PCT monitoring with immunoassays using f-BPM

van Wetten, Stan P.M.

Award date:
2021

Molecular Biosensing for Medical Diagnostics

3MA60

PCT monitoring with immunoassays using f-BPM

Stan van Wetten 0949191

Committee: Dr. ir. A. M. de Jong
Dr. P. Zijlstra
Dr. ir. P. A. H. Mutsaers

Eindhoven, October 2021

Abstract

In this paper the possibilities are explored to use a Free Bio-sensing by Particle Motion (fBPM) immunoassay to detect low concentrations of Procalcitonin (PCT). The antibody pair PPC3-B27A3 has shown positive response for a PCT concentration as low as 100 pM. This is useful in the prevention and treatment of bacterial infections and eventually sepsis. A PCT concentration of 100 pM is when local infections start to become more severe, thus when treatment is due. This paper also looks at the post-processing of experimental data and compares the current thresholding method with a recently designed post-processing technique, based on deep learning. During experiments an artifact in the imaging software has been found. We found that fBPM immunoassays are prone to agglomeration of detection particles. This artificially increases the activity and lowers the state lifetime. An extra step in the post-processing has been implemented to filter out this agglomeration, with success, making the filtered results more reliable.

Contents

1 Introduction

The goal of this project is to be able to continuously measure the Procalcitonin (PCT) concentration in the blood of hospitalized patients. In hospitals it can be a struggle to distinguish a viral infection from a bacterial infection. PCT is chosen as bio marker because it is a good marker of bacterial infection. Healthy people have a very low concentration of PCT in their blood (4 pM), as do patients with a viral infection[1]. The PCT concentration however increases significantly when the patients have a bacterial infection. There is a close correlation between the PCT concentration and the severity of inflammation. The concentration regime for this project is 4 to 160 pM [2]. The concentration of PCT in healthy individuals is around 4 pM, when local infections occur the PCT concentration is around 40 pM and the onset of sepsis is reflected by a PCT concentration of 160 pM, at 800 pM of PCT the patient is likely to enter septic shock. We intend to prevent sepsis from happening, so the relevant regime would be between 40 to 800 pM of PCT. If an increase in PCT concentration can be observed early, treatment can be started earlier, improving the patient's health.

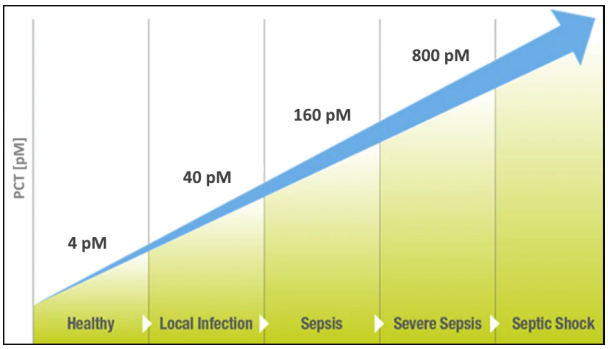

Figure 1.1: Graphical illustration of the relation between PCT concentration in blood and the severity of infection.

The fact that PCT concentrations increase with bacterial infection, but not with viral infection gives PCT diagnoses an edge over other bio markers. This allows for more effective use of antibiotics, as a viral infection can not be combated by the usage of antibiotics and only increase antimicrobial resistance. As a result of drug resistance, antibiotics and other antimicrobial medicines become ineffective and infections become increasingly difficult or impossible to treat[3], increasing the risk of disease spread, severe illness and death.

In order to be able to measure PCT accurately, this research is set up to use a new bio-sensing technique called free bio-sensing by particle motion (fBPM). This is an unobtrusive technique which will hopefully be used alongside other vital signs of patients in a medical ward. The eventual end product will work by drawing a little bit of blood and running that through a fBPM setup, the PCT concentration will be given, either real-time or at intervals.

This paper will focus on improving the accuracy of the fBPM system by comparing different post-processing techniques. The effectiveness of the current post-processing method, thresholding the diffusion coefficient, is compared to a novel deep learning model. On top of that this research will look into filtering out experimental artefacts.

2 Theoretical Background

2.1 Brownian Motion

When looking at particles in the length scales of microns or smaller one will notice that nothing stands still. Particles will move up and down left and right, in no particular direction. This phenomena is called Brownian motion (named after Robert Brown), and is caused by molecules bumping into another particles, which in turn bump into other particles. This happens for medium particles, like water molecules in water, but also with other particles dispersed in a medium. When a single particle is dispersed in a medium, the trajectory of this particle is determined by the collisions this particle makes with incoming medium particles. These collisions are random and discrete. As long as this particle is small enough, its motion can be defined as a 3D random walk.

This random walk can be defined as a normal distribution around $r = 0$, r being the radial distance from the origin, with a variance $\sigma^2 = 2dDt$, according to Einstein's theory. Here D is the diffusion coefficient, d the dimension From this the second moment can be calculated to be $\langle r^2 \rangle = 2dDt$. This means that the value of $\langle r^2 \rangle$ scales linearly with time, if the measured value of $\langle r^2 \rangle$ does not, that would indicate confinement or forced motion. From the Stokes-Einstein equation, the value of the Diffusion coefficient can be calculated, as long as the flow is not turbulent and the particles are assumed smooth, spherical and non-interacting. The Stokes-Einstein equation yields:

$$D = \mu k_B T = \frac{k_B T}{6\pi\eta r}. \tag{2.1}$$

Where μ is the mobility, k_B the Boltzmann coefficient, T the temperature (in Kelvin), η the dynamic viscosity of the medium and r the particle radius. These are all experimental parameters. Under typical experimental conditions used in this research, this would give an theoretical diffusion coefficient of approximately $D = 0.5\mu m^2 s^{-1}$. This diffusion coefficient decreases however as the particle is in close proximity to the substrate surface, due to the no slip condition at the interface between substrate and medium. This can amount up to a factor 3 in lateral diffusion [4].

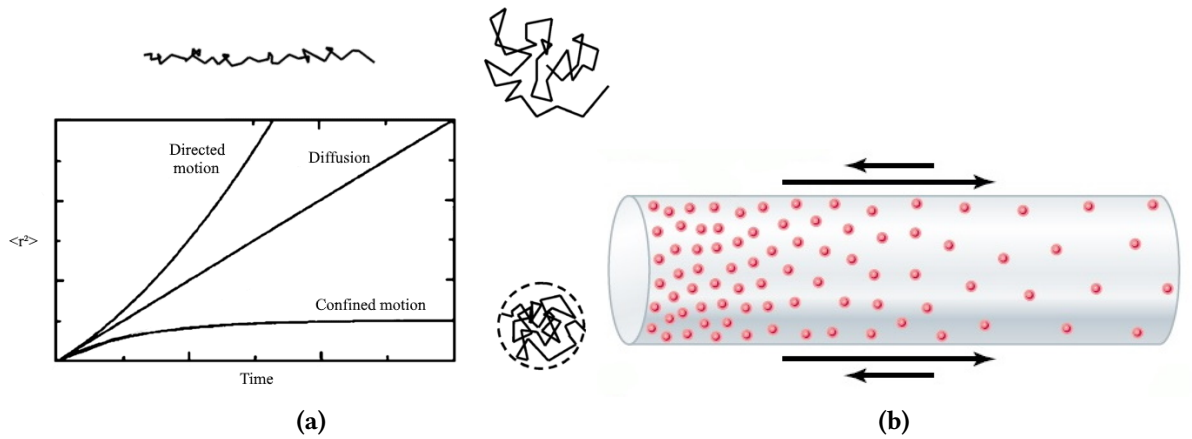

Figure 2.1: a) The dependence of D on $\langle r^2 \rangle$ as a function of time, along with illustrations of the three major types of diffusive behavior. b) Illustration of how the individual collisions make up the macroscopic manifestation of diffusion.

Diffusion is a direct consequence of Brownian motion. Consider a thought experiment where a container is half filled with particles, while the other half is empty. Diffusion would state that once an equilibrium is reached, the particles will be spread homogeneously. This can be understood by thinking of the collisions the particles make on the filled side of the container. The outer particles would than only be receiving collisions from the

high concentration side, resulting in a net force driving them outwards. This is visualized in 2.1b. Similarly, particles with a higher temperature vibrate more, causing the temperature to spread homogeneously.

These collisions are discrete events, however, the larger the particle, the less discrete this appears. When the particle becomes too large, the collisions pushing it into one direction are counterbalanced by collisions on the other side pushing it in the opposite direction. Eventually these minor fluctuations in locations are insignificant compared to the particle size. This can be seen in the mean squared displacement of a Brownian particle, which is proportional to D, and thus via Equation 2.1, scales with $\frac{1}{r}$.

2.2 Procalcitonin

This research is set up to find the concentration of procalcitonin (PCT) in blood of hospital patients. PCT is the precursor of calcitonin, which helps regulate the calcium levels in blood. It is quite a small protein, made up of 116 amino acid residues and weighing in at 13 kDa of molecular weight[5]. It consists of three parts, premature calcitonin, surrounded by the N-terminus and katacalcin, see Figure 2.2. These last two are cleaved off when PCT is converted into calcitonin. In a healthy person, the PCT is almost completely converted into calcitonin and therefore normally only a very low concentration is present. This changes however when a bacterial infection arises. An increase in PCT levels can signal immune system activation, which is often a result of an infection. Infections can be caused by multiple infectious agents such as parasites, bacteria or viruses.

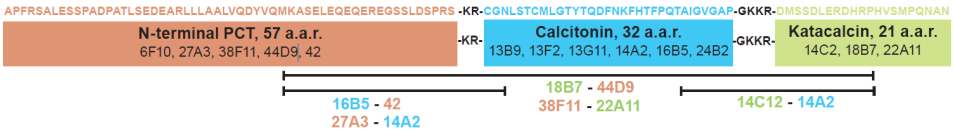

Figure 2.2: Amino acid sequence of human procalcitonin , epitope specificities and pairs of mAbs recommended for PCT sandwich immunoassay

Inflammation is one of the first responses of the immune system to infection or irritation. Inflammation is stimulated by chemical factors released by injured cells and serves to establish a physical barrier against the spread of infection, and to promote healing of any damaged tissue following the clearance of pathogens. If the immune response is not adequate enough, the patient can develop sepsis, which can lead to organ failure and death, if not treated properly. This is especially true for the more vulnerable patients, such as elderly and neonates.

Nowadays, infections can be treated rather efficiently. Depending on the type of infection (the type of pathogen causing the infection) the adherent medication can be administered. But in order to find the right medication, the type of infection does need to be diagnosed. Here time is of the essence, the faster the medication is administered, the less severe the symptoms. Bacterial and viral infections can both cause the same kinds of symptoms, it can be difficult to distinguish which is the cause of a specific infection. This is where PCT plays a role. Measuring the PCT concentration is especially useful in detecting bacterial infections, since the PCT concentration is closely correlated to the severity of inflammation. While the PCT concentration hardly changes in the case of a viral infection [6]. On top of that is the concentration of PCT in the blood of healthy individuals much lower (4pM) than that of bacterial infection patients (40 - 160 pM), meaning there is a high signal to noise ratio.

2.3 Bio-sensing by Particle Motion

The method of measuring the PCT concentration chosen in this study is called free bio-sensing by particle motion (fBPM), the free referring to free diffusion. Bio-sensing by particle motion is a fancy way of saying "extracting information by looking at the movement of particles". This technique is based on Brownian motion and is in a way very similar to tethered particle motion (TPM).

