient is varied, whereas in (b) the average particle number changes. In (a) the red lines indicate the diffusion correlation time $\tau_{D,\chi}$ as defined by Eq. (2.40) for each of the curves. Fixed parameters: $\langle N_\chi \rangle = 10$, $D_\chi = 100\,\mu m^2/s$; MDE parameters: $w_\gamma = 250\,nm$, $\kappa_\gamma = 6$.

The factor for the z direction is of the same form, but with w_γ replaced by z_γ. From this the non-normalized correlation function is easily calculated:

$$G_\gamma^\chi(\tau) = \frac{\langle c_\chi \rangle}{\pi^{3/2} w_\gamma^2 z_\gamma} \cdot \left(1 + \frac{4D_\chi \tau}{w_\gamma^2}\right)^{-1} \cdot \left(1 + \frac{4D_\chi \tau}{z_\gamma^2}\right)^{-1/2}. \tag{2.37}$$

The final FCS normalized correlation function is then given by:

$$g_\gamma(\tau) = \frac{1}{\langle c_\chi \rangle \cdot \pi^{3/2} w_\gamma^2 z_\gamma} \cdot \left(1 + \frac{4D_\chi \tau}{w_\gamma^2}\right)^{-1} \cdot \left(1 + \frac{4D_\chi \tau}{z_\gamma^2}\right)^{-1/2}, \tag{2.38}$$

which can be reformulated to

$$g_\gamma(\tau) = \frac{1}{\langle N_\chi \rangle} \cdot \left(1 + \frac{\tau}{\tau_{D,\chi}}\right)^{-1} \cdot \left(1 + \frac{\tau}{(z_\gamma/w_\gamma)^2 \cdot \tau_{D,\chi}}\right)^{-1/2}. \tag{2.39}$$

Here $\langle N_\chi \rangle$ is the average number of particles in a focal volume as given by Eq. (2.22) and $\tau_{D,\chi}$ is the diffusion correlation time with

$$\tau_{D,\chi} = \frac{w_\gamma^2}{4D_\chi}. \tag{2.40}$$

This $\tau_{D,\chi}$ is the average dwell time of particles in the observation volume and also the half decay time of $g_\gamma(\tau)$. Figure 2.4 illustrates the function $g_\gamma(\tau)$ for a single species χ for different the diffusion coefficients and the particle numbers. The red lines in Fig. 2.4(a) indicate $\tau_{D,\chi}$ for the different cases. They demonstrate that $g_\gamma(\tau_{D,\chi}) = g_\gamma(0)/2$. Figure 2.4(b) illustrates the general dependence $g_\gamma(0) = 1/\langle N_\chi \rangle$ of the zero-lag amplitude on the average particle number $\langle N_\chi \rangle$ in the focal volume.

Lower-dimensional diffusion: Using the separation of the confocal FCS autocorrelation function into directional components, also lower-dimensional diffusion models are easy to set up. In Eq. (2.35) one can easily omit the dimensions, along which no motion takes place. The concentrations $\langle c_\chi \rangle$ then have to be reinterpreted as an areal density, or a line density, depending on the diffusion model.

