FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA

Friedrich-Schiller-Universität Jena

Faculty of Mathmatics and Computer Science
Institute for Computer Science

Virtual Fluorescence Recovery after Photobleaching in Spatial Rule-based Modelling

for attainment of the academic degree of

Bachelor of Science

Author: Laura Ölsner
MatNr. 163528

Date and Location September 8, 2019, Jena

1. Supervisor: Prof. Dr. Peter Dittrich
2. Supervisor: Prof. Dr. Bashar Ibrahim

Abstract

In the field of systems biology the virtual simulation of biological processes on the computer has become an important part of scientific research. Recently, rule-based modeling has become more significant, especially the field of spatial simulation. But there is the ongoing problem, how to calibrate these models to gain quantitative data.

The objective of this thesis is the simulation of *fluorescense recovery after photobleaching* (FRAP) measurements and the simulation of the effect of certain parameters involved in this process, such as the radius of the bleaching area, the mass of atoms, binding rate, and dissociation rate. The simulations were performed with the help of the tool SRSim, which was extended with two self-written commands.

The results – in form of so called FRAP curves – have presented a realistic virtual FRAP. They have shown that the use of bigger radii has a negative effect on the expected upper bound of the final fluorescence intensity, and the use of higher masses have a distinct influence on the immobile fraction. But a contradiction between the results of the radii and the masses has occured that needs further investigation. The dissociation rate has indicated an effect on the recovery time, whereas the binding rate seems to have no significant influence. But since only three different values were considered, further validations with smaller binding rates are necessary.

With these first insights, future research could continue to work on a suitable calibration of the simulation, to determine and further explore the underlying biological system based solely on a FRAP curve.

Zusammenfassung

Die Simulation von biologischen Systemen ist in den letzten Jahren immer weiter in den Fokus der Forschung gerückt. Besonders die räumliche Simulation im Bereich der regel-basierte Modellierung gewinnt immer mehr an Bedeutung. Dabei gibt es aber noch viele offene Probleme, wie z.B. die richtige Anpassung der Parameter, um quantitative Daten zu erzeugen.
Das Ziel dieser Arbeit ist die Simulation von *Fluorescence Recovery after Photobleaching* (FRAP) Messungen und die Simulation des Einflusses bestimmter Parameter, die bei diesem Experiment eine Rolle spielen. Dafür wurden der Radius des Bleichbereichs, die Masse der Atome, die Bindungsrate sowie die Dissoziationsrate ausgewählt. Die Simulationen wurden mit Hilfe des Tools SRSim durchgeführt, welches mit zwei Methoden erweitert wurde.
Die Ergebnisse dieser Simulationen haben die Form einer sogenannten FRAP-Kurve. Diese zeigen, dass eine realistische Simulation eines solchen Experimentes möglich ist. Die Analyse der Parameter hat ergeben, dass der Radius einen Effekt auf die erwartete obere Schranke der Fluoreszenzintensität hat, und die Masse hat einen großen Einfluss auf die Immobile Phase, die in einer solchen Kurve sichtbar wird. Bei diesen zwei Parametern ist in den Ergebnissen ein Widerspruch aufgetreten, der mit den gegebenen Daten nicht erklärt werden konnte und eine weitere Untersuchung erforderlich macht. Die Dissoziationsrate zeigt eine starke Auswirkung auf die Wiederherstellungszeit, wohingegen die Bindungsrate keine Auswirkungen auf die Ergebnisse zu haben scheint. Da aber nur drei verschiedene Werte für die Bindungsrate betrachtet wurden, sind auch hier weitere Untersuchungen mit kleineren Werten nötig.
Mit diesen Erkenntnissen könnte in Zukunft weiter daran gearbeitet werden, herauszufinden welchen Einfluss bestimmte Faktoren auf die Fluoreszenzwiederherstellung haben, um das zugrundeliegende biologische System einer FRAP-Kurve zu bestimmen und weiter zu erforschen, und um eine passende Einstellung der Parameter für komplexe Simulationen zu finden.

Contents

1 Introduction

1.1 Motivation and Objective

The field of systems biology has become more and more important over the last few years [1]. It is used to study biological systems and processes on a systems level to predict how these systems change over time and under varying conditions [3]. To accomplish this, it combines several scientific disciplines: biology, computer science, bioinformatics, physics and others [3]. Systems biology is responsible for many important developments not only in human health but in environmental sustainability [3].
Living systems are very complex. There are interactions between a large variety of components. Therefore, it is hard to predict their behavior by studying only isolated components or subsystems [1]. Instead, one must examine the structure and dynamics of cells and the whole organism. Systems biology tries to deal with this problem, by summarizing current knowledge into manageable quantitative or qualitative descriptions and finding an appropriate level of abstraction [1]. With the help of that, modeling and virtual simulation of such systems provide powerful tools to gain insight into the complexity of living systems [16].
Due to the increasing presence of simulation software, such as LAMMPS [22] and SRSim [7], the simulation of biological systems on the computer has become an important part of the scientific research. Not only to answer the question of whether one can reproduce a biological system on the computer but also because it brings many advantages, such as the ability to adjust individual parameters easier or to perform an experiment several times without much effort.

This bachelor thesis deals with the topic of virtual simulation. Its goal is to simulate the multifunctional tool *fluorescence recovery after photobleaching* (FRAP) with the help of the simulating software SRSim, which will be extended by a self-written method. Since the 1970s, FRAP has been an instrument for analyzing molecular transport on the micrometer scale [19]. But since it is a very complex procedure and it is rather difficult to interpret the resulting data, it would be reasonable to simulate such an experiment.
Hence, in addition to extend SRSim, the objective of this thesis is to provide an overview of existing literature on the tool SRSim and FRAP and to simulate the effects of parameters on the fluorescence recovery that are crucial for the experiment. This includes the radius of the bleaching area, the mass of the binding molecules and the binding- and dissociation rates. For this purpose, various simulation models

were designed, such as the diffusion-related model and the reaction-related model. The result of each simulation has the form of a so called FRAP curve or recovery curve, which represents the fluorescence recovery depending on the time.

Essentially, this bachelor thesis can be divided into four parts. At the beginning theoretical and technical principles concerning FRAP and the tool SRSim are explained. Chapter two deals with the extension of SRSim and the structure of the method used to simulate FRAP. Based on this, in the next section, three distinct simulation models are implemented and explained in detail. This section also involves each simulation's outcomes, which will then be discussed and interpreted in the last section.

1.2 Fluorescence Recovery After Photobleaching (FRAP)

1.2.1 Principle of FRAP

Fluorescence recovery after photobleaching (FRAP) is a multifunctional tool for determining diffusion and interaction properties in biological and material science at the micrometer scale [19,21]. It was developed in the 1970s and became increasingly popular in the 1990s. FRAP is based on a simple approach [19]: At first the translational diffusion coefficient of a fluorescence labeled protein is measured by bleaching molecules that diffuse into a limited volume of a light beam [20]. As a result, the fluorescence intensity is decreased in the bleached region, as illustrated in Fig. 1(b). This process is called photobleaching [19]. In the next step the fluorescence recovery can be measured with a highly attenuated light beam on grounds of the diffusion of fluorescent labeled molecules from the adjacent unbleached areas into the bleached area [20]. The results of the varying intensities of the fluorescence measured over time are represented by a recovery curve (Fig. 1(i)) [19]. The recovery rate of the fluorescence intensity depends on the mobility of the molcules. Meaning that, if all molecules are essentially mobile the final fluorescence intensity - after a certain amount of time, ensuing from the bleaching process - will reach nearly the same intensity level as before the bleach, see Fig. 1(c)(d)(i). A divergence between the final fluorescence intensity, and the one before the bleach, indicate that there is a fraction of immobile bleached particles (Fig. 1(i)). With a suitable mathematical method one can analyze the evolution of the fluorescence recovery and extract quantitative information on the molecular dynamics, such as diffusion and binding [19]. Next to calculating the diffusion coefficient D, which is a measurement for the mobility of atoms and requires a high level of knowledge of FRAP theory, one can quantify

FRAP measurements in terms of the half-time of recovery $t_{1/2}$ [14], defined as the time to reach half of the final recovered fluorescence. It can be easily extracted from FRAP recovery curves [10,14]. For example, the simplest way to describe a recovery curve is the use of a simple expontential function [15, 19].

$$\frac{F(t)}{F_0} = 1 - be^{-t/\tau} \tag{1}$$

F_0 is the initial fluorescence intensity, $F(t)$ is the fluorescence intensity in the bleached area after time t, τ is a characteristic revovery time which is related to $t_{1/2}$ through $t_{1/2} = \tau \ln 2$ and b is the fraction of fluophores that are bleached. With the help of Eq.1 one can compare half-times and make relative statements about the diffusion rate.
However, $t_{1/2}$ is an empirical parameter, meaning that it is dependent on experimental parameters such as the radius of the bleaching area and molecular dynamics, and cannot be compared across studies [14].
In this thesis, the value of $t_{1/2}$ is defined slightly different. Based on the radius of the bleached area, the maximum possible fluorescence intensity is calculated and used as an expected upper bound for the fluorescence recovery of each simulation. This will then be used to determine $t_{1/2}$. It can be assumed that each model can reach this upper bound after a finite amount of time.

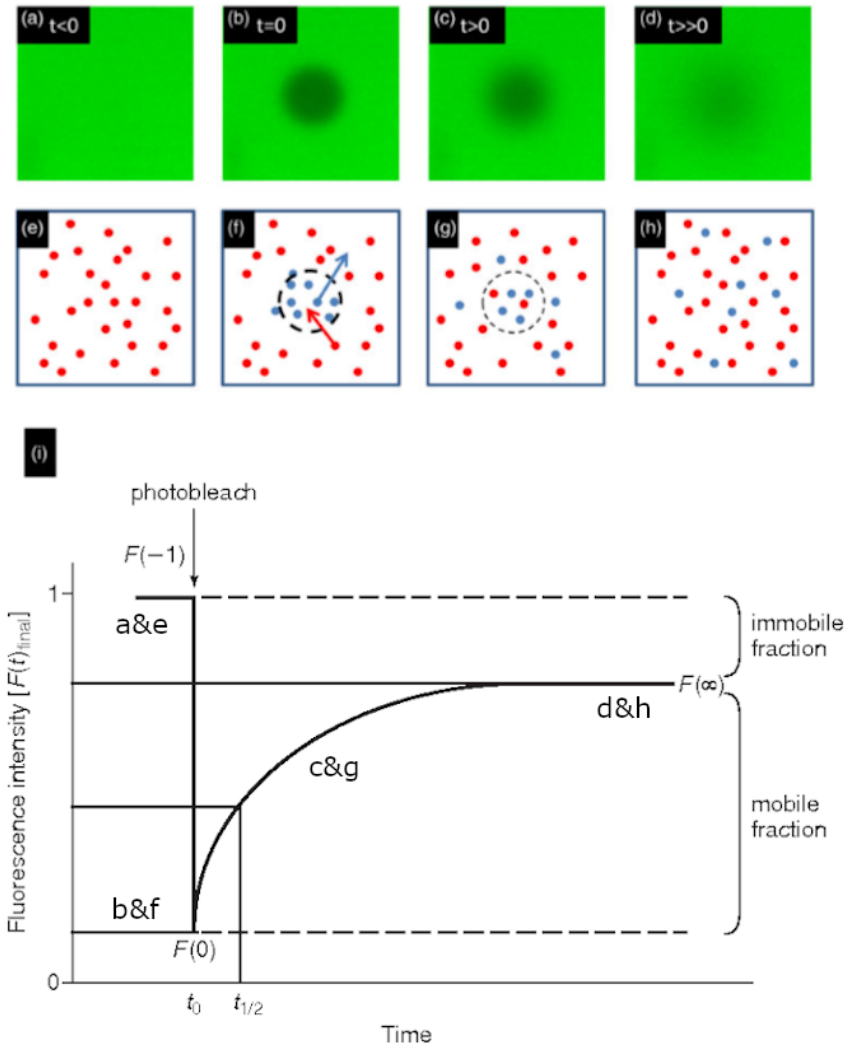

Figure 1: **The Principle of FRAP**. In panels (a)-(d) a FRAP experiment in a glycerol solution is shown. The panels (e)-(h) illustrate the corresponding actions abstractly at the molecular level, whereas blue and red spots are bleached and non-bleached molecules, respectively. The graph in (i) shows a schematic recovery curve that corresponds to (a)-(h).
F(-1) is the initial fluorescence. At t = 0 the photobleach takes place and causes a drop in the fluorescence intensity, here pictured as F(0). $t_{1/2}$ is the recovery half-time, i.e., time to reach half of the final recovered fluorescence. At the end of an experiment with $t = \infty$, fluorescence intensity has recovered to $F(\infty)$, which is either a full recovery or less than F(-1). ((a-h) are taken from Lorén et. al. [19] and (i) from Hardy et. al. [10])

1.2.2 Basic Instrumentation

The basic instrumentation consists of an optical microscope, a light source and a fluorescent sample. Since FRAP was developed in the 1970s a wide variety of instrumentation has been established [20]. In every setup used, there must be one light source with bleaching power and another to monitor fluorescence before and during the fluorescence recovery process. Mostly, intense laser light is used for bleaching, whereas monitoring is done either by laser light or by a light from a mercury vapor lamp. If a laser is used, one can use a single laser source for both, bleaching and monitoring [20].

There are two different approaches for recording the fluorescence intensity. It is recorded either directly by a photomultiplier (PMT) signal or by analysis of camera images taken during recovery. In recent years, the second method has become more popular due to the rapid development in digital photography [20]. The measurement process is performed with a conventional or confocal fluorescence microscope.

1.2.3 Fluorescence and Diffusion

The main principles explaining FRAP are fluorescence and diffusion.
Fluorescence is the emission of light by matter or molecules subsequent to the absorption of light or other electromagnetic energy [25]. In that process atoms are put into an excited singlet state by the exposure of light and then relax to the ground state with spontaneous emission of light. The emitted light usually has a longer wavelength than the absorbed radiation [17, 23, 25].
In FRAP fluorescence labeled particles are bleached, caused by oxidation of the excited molecules. Hence, the bleach rate is dependent on the collision rate of molecular oxygen with the excited fluorophore [19]. This implies that the bleach rate depends on environmental factors like temperature, viscosity, and oxygen content. FRAP generally assumes that the fluorescence is directly proportional to the concentration of fluorescent molecules and that the bleaching is irreversible [20].