2.3.1 Tethered Particle Motion

TPM is a bio-sensing method where a large bead is connected to a substrate via a tether (often DNA or a polymer), in order to study the behavior of said tether. The tether itself is oftentimes to small, or thin, to be able to properly see with regular microscopes, whereas the bead is not. To this extend, the location of the bead is determined. In the case of TPM, the bead does not display Brownian motion, but rather confined motion, given by Equation 2.2:

$$\langle r^2 \rangle = R_{conf}^2 (1 - e^{-\frac{t}{\tau}}). \tag{2.2}$$

Where the mean squared radial displacement $\langle r^2 \rangle$ starts out as Brownian motion, but starts to feel the effects of confinement after a characteristic confinement time τ, see also 2.1a. Eventually to converge to the square root of the confinement radius R_{conf}^2, in this case the radius of the tether. By equating the diffusive speed of a confined particle at $t = 0$ to the diffusive speed of a free particle, an expression for τ can easily be found:

$$\left.\frac{\partial \left\langle r_{conf}^2 \right\rangle}{\partial t}\right|_{t=0} = \frac{R_{conf}^2}{\tau} = \left.\frac{\partial \left\langle r_{BM}^2 \right\rangle}{\partial t}\right|_{t=0} = 2dD \tag{2.3}$$

$$\tau = \frac{R_{conf}^2}{2dD}. \tag{2.4}$$

A scatter plot of the projection of its 2D coordinates over time typically yields a spherical image with a radius. This radius is $\sqrt{l_{\text{tether}}^2 + 2l_{\text{tether}}R_{\text{bead}}}$ (Pythagorean theorem), where the length of the tether is l_{tether}. If the tether binds to the substrate, it effectively shortens its length, reducing the radius of the scatter plot while still remaining circular. Whereas when the bead is to be bound by two tethers, the confinement region of the bead is limited to the overlap of two circles. This results in a stripe like pattern. This is useful to distinguish whether a bead is bound to another tether or the substrate, when its region of confinement decreases.

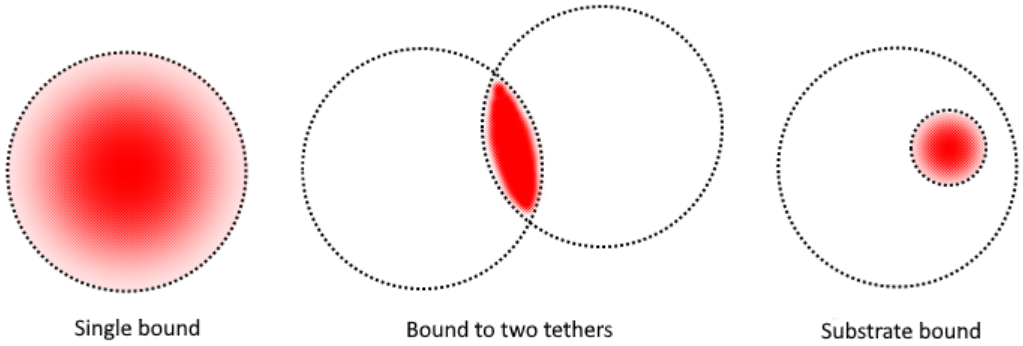

Figure 2.3: The three different states the TPM system can be in, animated as the measured center of the bead after a certain time. On the left the normal state, the bead is confined by the tether; in the middle the bead is confined to a stripe-like pattern due to the overlap in confinement of two tethers; on the right the bead is bound to the substrate.

2.3.2 Free Particle Motion

fBPM is in essence very similar to TPM, the difference being that the bead is not necessarily bound to a tether. The starting position of fBPM is with freely diffusing particles in a flow cell. This flow cell contains a surface on which binders are homogeneously dispersed, connected to the substrate on one end, free on the other. Similar to a long-pile carpet, but not so dense. A schematic overview is given in Figure 2.4

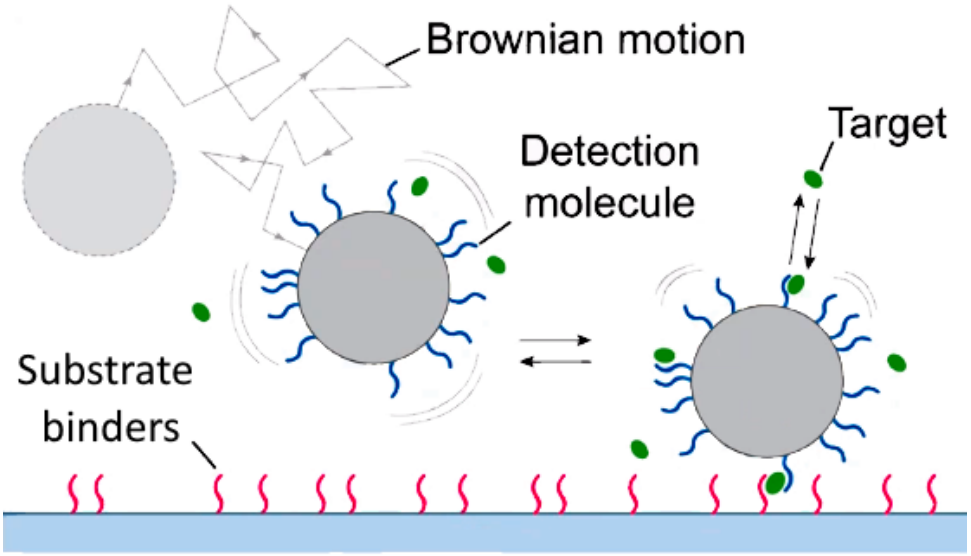

Figure 2.4: A simple illustration of how fBPM works

The particles in the flow cells display Brownian motion when they are unbound, but once they get close to a binder, they can bind. The bead motion changes into confined motion, like in a TPM system, once it is bound to a binder. The free diffusion changes into circular motion. If a stripe like pattern is seen, like in Figure 2.3, the bead is clearly bound by two binders. Likewise a smaller circle or dot might indicate a that it is stuck to the substrate. This way allows one to classify beads into categories: unbound, single bound state, double bound state, and stuck. As well as the bound state life times of said states. All of this is possible by only looking through a microscope, determining the center of beads and tracing the motion patterns of these beads.

In order to bind to a binder, a bead must diffuse downwards to the bottom of the flow cell, and approach the substrate. The decrease in diffusion coefficient when near a planar surface, mentioned in subsection 2.1, means that not only the $\langle r^2 \rangle$ decreases, but also the variation on the measured diffusion coefficient. Because of this it is often difficult to distinguish single and double bound states. That is why the stripe-like patterns are a extremely useful indication of double bound states.

The advantages of this method is that it requires a rather simple (and thus cheap) experimental setup, basically only a microscope and a PC are required. The field of view of the microscope is large enough to capture several hundreds of particles simultaneous, this makes measurements more reliable, since there is a higher statistical population, without removing the possibility to look at individual particle motion and behavior.

2.4 Antibodies

Antibodies, also known as immunoglobulins, are proteins used by the immune system to identify and neutralize foreign objects such as pathogenic bacteria and viruses. The Y-shaped molecule composed of four poly-peptides: two heavy chains (H) and two light chains (L). The two tips of the Y-shape show considerable variation in amino acid composition and are referred to as the variable (V) regions to distinguish them from the relatively constant (C) regions. Both the heavy and the light chain consist of a variable domain, respectively HL and VL, and constant domains, CH and CL, see figure Figure 2.5. Together the VH and VL make up a paratope (analogous to a lock), this is the part of an antibody which recognizes and binds to one particular epitope (analogous to a key) on an antigen, allowing these two structures to bind together with precision. Each arm of the Y-shaped antibody has an identical paratope at the end.

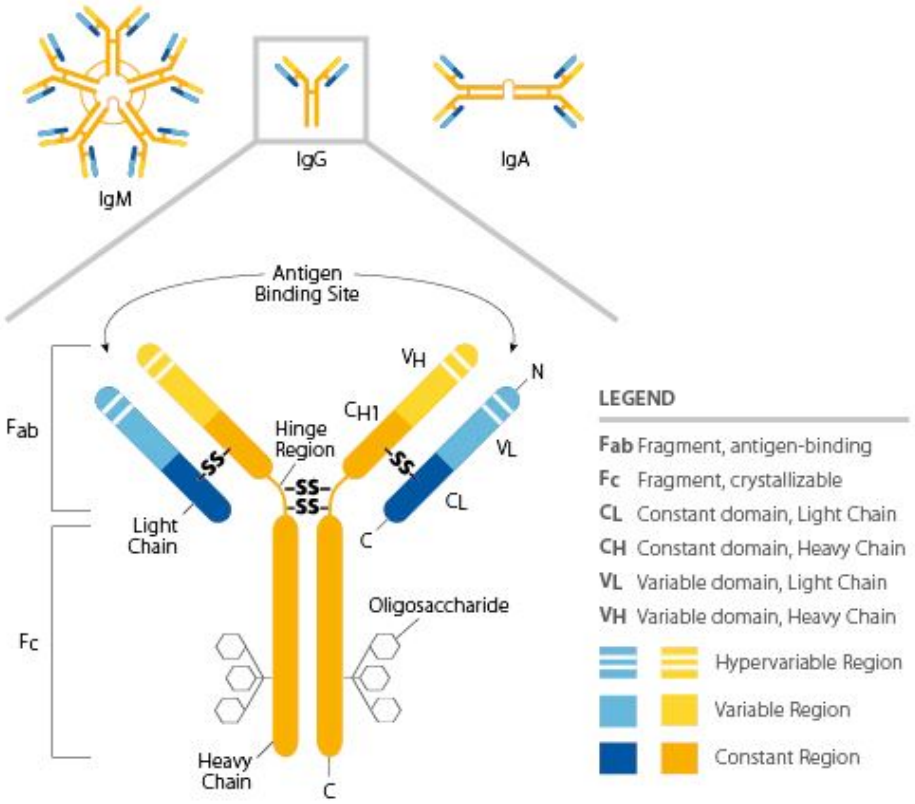

Figure 2.5: The generalized struction of IgG

The Y-shape is split into two by the hinge region, this held together by disulfide bonds and is flexible in nature, this allows the distance between the paratopes to vary. The CH domain beneath the hinge is called the Fc region. This region can bind to the receptor present on a phagocyte, which is essential for phagocytosis (the process of ingestion of pathogens). This is one of the main ways of the immune system to remove pathogens.

There are five different primary classes of antibodies, which are distinguished by the type of heavy chain they possess. These differences allow the different classes to function in different types and different stages of immune responses. The antibodies that can bind to PCT are all part of the immunoglobulin class IgG, the most common type of antibody found in the blood circulation. Because of its relative abundance and excellent specificity toward antigens, IgG is the most commonly used antibody in research and clinical diagnostics.

The antibody classes can be further divided into subclasses, based on minor differences in the heavy chain type of each Ig class. In humans there are four subclasses of IgG: IgG1, IgG2, IgG3 and IgG4 (numbered in order of decreasing concentration in serum). The subclasses differ in the number of disulfide bonds and the length and flexibility of the hinge region[7][8].

In this research all of the used antibodies are of the IgG1 or IgG2a subclass. Where generally speaking IgG1 has a high Fc affinity and IgG2a has an extremely low Fc affinity. But, the IgG affinity to Fc receptors is specific to individual species as well as the class. The structure of the hinge regions contributes to the unique biological properties of each of the four IgG classes. Even though there is about 95% similarity between their Fc regions, the structure of the hinge regions is relatively different.

There are two types of antibodies available that can be picked depending on the applicational requirements: polyclonal and monoclonal. Polyclonal antibodies are made using several different immune cells. They will have the affinity for the same antigen but different epitopes, whereas monoclonal antibodies are made using identical immune cells that are all clones of a specific parent cell, and thus only bind to one specific epitope corresponding to their paratope. Nowadays it is possible to produce monoclonal antibodies that bind to any epitope desired.

2.4.1 Affinity

Antibody affinity is defined as strength of the binding interaction between antigen and antibody. It depends on the closeness of the stereochemical fit between antibody sites and antigen determinants, the size of the area of contact between them, and the distribution of charged and hydrophobic groups. In stable condition, where the associated form of the antigen and antibody is favored, the antibody is referred to as being of higher affinity.

When the affinity is higher, the time bound is longer, making it possible to detect lower concentrations of PCT. This is at a cost of reversibility. If the binding strength is too high, the PCT will no longer unbind. This would make it impossible to measure a decrease in PCT concentration. In the end we would like to be able to continuously measure the PCT concentration in blood, so not only increase but also fluctuation. Polyclonal antibodies are less sensitive to pH or buffer changes, even to antigen changes. And because of them being able to bind to more than one epitope, they can help amplify the signal from target protein even with low expression level. The specificity however decreases, as well as the batch-to-batch consistency. As polyclonal antibodies can bind to any epitope, it is impossible to predict to which it will bind on beforehand, making the affinity impossible to predict.