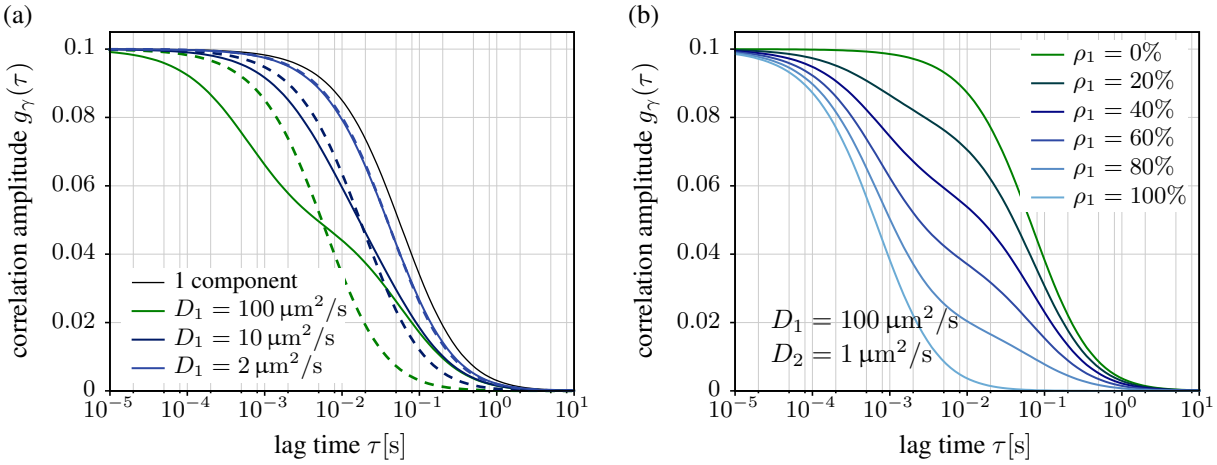

Figure 2.5: **Plots of the confocal FCS autocorrelation function for a two-component system $\chi = 1, 2$ with $\eta_{\gamma,1} = \eta_{\gamma,2}$, as defined by Eq.** (2.37)**.** In (a) one diffusion coefficient is fixed to $D_2 = 1\,\mu m^2/s$ and D_1 is varied, while $\langle N_1 \rangle = \langle N_2 \rangle = 5$. The black line is a one-component model with $\langle N_1 \rangle = 10$ and $D_1 = 1\,\mu m^2/s$. The dotted lines represent 1-component fits to the 2-component curves. In (b) the diffusion coefficients are fixed to $D_1 = 100\,\mu m^2/s$, $D_2 = 1\,\mu m^2/s$ in all curves, but the particle number fraction $\rho_1 := \langle N_1 \rangle / (\langle N_1 \rangle + \langle N_2 \rangle)$ is varied, keeping $\langle N_1 \rangle + \langle N_2 \rangle = 10$. MDE parameters: $w_\gamma = 500\,nm$, $z_\gamma = 1200\,nm$.

Multi-component diffusion: Figure 2.5 shows exemplary autocorrelation curves of a 2-component confocal FCS autocorrelation model. To simplify the situation, the molecular brightnesses of both species $\chi = 1, 2$ are set equal ($\eta_{\gamma,1} = \eta_{\gamma,2}$). Figure 2.5(a) shows a 2-component model with different combinations of diffusion coefficients (solid lines) and 1-component fits (dotted line) to these. The 1-component fit is hardly distinguishable from the 2-component data if the diffusion coefficients are too close to each other. If the assumption of equal brightnesses holds in a system, the multi-component diffusion model is typically written in terms of an overall concentration $\langle c_{all} \rangle$ and relative concentrations ρ_χ for each species, with:

$$\langle c_{all} \rangle := \sum_{\chi \in \mathbb{S}} \langle c_\chi \rangle, \qquad \rho_\chi := \frac{\langle c_\chi \rangle}{\langle c_{all} \rangle} \qquad \text{and} \qquad \sum_{\chi \in \mathbb{S}} \rho_\chi \stackrel{!}{=} 1. \tag{2.41}$$

With these definition, Eq. (2.29) can be simplified to:

$$g_\gamma(\tau) = \frac{1}{\langle c_{all} \rangle^2} \cdot \sum_{\chi \in \mathbb{S}} G_\gamma^\chi(\tau) \tag{2.42}$$

Chapter 3

Summary of FCS-Simulation in DIFFUSION4

3.1 Stepes of an FCS-Simulation

The software DIFFUSION4 was designed simulate FCS and FCCS correlation curves for different focus geometries and is closely related to the FCS/FCCS theory as presented in chapter 2 and Ref. [1] and especially the modeling of fluorescence detection described in section 2.2. It starts from a set of N particle trajectories $\{\vec{r}_i(t)\}$, where i numbers the particles and $t = 1, 2, ...$ numbers the equidistant timepoints with resolution Δt_{im}. The trajectories are either read from an external data file or are created internally by a configurable random walk. First $N_i \geq 1$ fluorophores are assigned to each particle, leading to an overall number of fluorophores

$$F = \sum_{i=1}^{N} N_i$$

fluorophores f, which are each characterized by the following set of properties (the functions $i(f)$ is the trajectory ID for every fluorophore f):

1. a position $\vec{r}_f(t) = \vec{r}_{i(f)}(t) + \Delta\vec{r}_f$, where $\vec{r}_{i(f)}(t)$ is the position of the particle and $\Delta\vec{r}_f$ is an arbitrary, but constant shift from this position. In the simplest case there is only one fluorophore per particle and $\Delta\vec{r}_f = 0$. Using $\Delta\vec{r}_f \neq 0$, moving finite-sized objects may be simulated that are e.g. labeled with a set of fluorophores on their surface or in their interior.

2. each fluorophore may be in one of S_f states. Each state may have different spectroscopic properties. The current state at time t is denoted by $s_f(t)$.

3. a wavelength-dependent absorption crosssection spectrum $\sigma_{\text{abs},i}(\lambda)$.

4. a normalized fluorescence spectrum $\eta_{\text{fl},f}(\lambda)$ and a fluorescence quantum yield $q_{\text{fluor},f,s_f(t)}$

5. a dipole orientation vector $\vec{p}_f(t)$ with $\|\vec{p}_f(t)\| = 1$.

The state trajectory $s_f(t)$ for each fluorophore either does not change (the default case), is read from an external file, or is simulated using a matrix of transition rates and a random decision in each simulation step. In this way photophysical blinking transitions may be simulated, if e.g. $s_f(t) \equiv 1$ is a bright state and $s_f(t) \equiv 2$ is a dark state with $q_{\text{fluor},f,2} = 0$. Also a simple bleaching process is implemented, by switching off (but never on again) a fluorophore with a certain low probability.