The second important principle in FRAP experiments is diffusion. One can divide this process into two mechanisms: diffusion and self-diffusion.
Diffusion in general is a process with which matter is spread from one part of the system to another by random motions. The process does not require any extra energy except thermal energy, i.e., it is a passive transport mechanism [6, 19].
Self-diffusion is also an inherently random motion displacement process and requires

no extra energy. However, it is a process at thermodynamic equilibrium, i.e., individual molecules are randomly displaced by collision with surrounding molecules rather than shifting the gravity center of a material mass [19].

A special case of sub-diffusive motion is when the solute diffuses in the presence of a binding partner. By labeling the solute with a fluorophore one can retrieve information about binding and unbinding events with the help of FRAP.
According to Lorén et.al. [19] the reversible reaction rule can be described with

$$S_f + \mathrm{B} \underset{k_{off}}{\overset{k_{on}}{\rightleftharpoons}} C_f, \tag{2}$$

where S_f represents the free fluorescently labeled solute molecules, B the unlabeled binding partners, and C_f the (fluorescent) molecular complex between S_f and B. $\mathrm{k_{on}}$ is the second-order rate constant of the binding reaction and $\mathrm{k_{off}}$ is the first-order rate constant for the dissociation of the complex [5], illustrated by Fig. 2. The FRAP recovery $C_{FRAP}(t)$ reflects the changing concentrations of the free solute $\mathrm{C}_s(\mathrm{t}) = [\mathrm{S}_f]$ and the solute-binding partner complex $\mathrm{C}_c(\mathrm{t}) = [\mathrm{C}_f]$ [19].

$$C_{FRAP}(t) = C_s(t) + C_c(t) \tag{3}$$

Figure 2: **Binding and Dissociation**. A schematic representation of the binding process. A free fluorescence tagged molecule (green) diffuses and eventually binds to a binding site (blue). The binding and dissociation is paramerterized by the rate constant k_{on} and k_{off}, respectively. The binding site is either mobile or immobile. In both cases it has an effect on the motion of the fluorescent molecule. This figure is from [19].

1.2.4 Areas of Application

Since the first FRAP experiment in 1974 by Peters et al. [21], who demonstrated the mobility of membrane proteins in erythrocyte ghosts, this method has found applications in many other areas. For instance, it is closely associated with cell biology, in which it is used to study the mobility or binding of membrane proteins [19]. Based on that, one can make conclusions about the structure and function of the cellular membrane and its components [19]. A typical practical example in this field is the study of PML NBs – promyelocytic leukemia protein nuclear bodies [24].
PML NBs are internal domains in the mammalian nucleus [18] and are expressed in all tissues and cell lines [24]. They are most likely involved in apoptosis, DNA damage response, cellular senescence, and angiogenesis, though their biochemical function is still unclear [24]. During process of FRAP, PML proteins are tagged with fluorescence proteins like GFP and the mobility of the individual intranuclear components can be examined [24].

Only a short while after the first FRAP experiment, this technique was applied to pharmaceutical and biomedical research. In these fields, it is used to study the diffusion of therapeutic macromolecules and their binding kinetics with receptors within a cell or other systems [19]. Furthermore, FRAP can be used in the areas of food science and food technology. [19].

1.3 Spatial Rule-based Simulation

SRSim (version 05-03-2014) is an integrated spatial and rule-based simulator software [7–9]. It was designed for the simulation of biological systems, especially reaction networks that are combinatorially complex as well as spatially inhomogeneous.

Rule-based modeling is based on the idea of subdividing molecules into their components, e.g., active sites, protein domains etc. These sites or domains can then bind to other proteins or can be modified post-translationally. This concept is called domain-orientated approach [4, 12]. A complex molecule can be described by a molecule graph, which consists of elementary molecules that are connected with each other as illustrated in Fig-3(b). The subdomains of a particle are called components and they either serve as connectors or can be modified (Fig-3(a)). A set of possible reactions is implicitly defined by a rule-based reaction system (R, P, S). R is the set of rules based on groups of chemical species $s \in S$ (A and B in Fig-3(a)). They are sharing a common property (or pattern) $p \in P$, e.g. the containment of a

similar subgraph structure [8]. Fig-3(c)(d) illustrates an example.
In principle, in a simulation with SRSim, individual particles are used for each elementary molecule. These particles are represented by their position, velocity, species, modification state and other characteristics. They can diffuse through reaction space - a cuboid box of selectable dimension [7] - and push away other molecules. During that process, biomolecular reactions can occur if the particles, that have collided, match a reaction pattern specified by the reaction rules. The biochemical reaction also depends on the geometrical properties of the atoms, such as size or the position and angle of binding sites.

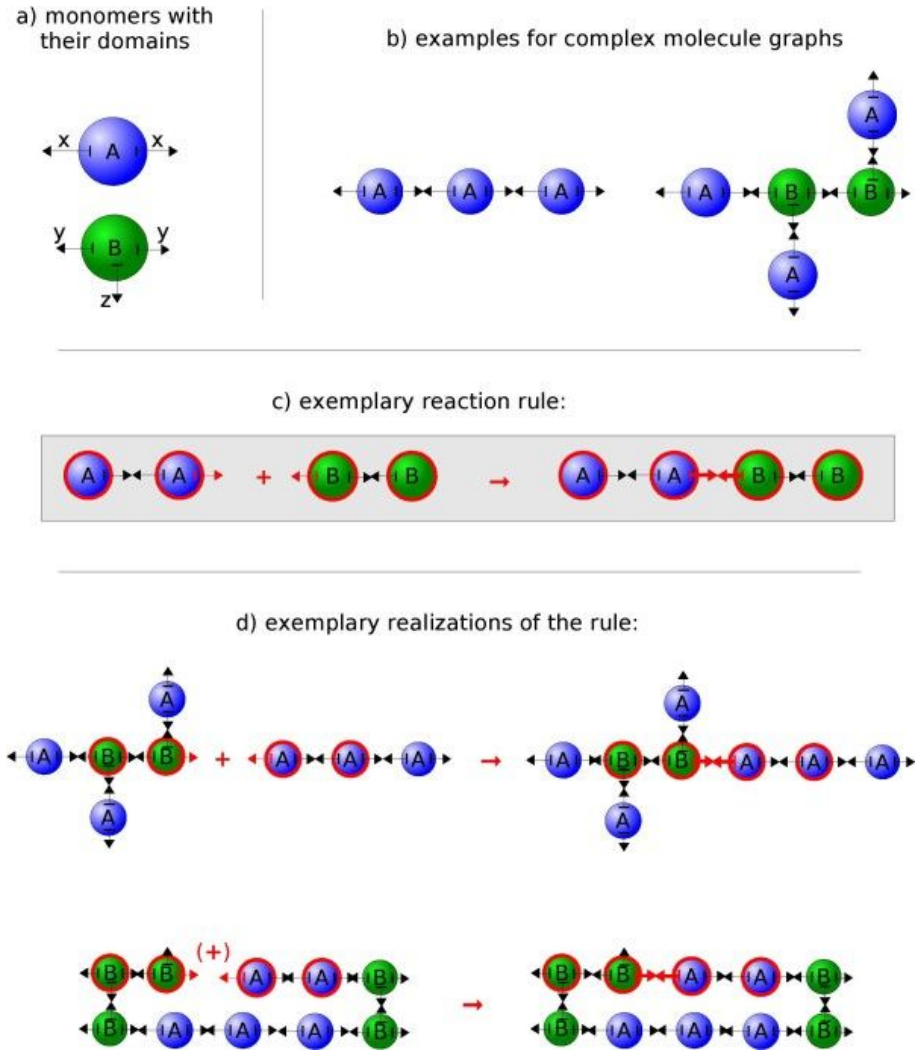

Figure 3: **Exemplary Rule-based System**. In (a) two elementary molecules with their domains (or components) are displayed. Multiple elementary molecules can be connected at their domains to form a complex molecule graph (b). Panel (c) shows a binding reaction rule in form of a pattern graph (or reactant patterns). An example for a realization of the rule is displayed in (d) [8].

SRSim is an extension of the molecular dynamics simulator LAMMPS (version 07-07-2009) [8,9,22] so it can be started in the same way: $lmp\_srsim < input\_script.in$. The input script *.in is used to start a SRSim simulation and contains references to three other input files, **.bngl, *.geo* and **.tgeo* and two or more output files (Fig-4). The input script is parsed line by line, where each line contains a command for modifying the simulation system. There is a set of basic parameters set in the first phase of the file, e.g. units, size of reaction volume, maximum number of molecule species, and initial configuration of the system [9]. In the second phase so called 'fixes' are selected, which are computations that influence each molecule's data, e.g. position, velocity or binding state. In the last section there are definitions of the output types and length of simulation runs [9].

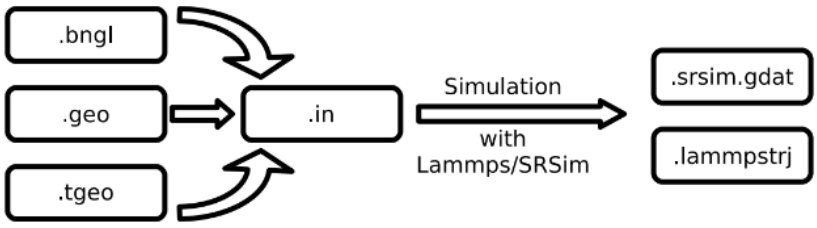

Figure 4: **File Structure of SRSim**. An overview of input and output files from [9].

The **.bngl* file specifies the used rule-based reaction system in the BioNetGen language (BNG) [11], which enables an easy import and export of models from BioNetGen [8]. It includes information like rate constants, molecular types, initial concentration of all molecules and possible reactions. Information about the molecular geometry, like mass radius and attributes for each component, are stored in the **.geo* and **.tgeo* files, the latter contains the template geometry for more complex molecules. Both files use the Extensible Markup Language (XML) [9].
The standard output files are a *.srsim.gdat file containing the concentration of the observed species and the *.lammpstrj file with all the molecular coordinates, which can be used for visualization and analysis of the simulation run. There are various programs to observe the results, e.g. GNUplot and VMD[1] [13].

[1]Visualization program. See https://www.ks.uiuc.edu/ for more details and tutorials.

2 SRSim Extension Virtual FRAP

Since the goal of this project is the simulation of FRAP experiments, it is necessary to extend LAMMPS, or rather SRSim, by two commands "*bleach*" and "*count*" as a realization of the bleaching and measurement steps of such an experiment, respectively. The two commands have the following form:

```
                name     coordinates    radius   Intensity
start_frap      bleach   x   y   z        r        I

start_frap      count    x   y   z        r
```

in the input script. First of all, the bleaching process - carried out with *bleach* - requires a certain area, in which the bleaching will occur. In this study, this area has the form of a sphere with position *x, y, z* and radius *r*.
The bleaching is implemented by defining a specific modification state for each atom, which declares if the fluorescence tag is still intact, see chapter 3. In this implementation, the modification site must be the first site of the atom, i.e., if one wants to have an atom with a tag and some binding sites *a, b, c* the declaration has to look like this: $A(tag, a, b, c)$ (see chapter 3). Additionally, the marked atom must first be defined in the **.bngl* file. If the atom gets bleached this state will be changed. In addition, there is a certain intensity $I \in [0, 1]$, which specifies the probability for a particular atom to be bleached within in the sphere, based on the fact that a laser also has certain intensity and does not bleach all the particles.
The measurement of the fluorescence intensity is done by simply counting the number of fluorescent atoms - with the command *count* - in the sphere and checking the modification state. Eq. 4 and 5 are used again, to determine the location of an atom.

Implementation

The following is a brief description of the implementation of the new command *start_frap*. SRSim is an extension of LAMMPS, which is written in the programming language C++ and was specially designed to be easily modified and extended with new functionalities [1]. At first, it is useful to find similar features that already exist, and take a closer look at the source code and header files to further the understanding of said programs.

The next step is to create a new class *StartFrapSRSim*. This requires two files in C++. On one side there is a source code file **.cpp* and on the other side a header file **.h* that contains the declarations and other components of the source code.
This new class must be derived from the base class itself, or from a class that already exists. *StartFrapSRSim* was derived from the class *Pointers*, which contains references to certain data structures that represent e.g. atoms, their binding sites and coordinates.
StartFrapSRSim implements three functions, one of them in command style, i.e., it can be called by using the command name in the input script. To add such a command one has to append it to the header file *style_user.h*, in order for it to be recognized by the program as a new command.
The command *start_frap* has two options: *bleach* and *count*. Depending on which command name is given in the input script, *start_frap* will either bleach a specific area or measure the fluorescence recovery in that area. Both options need three coordinates, a radius and, in case of *bleaching*, an intensity with a value between 0 and 1.
The coordinates are used to define a sphere k with a radius r, which represents the bleaching area. When using the *bleach* command the sphere is defined and then all atoms - that are tagged - are checked to see if they are in that sphere.