2.4.2 Immunoassays

In this project f-BPM is applied in an immunoassay. This is a method of measuring the presence of certain molecules, by use of antibodies. In the case of immunoassays, the binders are antibodies working as a pair, a substrate and a detection antibody. These antibodies are chosen such that they both bind to PCT, but not to each other. The detection antibody is attached to the surface of the bead, while the substrate antibody is physisorbed to the substrate's surface. After the antibodies are introduced, both the surface of the bead and that of the substrate are blocked, to remove residual interaction. The bead with the detection antibody will find and bind to the PCT molecules. This will than travel as a compound, until it meets and binds to a capture antibody, immobilizing the bead. This confinement can be measured. This way the binding frequencies and lifetimes can be measured. A schematic of such a structure can be found in Figure 2.6

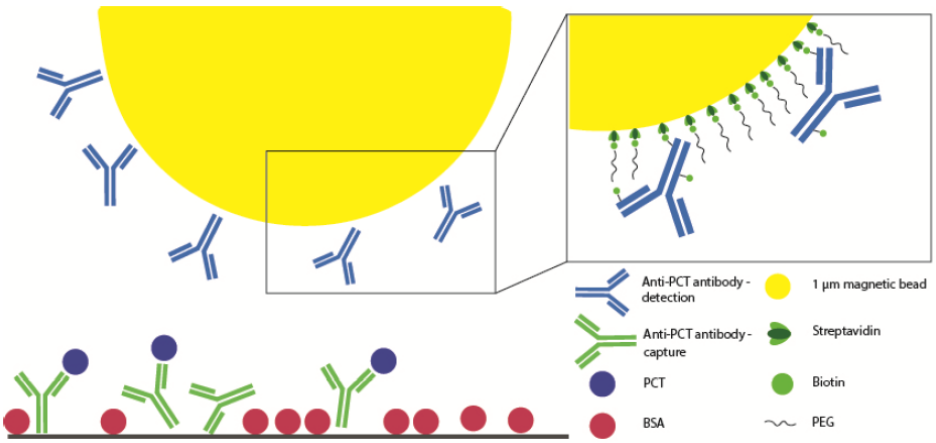

Figure 2.6: A typical illustration of an fBPM immunoassay

2.5 Response Measurements

To check the concentration of a sample with unknown concentration, one needs to compare its results with a dose response curve, made beforehand for that certain sample. This can be done for the bound fraction, the activity or the state lifetime. For this to be possible the dose response curves needs to be reproducible, there should not be any significant batch-to-batch variations in the dose-response curve, as this would significantly reduce the accuracy of the concentration measurement. The latter is also true when the slope of the curve is not steep enough or too steep, since a small error in the readout of a x value with a very steep slope will lead to a big error in the determined y value, and vice versa for a gentle slope. The dose-response curve of bound fraction and activity are sigmoid shaped, meaning that only the linear part of that curve can be used for accurate concentration measurements. This means that the accuracy is dependant on size of the linear regime, so ideally the desired concentration regime lies in this linear regime of the response and this regime is as large as possible.

2.5.1 Bound Fraction

The key element of experimental data is state prediction. During post-processing, a time dependent state is coupled to each individual particle tracked. Every particle at every analyzed frame of a measurement has a state assigned to it. These states can either be unbound, single bound or double bound. The bound fraction is the fraction of frames where a particle is in the bound state, divided over the total amount of frames, as seen in Equation 2.5. This can further be categorized into the single bound and double bound fraction. Logically there also exists an unbound fraction, which is one minus the bound fraction. These fractions correspond to the time averaged fraction of frames where particles are in said state over the total amount of frames.

$$BF_s = \frac{\sum_{n=1}^{N_p} F_{s,n}}{N_p F_{max}} \tag{2.5}$$

Here BF_s is the time averaged bound fraction of state s, N_p the total number of particles, $F_{s,n}$ the total number of frames particle n is in state s and F_{max} the total amount of frames in the measurement time.

2.5.2 Activity

A measure to analyze the concentration is to look at the activity. The activity is the number of events per particle per second. To do this analytically, all the events are summed and divided over the amount of active particles (so particles with an average D above the stuck limit) and over the measurement time in seconds. There are however two ways to count events: by counting the amount of times a state appears, and by looking how often a particle switches states.

There are two types of switching activity, the so called '01' and the '12' activity. Respectively the amount of times a particle switches from the unbound to the single bound state or vice versa and how often it switches back and forth between single bound and double bound state. All of this is illustrated in Figure 2.7, where the

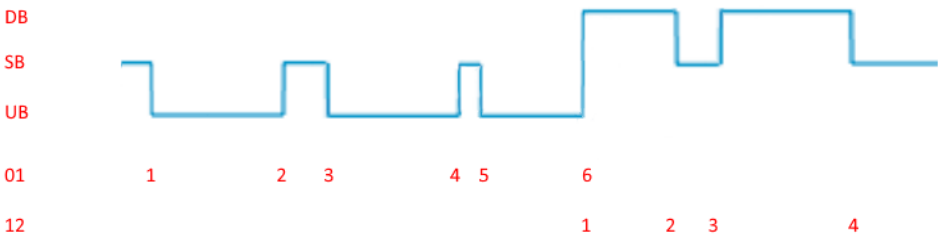

Figure 2.7: An illustration of how the activity is counted, where UB $= 2 \times n_{01}$ and DB $= 2 \times n_{12}$.

blue line is a simplistic representation of what state a certain particle is in as function of time. Below that the 01 and 12 events have been counted at the moment such an event occurs. If one were to count the times this particle is in the unbound (UB) state this would be exactly double as the number of 01 events. Likewise the number of double bound (DB) events is double the number of 12 events.

However is this same state prediction is cropped, like in Figure 2.8, this is no longer the case. If one were to compare 01 and 12 events to be twice the UB and DB events, there is now 1 fewer 01 event and 1 fewer 12 event. Bearing in mind that there are usually hundreds of particles in one measurement, this seemingly insignificant mistake might rack up to be larger than expected. Therefor it is important to be consistent in the determination of the term 'event'.

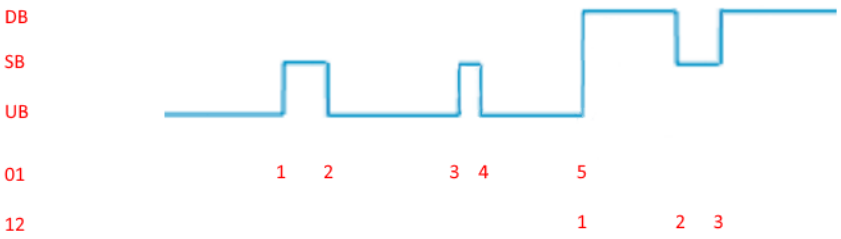

Figure 2.8: An illustration of how the activity is counted, where UB $\neq 2 \times n_{01}$ and DB $\neq 2 \times n_{12}$.

In order to get from the number of events to the actual activity of a measurement, the total number of events N_{events} are is fitted with a normal distribution, from which the mean number of events μ_{events} is derived and divided by the measurement time T_{meas}, like so [4]:

$$Activity_j = \frac{\mu_{events,j}}{T_{meas}}. \tag{2.6}$$

Where j is either 01 or 12, for the desired type of activity.

Ultimately, the activity will follow the Hill equation:

$$R = R_{bg} + R_{ampl}\frac{[\mathrm{T}]^n}{EC_{50}^n + [\mathrm{T}]^n}. \tag{2.7}$$

This is the response R as a function of background response R_{bg}, an amplification factor R_{ampl}, concentration [T], the half maximal effective concentration EC_{50} and the Hill coefficient n which is a way to quantify the degree of interaction between ligand binding sites. The activity per concentration is an experimental result, therefore the dissociation constant can extracted from this fit. This Hill equation is in essence nothing different from a 4 parameter logistic regression curve.[9]

2.5.3 State Lifetime

Another property one can extract data from are the state lifetimes. Simply put, a particle is either bound or unbound. The time a particle resides in either of those states, is called the state lifetime, respectively τ_B and τ_{UB}. Binding and unbinding happen at a certain rate, these rate constants are called k_{on} and k_{off}.

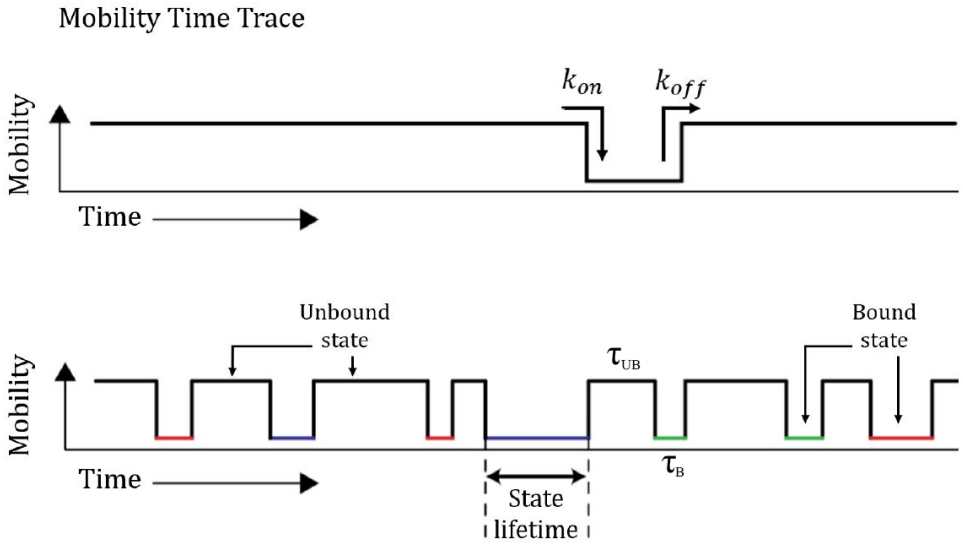

Figure 2.9: An illustration of a typical mobility time trace to illustrate τ_B, τ_{UB}, k_{on} and k_{off}

In order to get these values from experimental results, the state lifetimes are sorted by length in increasing order. The survival fraction is then plotted, which is nothing else than 1 - the cumulative distribution function. This is then plotted and fitted with an exponential fit, like in Equation 2.8.

$$\begin{aligned} 1 - F_{UB}(t) &= \exp\left(-\frac{t}{\tau_{UB}}\right) = \exp\left(-k_{\mathrm{on}}\, t\right) \\ 1 - F_B(t) &= \exp\left(-\frac{t}{\tau_B}\right) = \exp\left(-k_{\mathrm{off}}\, t\right) \end{aligned} \tag{2.8}$$

This is an oversimplified approach, as it is already known that there are three states in the fBPM system. The bound state is an overarching state comprised of the single and double bound state. This results in two new state lifetimes: τ_{SB} and τ_{DB}, see Figure 2.10.

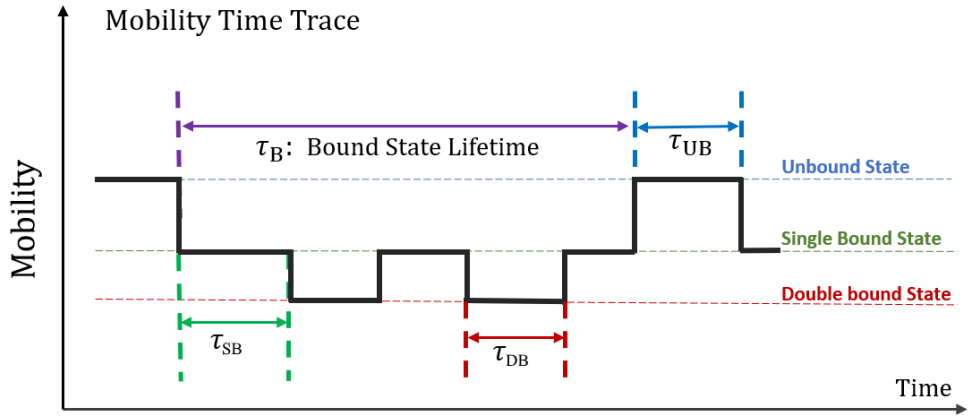

Figure 2.10: A state lifetime illustration including the single and double bound state

The problem now is that the bound state lifetime can no longer be fitted with a single exponential. Therefor a double exponential is introduced:

$$1 - F(t) = A_1 \exp\left(-\frac{t}{\tau_1}\right) + A_2 \exp\left(-\frac{t}{\tau_2}\right). \tag{2.9}$$

This double exponential fit consists of two exponents, with two different decay times, and two different magnitudes A_1 and A_2. The constraint is that $A_1 + A_2 = 1$, thus:

$$1 - F(t) = A_1 \exp\left(-\frac{t}{\tau_1}\right) + (1 - A_1) \exp\left(-\frac{t}{\tau_2}\right). \tag{2.10}$$

Due to the two different decay times, this double exponential fit is able to account for two populations of lifetimes, short lifetimes τ_1 and long lifetimes τ_2. Equation 2.10 shows the difference in performance between the single and double exponential fit.