The simulation proceeds in steps of Δt_{sim}. For each time step and each focus in the simulation, first the expected number of fluorescence photons is calculated:

$$\overline{N}_{\text{phot}}(t) = \sum_{f=1}^{F} q_{\text{fluor},f,s_f(t)} \cdot \sigma_{\text{abs},i}(\lambda_{\text{ex}}) \cdot q_{\text{det}} \cdot \frac{\Delta t_{\text{sim}} \cdot I(\vec{r}_f(t))}{hc_0/\lambda_{\text{ex}}} \cdot \Omega(\vec{r}_f(t)) \cdot h_{\text{pol}}(\vec{p}_f(t)), \tag{3.1}$$

where h is Planck's constant, c_0 is the velocity of light in vacuum and λ_{ex} is the excitation wavelength.

The shape of the illumination profile is described by the function $I(\vec{r})$ and the respective shape of photon collection efficiency by $\Omega(\vec{r})$. Several models are implemented for them:

1. **Gaussian**: The shapes of $I(\vec{r})$ and $\Omega(\vec{r})$ are cigar-like Gaussian functions with equal x- and y-width w_0 and z-width z_0:
$$I(\vec{r}), \Omega(\vec{r}) \propto \exp\left(-2 \cdot \frac{x^2+y^2}{w_0^2} - 2 \cdot \frac{z^2}{z_0^2}\right) \tag{3.2}$$

2. **Gaussian beam**: The illumination/detection focus is described by a Gaussian beam, which has a lateral width $w(z)$ increasing with distance z from the focus and a Laurentzian shape in z-direction:
$$I(\vec{r}), \Omega(\vec{r}) \propto \left(\frac{w_0}{w(z)}\right)^2 \cdot \exp\left(-2 \cdot \frac{x^2+y^2}{w^2(z)}\right), \quad \text{with} \quad w(z) = w_0 \cdot \sqrt{1+\left(\frac{z}{z_0}\right)^2} \tag{3.3}$$

3. **Gaussian light sheet**: A simple model for a lightsheet is a Gaussian in z-direction, which does not depend on x or y:
$$I(\vec{r}) \propto \exp\left(-2 \cdot \frac{z^2}{z_0^2}\right) \tag{3.4}$$

4. **Slit pattern light sheet**: To model the sidelobes observed in typical light sheets a slit function can be used:
$$I(\vec{r}) \propto \left(\frac{\sin(\pi \cdot z/z_0)}{\pi \cdot z/z_0}\right)^2 \tag{3.5}$$

The first two patterns can be used for both, the illumination and detection foci, whereas the last two are designed to model the light sheet illumination.

The remaining influence of the detection process (signal loss at optical interfaces and filters, detector quantum efficiency, ...) is described by the factor

$$q_{\text{det}} = q_{\text{det},0} \cdot \frac{\int_{\lambda_{\text{det,min}}}^{\lambda_{\text{det,max}}} \eta_{\text{fl},f}(\lambda)\, \mathrm{d}\lambda}{\int_0^{\infty} \eta_{\text{fl},f}(\lambda)\, \mathrm{d}\lambda}, \tag{3.6}$$

summarizing the loss of light due to optics and detector quantum efficiency $q_{\text{det},0}$, as well as the spectral width of the fluorescence detection window $\lambda_{\text{det,min}}...\lambda_{\text{det,max}}$. This detection window allows to also take into account crosstalk between two detection channels. The influence of the dipole direction $\vec{p}_f(t)$ and a possible laser polarization is modeled by the factor

$$h_{\text{pol}}(\vec{p}_f(t)) = (1-\theta_{\text{pol}}) + \theta_{\text{Pol}} \cdot (\vec{\epsilon}_{\text{ex}} \bullet \vec{p}_f(t))^2, \tag{3.7}$$

where $\bullet$ is a scalar product, $\theta_{\text{Pol}} \in [0,1]$ is the fraction of linear polarization of the excitation light source and $\vec{\epsilon}_{\text{ex}}$ (with $\|\vec{\epsilon}_{\text{ex}}\| = 1$) is the linear polarization direction of this light source.

From the average number of detected photons, the measured number of photons $N_{\text{det}}(t)$ is calculated, taking the detector statistics into account. In the simplest case of a photon counting detector, $N_{\text{det}}(t)$ is drawn from a Poissonian distribution with mean (and variance) $\overline{N}_{\text{phot}}(t)$. Other detection statistics are possible, such as a linear detector, where $N_{\text{det}}(t)$ is drawn from a Gaussian distribution with mean $\overline{G} \cdot \overline{N}_{\text{phot}}(t)$ and a variance:

$$\sigma_{\text{det}}^2 = \overline{G}^2 \cdot \mathcal{F}^2 \cdot N_{\text{det}}(t) + \sigma_{\text{read}}^2, \tag{3.8}$$

where $\overline{G}$ is the average detector gain, $\mathcal{F}^2$ the excess noise factor and σ_{read}^2 the read noise variance, summarizing all contributions, not depending on the number of incident photons. Also artifacts, such as a background intensity offset may be included in the detector simulation. Although intermediate results may

be floating-point numbers, the finally detected number of photons (or ADU counts in a linear detector) is always an integer number.

Finally the time series $N_{\text{det}}(t)$ is post processed to yield count rate traces with arbitrary binning, auto- and cross-correlation functions (between different foci on the simulation) and other statistical properties. Also several test data sets are saved by the simulation program, such as particle mean squared displacements (MSDs), the raw detector statistics etc.