Listing 1: start_frap: bleach

```
void StartFrapSRSim::command(int argc, char **argv )
  {
    int countOn = 0; //variables to count the atoms in de sphere, fluorescent
    int countOff = 0; // bleached
    // cast of a LAMMPS-atom to a SRSim-atom
    AtomVecSRSim              *avecsrsim  = dynamic_cast<AtomVecSRSim*>(atom->avec);
    int nlocal = atom->nlocal; //total number of atoms

    if (argc < 1) {error->all("StartFrapSRSim:: command: start_frap should have at
        least 1 Paramater for a command {bleaching, count} and parameters for more
        setting {x-coordinate, y-coordinate, z-coordinate, radius, intensity}");}

   if (argc == 6 ){ //bleach command

        double x = atof(argv[1]); //x coordinate for sphere
        double y = atof(argv[2]); // y coordinate
        double z = atof(argv[3]); // z coordinate
        int r = atoi(argv[4]); //radius of the sphere
        double intensity = atof(argv[5]); //intensity or probability for bleaching
            [0,1]

        double k[3] = {x, y, z}; // initialize sphere's center coordinates
        if (intensity < 0 || intensity > 1) error->all("Illegal SRSim command:
            intensity too small has to be between 0 and 1");
        //printf("number of  Atoms %d\n", nlocal);

```

```
23          if (strcmp(argv[0], "bleach" ) == 0) { // command for bleaching
24
25
26              for (int j=0; j < nlocal; j++){ // iterate through all atoms j
27                  if(atom->type[j] == 1){
28
29                      if(distance3(atom->x[j], k) < r){
30                          double random = (rand() % 100); // generate random number
                                between 0 and 100
31                          if(random < (intensity*100)) {bleaching(j);}
32                          //if this number is smaller than the Intensity, this atom
                                will be bleached
33                       }
34                  }
35              }
36          }
37          else
38               error->all("StartFrapSRSim::command: start_frap didn't recognize
                     command!");
39
40
41  ...
```

To determin if an atom is located in the sphere, one has to calculate the distance between the atom a and the sphere k with the following equation, where k and a are vectors:

$$dist(k,a) = \sqrt{(k-a)^2} \tag{4}$$

To determine $(k-a)^2$ one has to calculate the scalar. The general formula is

$$\vec{x} \circ \vec{y} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \circ \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = x_1 \cdot y_1 + x_2 \cdot y_2 + x_3 \cdot y_3 \tag{5}$$

$dist(k,a)$ is then compared to the radius. If it is smaller, the atom is located in the sphere. The distance is calculated with the function *distance3()*:

Listing 2: Method distance3()

```
/**Method for calculating the distance between two vectors*/
double StartFrapSRSim::distance3(double a[], double k[]){

    double x = abs(k[0]-a[0]); //calculate distance for each entry in vector -->
        absolute value
    double y = abs(k[1]-a[1]);
    double z = abs(k[2]-a[2]);

    double abstand[3] = {x, y ,z}; //vector of distance

    double skalar = 0;
    double produkt;
    for(int i=0; i < 3; i++) //calculate scalar of distance vektor
    {
        produkt = abstand[i]*abstand[i];       // multiply values
        skalar+=produkt;              //add up
    }
    double wurzel = sqrt(skalar);
    return wurzel;
}
```

If the particles are in the sphere, they get bleached with a specific probability I. Bleaching is performed by setting the modification state of the atom to 1 using the function *bleaching()*:

Listing 3: Method bleaching()

```
/**Method for bleachig an atom*/
void StartFrapSRSim::bleaching(int t){ // t is the current atom,
                                       // which position was checked
    AtomVecSRSim         *avecsrsim   = dynamic_cast<AtomVecSRSim*>(atom->avec);
    // cast of a LAMMPS atom to a SRSim atom

    int newModif = 1; // set site to 1 = unmodified, meaning,
                      // this atom was bleached and is no longer fluorescent
    avecsrsim->site_modified[t][0] = newModif;

    avecsrsim->updateSubgraphTemplateData(t); //has to be updated

}
```

In SRSim, all modification states are set to a specific value depending on which state the atom is in. The default is -1, in this case 0 means the atom is still fluorescent while 1 is the state of said atom after the bleach.
The *count* command works like *bleach*, but instead of calling *bleaching()* it counts how many of the atoms still have their fluorescence-tag and are located in the sphere. These values can then be used to calculate FRAP curves with e.g. the tool GNUplot.

3 Results

There are different kinds of diffusion models, as already mentioned in the section 1.2.3 'Fluorescence and Diffusion'. The simplest variant is a diffusion-related model, i.e., there is only one type of atom that diffuses through the reactor without any binding reactions. The second model is the reaction related model, meaning there is not just one kind of particle, but at least two, and those can bind. For the second model, two variants are designed in which the binding partner of the labeled atom is (almost) completely immobilized in two different ways. On one side, the binding partner will be immobilized by modifying its mass, on the other side, it can enter a bond with other atoms of the same type and forms a network-like fixed structure. In order to test the new method *start_frap_srsim* in SRSim and to simulate the influence of certain parameters, those two diffusion models are devised.

3.1 Diffusion-related Model

For the first model, one type of atom is designed. From now on this atom will be called *GFP*. It has only one modification site, which represents the fluorescence tag. This site can be modified by the bleaching process, which can be expressed with the following irreversible formula:

$$GFP_{on} \xrightarrow{\text{bleach}} GFP_{off} \tag{6}$$

There are no additional binding sites, since one atom can not form a bond with other *GFP* atoms. According to this, there are no binding reaction rules defined. *GFP* is of small size and mass, allowing it to diffuse freely and quickly, based on the fact that atoms with a lower mass diffuse faster, as mentioned in the paper by Callister et. al. [2].

	size	mass	sites	initial number
GFP	1.0Å	1.0g/mol	1	5000

	reactor size	temperature	# runs	time of bleach
simulation	50x50x50Å	300K	17,000 timesteps	after 2,000 timesteps

	radius	location (coordinates)	bleaching intensity
sphere	10Å 15Å 20Å	(25, 25, 25)	90%

Table 1: Basic Setting of the Diffusion-related Model

3.1.1 Implementation of the System

There are four input files implementing the system, which are partly shown here. The complete files can be reviewed in the supplements.
The initial number of atoms (*GFP0*) and the definition of the atom *GFP* are set in the **.bngl* file (Listing 4). *GFP0* is 5000 to guarantee a higher number of atoms in the sphere. As required by the current implementation, the tagged proteins are the first defined and their first site is the one where the modification can take place.

Listing 4: Diffusion.bngl File

```
# Simulation of diffusion without binding processes

begin parameters
  GFP0      5000      # initial number of atoms
end parameters

begin species
  GFP(f~Fon)                GFP0    # definition of labled atom
end species


begin reaction rules

# no reaction rules defined

end reaction rules

begin observables
  Molecules    nGFP0    GFP(f~Fon)      # molecule that can be displayed later
end observables
```

The geometry settings of *GFP* are set in the **.geo* file (Listing 5). The parameters in *GeneralProperties* are not yet relevant and are set to their default value. The initial coordinates are set in the **.tgeo* file (Listing 6).

Listing 5: Excerpt of Diffusion.geo File

```
<?xml version="1.0"?>
<molecule-geometry-definition>
   <version value="1.01" />

  <GeneralProperties>
    ...
  </GeneralProperties>

  <molecule name="GFP" >
     <mass    value="1.0" />
     <radius value="1.0" />
     <site name="f" phi="0" theta="0" dist="1.0" />
  </molecule>

</molecule-geometry-definition>
```

Listing 6: Diffusion.tgeo File

```
<?xml version="1.0"?>
   <template-geometry-definition>

      <template id="0" name="GFP(f~Fon)">
         <mol id="0" x="0.0" y="0.0" z="0.0" />
      </template>

   </template-geometry-definition>
```

The settings for the simulation itself are defined in the **.in* file (Listing 7).

Listing 7: Diffusion .in File

```
# diffusion .in file

dimension   3
boundary    f f f   #use fixed boundary conditions
units       real    #timescale: fs, distances: Ansgtrom
newton      on

atom_style  srsim just_diffusion.bngl just_diffusion.geo just_diffusion.tgeo 123456
```

```
9
10 lattice none
11 region            Nukleus block 0 50.0 0 50.0 0 50.0 units box # size of reactors
12 create_box        10 Nukleus
13
14 #                   name
15 start_state_srsim     coeffs
16 start_state_srsim     atoms
17
18 # Stimulation:
19 #                        Brownian Temp Temp damping randomSeed
20 fix           2    all   langevin  300  300  200.0   1234567
21 fix           1    all   nve
22
23
24 #  SRSim Reactions:  id  group fixName | nEvery  randomSeed  preFactBindR
       preFactBreakR preFactExchangeR preFactModifyR_1 preFactModifyR_2
25 #  fix             3    all   srsim   1   45678   4.7   1e-3   1.0   1.0e-3   4.7
26 fix               4    all   wall/reflect xlo xhi ylo yhi zlo zhi
27
28 # Dumps:
29 timestep     1.0
30 thermo       500
31
32 dump           2 all srsim 50 diffusion.srsim.gdat
33
34 dump           1 all atom 10 one.lammpstrj
35 dump_modify    1 scale yes
36
37 # Stimulation 1
38 run          2000
39
40 #           command     x    y    z    r    l
41 start_frap  count       25.0 25.0 25.0 10
42 start_frap  bleach      25.0 25.0 25.0 10 0.90
43 start_frap  count       25.0 25.0 25.0 10
44 #2
45 run      500
46 start_frap  count 25.0 25.0 25.0 10
47 #3
48 run      500
49 start_frap  count 25.0 25.0 25.0 10
50
51 ...
52
53 #30
54 run      500
55 start_frap  count 25.0 25.0 25.0 10
```

In the first phase settings for basic parameters like size of reaction volume, units to be used, maximum number of molecular species, and initial configurations of the simulation system are implemented. After that, the other input files *.bngl, .geo and .tgeo* are included. With *start_state_srsim coeffs* and *atoms* all settings for forces

and geometry are adjusted, and a set of molecule graphs depending on the amount of particles and reactant template conformations are created.
In the second phase, different *fixes* are defined. The fix *nve* and *langevin* [7] are essential for the movement of molecules, resulting in diffusion. The fix *wall/reflection* is applied to prevent those molecules which might move out of the reactor from being lost, this being a result of previously choosing fixed boundaries [7]. Since there are no reaction rules defined, the fix *srsim* is not yet used. After the initial settings and the first 2000 steps, *start_frap* is called with the bleach command.

3.1.2 Affect of Different Radii

The goal of the first model was to provide a first impression of the FRAP method. Initially, three different radii were used for the sphere to see how it affects the fluorescence recovery. Measuring the fluorescence recovery is done 30 times every 500 steps. For each of the three cases, 20 runs are performed and from this the arithmetic mean and standard error[2] are calculated.
The resulting recovery curves are illustrated in Fig. 5. All three graphs show the average shape of a FRAP curve. The proportion p of the sphere at the reaction space was calculated using the formula $p = \frac{\frac{4}{3} \cdot \pi \cdot r^3}{Vol_{reactor}} \cdot 100$. At a radius r of 20A, the bleached area takes up over 25% of the reactor, whereas in the other two cases it is only 3% and 11%, respectively. Based on that, the upper bound for the fluorescence intensity is 97%, 90% and 75% (gray dashed line in Fig. 5).

[2] was calculated with a proprietary Python script, see attachment

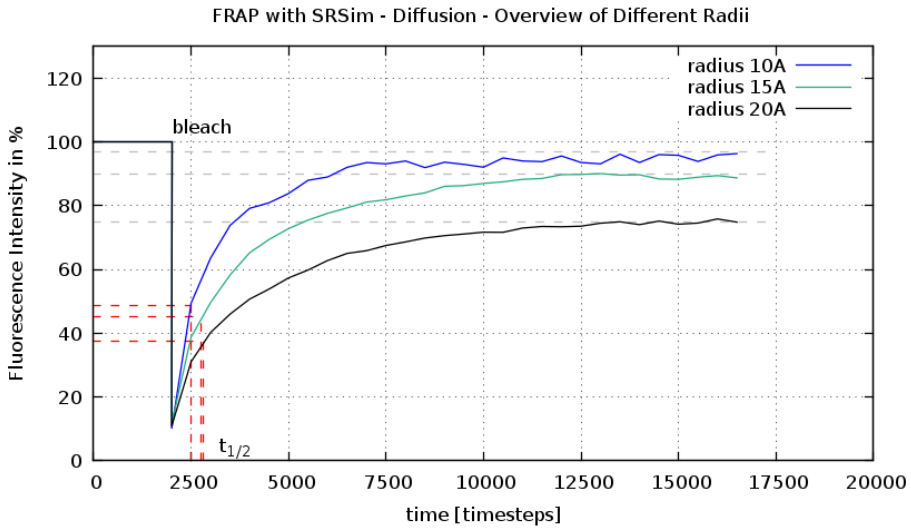

Figure 5: **Avarage FRAP Curves of the Diffusion-related Model with Different Radii**. The plot shows three FRAP recovery curves of the diffusion-related model, with different sphere sizes: r =10Å(blue), r = 15Å(green), r = 20Å(black). The total number of atoms inside the sphere just before the bleach process is set at 100%. For these simulations, a temperature of 300K, 5000 atoms, a reactor size of 50x50x50Å and a bleaching intensity of 90% are chosen. In each case, 20 simulations with 30 measurements were carried out and the bleach occurs after 2000 timesteps. The standard error of mean is between 0.4% to 2%. The recovery half-time $t_{1/2}$ (red) is reached at $2.5 \cdot 10^2$(blue), $2.7 \cdot 10^2$(green) and $2.8 \cdot 10^2$(black) timesteps, respectively.

The radius 10Å indicates the highest and fastest fluorescence recovery. The recovery half-time $t_{1/2}$ is reached at $2.5 \cdot 10^2$ steps, i.e., the time at which the recovery rate reaches 50% (of the upper bound). Up to $4 \cdot 10^2$ timesteps, the curve rises very quickly, then it slowly converges towards 96%. It does not quite reach 100%. In this case, the error rate is the highest, resulting in the small fluctuations in the graph.
In the case of a radius of 15Å, the recovery rate is a little lower. The maximum recovery amounts to 90%, resulting in a bigger 'immobile fraction'. $t_{1/2}$ is reached at $2.7 \cdot 10^2$ timesteps. After approximately 1.000 timesteps, it converges towards 90%, where only small fluctuations occur.
The last case – radius 20Å - has a much lower recovery rate. It takes about $2.8 \cdot 10^2$ timesteps to reach 50% of the primary intensity and it has a maximum rate of 75%. There are almost no variations in the curve and the error rate is - like in the second case - low.

3.2 Reaction-related Model

The second model is the reaction-related model, which has an additional type of atom to serve as a binding partner. It is divided into two different variants.