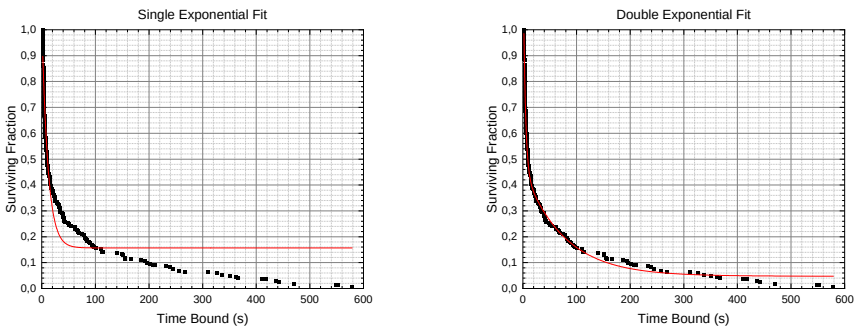

Figure 2.11: The survival curve of bound state lifetime of the 200 pM measurement of the PPC3-B27A3 sample, as a demonstration why the double exponential fit is required.

It is to be expected that the single and double bound state lifetime still correspond to a single exponential decay, while the bound lifetimes do not. However, deviations from this behavior might be explainable when using polyclonal antibodies, due to the heterogeneity in paratopes. After looking at the results, all states seem to be better suited for a double exponential fit. Therefore, the short lifetime is attributed to low affinity paratopes and the long lifetime is attributed to the high affinity paratopes.

3 Experimental Setup

3.1 Antibody Pairs

For this research the antibody pair PPC3-B27A3 was chosen and tested. PPC3 is a polyclonal antibody, meaning a higher overall antibody affinity and therefor quicker binding and high sensitivity for detecting low quantities [10]. Also the binding is less dependent on particle deformations and irregularities. This comes at the cost of some batch-to-batch reproducabitity, as the bond formed can be one of several options of monoclonal bonds. This difference in bond also leads to a loss in accuracy, as different bonds have different affinities and thus different dissociation rates. B27A3 is the biotinylated version of 27A3, a monoclonal antibody particularly binding to the N-terminus of PCT, see Figure 2.2.

For the comparison of DLM vs THM, results of this research can be compared with the results of other experiments, such as the 13B9-B27A3 antibody pair. 13B9 is a monoclonal antibody, and is chosen because of its low affinity to B27A3. The result of this is lower non-specific binding, a delay the saturation of dose response curves and evading irreversible binding. The antibodies compatible with PCT have been screened for affinity [11], see Figure 3.1, to get a general sense which antibodies to use.

We expect the responses of PPC3-B27A3 to be different than that of 13B9-B27A3, because of the higher affinity. This would translate into more response at lower concentration. The difference in polyclonal versus monoclonal antibodies is expected to increase this low concentration response.

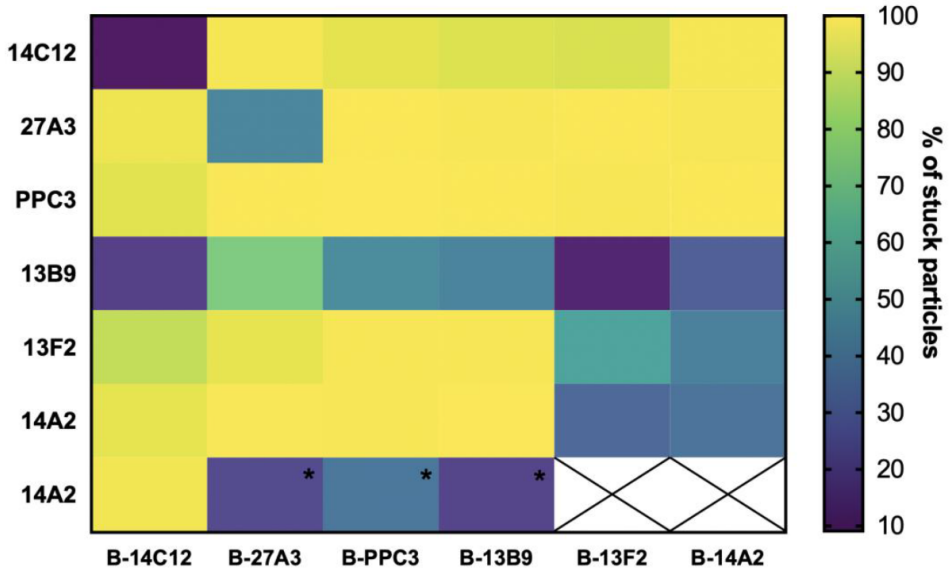

Figure 3.1: Heat map of the antibody screening of all available antibodies against each other. An antibody concentration of 100 nM was used together with a target concentration of 10 nM. No target was added to three negative control samples, marked with '*'.

3.2 Assay Preparation

In this research the main focus is on PCT fBPM immunoassays. This was done for different antibody pairs, however the preparation procedure does not significantly change for different pairs. The general procedure basically consists of three steps: functionalization, blocking, particle cleaning.

Firstly the antibodies are diluted in phosphate-buffered saline (PBS) such that the antibody solution is 100 nM. The capture antibody solution is suspended and incubated for 60 minutes in the flow cell at room temperature, such that they will physisorb to the substrate. This flow cell is made by sticking a flow cell sticker onto a glass plate. The detection antibody solution is pipette mixed with 1 μm streptavidin coated Dynabeads MyOne and incubated for 45 minutes at room temperature in a HulaMixer Sample Mixer.

Once this is done, the blocking step can commence. The particles are blocked by biotinylated methyl-polyethylene glycol (mPEG), diluted in PBS, during a 10 minute incubation step at room temperature in a HulaMixer. The flow cells are blocked by a 1% bovine serum albumin (BSA) solution, diluted in PBS, during a 60, minute incubation step at room temperature.

After the particle incubation step, the magnetic property of the Dynabeads is utilized. In a magnetic rack the beads are pulled towards the magnet, such that the remaining solution can be removed, without the loss of particles. This effectively washes the particle solution. After rinsing three times with PBS, the particles are isolated and placed in a 1% BSA (in PBS) solution for 60 minutes at room temperature. For a schematic overview of the immunoassay structure, see Figure 2.6.

After which the particle solution is sonicated, and diluted to the desired imaging concentration with the assay buffer (which consists of 0.1% BSA solution in PBS). Before imaging, the flow cells are injected with the diluted particle solution, which is incubated there for 20 minutes at room temperature, such that the particles have plenty of time to sediment. After which imaging can begin.

3.3 Imaging

3.3.1 Microscope

Measurements are done on a movable stage a Nikon Ti Confocal Microscope or the Leica inverted microscope with an objective with 20x magnification in darkfield. The particles were recorded for 10 minutes at 60 frames per seconds using the FlyCapture Software Development Kit (FLIR) after calibration with the NIS-Elements Microscope Imaging Software (Nikon).

3.3.2 Software

In order to get the particle locations from a visual feed tracking software is required. To this end the ”Biosensing by particle diffusion software” is used, which has been developed by Max Bergkamp [12]. This is a real-time particle tracking software made especially for BPM in `C++`. By having accurate particle identification and localization, particles can be tracked and their xy locations can be found with sub-pixel precision, after which it is noted down as a `.txt` file, called a xy-list. These xy-lists are saved and used for post-processing.

The software identifies particles in a three step process, illustrated in Figure 3.2.

- First, the image is filtered with an intensity thresholds to find possible particles against a dark background.
- Then the distance between two intensity peaks is checked to remove larger shapes and minimize false positives. This distance filter includes also a distance to the boundary of the field of view, as particles leaving the field of view can no longer be tracked.

Lastly, the particles are filtered on shape, this exists of two components, symmetry and deviations from a particle template is added to ensure that all particles are spherical and thus indeed particle, not artifacts.

- Symmetry filtering is done by calculating the covariance matrix, and retrieving the major and minor motion amplitude as a result of the square root of the largest and smallest eigenvalue of the covariance matrix. From the fraction of the minor over the major motion amplitudes, the symmetry of the motion pattern can be found. For each particle its symmetry is calculated and particles with a symmetry below a certain threshold will be rejected.
- The second part of the shape filtering step is based on particle template deviation. A particle template is gathered from simulation, which is subtracted from the particle's location. The sum of the remaining absolute pixel intensity values is used to calculate the deviation from the particle template. Again a threshold value is set to filter out deviating particles.

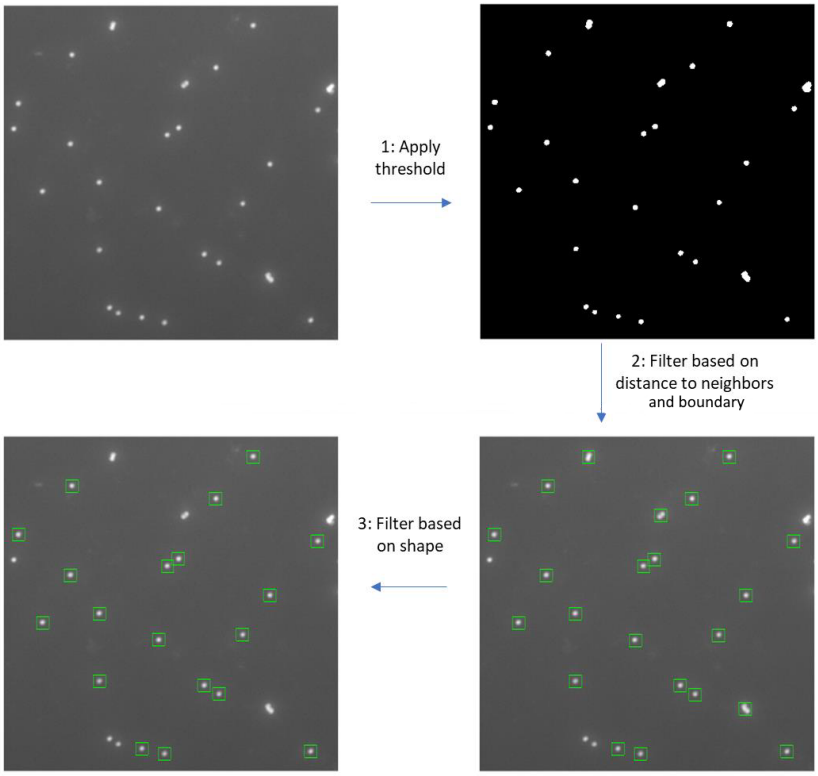

Figure 3.2: The three step particle identification process.

Once the particle identification is done, the localization can commence. For reasons of sensitivity and speed, phasor localization is used. This algorithm transforms the intensity of the point spread function into phase vectors using the first Fourier coefficients in the x and y direction. The angles of these phase vectors are then inverse Fourier transformed to give the x- and y- location of the center of the point spread function[13].