The complete program is split into modules that may be combined in different ways for a simulation. All these modules are either trajectory sources or sinks. In each time step first all sources generate a new set of fluorophore properties, e.g. by reading a new data set from a file or advancing a random walk simulation. The these new particle properties are forwarded to the sink objects, which simulate the actual detection process, as described above, or generate MSDs and other trajectory statistics. Every sink may be connected to several sources, and one source can feed several sinks. This can be used e.g. for simulations of fluorophore reservoir depletion, where a single trajectory source is fed into intermediate objects that simulate different bleaching rates on the same particle positions and finally detected by a set of identical sinks, which simulate FCS detection.

This software was developed during my PhD thesis. It has already been used to simulate different aspects of FCS/FCCS for several publications [6, 11–13].

3.2 Organization of DIFFUSION4

The software DIFFUSION4 has been designed to mainly follow the steps, presented in the last section, but in a very configurable way. The simulation is set up, as a combination of two classes of objects:

- **(fluorophore) dynamics objects** that simulate the motion of detectable particles. Each such object outputs a list of the positions and properties of all the particles it simulates. These properties may be generated by a dynamic simulation, or they can be read from external data-files (e.g. the output of another simulation software, as described in [11]). It is even possible to base the output of a trajectory generator object on the trajectories from another object, e.g. adding a fluorophore of a second color to a particle.

- **(fluorescence) detection objects** that accept the trajectories as input and generate some kind of output from them (e.g. FCS correlation curves from the fluorescence in a virtual microscope focus, MSD-curves, ...).

Of both classes, several different object types exist. They are listed and explained in more detail in chapter 4. You can even write your own new classes using C++ and use them in a simulation. This is explained in chapter 4. Then the simulation is set up by instantiating at least one trajectory generator and at least one detection object and connecting them in an appropriate way. The setup of a simuation is denoted in a configuration file (often also called INI-file). The next section 3.3 will introduce these concepts with a simple (and basic) example.

3.3 A Minimal Example

A very simple configuration for DIFFUSION4 could look as follows:

```
######################################################################
# simulation setup
######################################################################
simulation.duration=0.1      # duration of the simulation in seconds
simulation.timestep=1e-6     # simulation timestep in seconds
# prefix for all output files
#    tosystempathseparator() converts \ and / to your locally
#    valid filename separator
simulation.basename=tosystempathseparator("./diffusion4_test/")
```

```
#################################################################
# setup all brownian objects ("supergroup")
#################################################################
brownian.volume_shape=sphere              # spherical volume
brownian.sim_radius=3                     # with 3 micrometer radius
brownian.init_fluorophor=atto488          # fluorophore is al488


#################################################################
# setup all fcs objects ("supergroup")
#################################################################
# resolution of the "FCS-data acquisition"/minimum lag-time [seconds]
fcs.corr_taumin=simulation.timestep*2
fcs.corr_S=20               # number of blocks in correlator
                            # taumax ~ corr_taumin*8*2^S
fcs.P0=150                  # laser power in microWatt above objective
fcs.lambda_ex=488           # excitation laser wavelength in nanometers
fcs.expsf_r0=0.33           # 1/e^2 width of excitation volume [micrometers]
fcs.expsf_z0=6*0.33         # 1/e^2 width of excitation volume [micrometers]
fcs.detpsf_r0=fcs.expsf_r0 # detection volume size = excitation volume size
fcs.detpsf_z0=fcs.expsf_z0
fcs.q_det=1                 # detection efficiency
fcs.save_binning=true       # save binned countrate
fcs.save_binning_time=0.005# 5ms binning


#################################################################
# dynamics object "brownian1"
#################################################################
brownian1.diff_coeff=50     # diffusion coefficient [micron^2/sec]
brownian1.c_fluor=10        # concentration [nM]


#################################################################
# detection object "fcs1"
#################################################################
fcs1.sources=brownian1        # get trajectories from brownian1
fcs1.object_name=particlesA  # a custom name for the fcs1-object
```

As you can see, the configuration has a simple structure:

- comments are started with the character "#" and stop at the end of the line
- each line is an assignment to a property:

  `property_group.property_name =` *value*

- Each property belongs to a group of properties, such as:
  - `simulation`: basic simulation settings, such as the duration, or timestep
  - `fcs`: presets for all "fcs" detection-objects in the simulation
  - `brownian`: presets for all "brownian" dynamics-objects in the simulation
  - `fcs1`: properties of the concrete detection-object "fcs1" (to instantiate a detection object, you have to set at least one of its properties!)
  - `brownian1`: properties of the concrete dynamics-object "brownian1" (to instantiate a dynamics object, you have to set at least one of its properties!)