3.2.1 Immobilization by Molecular Mass

First of all there is the fluorescent atom *GFP*, which has one modification site and one binding site. It is able to form a complex with the second type of atom, called *B*, but not with atoms of the same type. *B* has also one binding site, but no modification site, which means that it is not fluorescent. According to that, only *GFP* and the complex between those two atoms are relevant in the fluorescence measurement. The binding reaction between *B* and *GFP* is reversible and C_f represents the resulting complex (see also Eq.2).

$$GFP_f + \mathrm{B} \underset{k_{off}}{\overset{k_{on}}{\leftrightarrows}} C_f \tag{7}$$

k_{on} and k_{off} are set to 0.01 and 0.1, respectively. Those values specify the binding and dissociation rates of the reaction. In addition, *B* is slightly bigger than *GFP* and has a higher weight to simulate a immobile fraction. The mass is modified in three independent simulation runs.

	size	mass	sites	initial number	k_{on}	k_{off}
GFP	1.0Å	1.0g/mol	2	2,000	0.01	0.1
B	2.0Å	10g/mol 50g/mol 100g/mol	1	4,000		

	reactor size	temperature	# runs	time of bleach
simulation	50x50x50Å	300K	140,000 timesteps	after 6,000 timesteps

	radius	location (coordinates)	bleaching intensity
sphere	15Å	(25, 25, 25)	90%

Table 2: Basic Setting of the Mass Model

3.2.2 Immobilization by Complex Formation of the Binding Partner

In the second case, *GFP* is the same as before, but B has four binding sites instead of one. It is able to bind with *GFP* (Eq. 7) as well as with other Bs, for which the three additional binding sites have been defined. The reaction between atoms of type B is irreversible, i.e., once they have formed a bond they cannot dissociate again. Furthermore, B is smaller and lighter than before, but still bigger than *GFP*. Instead of modifying the mass, the k_{on} and k_{off} rates are changed in several simulation runs. The general properties of the simualtion are the same as before.

	size	mass	sites	initial number
GFP	1.0Å	1.0g/mol	2	2,000
B	2.0Å	10g/mol	4	4,000

Reactions	$GFP + B$	$B + B$
k_{on}	0.1, 0.01 and 0.2	irreversible
k_{off}	0.1, 0.2 and 0.01	0.0001

	reactor size	temperature	# runs	time of bleach
simulation	50x50x50Å	300K	165,000 timesteps	after 6,000 timesteps

	radius	location (coordinates)	bleaching intensity
sphere	15Å	(25, 25, 25)	90%

Table 3: Basic Setting of the Network Model

3.2.3 Implementation Details

In the first variant, *GFP* has the modification site f and an additional binding site a. B has one binding site b, which can bind to a. For this process, a reaction rule is defined in which *GFP* and B bind or separate.

```
GFP(a) + B(b)   <-> GFP(a!1).B(b!1) k_on, k_off
```

Setting an initial total number of 6000 atoms, the number of B has now twice the volume of the GFP. Some observables such as *GFP*, bound and unbound B, which

can be plotted later, are defined to monitor the complexing.

Like in the diffusion model, the geometry[3] of the atoms is set in the **.geo* file. GPF is small and has a low weight, whereas B is larger[4] and has a much higher weight to ensure that B is (almost) immobile. The mass was set to 10Å, 50Å and 100Å in three independent simulation runs. The variable *GPT_DEVI_ANGLE* is set to 180°, which allows *GFP* to bind to B from any direction. *GPT_Refractory* defines the waiting time for an atom to ensure that two particles cannot bind immediately after being broken. The remaining parameters were set to their default value.

Listing 8: Binding.geo File

```
<?xml version="1.0"?>
<molecule-geometry-definition>
   <version value="1.01"/>

  <GeneralProperties>
    <property name="GPT_Devi_Dist"   value="0.2"/>
    <property name="GPT_Devi_Angle"  value="180"/>
    <property name="GPT_Mol_Mass"    value="1"/>
    <property name="GPT_Mol_Rad"     value="1"/>
    <property name="GPT_Site_Dist"   value="1"/>

    <property name="GPT_Refractory"          value= "50.0"/>
    <property name="GPT_Force_Repulsion"     value= "1.5"/>
    <property name="GPT_Force_Bond"          value= "1.5"/>
    <property name="GPT_Force_Angle"         value=" 1.5"/>
    <property name="GPT_Force_Dihedral"      value= "1.5"/>
    <property name="GPT_Temperature"         value="300"/>

    <property name="GPT_Option_Dihedrals"    value="0"/>
    <property name="GPT_Option_Impropers"    value="0"/>
    <property name="GPT_Option_Rigid"        value="0"/>
  </GeneralProperties>

 <molecule name="GFP" >
     <mass    value="1.0" />
     <radius value="1.0" />
     <site name="f" phi="0" theta="0" dist="1.0" />
     <site name="a" phi="0" theta="100" dist="1.0" />
  </molecule>

  <molecule name="B" >
     <mass    value="100.0" />
```

[3] Comment: the variable *dist* is the distance of a binding site to the particles center. It depends on the size of an atom and should not be bigger than the radius. In this simulation it was set to 1 for *GFP* and 2 for B.

[4] the radius has a big influence on binding and dissociation processes. The larger the radius, the more dissociation occurs.

```
33      <radius value="2.0" />
34      <property name="GPT_Mol_Rad" value="2" />
35      <property name="GPT_Mol_Mass" value="100" />
36      <site name="b" phi="  0" theta="0" dist="2.0" />
37    </molecule>
38
39 </molecule-geometry-definition>
```

The setting in the input file **.in* are almost the same as in the diffusion simulation. Since there is a reaction rule defined, the *srsim* command is applied. It checks if any molecular collisions have occurred, analyzes which rules are practicable and executes them [9].
This time, the bleach command is called after 6000 steps, because bleaching is assumed to be after a binding equilibrium has been established [19].

In the second variant, B has several binding sites. One that can bind to *GFP* and three that can form a fixed structure only consisting of B.

```
GFP(f~Fon,a)     GFP0      # tagged protein A, one binding site
B(a,b,b,b)       B0        # binding protein for A, three additional
                           # sites
```

For the binding process two reaction rules are defined, one in which *GFP* and B are binding or breaking apart, and one in which B can form a complex with other Bs but does not dissolve again. This reaction was assigned to a new $k2_{on}$. In each simulation different values for the binding and dissociation rates are used.

```
  k_on    0.01    # normal
  k_off   0.1
  k2_on   0.0001

begin reaction rules
  1 GFP(a) + B(a)        <-> GFP(a!1).B(a!1) k_on, k_off
  2 B(b) + B(b)          ->  B(b!1).B(b!1)      k2_on   # reaction rule
                                                  # to form network
```

In the **.geo* files the parameters *phi* and *theta* are angles, and *dist* is the distance from the particle's center. They define the orientation of the atoms binding sites as spherical coordinates. Phi and theta can be imagined as the geographic longitude and latitude, respectively. Theta = 0° specifies one pole and 180° the other pole,

while 90° is the equatorial plane [9]. In this model, the binding sites of B are facing each other.

Listing 9: Excerpt of Network.geo File

```
1
2 ...
3
4 <molecule name="B" >
5       <mass    value="10.0" />
6       <radius value="2.0" />
7       <property name="GPT_Mol_Rad" value="2" />
8       <property name="GPT_Mol_Mass" value="10" />
9       <site name="a" phi="  0" theta="0" dist="2.0" />
10      <site name="b" phi="  0" theta="90" dist="2.0" />
11      <site name="b" phi="180" theta="90" dist="2.0" />
12      <site name="b" phi="0" theta="180" dist="2.0" />
13   </molecule>
```

3.2.4 The Influence of Immobile Atoms

The aim of the second simulation model was to simulate a (nearly) immobile fraction in two different ways. For this purpose, at first different masses were tested for the binding partner B - which is also twice as big as GFP - in order to investigate their effect on fluorescence recovery. Apart from that, no other parameters were changed in each simulation. Measuring the fluorescence recovery is done 35 times every 500 or rather 5000 steps. One simulation is done 20 time each, from which the arithmetic mean and standard error are calculated.
The resulting FRAP curves are illustrated in Fig. 6. Especially the graph at a mass of 10g/mol has the typical form as a FRAP curve resulting of free diffusion, whereas the other two do not have a steep of an increase. Furthermore, the fluorescence intensity reaches 30% very quickly within the first 10^4 timesteps in all three cases. The upper bound for each case is set to 90%, based on a radius of 15Å, that is about 10% of the total reactor, meaning that 10% of the atoms were bleached. 98% of all GFP are bound, as one can see in the Figures 13-15.

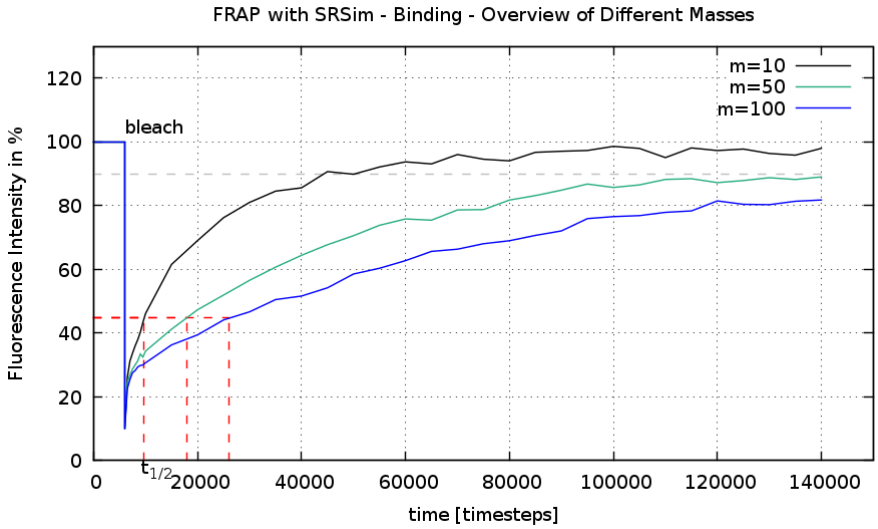

Figure 6: **Average FRAP Curve of the Reaction-related Mass Model with Different Molecular Masses.** The plot shows three FRAP recovery curves of the reaction-related model, with different masses of the binding partner: $m = 10g/mol$ (black), $m = 50g/mol$ (green), $m = 100g/mol$ (blue). The total number of atoms just before the bleach process is set at 100% ($\sim$ 215 atoms) and the radius of the sphere is 15Å. For these simulations, a temperature of 300K, 6000 atoms, a reactor size of 50x50x50Å and a bleaching intensity of 90% are chosen. In each case 20 simulations with 35 measurements were carried out and the bleach occurs after 6000 steps. The standard error of mean is between 0.5% to 3%. The approximate characteristic diffusion time $t_{1/2}$ (red) is reached at $9.7 \cdot 10^3$(black), $18 \cdot 10^3$(green) and $26 \cdot 10^3$(blue) timesteps, respectively.

The first case is defined with a mass of $10g/mol$. It reaches the 50% of the final recovery after $9.7 \cdot 10^2$ timesteps and then continues to rise quickly. It starts to converge towards nearly 100% after $60 \cdot 10^3$ timesteps.
The second case already shows some differences, particularly the recovery half-time. It is reached after $18 \cdot 10^3$ timesteps. The graph rises slower than the first graph and reaches a steady state at $100 \cdot 10^3$ timesteps. The recovery rate does not quite attain 90%.
At a mass of $100g/mol$ the immobile fraction is the most distinct. Reaching 50% of recovery requires much longer and the graph is much flatter than the others. The peak intensity of fluorescence is 82%.
For all cases additional plots were created with the complex formation processes, see attachment Figures 13-15. Based on these diagrams, one can understand how fast and how many complexes have formed, and how many of the particles are unbound on average. These plots do not show any distinct differences between various masses.

The second alternative of immobilization is the production of a complex network

consisting of the binding partner B, as described in Section 3.2.2. In addition, various values for k_{on} and k_{off} were used to analyze their influence on the fluorescence recovery. In this model measuring the fluorescence recovery is done 40 or 45 times every 500 or rather 5000 steps.
There are two resulting plots - Fig. 7 and Fig. 8 - each with various binding or rather dissociation rates. Due to theoretical considerations, the parameters $k_{on} = 0.01$ and $k_{off} = 0.1$ were chosen for the first simulation and are shown in both diagrams as a comparison. Only one value - k_{on} or k_{off} - has been modified[5] in each event, while the other value is the same as in the first setting.
As before the upper bound of the fluorescence intensity is at 90%, due to a radius of 15Å.

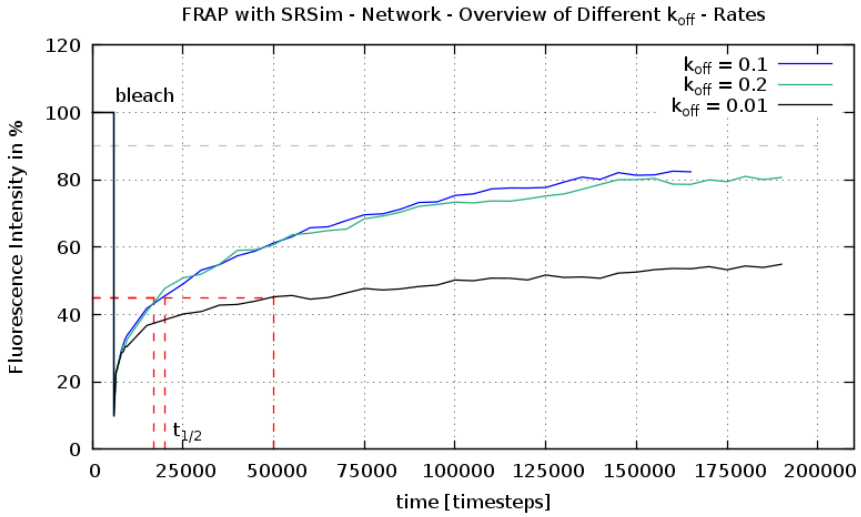

Figure 7: **Average FRAP Curve of the Reaction-related Network Model with Different Dissociation Rates.** The plot shows three FRAP recovery curves of the reaction-related model, with different k_{off} - rates for the reaction between *GFP* and B: $k_{off} = 0.1$ (blue), $k_{off} = 0.2$ (green), $k_{off} = 0.01$ (black). k_{on} is set to 0.01. The total number of atoms just before the bleach process is set at 100% (~ 210 atoms) and the radius of the sphere is 15Å. For these simulations, a temperature of 300K, 6000 atoms, a reactor size of 50x50x50Å and a bleaching intensity of 90% are chosen. In each case 20 simulations with 40-45 measurements were carried out and the bleach occurs after 6000 timesteps. The standard error of mean is between 1% to 3%. The recovery half-time $t_{1/2}$ (red) is reached at $20 \cdot 10^3$ (blue), $17 \cdot 10^3$ (green) and $50 \cdot 103$ (black) timesteps, respectively.

In Fig. 7 different k_{off} rates are shown. All three graphs show an equally fast recovery in the first $9 \cdot 10^3$ timesteps, probably caused by freely diffusing *GFP*. With a value of 0.1 and 0.2 no significant differences occur after these $9 \cdot 10^3$ timesteps steps. In both cases, the fluorescence intensity reaches 45% around $20 \cdot 10^3$ timesteps. The development of both graphs is almost the same, they start to converge after

[5] slightly modified: twice as big; heavily modified: 10 times as big/small

$140 \cdot 10^3$ timesteps. At $k_{off} = 0.1$ it reaches a slightly higher intensity, but both do not reach 90%.
$k_{off} = 0.01$ is very different to the other curves. It takes about $50 \cdot 10^3$ steps to reach $t_{1/2}$. The further run of the curve is very slow and does not reach 60%, even after $190 \cdot 10^3$ steps. The immobile fraction is accordingly large.