3.4 Post-processing

3.4.1 Threshold Method

The threshold method (THM) is a post-processing script that tries to analyse what is happening at the microscopic size. It runs on the input of the 'xy-lists' created by the imaging software. For every identified particle a new row is added to this text file. The first two columns are the starting x and y positions of each particle, in pixels. The rest of the columns are the difference in location at other frames, alternating in x and y. This difference (in pixels) is that between the position at that time, compared to the starting position. This means that the xy-list is a (m x n) matrix with m the number of particles and n two times the number of frames plus two, for it has two columns per frame (one for x, one for y) plus two starting columns. This is illustrated in Figure 3.3.

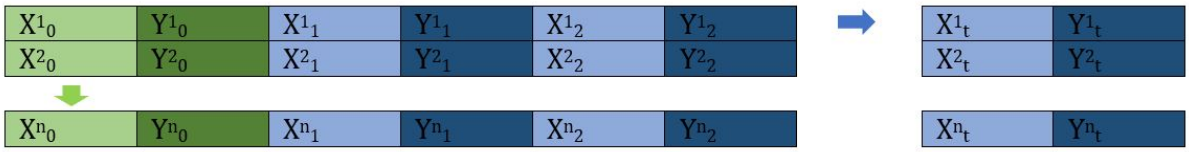

Figure 3.3: Illustration of a xy-list with the x y coordinates for particles 1 to n, for time 0 to t.

The THM predicts the particle's state based on the calculated diffusion coefficient, and compares this to a certain threshold value and assigns states to particles on the bases of these regimes. There are some filtering steps, mainly to filter out noise and stuck particles. The process is described below and illustrated in Figure 3.4.

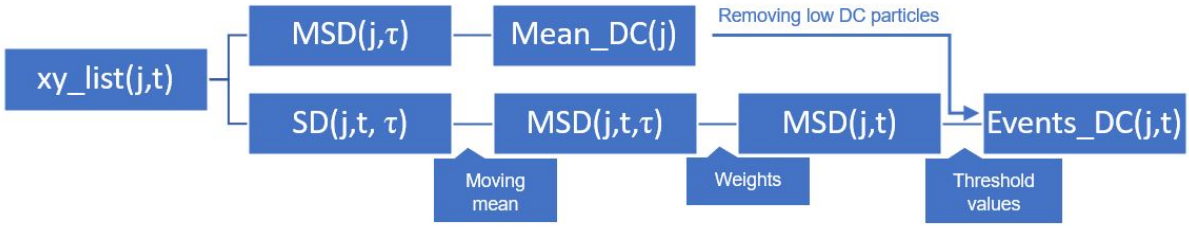

Figure 3.4: Schematic of the THM state prediction process.

Firstly the average diffusion coefficient for each particle (Mean_DC(j)) is calculated. This is done using the mean squared displacement (MSD(j,τ)) of particles at time step t versus at time $t + \tau$, with τ ranging from 1 to 10. This is then fitted with a linear fit, where the slope represents the time averaged diffusion coefficient.

There is also a process to get the time-dependent diffusion coefficient, this is calculated using the mean square displacement as a function of time with a moving mean. A moving mean transformation with a sliding window of size k, which averages each value of an array by averaging the value of the current position over the values of its $k - 1$ neighbors. This is a good method to even out short-term fluctuations on a signal. See Figure 3.5 for an illustration.

The calculation of the time dependent diffusion coefficient starts with calculating the squared displacement (SD(j,t,τ)) from one point at time t versus at time $t + \tau$, where *tau* is an integer increasing from 1 to 10. This gives 10 different sets of values for the squared displacement, for every value of τ a set of time-dependent values. This squared displacement is averaged with a moving mean with a window size of 60, resulting in a mean square displacement as function of particle number, time and τ (MSD(j,n, τ)).This is then weighted over τ such

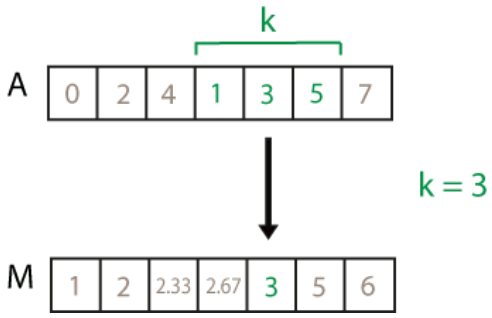

Figure 3.5: Illustration of a moving mean[14]

that further time-steps influence the mean to a lesser degree, reducing it to one single set of values per particle (MSD(j,t)).

The time averaged diffusion coefficient is then used to filter out all stuck particles, by removing particles of which the average diffusion coefficient is not higher than the stuck threshold value, which is in this research set at 0.02. In a for loop over all remaining frames, the value of the weighted diffusion coefficient is once again averaged with a moving mean, to smooth out the signal. The output of this is compared with certain threshold values per time-step, after which states are assigned to frames. These threshold values distinguish three different regimes: unbound, single bound and double bound. Lastly the length of said states are checked, if states are shorter than a specified lifetime (120 frames), the state is discarded. The end result is a matrix for all particles with the particles' state as a function of time (Events_DC(j,t)).

3.4.2 Deep Learning Algorithm

In order to improve the state determination of tracked particles, a deep learning algorithm was applied for the post-processing of the experimental data. Deep learning is a type of machine learning, which is in turn a form of artificial intelligence that enables a system to learn from data rather than through explicit programming. Machine learning uses a variety of algorithms that iteratively learn from data to improve data and predict outcomes. If a machine learning algorithm is trained with data, it generates a machine learning model. This model can than be used on real data to predict the outcome.[15]

In order to get a machine learning model to work, it needs to be trained. This means that one is to feed the model a set of training data. In the case of the deep learning model (DLM) used in this research, this means creating a set of simulated data and feeding that to the model. This simulated data also has the correct outcomes. The model starts looking for patterns and crudely predict states for this training data. The predicted states are compared to the true states, and false states are rejected. This way the model only keeps the correct prediction patterns and reject patterns that lead to incorrect outcomes. By digesting more data the accuracy of the predictions increase. When the model is sufficiently trained, it can be used on experimental data where the outcome is unknown.

Deep learning is a specific method of machine learning that goes through successive layers in order process data.This is especially useful when trying to learn patterns from unstructured data and problems that are poorly defined. A deep learning model consists of three types of layers: one input layer, many hidden layers, and an output layer. Each layer consists of multiple, interconnected, nodes who each do a simple processing step. Data is fed into the input layer. Then the data is modified in the hidden layer and the predicted outcome is given in the output layer. Every connection between the neurons consists of weights, it denotes the significance of the input values. These weights are calculated during the training of a model. The hidden layers is what makes

transforms the raw data into values for the desired parameters.

The DLM used in this research is used to distinguish in what state colloidal particles are, based on coordinates measured in fBPM. In order to do so, the model uses a combination of a bidirectional Long Short-Term Memory (LSTM) layer and a 1D convolutional neural network (CNN)[16]. This breaks down to the following:

A LSTM layer is a type of recurrent neural network (RNN), designed for processing sequential data. This means that it includes the output of the previous step along the input at the current position for the output of the current step, see Figure 3.6 for a schematic. LSTM is a specific type of RNN which includes a separate memory cell, designed to learn long-term dependencies. Not only is this RNN dependent on the output of the previous step and the input of the current step, it is also dependant on the memory cell state. Each step the information of the current step is can be stored in the memory cell and old information can be forgot. This way the LSTM can save relevant information for a longer period of time. This LSTM is used bidirectional, this refers to an adaptation of tradition RNN algorithms. Since, these types of algorithms are very dependant on the input, the order in which it is ran affects the outcome. To this end the bidirectional LSTM is made of two LSTM's, one of the two processes the inputs backwards. In the end, both representations are merged into a single output. By looking at it backwards, it might find patterns that otherwise would have been overlooked.

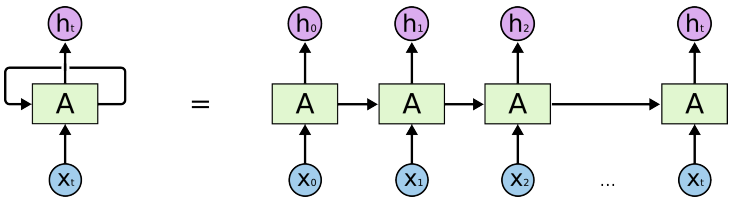

Figure 3.6: Schematic overview of a RNN[17]

A CNN is an algorithm often used in image recognition and excels in picking out local patterns, or 'patches'. In terms of image recognition, it would easily be able to find the edges of objects from which it tries to recognise categorized objects. For example, by looking at the edges making up a human nose, it can tell if an object is a nose. It maps features to the next layer, which has one neuron for every different feature. This layer is used as filter for the whole grid of inputs to quickly recognize features.

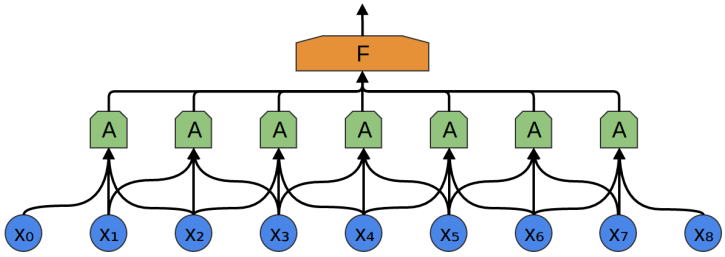

Figure 3.7: Schematic overview of a 1D CNN[18]

The 1D stands for 1 dimensional. This means it looks at 1D inputs, like time traces. In Figure 3.7, a schematic of a 1D-CNN is shown. Here **x** is an array of inputs, which are ran through a the convolutional layer made up of a set of neurons, **A**. These neurons look at small time segments of the data. **A** looks at all such segments, computing certain features. Then, the output of **A** is fed into a fully-connected layer, **F**, which links the output of **A** to a state prediction.

For this project the 1D CNN is used as a pre-processing step for the bidirectional LSTM. This way the input of the LSTM is reduced to a shorter sequence of meaningful features. This way the computation speed is greatly increased without sacrificing on performance.

4 Results PPC3-B27A3

This researched is mainly based on the results of the PPC3-B27A3 antibody pair. To goal is find positive response at low PCT concentration. This pair is tested by both the deep learning model (DLM) and the thresholding method (THM). A secondary goal is to look at the difference in response as result of post-processing method. To validate and possibly quantify these differences, results of other pairs are also looked at. For this purpose, the results of the 13B9-B27A3 pair is used, because we have three separate measurements of this pair.

The results of the THM are very dependent on input parameter, such as the averaging window size or the set threshold values. The DLM is more robust and more complex to alter, therefor changes were made to the THM script and the results were compared to previous results or the DLM results.

4.1 State fractions

In Figure 4.1 the unbound, single and double bound fractions are displayed. It is clear that there is a response as result of increasing PCT concentration. The concentration regime seems to be chosen well, as this captures the linear regime of response perfectly. Note the saturation around 800 pM. This coincides with the regime where sepsis occurs up to septic shock. Preferably the limit of detection would lie around 4 to 40 pM. Unfortunately, there was no measurement done at this concentration, so we are unable to tell if it is possible to measure here.

There is hardly any difference between DLM and THM for the unbound fraction. This means that both DLM and THM are equally capable to distinguish bound from unbound, even though the state determination method is different, see subsection 3.4. This can also be seen in the EC_{50} values, displayed in Table 4.1.

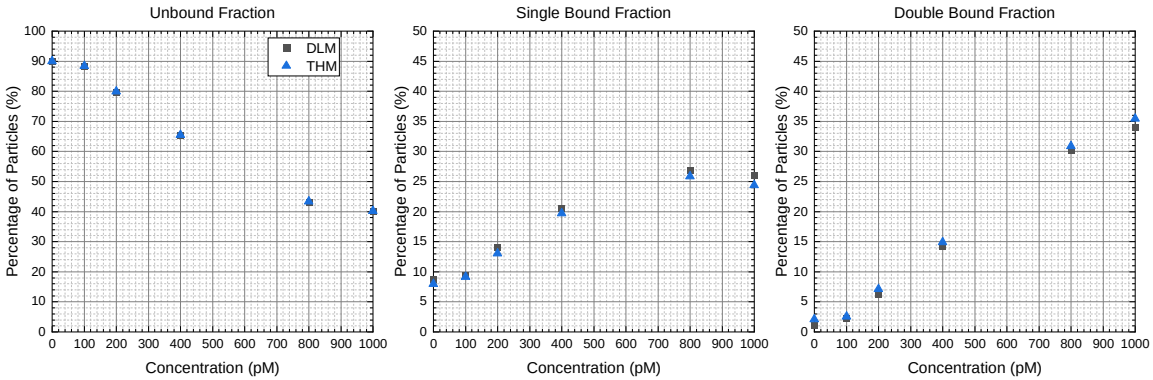

Figure 4.1: Unbound, single bound and double bound fraction for PPC3-B27A3 for the DLM and THM.