- as *value* you can either give numbers (“.” as decimal separator), simple strings (potentially in double-quotes), or a boolean value `true` or `false`.
- In addition it is possible to denote simple mathematical expressions as *value*. In these you can calculate with the three basic datatypes number, string and boolean. All properties that have already been defined in the file can be used. Here is an example from the file:

  ```
  fcs.corr_taumin = simulation.timestep*2
  ```

  This sets `fcs.corr_taumin` to 2-times the simulation timestep `simulation.timestep`. In these expression, you can use the standard C-functions, like `sin(), cos(), exp(), sqrt(), pow(), ...` and several special functions like `floattostr(), booltostr(), if()`. Strings can be concatenated with the operator `+`.

The configuration script, given above, will run a simulation of one species of particles (`brownian1`) with a diffusion coefficient of $50\,\mu\mathrm{m}^2/\mathrm{s}$ and at a concentration of $10\,\mathrm{nM}$. An Alexa-488-fluorophore (`al488`) is attached to each particle and they are excited by a laser at $488\,\mathrm{nm}$, which forms a virtual Gaussian focus with a $1/\,\mathrm{e}^2$-halfwidth of $w_{\mathrm{xy,ex}} = 0.33\,\mu\mathrm{m}$ and a $1/\,\mathrm{e}^2$-halflength of $w_{\mathrm{z,ex}} = 6{\cdot}0.33\,\mu\mathrm{m} \approx 2\,\mu\mathrm{m}$. A detection focus with the same position and sizes ($w_{\mathrm{xy,det}} = w_{\mathrm{xy,ex}}$,$w_{\mathrm{z,det}} = w_{\mathrm{z,ex}}$) is used to detect the photons. therefore the effective focus used in the FCS-equations will have a size of:

$$w_{\mathrm{xy}} = \sqrt{\frac{1}{1/w_{\mathrm{xy,det}}^2 + 1/w_{\mathrm{xy,ex}}^2}} \approx 0.23\,\mu\mathrm{m}, \qquad w_{\mathrm{z}} = \sqrt{\frac{1}{1/w_{\mathrm{z,det}}^2 + 1/w_{\mathrm{z,ex}}^2}} \approx 1.38\,\mu\mathrm{m}$$

The simulation runs for $0.1\,\mathrm{s}$ with a timestep of $\Delta t_{\mathrm{sim}} = 1\,\mu\mathrm{s}$. The minimum lag-time of the correlator is set to $\tau_{\mathrm{min}} = 2 \cdot \Delta t_{\mathrm{sim}}$. Finally, the correlator has $S = 20$ linear correlator blocks, which amounts to a maximum lag-time of approximately:

$$\tau_{\mathrm{max}} \approx m^S \cdot P/2 \cdot \tau_{\mathrm{min}} \approx 17\,\mathrm{s} \qquad [\text{default: } m = 2, P = 16]$$

The configuration of the simulation can be drawn as a simple diagram, as shown in Fig. 3.1 The resulting correlation curves will look approximately like the one shown in Fig. 3.2.

The simulation outputs its results into files that are composed of:

`simulation.basename` + *object-name* + ***description.extension***

In the file above, the `simulation.basename` will be `./diffusion4_test/`, which indicates a sub-directory “diffusion4_test” in the execution directory of DIFFUSION4. The *object-name* can either be “brownian1” for output by the dynamics object, or “particlesA” for output from the detection object fcs1 (it would be “fcs1”, if `fcs1.object_name` wouldn’t have been given). Note that the *object-name* is optional and there will also be some output files with only *description.extension*, e.g. the files `config.txt` and `log.txt`, which contain general log-output and properties of the simulation. There is also a file `config.gv`, which can be interpreted by GRAPHVIZ (see section 1.6) to draw the structure in Fig. 3.1. Finally the file `config.ini` contains an extended copy of the simulation configuration files.

The output-files from brownian1 and fcs1 contain different checks of the simulation software and also different types of additional outputs. The majow files from fcs1/particlesA are:

- particlesA**alvacf.asc**, particlesA**.qf3acorr**
  output correlation curves (together with countrate traces) either in the ALV-5000 ASCII-format, or in a generic ASCII-format that is understood by QUICKFIT 3.0
- particlesA**bts.dat**, particlesA**btsplot.plt**
  countrate-trace as CSV-file and a GNUPLOT-script to plot it
- particlesA**corr.dat**, particlesA**corrplot.plt**
  correlation curve as CSV-file and a GNUPLOT-script to plot it