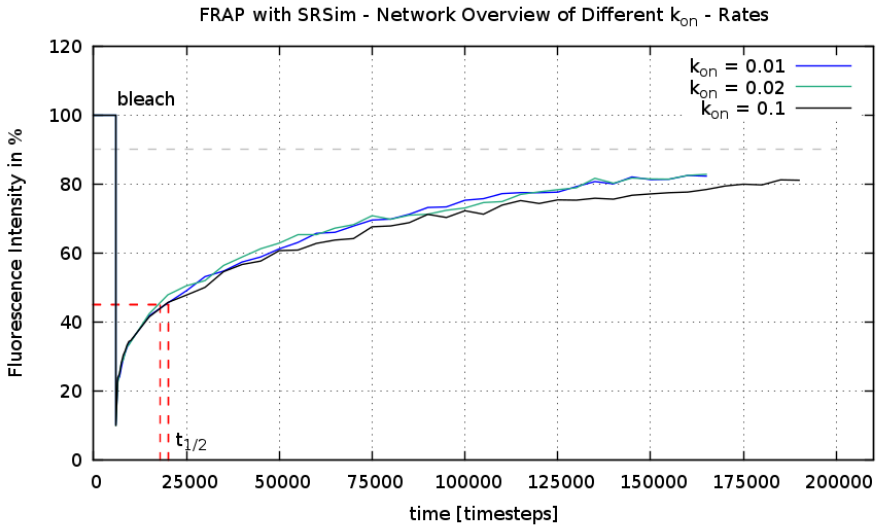

Figure 8: **Average FRAP Curve of the Reaction-related Model with Different Binding Rates.** The plot shows three FRAP recovery curves of the reaction-related model, with different k_{on} - rates for the reaction between *GFP* and B: $k_{on} = 0.01$ (blue), $k_{on} = 0.2$ (green), $k_{on} = 0.1$ (black). k_{off} is set to 0.1. No parameters were changed (see Fig. 7). In each case 20 simulations with 40-45 measurements were carried out and the bleach occurs after 6000 steps. The standard error of mean is in a range of 1% to 3%. The recovery half-time $t_{1/2}$ (red) is reached at $20 \cdot 10^3$(blue), $18 \cdot 10^3$(green) and $20 \cdot 10^3$(black) timesteps, respectively.

As with the k_{off} graphs, all curves in Fig. 8 increase rapidly at the beginning, but in this case the further progression of all three cases is very similar. The recovery half-time is reached around $20 \cdot 10^3$ timesteps. At a k_{on} of 0.01 and 0.02 both curves have a maximum intensity of 83%.
The third case reaches 82% and it converges after $175 \cdot 10^3$ timesteps, whereas the other two level out at $140 \cdot 10^3$ timesteps.
For all cases additional plots were created with the complex formation processes, like in the second model. These plots do not show any distinct differences. The plot (Fig. 20) with $k_{off} = 0.01$ shows a slightly faster complexing process.

The tool *VMD* was used to visualize the first simulation ($k_{on} = 0.01$ and $k_{off} = 0.1$). The results are shown in Fig. 9. The high number of atoms make it rather diffcult to observe any bonds or network-like structures. As a comparison, another simulation - with less atoms - was performed and visualized with *VMD*, see Fig. 21.

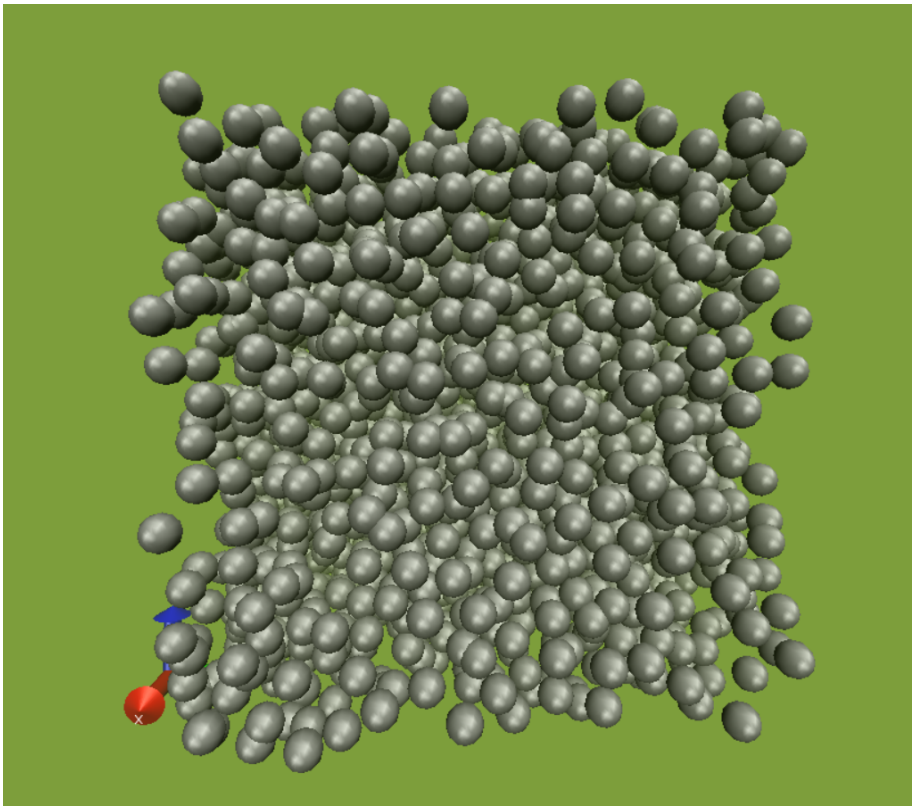

Figure 9: **3D-Visualization of the First Network Simulation** The first simulation with $k_{on} = 0.01$ and $k_{off} = 0.1$ was visualized with VMD, but just the binding partner B is pictured. Only a few dimers can be observed (two atoms touching).

4 Discussion

The effects of the parameters on the FRAP results were tested with the help of several simulations. In order to obtain representative results, the simulations were carried out several times and the mean value was determined. Since the scope of this thesis is limited, only a defined number of experiments and runs could be performed.
The biggest influence on the diffusion velocity, and correspondingly to the fluorescence recovery, showed the second simulation model: the presence of a binding partner, whose mobility has been manipulated with different masses. The first model also showed a significant effect on the recovery as the radius increased. The dissociation rate has only shown an influence when it was greatly decreased, whereas the binding rate seems to have no significant influence on the fluorescence recovery after photobleaching. This could be explained by the fact that only values between 0.01 and 0.2 were considered.

4.1 Diffusion-related Model

First and foremost, the diffusion-related model demonstrates the proper functionality of the *start_frap_srsim* method and confirms that it provides similar results to a real FRAP experiment.
The radius of the bleached area has a big influence on the maximum possible fluorescence intensity. By calculating the sphere proportion on the reactor, this can be confirmed. At a radius of 10A, only 3% is bleached, while at 20Å it is already one quarter of the total reactor. For this reason – in the case of 15Å and 20Å – the proportion of bleached atoms in the total number is very high, which should make it impossible for the intensity to return completely. This is supported by the fact that the immobile fraction of the individual curves correlate (almost) to the proportion of the sphere on the reactor.
The recovery half-time probably corresponds to the number of atoms that have not been bleached, since it was slightly higher at 20Å. The fewer fluorescent particles are left, the longer it takes to reach 50% of intensity. However, the difference was not significant compared to the other models.

4.2 Reaction-related Model - Mass

Initially, different masses for the binding partner B were tested in the reaction-related model. The weight was also increased to enhance the effect of immobilization. Since the mass of a binding partner has a negative influence on the recovery

rate it was assumed that the recovery curves will show distinct immobile fractions. This expectation was confirmed in the evaluation of the survey.

The results show that a higher mass causes a deceleration of the recovery curves. That is supported by the fact that atoms with a higher mass diffuse slower, as mentioned in the paper by Callister et. al. [2]. As expected, the immobile binding partner causes an immobile fraction, which becomes observable in the FRAP diagrams. This is, as already mentioned, the proportion of the fluorescence intensity that is not completely restored [19].
This fraction is not yet visible at a mass of 10g/mol. In fact, as showen in the results, the graph nearly reaches 100%, although the radius was set to 15Å. The results of the diffusion-related model have shown that the fluorescence recovery cannot be restored entirely at this point. Possible error sources have already been checked, such as calculation errors or incorrect settings. With the data given, the results cannot be clarified.
On the contrary, the immobile fraction is more distinct at a mass of 50g/mol. However, the curve shows a similar development to the diffusion-related simulation with a radius of 15Å, with the exception of a larger amount of time to reach a maximum of almost 90%. This suggests that the diffusion velocity was compromised by B.
The most distinct immobile fraction can be observed at 100g/mol. This is an indication of an (almost) complete fixation of atom B. According to this, the GFP atoms can only diffuse in an unbound state, which significantly slows down the process of FRAP. It has not yet reached 90%, but it can be assumed that this is possible if the simulation is done with more runs. The value of $t_{1/2}$ is also the highest compared to the other cases. Thus, the recovery half-time is also affected by the different masses and is linked with the diffusion rate slowing down.
The evolution in the first 10^4 timesteps - all three diagrams reach 30% with the same speed - cannot be explained by diffusion of GFP, because after $5 \cdot 10^2$ steps the complexing process (of B and GFP) has reached an equilibrium, see Fig. 13-15. So, there is only a small amount of free GFP. To find a possible explanation, further analysis is required.
Another indication of the immobilization is the much longer recovery time (time until the FRAP graphs converge). It is higher than the diffusion-related model, which only took 10^4 timesteps. This can be traced back to the binding partner B, which was not yet present in the first model.

4.3 Reaction-related Model - Network

The aim of the third model was the immobilization of B via building a fixed network and to simulate the effect of different binding and dissociation rates.

To what extend this fixed structure was build can be determined with the help of the observable-diagrams, see attachment Figures 16-20. The diagrams show how fast and how many of the atoms have formed a bond. As already stated in the results, there were no significant differences between these plots, no matter which k_{on} and k_{off} were used. But all of them show that after a small amount of time all the atoms B are bound, i.e., all three binding sites are bound, and only a small proportion is still free. This is an indication for the formation of a network-like structure, only consisting of B. Whether this structure is completely connected and immobilized needs further investigation.
The visualization with *VMD* (Fig. 9) has shown no clear complexes consisting of B. This could be explained by the large number of atoms or other settings in VMD that are needed to represent such a simulation. A test simulation with a lower number of atoms was conducted to demonstrate that the complexing process works (Fig. 21). Like in the mass model, the graphs increase faster at the beginning. Comparing the observable-plots with the FRAP curves reveals that at the time of the bleach, only half of all the atoms B are completely bound, probably because there is still enough free B left to diffuse. However, after a short time, almost all of the *GFP* atoms are bound, which would mean that the diffusion depends more on B than on *GFP* at the time.
Additional signs of immobility are that none of the simulations have reached 90% yet could be verified with a longer simulation, and the correspondingly large immobile fractions that can be observed in all runs.

The results of the k_{off} variations have shown that a small value - $k_{off} = 0.01$ - has the biggest influence of the considered quantities on the recovery rate. It can be assumed that $k_{off} < 0.01$ has an even bigger influence. The probability is much smaller that *GFP* and B dissociate, i.e. most of the *GFP* atoms are fixed and can only diffuse slowly (or not at all). According to this, it has the largest half-time recovery, which suggests that k_{off} has the greatest impact on FRAP.
There were no significant differences in the case of 0.1 and 0.2. However, it would have been expected that 0.2 would reach the maximum intensity faster than 0.1, because it is more likely that B and *GFP* would disassociate and more *GFP* could diffuse.

On the contrary the variation with the binding rate hardly shows any differences for $k_{on} > 0.01$. At 0.1, it takes a little longer before the graph converges, but the maximum intensity is almost the same. In this case, it is more likely that *GFP* and *B* bind, which should affect the diffusion rate negatively, because more *GFP* is bound. $t_{1/2}$ is pretty similar in each simulation, which also indicates that the binding rate has barely an effect on FRAP. However, since only three different k_{on} were used, these results need further investigation with smaller values.
It has to be taken into account that the objective of this thesis is not a life-like simulation of FRAP, but an abstraction of this method and a first insight into a virtual simulation. In addition, examining all possible parameters involved in FRAP or analyzing their impact in combination with each other is beyond the scope of this research. The visualization with *VMD* served only as a first pictorial representation of the results, but were not further considered in the evaluation.
Further adjustments of the method and the performance of additional simulations could provide different results and could be a topic for further research. The FRAP – method could be improved by using a different way of measuring the fluorescence intensity or a different shape of the bleaching area. The analysis of additional parameters could be performed in order to determine the underlying biological system of a FRAP curve without further information, since it is unknown in most cases. From the knowledge how each parameter effects the shape of the curve, one might draw a conclusion which properties the analyzed system has, e.g. in relation to mass or binding and dissociation rate.

5 Summary and Conclusion

The aim of this thesis is to extend the SRSim simulation software with a new method to simulate FRAP measurements, followed by its validation through an analysis of the influence of crucial parameters. These include the radius of the bleaching area, which has to be chosen at the beginning of an experiment and has a great impact on the number of atoms that will be bleached. The mass and size of the binding partner are also important factors and effect the diffusion of the tagged atoms. The binding and dissociation process of two atoms depends on the on- and off-rate. These quantities provide information about the probability that two atoms bind/separate.
Three different simulation models were developed. The diffusion-related model (Section 3.1) was used to demonstrate the functionality of the new method *start_frap_srsim* and to show the effect of different radii. As expected, the radius has an influence on the maximum fluorescence intensity that can be restored. Recovery rate and recovery half-time were only slightly affected.
The second model - reaction-related (Section 3.2) - was used to simulate an immobile fraction in two different ways. First, the immobilization of the binding partner with the help of different masses. The results confirmed that the higher the weight, the larger recovery the half-time. This has demonstrated a possible way to simulate an immobile fraction that is caused by a binding partner that has been almost entirely immobilized by increasing its mass.
In the second case, the immobilization was implemented by building a fixed network consisting of the binding partner. On the one hand, this simulation was used to demonstrate another way of simulating an immobile fraction and, on the other hand, to evaluate the impact of varying binding and dissociation rates of GFP. The result has shown a great impact of the dissociation rate. If it is significantly decreased, the recovery rate is much slower and the recovery half-time larger than with bigger k_{off}. Whereas the results of the binding rate where inconclusive and it seems to have no significant influence. This would be good starting point for further research. To gain more meaningful information one could repeat this simulation with more than three different values. Additionally, in the simulation with different masses a contradiction had occurred, possibly caused by an error in the simulation. At a radius of 15Å the recovered intensity almost reached 100%, which should not be possible due to the expected upper bound calculated in the diffusion model. But with the available data no solution could be found and requires further investigation.
In conclusion, the simulation of virtual FRAP in a spherical volume was successful and it generated realistic recovery curves, which are comparable to life-like experiments, though some unexpected simulation results require further investigation and

more elaborate validation. It is possible to modify this simulation method by choosing different radii, bleaching intensities or parameters that effect the velocity and chemical reactions of an atom. This first insight into the simulation of FRAP could be used in future work to calibrate quantitative rate constants of rule-based models or evaluate FRAP curves with the help of algorithms and determine the underlying biological system, since this is still a major problem in the analysis of FRAP experiments.