There is a very small difference in the bound states. At higher concentrations the THM has a relatively higher double bound fraction and lower higher bound fraction compared to the DLM. This difference is negligibly small. This is also reflected in the EC_{50}'s in Table 4.1, the UB and SB fraction have a nearly identical Hill curve. The DB fraction has a slightly higher EC_{50} for DLM, but the both DLM THM share the same confidence interval. The fact the the standard deviation for the DB fraction THM Hill fit is significantly higher than that of DLM, indicates that the DLM DB fraction values align better with the expected dose-response curve. Indicating that the DLM might be more reliable in double bound state recognition.

For the 13B9-B27A3 pair the state fractions were also calculated and plotted in Figure A.1. Here too the conclusion is that there is no significant difference in state fraction between DLM and THM. This is remarkable,

Table 4.1: The corresponding EC_{50}'s per method as result of fitting the bound fractions with the Hill equation.

	UB	SB	DB
DLM	480 ± 71,0	314 ± 39,6	651 ± 109
THM	482 ± 69,7	304 ± 42,5	695 ± 157

since the DLM is supposed to have better DB state detection. The two methods have different state prediction techniques, but yield similar results.

The main difference in state prediction between DLM and THM is the overlap region in between the single and double bound states. The only possibility for particles to be assigned to the double bound state while having a D above the double bound state threshold (and vice versa) is when the state transition is rejected because of its length is too short. If this rejection was not incorporated, the border between the two bound states would be a straight vertical line.

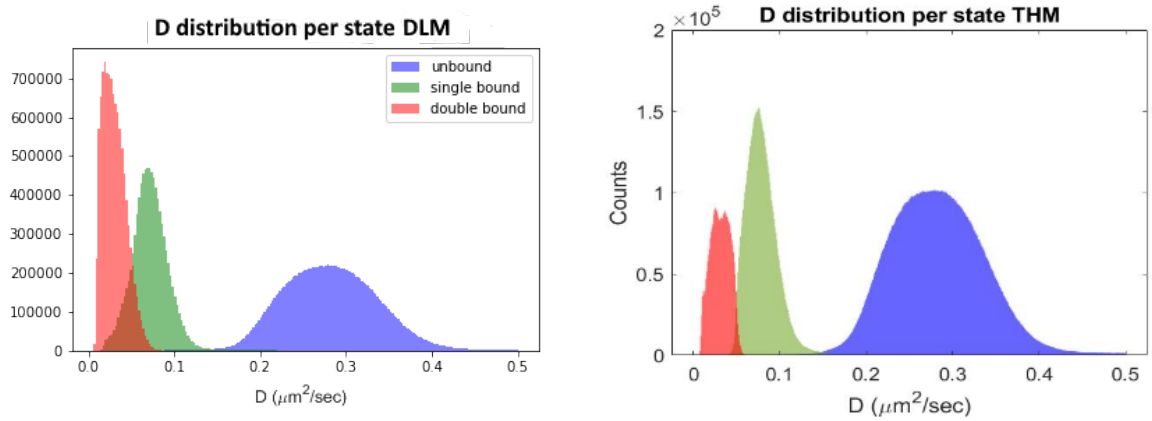

Figure 4.2: Diffusion coefficient distribution per state for both post-processing methods plotted from experimental data. Data from 800 pM PPC3-B27A3 antibody pair.

In Figure 4.2, the D distribution per state is displayed for both the DLM and the THM result. Both images represent the same measurement, the 800 pM measurement for the PPC3-B27A3 antibody pair. Like mentioned before, the overlap between the two bound states is distinctly shaped. The DLM has greater overlap. This is the reason the DLM was introduced to this system, there is a lot of information between state changes lost in that overlap regime, which would otherwise be lost. Heterogeneity in particles and bonds translate to different values of D where states occur, so much so that in some instances the double bound state of a certain particle has a higher D than the single bound state of another particle. This is information lost by simply assessing states based on a threshold.

Another thing that we notice is that the DLM has a much higher fraction of double bound states, compared to THM. Its peak is shifted to the left compared to THM. This is probably due to the low mobility filtering in THM that removes particles with a too low average D, intended to remove stuck particles. This is shown in Figure A.4. Though this only slightly affects the bound states, it does filter out some non-specific binding. To prove this, we plotted the D distribution for the control measurement with and without this filter in Figure A.5. For that reason, the low mobility filter remains implemented in the THM. It might be useful to implement this also in the DLM, but in view of time, this has not been done in this research.

4.2 Activity

The activity for both the DLM and the THM is measured as the average number of state switches per second, see subsubsection 2.5.2 for more information.The activity for the PPC3-B27A3 pair is illustrated in Figure 4.3. This plots the unbound to single bound (01), the single to double (12) activity as well as the total activity, the sum of 01 and 12.

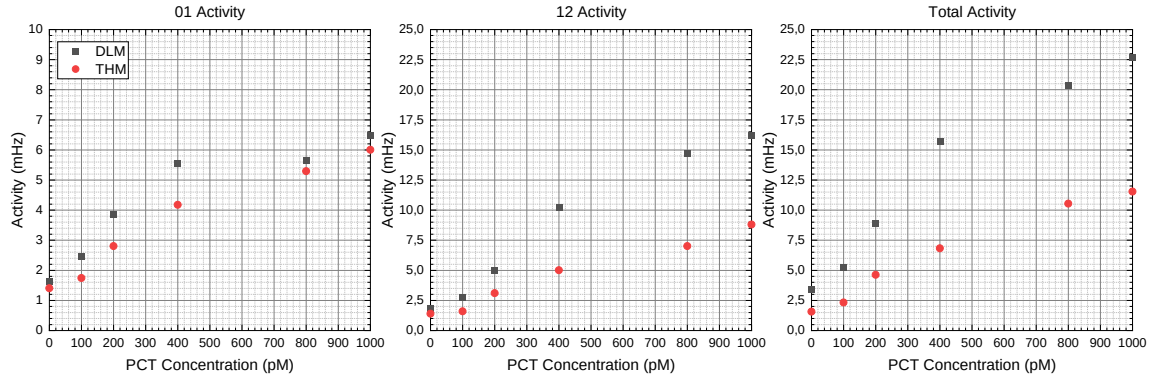

Figure 4.3: The 01, 12 and total activity for the PPC3-B27A3 antibody pair, comparing DLM to THM results.

The 01 activity follows the expected sigmoid curve, which seems to saturate once again at 800 pM. The 12 activity might not be saturated, but this is no problem. The fBPM technique works best with reversible binding, which is less often the case with 12 activity. Looking at the difference between methods, the slope at which 01 activity increases in the linear regime seems similar. The 01 activity is just a little bit higher for DLM than for THM. Assuming that both post-processing tools are equally capable of distinguishing bound from unbound seen Figure 4.1, this would mean that the states are generally shorter for DLM, in the low concentration regime.

The main difference in the 01 activity lies beyond the low concentration regime, the DLM saturates, while the THM still increases. This is probably due to the exclusion of the low mobility filter (LMF). This filter removes particles that are double bound for most of the time. This reduces the number of particles to divide the events over, increasing the 01 activity. In order to truly say something about the difference in state determination, this LMF needs to be taken in account. For that reason, Figure 4.4 displays the activity for DLM, THM with and THM without LMF. We see the sudden saturation in 01 activity after 400 pM for THM without LMF as well. This indicates the start of the multivalent regime.

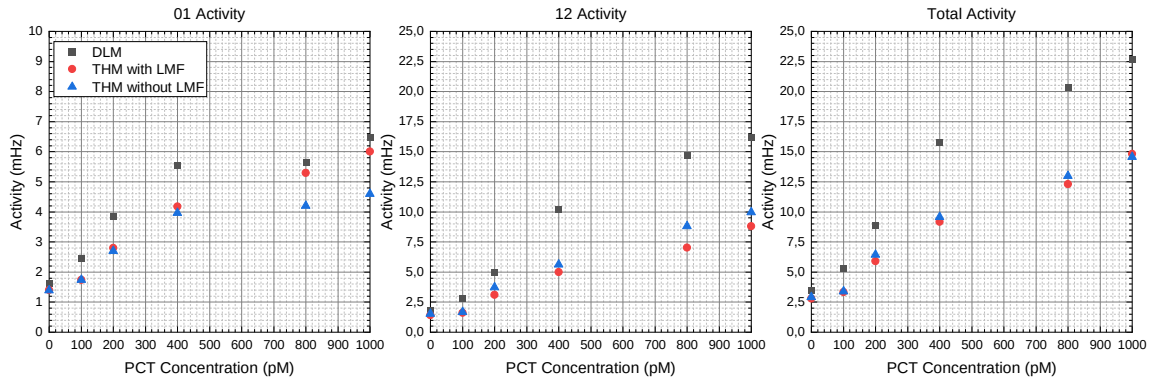

Figure 4.4: The respective 01, 12 and total activities for the PPC3-B27A3 antibody pair, comparing DLM and THM with and without the low mobility filter.

Clearly there are a lot of particles stuck at concentrations above 400 pM, who have been removed by the LMF. These particles do not contribute to the activity, except for the number of particles the events need to be divided by. The activity without the LMF is probably more representative of the real situation but is also more prone to irreversible bonds to contribute. With increasing concentration of PCT a new subculture of particles arises, particles who are multivalently stuck, so much so that they cannot unbind. This can also be seen in the slight increase in double bound lifetime in Figure 4.7.

These particles disfigure the dose response curve of the 01 activity. On top of that, they inflate the double bound fraction, as they would be assigned as double bound. This problem is fixed with the introduction of the LMF. After this filtering, the 01 activity does seem to better follow the expected sigmoid shape. The while the other 01 activities saturate much faster. This can also be seen in Table 4.2 where the EC_{50}'s of the THM with LMF is much higher for THM with LMF than the others, indicating that the activities saturate later.

Apart for the dip in the 01 activity at high concentration, there is virtually no difference in the THM with or without LMF. This implies that the LMF successfully removes stuck particles filtering away too much 12 events. The fact that the THM with and without LMF have very similar activities at 0 pM PCT indicate that most non-specific bonds are only temporary.

We expect the bound fraction to have the approximately the same EC50 as the 01 activity, since this is solely about binding and unbinding. The bound fraction is 1- the unbound fraction, thus has the same EC50. We see however, that the unbound fraction EC50 double that for the DLM and the THM without LMF, whereas the THM with LMF does approach it somewhat close but is still too low. At least it is clear that an LMF yields better results here. This makes very interesting to look at DLM with an LMF.

Table 4.2: The corresponding EC_{50}'s per method as result of fitting the activity with the Hill equation.

	01 Activity	12 Activity	Total Activity
DLM	204 ± 38	400 ± 28	350 ± 44
THM with LMF	338 ± 73,4	465 ± 328	403 ± 179
THM without LMF	225 ± 24	605 ± 346	414 ± 126

What strikes the eye is the huge difference in 12 activity. The 12 activity of the DLM reaches up to 150% of the magnitude of the 12 activity of the THM without LMF. The 12 activity of THM with LMF is even smaller still. The shape of the curve and the EC_{50}'s are very similar for the two THM's, even to some extent to that of DLM, but the DLM has a much lower standard deviation. Confirming that the DLM is better at distinguishing 12 events.

4.2.1 Reproducibility of activity

To be able to say something about the differences in results of two methods, we need to compare other pairs as well. The 13B9-B27A3 pair has been measured three times, once at low concentration (set A) and twice at higher concentration (set B and C). Since set B and C have the same concentration range, the activities of these experiments are plotted in Figure 4.5.