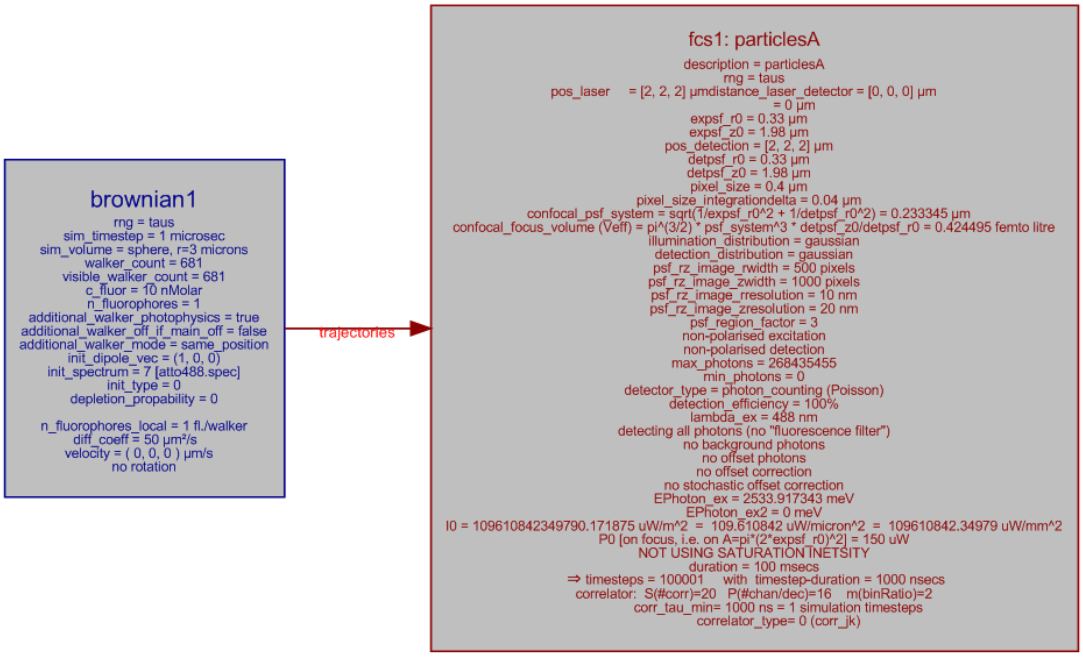

Figure 3.1: Project structure of the simple example

- `particlesA`**`detbackgroundtest.plt`**
  GNUPLOT-file with different checks of the background countrate in the detector of the FCS-simulation

- `particlesA`**`dettest.plt`**
  GNUPLOT-file with different checks of the detector statistics of the FCS-simulation

- `particlesA`**`psf.plt`**
  GNUPLOT-file with different characteristics of the excitation and detection focus of the FCS-simulation

3.4 A Simple 2-Component Diffusion Simulation

Starting from the simple example in the last section, we can extend it to show more features of DIFFUSION4. First we will do a 2-component diffusion simulation with two types of walkers with different diffusion coefficients. To do so, we will instantiate two instead of one `brownian` objects (the rest of the simple

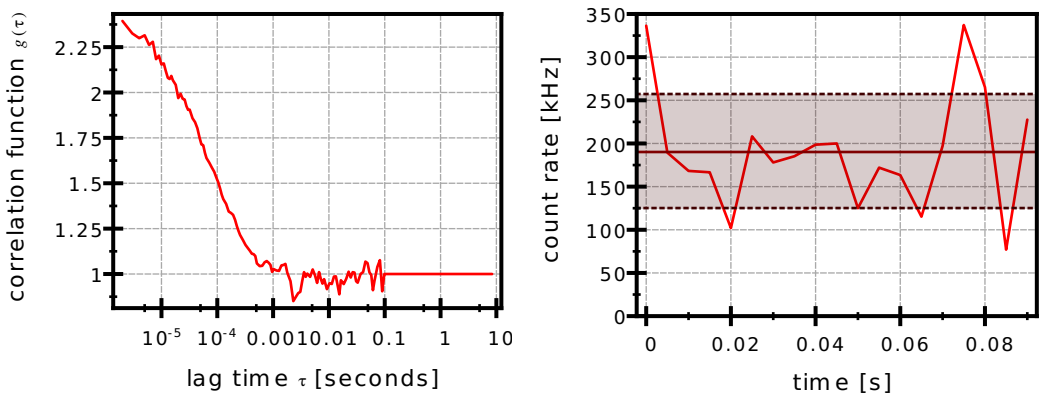

Figure 3.2: Correlation and countrate curves from the simple example script

script stays un-altered, except potentially `simulation.duration=0.5`):

```
#################################################################
# dynamics object "brownian1"
#################################################################
brownian1.diff_coeff=100    # diffusion coefficient [micron^2/sec]
brownian1.c_fluor=2         # concentration [nM]

#################################################################
# dynamics object "brownian2"
#################################################################
brownian2.diff_coeff=1      # diffusion coefficient [micron^2/sec]
brownian2.c_fluor=2         # concentration [nM]
```

Now we have two populations of (independently) diffusiong particles: fast ones ($D = 100\,\mu\mathrm{m}^2/\mathrm{s}$) in `brownian1` and slow ones ($D = 1\,\mu\mathrm{m}^2/\mathrm{s}$) in `brownian2`. For both of them, the same abundance of $c = 2\,\mathrm{nM}$ was set. For the detection we again use a single `fcs`-object, but now it takes the trajectories from two sources:

```
#################################################################
# detection object "fcs1"
#################################################################
fcs1.sources=brownian1,brownian2 # get trajectories from brownian1
                                 # and brownian2
```

This constructs a project with a structure as shown in Fig. 3.3. The results of such a simulation (also in comparison to a simple 1-component simulation with $D = 100\,\mu\mathrm{m}^2/\mathrm{s}, c = 4\,\mathrm{nM}$ is shown in Fig. 3.4.