Bibliography

[1] Rainer Breitling. What is systems biology? *Frontiers in physiology*, 1:159, 2010.

[2] William D Callister, David G Rethwisch, et al. *Materials science and engineering: an introduction*, volume 7. John wiley & sons, New York, 2007.

[3] Contributor of the Institute for Systems Biology. What is systems biology. https://systemsbiology.org/about/what-is-systems-biology/, 2019. [Online; accessed 5-August-2019].

[4] Holger Conzelmann, Julio Saez-Rodriguez, Thomas Sauter, Boris N Kholodenko, and Ernst D Gilles. A domain-oriented approach to the reduction of combinatorial complexity in signal transduction networks. *BMC Bioinformatics*, 7(1):34, 2006.

[5] Javier Corzo. Time, the forgotten dimension of ligand binding teaching. *Biochemistry and Molecular Biology Education*, 34(6):413–416, 2006.

[6] John Crank et al. *The Mathematics of Diffusion*. Oxford University Press, Oxford, New York, 1979.

[7] Gerd Gruenert. Spatial rule-based simulation of reaction networks. Diplom thesis, Friedrich-Schiller-Universität Jena, 2008.

[8] Gerd Gruenert, Bashar Ibrahim, Thorsten Lenser, Maiko Lohel, Thomas Hinze, and Peter Dittrich. Rule-based spatial modeling with diffusing, geometrically constrained molecules. *BMC Bioinformatics*, 11(1):307, 2010.

[9] Gerd Grünert and Peter Dittrich. Using the srsim software for spatial and rule-based modeling of combinatorially complex biochemical reaction systems. In Marian Gheorghe et.al., editor, *International Conference on Membrane Computing*, pages 240–256, Berlin, Heidelberg, Germany, 2010. Springer.

[10] Lori Redmond Hardy. Fluorescence recovery after photobleaching (frap) with a focus on f-actin. *Current protocols in neuroscience*, Chapter 2:Unit 2.17, 2012.

[11] Leonard A. Harris, Justin S. Hogg, José-Juan Tapia, John A. P. Sekar, Sanjana Gupta, Ilya Korsunsky, Arshi Arora, Dipak Barua, Robert P. Sheehan, and James R. Faeder. BioNetGen 2.2: advances in rule-based modeling. *Bioinformatics*, 32(21):3366–3368, 07 2016.

[12] William S Hlavacek, James R Faeder, Michael L Blinov, Alan S Perelson, and Byron Goldstein. The complexity of complexes in signal transduction. *Biotechnology and Bioengineering*, 84(7):783–794, 2003.

[13] William Humphrey, Andrew Dalke, and Klaus Schulten. Vmd: visual molecular dynamics. *Journal of molecular graphics*, 14(1):33–38, 1996.

[14] Minchul Kang, Charles A Day, Anne K Kenworthy, and Emmanuele DiBenedetto. Simplified equation to extract diffusion coefficients from confocal frap data. *Traffic*, 13(12):1589–1600, 2012.

[15] HG Kapitza, G McGregor, and KA Jacobson. Direct measurement of lateral transport in membranes by using time-resolved spatial photometry. *Proceedings of the National Academy of Sciences USA*, 82(12):4122–4126, 1985.

[16] Hiroaki Kitano. Computational systems biology. *Nature*, 420(6912):206, 2002.

[17] Joseph R Lakowicz. Introduction to fluorescence. In *Principles of fluorescence spectroscopy*, pages 1–23. Springer, Berlin, Germany, 1999.

[18] Angus I Lamond and William C Earnshaw. Structure and function in the nucleus. *Science*, 280(5363):547–553, 1998.

[19] Niklas Lorén, Joel Hagman, Jenny K Jonasson, Hendrik Deschout, Diana Bernin, Francesca Cella-Zanacchi, Alberto Diaspro, James G McNally, Marcel Ameloot, Nick Smisdom, et al. Fluorescence recovery after photobleaching in material and life sciences: Putting theory into practice. *Quarterly reviews of biophysics*, 48(3):323–387, 2015.

[20] Tom KL Meyvis, Stefaan C De Smedt, Patrick Van Oostveldt, and Joseph Demeester. Fluorescence recovery after photobleaching: a versatile tool for mobility and interaction measurements in pharmaceutical research. *Pharmaceutical research*, 16(8):1153–1162, 1999.

[21] Reiner Peters, Jutta Peters, Karl Heinz Tews, and Wolfgang Bähr. A microfluorimetric study of translational diffusion in erythrocyte membranes. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, 367(3):282–294, 1974.

[22] Steve Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of computational physics*, 117(1):1–19, 1995.

[23] Spectrum contributors. Fluoreszenz — Spektrum, an online-lexikon. https://www.spektrum.de/lexikon/biologie/fluoreszenz/25350, 2019. [Online; accessed 1-July-2019].

[24] Stefanie Weidtkamp-Peters, Thorsten Lenser, Dmitri Negorev, Norman Gerstner, Thomas G Hofmann, Georg Schwanitz, Christian Hoischen, Gerd Maul, Peter Dittrich, and Peter Hemmerich. Dynamics of component exchange at pml nuclear bodies. *Journal of Cell Science*, 121(16):2731–2743, 2008.

[25] Wikipedia contributors. Fluorescence — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Fluorescence&oldid=903606129, 2019. [Online; accessed 1-July-2019].

Listings

List of Figures

Appendix

Radius [Å]	10	15	20
Percentage of Reactor [%]	3	11	26
Immobile Fraction [%]	4	10	25
Bleached Atoms [%]	3	9.7	23.5
$t_{1/2}$ [timesteps]	2,490	2,700	2,800

Table 4: Summary of all results of the first simulation model.

Mass [g/mol]	10	50	100
Radius [Å]	15		
Immobile Fraction [%]	1	11	18
Bleached Atoms [%]	9.36	9.6	10
$t_{1/2}$ [timesteps]	9,700	18,000	26,000

Table 5: Summary of all results of the second simulation model, using different masses.

	Default	k_{off} variations		k_{on} variations	
k_{on}	0.01	0.01	0.01	0.02	0.1
k_{off}	0.1	0.2	0.01	0.1	0.1
Bleached Atoms [%]	9.6	9.45	9.65	9.3	9.2
Radius [Å]	15A				
$t_{1/2}$ [timesteps]	20,000	17,000	50,000	18,000	20,000

Table 6: Summary of all results of the third simulation model, using different binding and dissociation rates.

Listing 10: Python Script - Mean Value

```
# script to calculate the arithmetic mean, variance and
# standard deviation of a data set, stored in a table

import statistics
import sys
import os
import re
import math

indize = 0
means = [] # list with arithmetic mean per line
deviation = [] # list of standard deviation per line
variance = [] # list of variance per line
error = [] # list of error rate
```

```
15
16 def percent(liste):
17     """converts the mean value in percent"""
18     help = liste
19     resultsPer = []
20     for el in help:
21         result = (el / help[0])*100
22         resultsPer.append(round(result,2))
23     print(resultsPer)
24     return resultsPer
25
26 def parse(path, out):
27     """Method to calculate arithmetic mean, variance and standard deviation for
           FRAP data"""
28     file = open(path, 'r')
29     lines = file.readlines()
30     # calculate arith. mean for each line and write it into a new file
31     for line in lines:
32         help = line.split() # split by whitespace
33         values=[] # list for values, castet to float
34         for i in help:
35             values.append(float(i)) # cast values from string to float
36         print(values)
37         m = statistics.mean(values) # arithemtic mean
38         varRes = sum([(xi - m)**2 for xi in values]) / (len(values) - 1) # variance
39         devi = statistics.stdev(values)
40         errDevi = devi/math.sqrt(len(values))
41         means.append(round(m,2))
42         variance.append(round(varRes,2))
43         deviation.append(round(devi,2))
44         error.append(round(errDevi,2))
45
46     meanPer = percent(means)
47     #errPer = percent(error)
48     file.close()
49     countRuns = 6000 # number runs for binding and network
50     countRunsDiff = 2500  # number runs for diffusion
51     output = open(out, 'a')
52     output.write("#runs     mean     percentage  variance     deviation" + "\n")
53     for i in range (0, len(means)): # write all means in a file
54         if(i < 2):
55             output.write(str(6000) + "\t" + str(means[i]) + "\t" + str(meanPer[i])
                   + "\t" + str(variance[i]) + "\t" + str(deviation[i]) + "\t" +
                   str(error[i])+ "\n")
56         elif(i >= 2 and i < 10):
57             countRuns+=500
58             output.write(str(countRuns) + "\t" + str(means[i]) + "\t" +
                   str(meanPer[i]) + "\t" + str(variance[i]) + "\t" +
                   str(deviation[i])+ "\t" + str(error[i])+"\n")
59         else:
60             countRuns += 5000
61             output.write(str(countRuns) + "\t" + str(means[i]) + "\t" +
                   str(meanPer[i]) + "\t" + str(variance[i]) + "\t" +
                   str(deviation[i])+ "\t" + str(error[i])+"\n")
62         # diffusion
63         #if(i<2):
64             #output.write(str(2000) + "\t" + str(means[i]) + "\t" +
```

```
                str(meanPer[i]) + "\t" + str(variance[i]) + "\t" +
                str(deviation[i])+ "\t" + str(error[i])+ "\n")
65          #else:
66              #output.write(str(countRunsDiff) + "\t" + str(means[i]) + "\t" +
                    str(meanPer[i]) + "\t" + str(variance[i]) + "\t" +
                    str(deviation[i])+ "\t" + str(error[i])+ "\n")
67              #countRuns+=500
68          # complex
69          #output.write(str(means[i]) + "\n")
70      output.close()
71
72  def complexing_data(path, out):
73      """Method to calculate arithemtic mean for SRSim observables. """
74      file = open(path, 'r')
75      lines = file.readlines()
76      for line in lines: # calculate arith. mean for each line and write it into a
            new file
77          help = line.split() # split by whitespace
78          values=[] # list for values, castet to float
79          for i in help:
80              values.append(float(i)) # cast values from string to float
81          print(values)
82          m = statistics.mean(values) # arithemtic mean
83          means.append(round(m,2))
84      file.close()
85      output = open(out, 'a')
86      for i in range (0, len(means)): # write all means in a file
87          output.write(str(means[i]) + "\n")
88      output.close()
89
90  for arg in sys.argv: # argv = list of commands in the terminal
91      if(arg == '-path'):
92          path = sys.argv[indize + 2]
93          output = sys.argv[indize + 3]
94          print(path)
95          print(output)
96          #parse(path, output)
97          complexing_data(path, output)
98          indize +=3
```

Listing 11: start_frap_srsim.h

```
1  class StartFrapSRSim : protected Pointers
2    {
3    public:
4      StartFrapSRSim(class LAMMPS *lmp);
5     ~StartFrapSRSim();
6
7      void command(int, char **); //command for starting FRAP
8
9     private:
10     AtomVecSRSim *avec;
11     void bleaching(int); //method for bleaching
12     double scalar(int, double []); //method to calculate the scalar
13     };
```

```
14 }
```