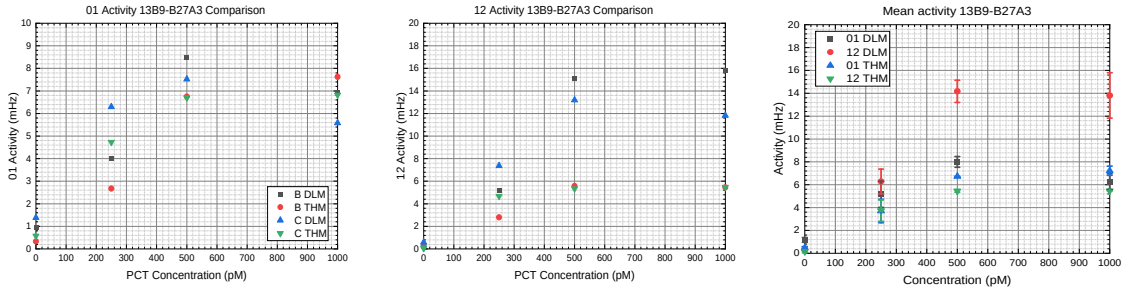

Figure 4.5: The activities of set B and set C of the 13B9-B27A3 pair, along with the mean activity per method and its standard deviation.

Unfortunately, we cannot say anything about the EC_{50}'s, since this requires a fully developed activity to be able to fit it with the Hill equation. Set A (Figure A.8) is unsuited for fitting the Hill curve, since the concentration regime is cut down too short to show saturation. The other sets do show saturation but has too few data points in the linear regime to properly fit the Hill curve. Making it hard to say something quantitative about the difference in performance of DLM versus THM (with LMF) regarding activity. But we can say something about patterns.

The drop in 01 activity for DLM due to a lack of LMF, mentioned earlier, is also visible for this antibody pair. What can also be concluded here, is that however the 12 activity is much higher for DLM than for THM, the THM result is much more consistent. This probably due to the lack of a low mobility filter for DLM. Like mentioned before, this reduces the inclusion of stuck particles. This would explain the deviation in the 12 activity. Even though set B and C are made and measured identically, they are not the same sample. There will be some batch-to-batch deviations in the amount of stuck particles. This is another reason the DLM will improve from an LMF.

4.3 State lifetimes

In Figure 4.6 the surviving fraction of the unbound state are plotted, along with the double exponential fit. What can be seen from this figure is that the fraction between short and long lifetimes is independent of PCT concentration. The same is true for the short lifetimes τ_1. With increasing concentration, the unbound τ_2 decreases exponentially, which is to be expected. The value of τ_2 is plotted in Figure 4.7, of all states.

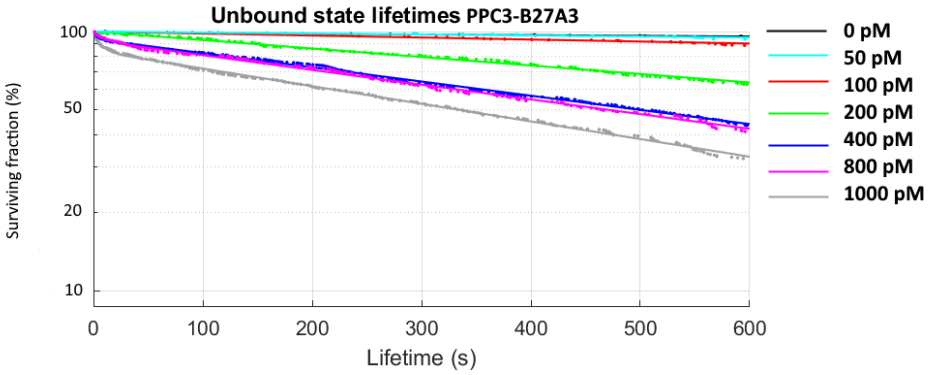

Figure 4.6: The THM unbound state lifetimes of the PPC3-B27A3 pair, along with a double exponential fit

The difference between the results of DLM and THM for the single and double bound τ_2's are insignificant, but there is a clear difference in the EC_{50} with which the unbound lifetime decays. This is $173 \pm 15,9$ s for DLM where it is only $119 \pm 2,71$ s for THM. This earlier plateau can also be seen in Figure 4.7, where the UB THM fit seems to bend off earlier. This is possible due to the fact that the unbound state lifetimes at 0 pM are much higher for THM. Both THM and DLM have values higher than the measurement time. This value is extrapolated from the lifetime fit, and therefore uncertain. Increasing the measurement time would make these results more trustworthy.

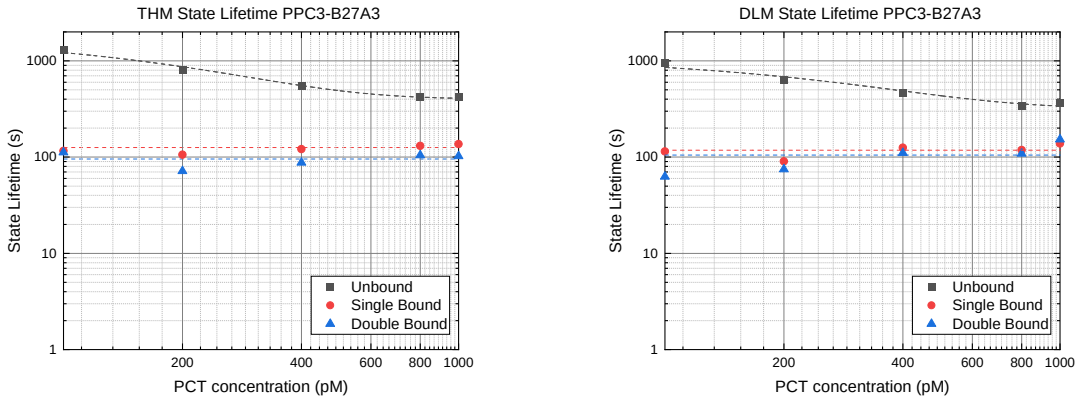

Figure 4.7: The characteristic state lifetimes τ_2 for unbound, single and double bound states as a function of concentration, plotted for THM and DLM

As expected, the bound states are largely concentration independent, whereas the unbound state lifetime displays an exponential decay. This corresponds with literature [19]. This means that only the τ_2 of the unbound lifetime can be used as calibration curve, but for this the result must be reproducible. To test this, the results of two different measurements of the same antibody pair were compared. This was done for the 13B9-B27A3 pair. The results can be seen in Figure 4.8.

We do see a slight increase in the double bound state lifetime of the DLM, this is due to the sub-population of particles that are immobilized. This is likely because they are multivalently bound to the extent that they can no longer unbind. These irreversible bonds only appear at high PCT concentrations. The LMF in the THM removes this sub-population, but the DLM does not.

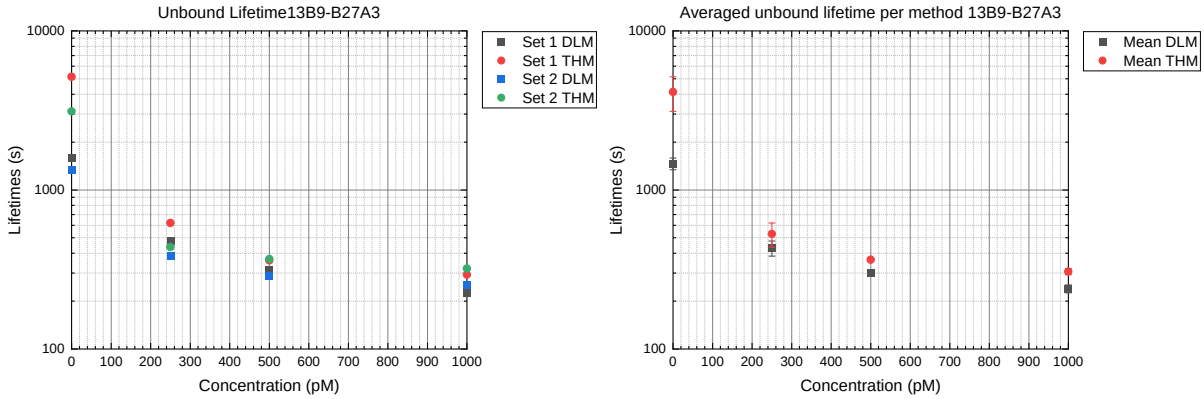

Figure 4.8: On the left, the unbound state lifetimes, derived from both THM and DLM, plotted for two different measurements of a 13B9-B27A3 antibody pair; On the right, the averaged values for the unbound state lifetime per method along with the standard deviation.

Once again, the exponential decay is clearly visible. The DLM has a slightly lower unbound lifetime. The difference in between sets, is lower for DLM, especially in the low concentration regime. From this we can conclude that the DLM analysis is more precise, and thus more reliable to make a calibration curve. A thing to note here is that, unlike the activity, the precision increases with concentration. Making this a more suitable dose-response for high concentrations.

4.4 Conclusion PPC3-B27A3 results

We investigated the possibility of using the PPC3-B27A3 antibody pair in a fBPM immunoassay to detect low concentrations of PCT. This pair show response for low concentrations, its linear regime coincides with the desired concentration regime to detect the onset of sepsis. The single and double bound fraction and the 01 activity show proper linear response without all too much deviation at low concentrations, while the unbound lifetime can best be used at higher concentrations.

This pair has been post-processed with DLM and THM, these results have been compared. The main difference between the two models is the determination of double bound states, since both are equally capable in distinguishing bound from unbound. The DLM is better in detecting double bound states. This is reflected in a vastly larger 12 activity. This leads to a higher total activity for the DLM, which makes this the better method to determine concentration from as the signal to noise ratio is decreased.

On the other hand, the reproducibility for the 12 activity is lower than that of THM. This can be attributed to low mobility filter build into the THM, which has a more accurate number of active particles. Clearly there is a large sub-population of particles immobilized. Introducing this filter to the DLM removes this sub-population, increasing the accuracy of the activity and possibly the reproducibility. This would ultimately give the DLM an edge over THM.

5 Agglomeration

Upon later inspection it became apparent that something odd was happening during measurements. When two particles enter each other's vicinity, there is a chance that somehow they will agglomerate. These agglomerations are then indistinguishable for the tracking software. The result of this is that the recorded positions of both particles is than identical. Those xy positions often display great leaps, this is presumably due to mislocalization. The agglomeration consists of two (or more) particles. The recorded xy position is that of the predicted center of a particle. Rather than following the center of the agglomeration, the tracking software might still track the center of one of the agglomerated particles but switch from one to the other. This would explain the great leaps in position, as the two particles are logically in two different locations. If at one moment the location of the perceived particle is that of particle 1, and an instant later that of particle 2, the perceived particle would have instantly 'travelled' the distance between the centers of the two particles.

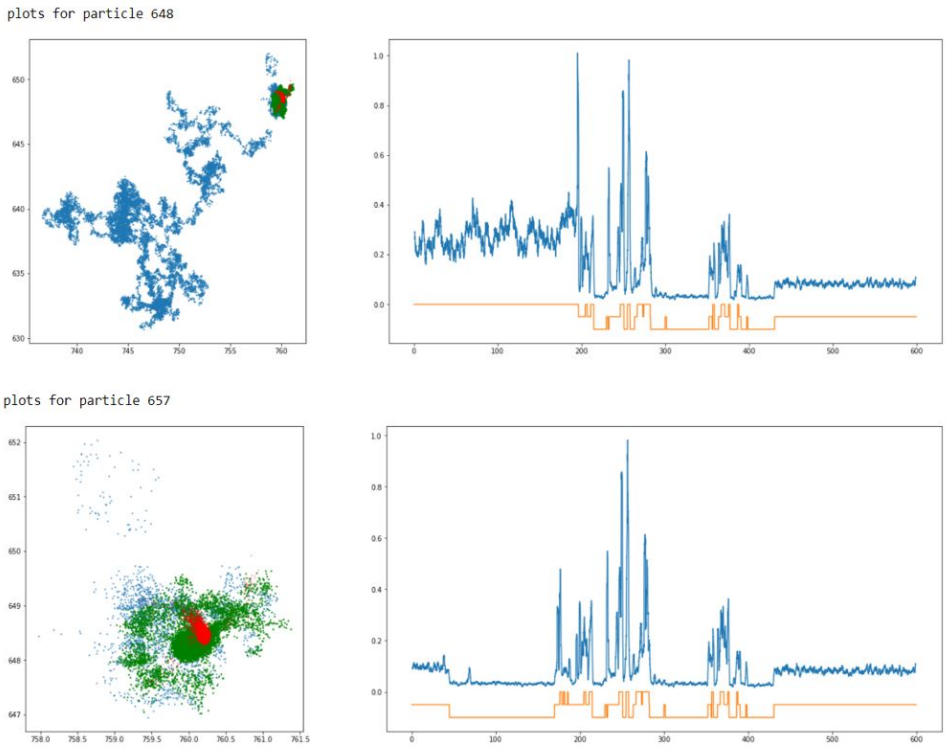

Figure 5.1: An experimental example of agglomeration taking place. On the left: their xy positions scattered, on the right: the calculated diffusion coefficient.