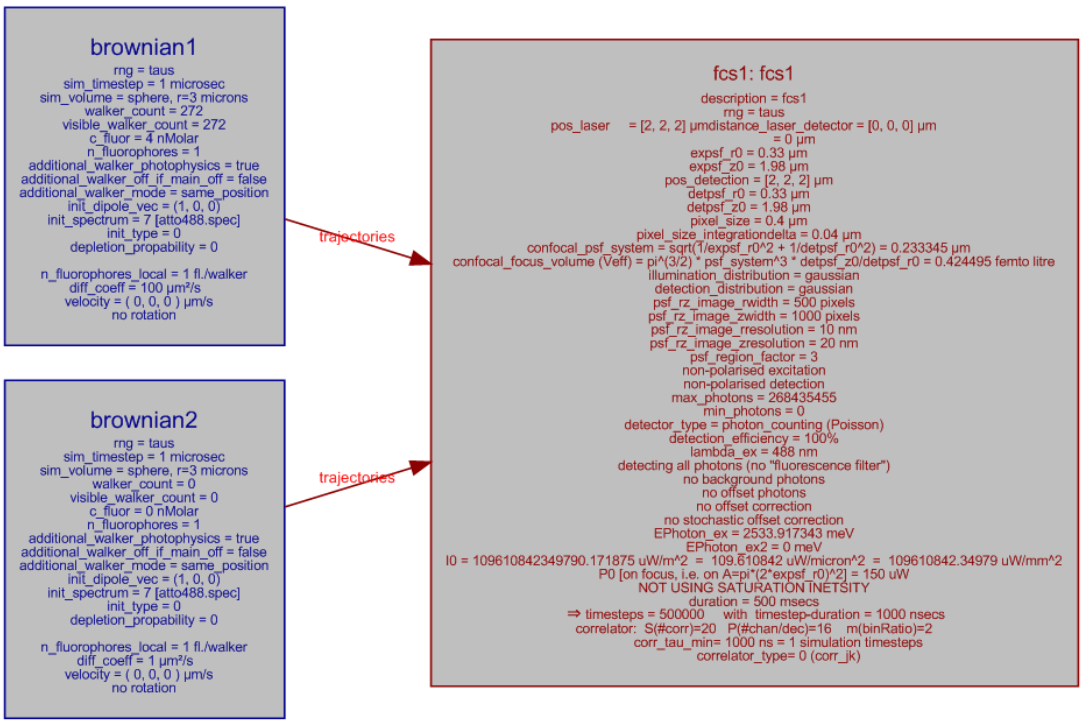

Figure 3.3: Structure of a 2-component diffusion configuration

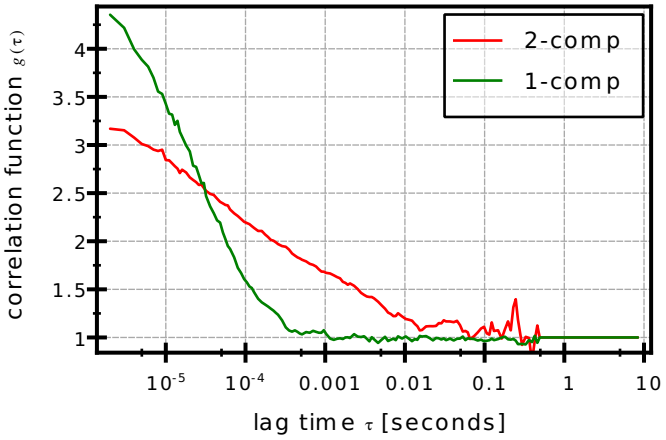

Figure 3.4: Results from a 2-component simulation

Chapter 4

Reference of the DIFFUSION4-Modules

4.1 Fluorophore Dynamics Modules

4.2 Fluorescence Detection Modules

Chapter 5

Extending DIFFUSION4

5.1 Introduction and Registration

You can extend DIFFUSION4 with your own modules by implementing classes of the type `FluorophorDynamics` for fluorophore dynamics modules, or `FluorescenceMeasurement` for fluorescence detection modules. these virtual base-classes each provide virtual functions that you have to implement in order to give you class a function. Finally you will have to register your new class in `main.cpp`:

- `FluorophorDynamics`-classes have to be added near the text

```
//////////////////////////////////////////////////
// add you custom dynamics classes here
//////////////////////////////////////////////////
```

  e.g.:

```
//////////////////////////////////////////////////
// add you custom dynamics classes here
//////////////////////////////////////////////////
} else if (lgname.find("mydyn")==0 && lgname.size()>5) {
        supergroup="mydyn";
        d=new MyDynamics(fluorophors, oname);
```

  Here you custom class is called `MyDynamics` and in the configration files it will have the prefix `mydyn`.

- `FluorescenceMeasurement`-classes have to be added near the text

```
//////////////////////////////////////////////////
// add you custom detection classes here
//////////////////////////////////////////////////
```

  e.g.:

```
//////////////////////////////////////////////////
// add you custom detection classes here
//////////////////////////////////////////////////
} else if (lgname.find("mydetection")==0 && lgname.size()>14) {
        supergroup="mydetection";
        m=new MyDetection(fluorophors, oname);
```

  Here you custom class is called `MyDetection` and in the configuration files it will have the prefix `mydetection`.