Listing 12: start_frap_srsim.cpp

```
1  //
2  //
3  #include "start_frap_srsim.h"
4  #include "atom_vec_srsim.h"
5  #include "atom.h"
6  #include "domain.h"
7  #include "comm.h"
8  #include "error.h"
9  #include "force.h"
10 #include "bond.h"
11 #include "angle.h"
12 #include "dihedral.h"
13 #include "pair.h"
14 #include "neighbor.h"
15 #include "update.h"
16 #include "respa.h"
17 #include "random_mars.h"
18 #include "memory.h"
19 #include "modify.h"
20 #include "compute.h"
21 #include "compute_reapot_atom.h"
22
23
24
25 #include <SRSim/defs.h>
26 #include <assert.h>
27 #include <sstream>
28 #include <string.h>
29 #include <cmath>
30 #include <cstdlib>
31 #include <time.h>
32 #include <stdio.h>
33
34 #include <SRSim/sr_model.h>
35 #include <SRSim/start_state_definition.h>
36 #include <SRSim/defs.h>
37 #include <SRSim/molecule_type_manager.h>
38 #include <SRSim/bng_rule_builder.h>
39 #include <SRSim/names_manager.h>
40 #include <SRSim/molecule_type_manager.h>
41 #include <SRSim/reactant_template.h>
42 #include "fix_srsim.h"
43 #include <SRSim/site_reactant_template.h>
44 #include "lammps_molecule.h"
45
46
47
48 /*#define  FORCE_OF_BOND              "2.0"
49 #define  FORCE_REPULSION            "2.0"
50 #define  FORCE_REPULSION_CUTOFF     "2.5"
51 #define  FORCE_AT_ANGLE             "52.0"*/
52
```

```
53
54 using namespace SRSim_ns;
55
56 namespace LAMMPS_NS {
57
58 StartFrapSRSim::StartFrapSRSim(class LAMMPS *lmp) : Pointers(lmp)
59   {
60   avec = dynamic_cast<AtomVecSRSim*>(atom->avec);
61   }
62
63
64 StartFrapSRSim::~StartFrapSRSim()
65   {
66   printf("Destruction! [StartFrapSRSim::~StartFrapSRSim]\n");
67   }
68
69
70 void StartFrapSRSim::command(int argc, char **argv )
71   {
72     int countOn = 0; //variables to count the atoms in the sphere, fluorescent
73     int countOff = 0; // bleached
74     AtomVecSRSim              *avecsrsim  =
          dynamic_cast<AtomVecSRSim*>(atom->avec); // cast of a LAMMPS atom to a
          SRSim atom
75     int nlocal = atom->nlocal; //total number of atoms
76
77     if (argc < 1) {error->all("StartFrapSRSim:: command: start_frap should have at
          least 1 Paramater for a command {bleaching, count} and parameters for more
          setting {x-coordinate, y-coordinate, z-coordinate, radius, intensity}");}
78
79     if (argc == 6 ){ //bleach command
80
81         double x = atof(argv[1]); //x coordinate for sphere
82         double y = atof(argv[2]); // y coordinate
83         double z = atof(argv[3]); // z coordinate
84         int r = atoi(argv[4]); //radius of the sphere
85         double intensity = atof(argv[5]); //intensity or probability for bleaching
              [0,1]
86
87         double k[3] = {x, y, z}; // initialize sphere's center coordinates
88
89
90         if (intensity < 0 || intensity > 1) error->all("Illegal SRSim command:
              intensity too small has to be between 0 and 1");
91         //printf("number of  Atoms %d\n", nlocal);
92
93
94         if (strcmp(argv[0], "bleach" ) == 0) { // command for bleaching
95
96
97             for (int j=0; j < nlocal; j++){ // iterate through all atoms j
98                if(atom->type[j] == 1){
99
100                   if(distance3(atom->x[j], k) < r){
101                       double random = (rand() % 100); // generate random number
                            between 0 and 100
102                       //printf("random number: %f\n", random);
```

```
103                         if(random < (intensity*100)) {bleaching(j);}
104                         //if this number is smaller than the Intensity, this atom
                                will be bleached
105                     }
106                 }
107             }
108         }
109         else
110              error->all("StartFrapSRSim::command: start_frap didn't recognize
                    command!");
111
112     }
113     else if(argc == 5){
114
115         int r = atoi(argv[4]);
116         double x = atof(argv[1]);
117         double y = atof(argv[2]);
118         double z = atof(argv[3]);
119
120         double k[3] = {x, y, z}; //initialize sphere
121
122         if(strcmp(argv[0], "count" ) == 0){ //command for counting
123
124             for (int j=0; j < nlocal; j++){//same as above
125                 if(atom->type[j] == 1){ //only count atom of type 1
126
127                     if(distance3(atom->x[j], k) < r){
128                         if(avecsrsim->site_modified[j][0] == 0){
129                             countOn += 1;
130                             //count all atoms which where bleached and are located
                                    in the sphere
131                         }
132                         else{
133                             countOff += 1;
134                             //count all atoms which are still fluorescent and are
                                    located in the sphere
135                         }
136                     }
137
138                 }
139             }
140             printf("GFP: nsteps: %d ntimestep: %d leuchten: %d gebleached: %d\n",
                      update->nsteps, update->ntimestep, countOn, countOff);
141
142         }
143         else
144             error->all("StartFrapSRSim::command: start_frap didn't recognize
                    command!");
145     }
146
147     else if (strcmp(argv[0], "sites") == 0) {
148          AtomVecSRSim             *avecsrsim =
                dynamic_cast<AtomVecSRSim*>(atom->avec);
149
150         for (int i=0; i < atom->nlocal; i++){
151
152             int numSites = avecsrsim->type2numSites[atom->type[i]]; //
```

```
153
154             for(int j=0; j<numSites; j++){
155                 printf("atom type=%d,     number Sites=%d\n",atom->type[i],
                        numSites);
156                 printf("sites of a atom: i=%d j=%d, modified=%d, bond_id=%d,
                        other_side=%d, tag=%d\n", i, j,
                        avecsrsim->site_modified[i][j], avecsrsim->site_bond_id[i][j],
                        avecsrsim->site_other_site[i][j],
                        avecsrsim->site_bound_tag[i][j]);
157             }
158         }
159     }
160
161     else if(strcmp(argv[0], "coords") == 0){
162         for (int i=0; i < nlocal; i++){ //atom->nlocal number of atoms
163             printf("x = %f y = %f z = %f\n", atom->x[i][0], atom->x[i][1],
                    atom->x[i][2]);
164             //atom->x[i][j] contains coordinates j (x,y,z) of atom i
165         }
166     }
167
168     else
169          error->all("StartFrapSRSim::command: start_frap didn't recognize
                command!");
170
171   }
172
173 /**Method for bleachig an atom*/
174 void StartFrapSRSim::bleaching(int t){ //t is the current atom, which position was
        checked
175     AtomVecSRSim         *avecsrsim  = dynamic_cast<AtomVecSRSim*>(atom->avec);
176
177     int newModif = 1;
178     //set site to 1 = unmodified, meaning, this atom was bleached and is no longer
            fluorescent
179     avecsrsim->site_modified[t][0] = newModif;
180
181     avecsrsim->updateSubgraphTemplateData(t); //has to be updated
182
183 }
184
185
186 /**Method for calculating the distance between two vectors*/
187 double StartFrapSRSim::distance3(double a[], double k[]){
188
189     double x = abs(k[0]-a[0]); //calculate distance for each entry in vector -->
            absolute value
190     double y = abs(k[1]-a[1]);
191     double z = abs(k[2]-a[2]);
192
193     double abstand[3] = {x, y ,z}; //vector of distance
194
195     double skalar = 0;
196     double produkt;
197     for(int i=0; i < 3; i++) //calculate scalar of distance vektor
198     {
199         produkt = abstand[i]*abstand[i];       // multiply values
```

```
200         skalar+=produkt;              //add up
201     }
202     double wurzel = sqrt(skalar);
203     return wurzel;
204 }
205
206
207
208
209
210
211 /**
212  * void StartFrapSRSim::command(int argc, char **argv)
213   {
```

Listing 13: style_user.h

```
1  ...
2
3  #ifdef CommandInclude
4    #include "start_state_srsim.h"
5    #include "runmodif_srsim.h"
6    #include "start_frap_srsim.h"
7  #endif
8
9  #ifdef CommandClass
10   CommandStyle(start_state_srsim,StartStateSRSim)
11   CommandStyle(runmodif_srsim,RunModifSRSim)
12   CommandStyle(start_frap, StartFrapSRSim)       //new command is added
13 //  CommandStyle(sss,StartStateSRSim)
14 #endif
15
16 ...
```

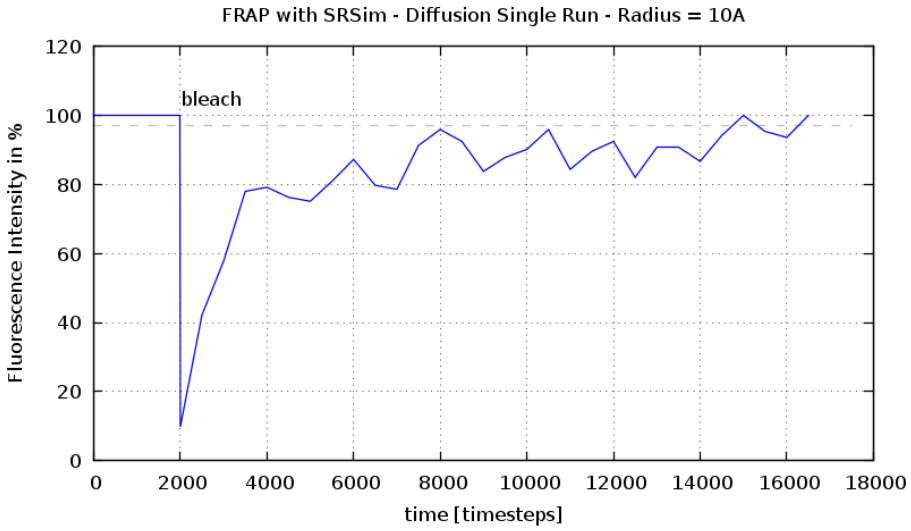

Figure 10: **FRAP Curve of a Single Simulation Run**. This figure illustrates the recovery curve of a single run of the diffusion-related model, with a radius of 10Å.

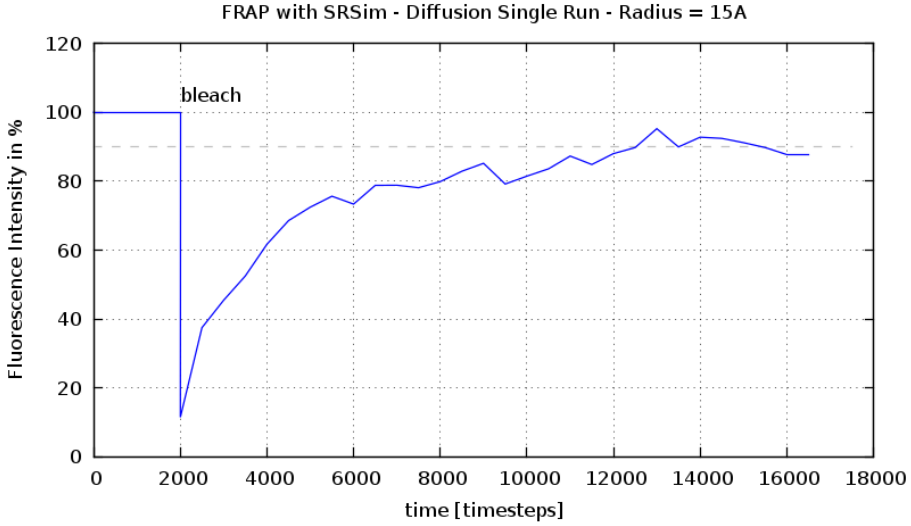

Figure 11: **FRAP Curve of a Single Simulation Run**. This figure illustrates the recovery curve of a single run of the diffusion-related model, with a radius of 15Å.

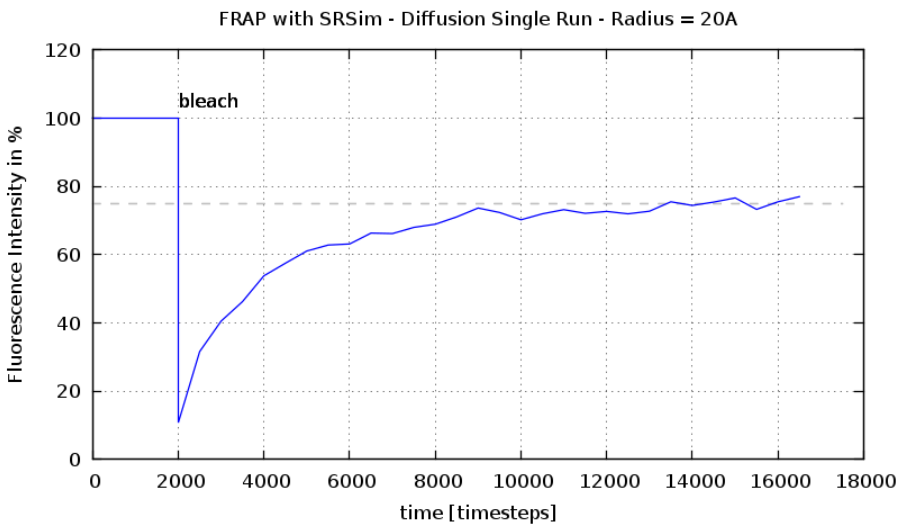

Figure 12: **FRAP Curve of a Single Simulation Run**. This figure illustrates the recovery curve of a single run of the diffusion-related model, with a radius of 20Å.

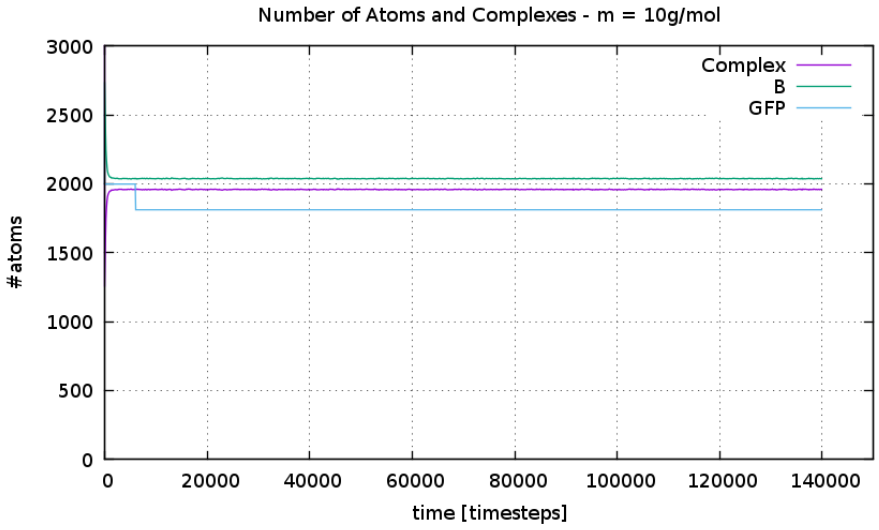

Figure 13: **Particle Quantity Trajectory**. This figure illustrates the number of atoms and complexes build in the simulation with a mass of 10g/mol. 'Complex' denotes the number of particles B that have their site 'b' bound. It can be observed that this value increases rapidly in the beginning and levels out before the bleach occurs.

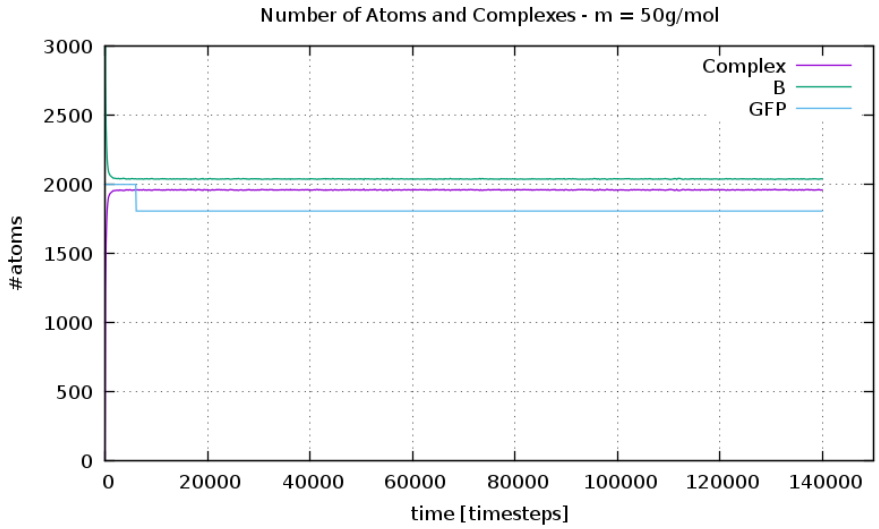

Figure 14: **Particle Quantity Trajectory**. This figure illustrates the number of atoms and complexes build in the simulation with a mass of 50g/mol. 'Complex' denotes the number of particles *B* that have their site 'b' bound. It can be observed that this value increases rapidly in the beginning and levels out before the bleach occurs.

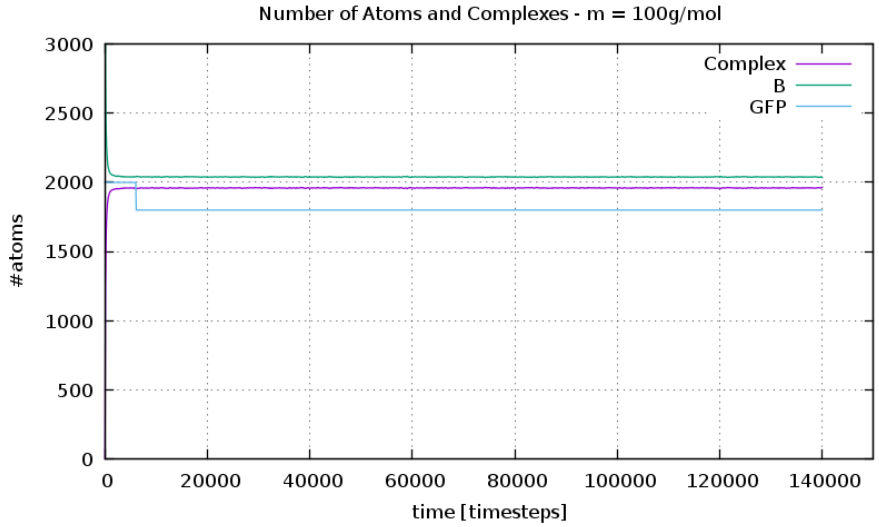

Figure 15: **Particle Quantity Trajectory**. This figure illustrates the number of atoms and complexes build in the simulation with a mass of 100g/mol. 'Complex' denotes the number of particles *B* that have their site 'b' bound. It can be observed that this value increases rapidly in the beginning and levels out before the bleach occurs.

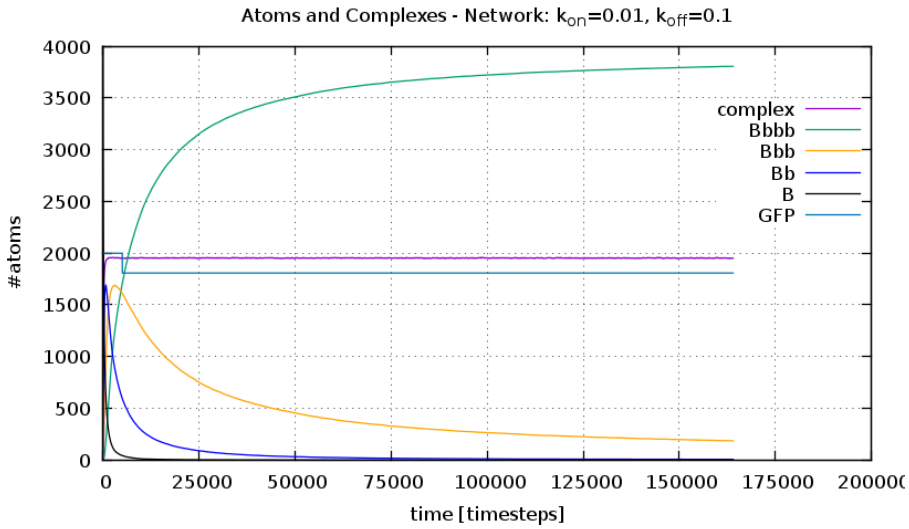

Figure 16: **Particle Quantity Trajectory**. This figure illustrates the number of atoms and complexes build in the simulation with $k_{on} = 0.01$ and $k_{off} = 0.1$. 'complex' denotes the number of particles B that have all their sites 'b' bound. It can be observed, that this value increases in the first 20.000 timesteps. At the time of bleach around 1700 atoms are completely bound. According to that the number of partially bound B ('Bbbb', 'Bbb', 'Bb') decreases rapidly.

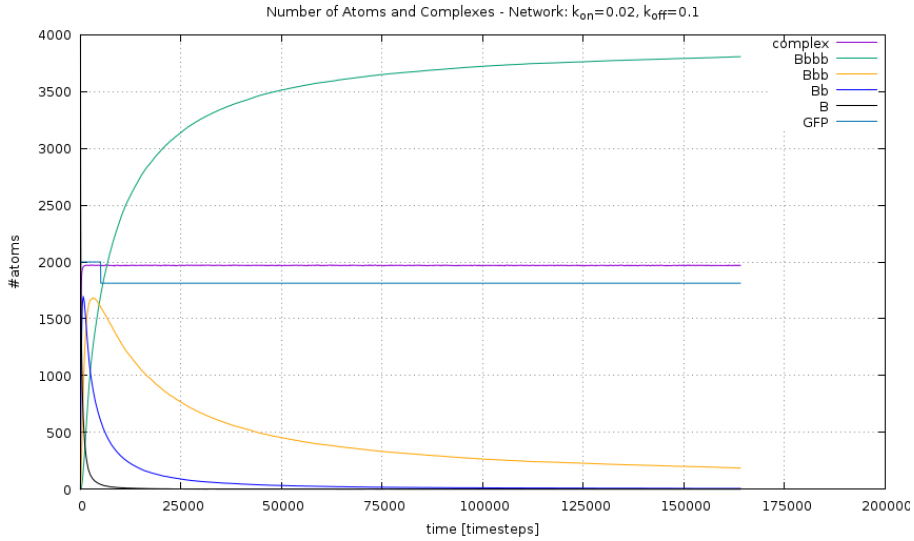

Figure 17: **Particle Quantity Trajectory**. This figure illustrates the number of atoms and complexes build in the simulation with $k_{on} = 0.02$ and $k_{off} = 0.1$. 'complex' denotes the number of particles B that have all their sites 'b' bound. It can be observed, that this value increases in the first 20.000 timesteps. At the time of bleach around 1700 atoms are completely bound. According to that the number of partially bound B ('Bbbb', 'Bbb', 'Bb') decreases rapidly.

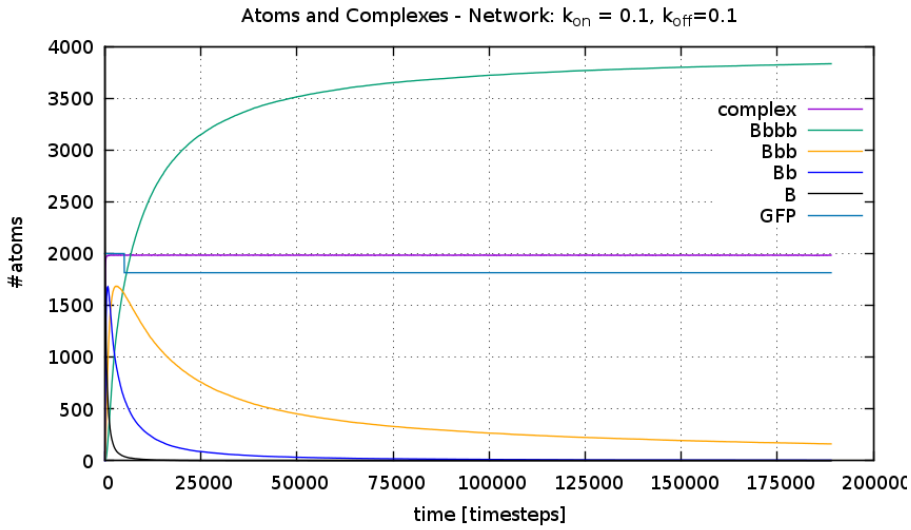

Figure 18: **Particle Quantity Trajectory**. This figure illustrates the number of atoms and complexes build in the simulation with $k_{on} = 0.1$ and $k_{off} = 0.1$. 'complex' denotes the number of particles *B* that have all their sites 'b' bound. It can be observed, that this value increases in the first 20.000 timesteps. At the time of bleach around 1700 atoms are completely bound. According to that the number of partially bound *B* ('Bbbb', 'Bbb', 'Bb') decreases rapidly.

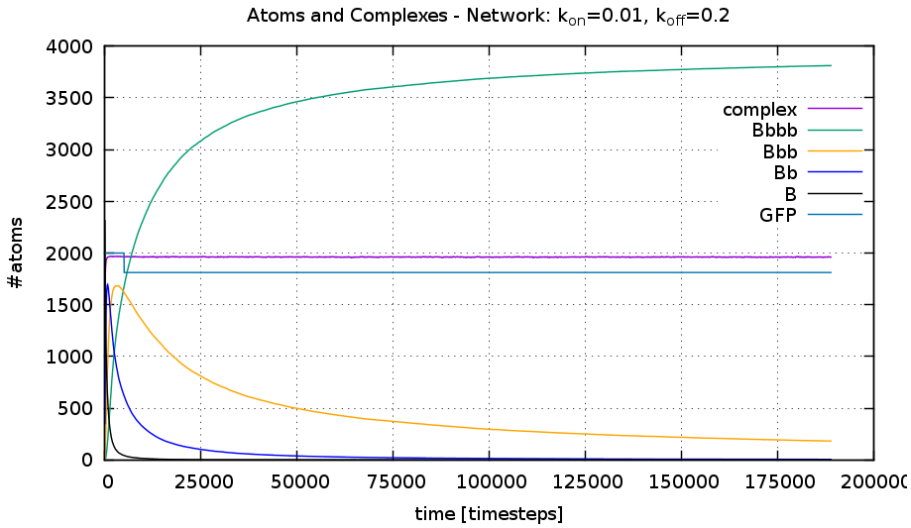

Figure 19: **Particle Quantity Trajectory**. This figure illustrates the number of atoms and complexes build in the simulation with $k_{on} = 0.01$ and $k_{off} = 0.2$. 'complex' denotes the number of particles *B* that have all their sites 'b' bound. It can be observed, that this value increases in the first 20.000 timesteps. At the time of bleach around 1700 atoms are completely bound. According to that the number of partially bound *B* ('Bbbb', 'Bbb', 'Bb') decreases rapidly.

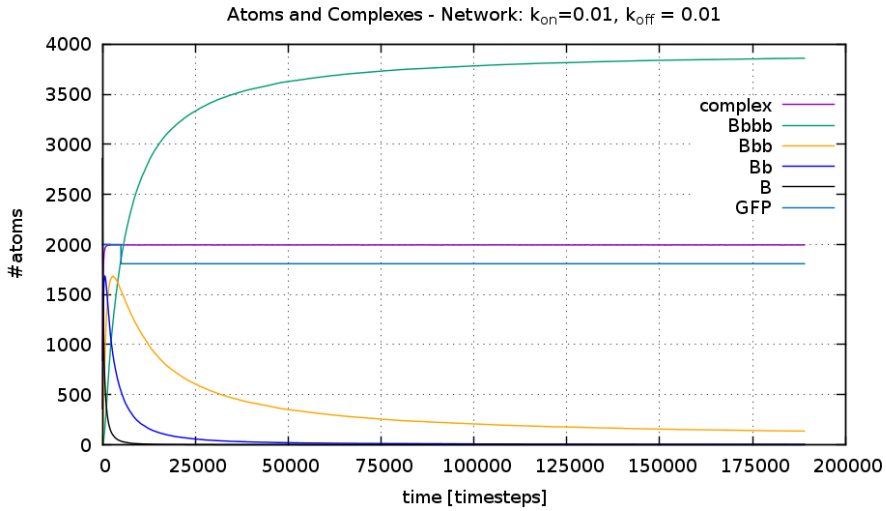

Figure 20: **Particle Quantity Trajectory**. This figure illustrates the number of atoms and complexes build in the simulation with $k_{on} = 0.01$ and $k_{off} = 0.01$. 'complex' denotes the number of particles B that have all their sites 'b' bound. It can be observed, that this value increases in the first 20.000 timesteps. At the time of bleach almost 2000 atoms are completely bound. According to that the number of partially bound B ('Bbbb', 'Bbb', 'Bb') decreases rapidly.

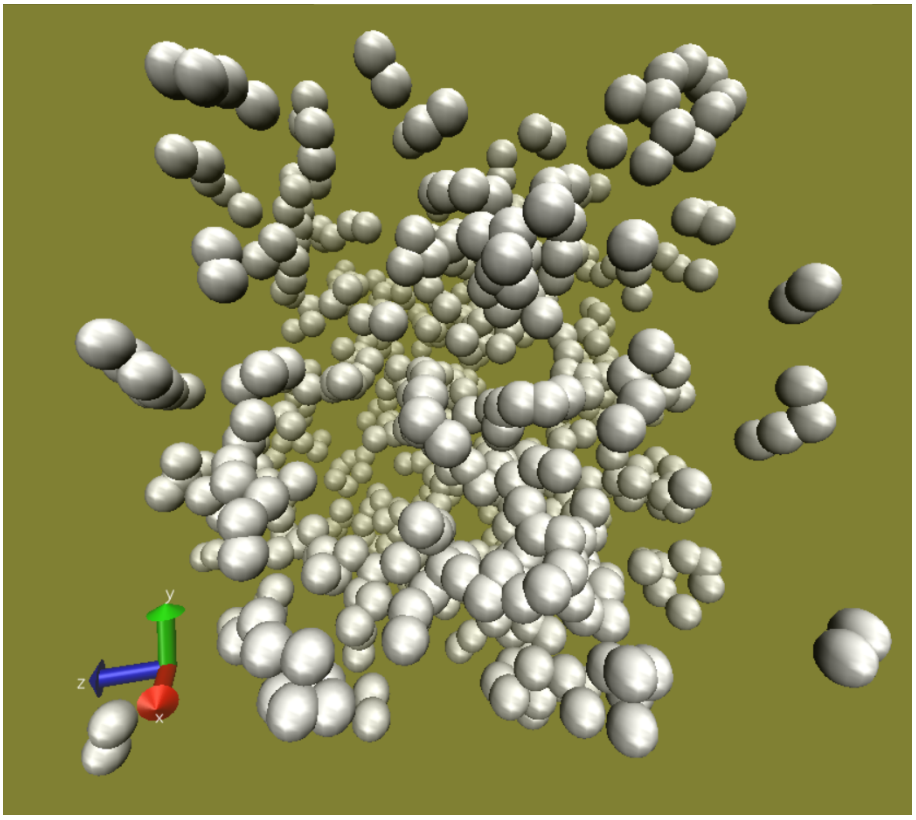

Figure 21: **3D-Visualization of the Test Model**. The image shows the result of a test model similar to the network simulations but with less atoms. Network-like structures and Dimers can be observed.

Eidesstattliche Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

$Unterschrift:$ $Ort, Datum:$