5.2 Implementing `FluorophorDynamics`-Classes

```
virtual void init();
```

This function is called before the simulation and is used to initialize the simulation object.

```
virtual void propagate(bool boundary_check=true);
```

This function implements the main functionality. It is called in every step and propagates the walkers that are stored in the array `walker_state`. If the parameter `boundary_check` is set `true`, the function body should perform a boundary-check at the borders of the sim-box. Otherwise the sim-box is assumed to be infinite (used for testing).

```
virtual std::string report();
```

This function reports the object-state in human-readable form.

```
virtual void read_config_internal(jkINIParser2& parser);
```

This function reads the object-configuration from the given parser, which is already cd'ed to the group to be read. This function is called twice for each object: Once for the super-group and once for the actual object-group.

Note that there are additional functions that can be used in special cases. See the implemented classes in the repository for details.

5.3 Implementing `FluorescenceMeasurement`-Classes

```
virtual void init();
```

This function is called before the simulation and is used to initialize the simulation object.

```
virtual void propagate();
```

This function implements the main functionality. It is called in every step and propagates the walkers that can be obtained from the dynamics-objects in the array `dyn`.

```
virtual std::string report();
```

This function reports the object-state in human-readable form.

```
virtual void save();
```

This function stores the simulation results.

```
virtual void read_config_internal(jkINIParser2& parser);
```

This function reads the object-configuration from the given parser, which is already cd'ed to the group to be read. This function is called twice for each object: Once for the super-group and once for the actual object-group.

Note that there are additional functions that can be used in special cases. See the implemented classes in the repository for details.

Index

Bibliography

[1] Jan W. Krieger. *Mapping Diffusion Properties in Living Cells*. doctoral thesis, Universität Heidelberg, 2014. URL `http://jkrieger.de/dissertation_final.pdf`.

[2] D Magde, E L Elson, and W W Webb. Fluorescence correlation spectroscopy i: Conceptual basis and theory. *Biopolymers*, 13(1):1–27, 1974.

[3] D Magde, E L Elson, and W W Webb. Fluorescence correlation spectroscopy. ii. an experimental realization. *Biopolymers*, 13(1):29–61, 1974.

[4] Douglas Magde, Watt W. Webb, and Elliot L. Elson. Fluorescence correlation spectroscopy. iii. uniform translation and laminar flow. *Biopolymers*, 17(2):361–376, 1978.

[5] Thorsten Wohland, Xianke Shi, Jagadish Sankaran, and Ernst H. K. Stelzer. Single plane illumination fluorescence correlation spectroscopy (SPIM-FCS) probes inhomogeneous three-dimensional environments. *Optics Express*, 10(8):10627–10641, 2010. doi:10.1364/OE.18.010627.

[6] Anand Pratap Singh, Jan Wolfgang Krieger, Jan Buchholz, Edoardo Charbon, Jörg Langowski, and Thorsten Wohland. The performance of 2D array detectors for light sheet based fluorescence correlation spectroscopy. *Opt. Express*, 21(7):8652–8668, 2013. doi:10.1364/OE.21.008652.

[7] Jean Pierre Hansen and Ian Ranald McDonald. *Theory of Simple Liquids*. Academic Press, 4 edition, 2013. ISBN 9780123870322. doi:10.1016/B978-0-12-387032-2.00015-5.

[8] Paul Hopkins, Andrea Fortini, Andrew J. Archer, and Matthias Schmidt. The van hove distribution function for brownian hard spheres: Dynamical test particle theory and computer simulations for bulk dynamics. *The Journal of Chemical Physics*, 133(22):224505, 2010. doi:10.1063/1.3511719.

[9] Felix Höfling and Thomas Franosch. Anomalous transport in the crowded world of biological cells. *Reports on Progress in Physics*, 76(4):046602, 2013. doi:10.1088/0034-4885/76/4/046602.

[10] M.L. Boas. *Mathematical methods in the physical sciences*. John Wiley & Sons New York, 1983.

[11] Tomasz Wocjan, Jan Krieger, Oleg Krichevsky, and Jörg Langowski. Dynamics of a fluorophore attached to superhelical DNA: FCS experiments simulated by brownian dynamics. *Physical Chemistry Chemical Physics*, 11(45):10671, 2009. doi:10.1039/B911857H.

[12] Jan Buchholz, Jan Wolfgang Krieger, Gábor Mocsár, Balázs Kreith, Edoardo Charbon, György Vámosi, Udo Kebschull, and Jörg Langowski. FPGA implementation of a 32x32 autocorrelator array for analysis of fast image series. *Optics Express*, 20(16):17767, 2012. doi:10.1364/OE.20.017767.

[13] Jan Wolfgang Krieger, Anand Pratap Singh, Christoph S. Garbe, Thorsten Wohland, and Jörg Langowski. Dual-Color fluorescence Cross-Correlation spectroscopy on a single plane illumination microscope (SPIM-FCCS). *Optics Express*, 22(3):2358, 2014. doi:10.1364/OE.22.002358.