In Figure 5.1 an experimental example is shown where two particles agglomerate. The top particle is diffusing freely until it meets the bottom particle. By looking at the diffusion coefficient on the left, the time at which they meet can be deduced. After 200 seconds their diffusion coefficient is exactly the same, this can only be the case if their lateral movement is exactly the same, proving the point that they are moving as one. This means that the tracking software detects them as one.

5.1 Filtering out agglomerations

In order to ensure that those experimental artefacts do not influence the measurement's result, those collisions need to be filtered out. For that a Matlab script was created, initially to remove particles that display agglomeration. Later a more sophisticated method was developed that removes frames once a particle is agglomerated, preserving the pre-collision frames.

There are two ways tried to go about detecting collisions, the first way was to scan the starting xy location of all particles. Then there is a list composed of every particle's nearest neighbors. The number of nearest neighbors is a parameter called 'k' and can easily be changed. 'Nearest' here refers to Cartesian distance, not necessarily label number. This is because the particles are labeled left-to-right top-to-bottom, meaning that two particles located in a vertical line can be very close and still have a large difference in label number. For every particle the measured diffusion coefficient of every time step is compared to that of its k nearest neighbors. If the difference between D's of neighbors is less than 10E-5 for 100 seconds, this will be registered as agglomerated.

The second way is to look to the coordinates themselves. This has the benefit that the calculated D dependent is on its method of determination and on system parameters, contrary to the raw and unedited coordinates. The way this works, is that if two or more particles have the same coordinates (within a margin of 10E-10) for 100 seconds. For this to work, it will register as agglomeration. the assumption that clustering only occurs when particles are close to each other, is required.

The difference in bound fraction for the PPC3-B27A3 antibody pair edited with the D approach compared to the xy edited data can be seen in Figure 5.2. It is clear that the differences are marginal, but that the xy based editing is a little more simplified. To that extent the xy based editing method is used from here on out.

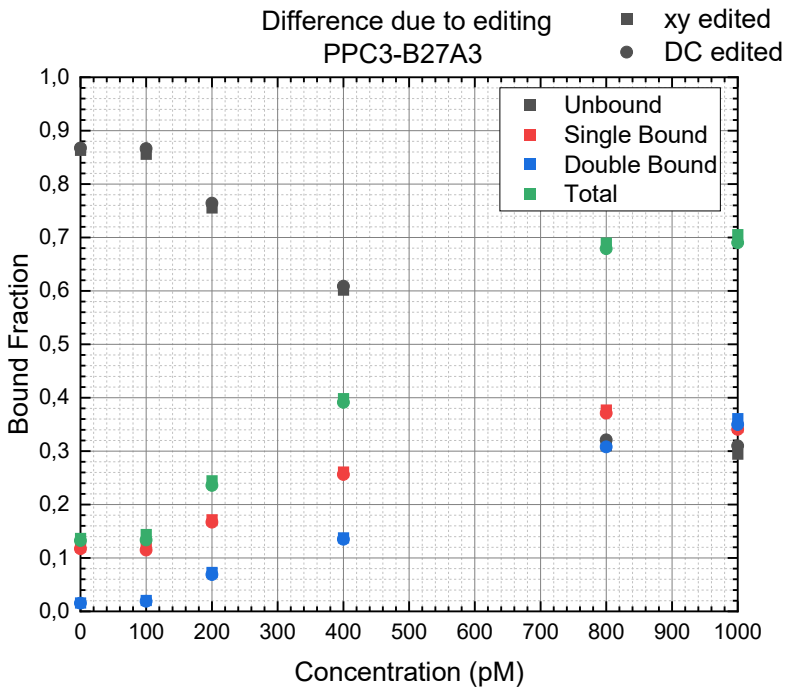

Figure 5.2: The unbound, single, double and total bound fraction plotted for both D and XY edited data.

In order to save computation time, both methods use the 'k' nearest neighbors technique. In order to find a proper value for k, some measurements were ran with different k, as function of time. The result can be seen in Figure 5.3. The result shown here is typical for antibody fBPM. First off, the number of particles with removed frames increases with k. This is logical as k increases, the area of inspection increases. Once k reaches 15 the number of registered particles does not increase anymore. This concludes that the particles do not travel beyond the distance of the fifteenth neighbor and that this would be an appropriate value for k. This might change with fluctuations in particle density. For this reason, most of the data analyzes were done with $k = 20$ or higher.

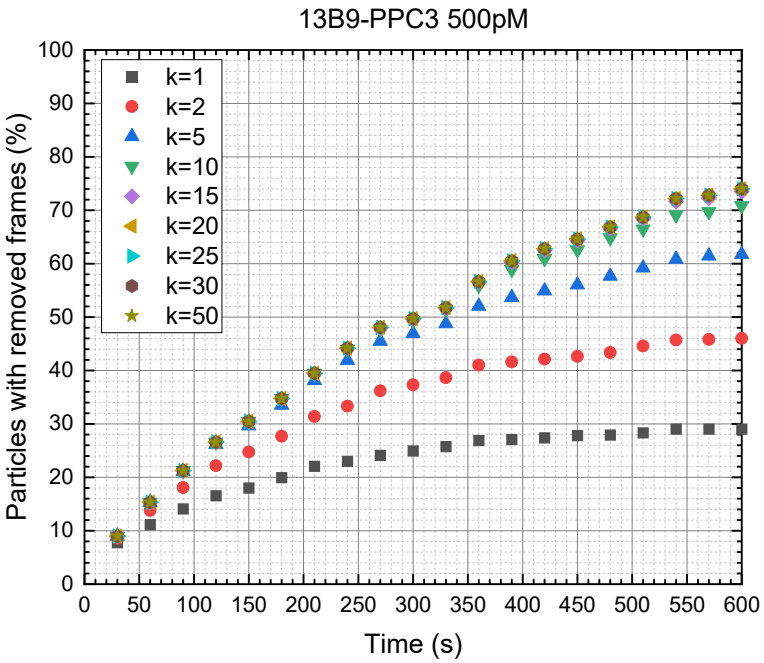

Figure 5.3: The percentage of particles tagged as agglomerated as a function of time, for different values of k nearest neighbors. Done on the experimental data of 13B9-PPC3 with a concentration of 500 pM PCT.

Secondly, it is evident that as more time elapses, more particles display faulty behavior. This is logical as there is more time for particles to find each other and agglomerate. This holds true as long as the timescale at which clusters disintegrate is larger than the measurement time.

After extensive testing, the conclusion can be drawn that once particles cluster together, the clusters do not disintegrate within the measurement time. To test this the starting frame number and the end frame number of the agglomeration is noted down in a list, along with what particle clusters with who. There are zero entries with a end frame number unequal to the total number of frames. This can also be seen in Figure 5.3, where there the number of particles with removed frames is ever increasing, signifying that clusters do not break up.

Having a list of the agglomerated particles including a time at which they start agglomeration ensures that agglomerated time steps can be trimmed. The way this is done is to set the values of the xy-lists on said times to 'Not a Number' (NaN). The THM can than be edited in such a way that it splits the xy-list into two categories: one intact list and one with removed frames. The script then calculates the D for the first list as usual. After which the second list is analyzed, particle per particle, each with a different end frame. Once both loops are concluded, the results are merged, and the script proceeds as per usual.

5.2 Editing data for THM

This editing is an useful addition to the original script, as one can easily see in Figure 5.3, the vast majority of particle at some point cluster with immunoassay fBPM. This way a lot of useful data can be saved.

5.2.1 Bound Fraction

To demonstrate this the results of the original data was compared to that of the edited data. Below the unbound, single and double bound fraction are given for the PPC3-B27A3 antibody pair, are displayed in Figure 5.4. From these images it is plain to see that editing for agglomeration reduces the unbound fraction and thus increases the bound fraction. Editing by removing particles (RP) reduces the unbound fraction to a larger extent than editing by removing frames (RF).

From this the conclusion can be made that the 'faulty' frames mostly consist of unbound and double bound states. Leaving them in would underestimate the single bound fraction. The particles saved by RF compared to RP significantly increase the unbound and single bound fraction, at the cost of the double bound fraction, meaning that these frames consist mostly of single bound and unbound states. In other words, frames that could contain 01 activity. Only removing particles would overestimate the double bound fraction, because the double bound fraction is much higher of RP edited data than that of the original or RF edited.

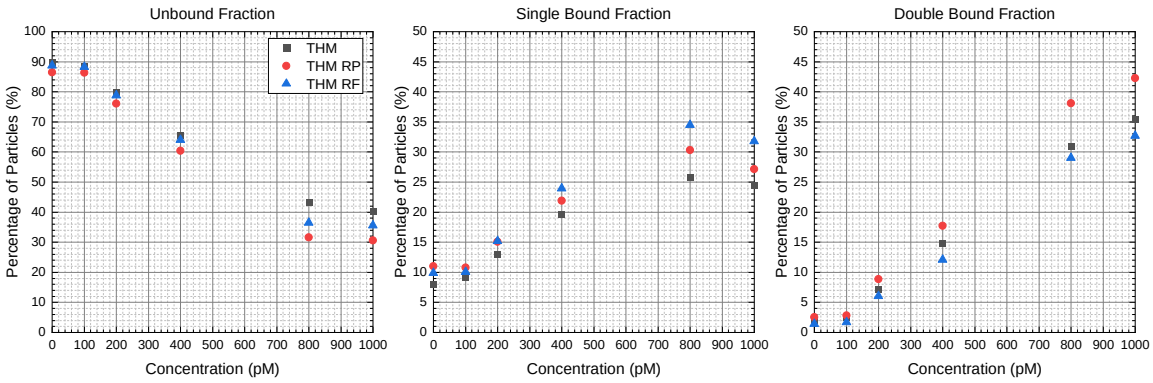

Figure 5.4: A comparison of THM values of the unbound, single and double bound fraction (for the PPC3-B27A3 antibody pair), for the original data compared to that of filtering out frames, or removing particles listed as agglomerated.

Note that all three methods have a distinct drop in single bound fraction at 1000 pM, indicating that the transition from single bound to double bound is more predominant than the unbound to single bound. Since the sandwich immunoassay is a three-phase system and the single bound state being a intermediate state, eventually all particles will end up in the double (or multiple) bound state, if the concentration is high enough. This indicates that for reversible binding the concentration regime should be restricted to 800 pM. Linking this back to the context, 800 pM is a sufficiently high concentration, as the patient would have entered septic shock by then.

5.2.2 Activity

The same comparison is also made for the activity, as can be seen in Figure 5.5. What we would expect is that both the 01 and the 12 activity eventually will saturate, first 01 then 12. The edited data has a much clearer saturation of 01 activity, around 800 pM. This is in accordance with the saturating of single bound fractions seen in Figure 5.4. The double bound fraction does not saturate in the measured concentration interval, so we would expect the 12 activity not to saturate either. This is true for the RP edited data, but the RF edited data seems to saturate at 800 pM. This is based on two data points, so this statement is quite fragile.

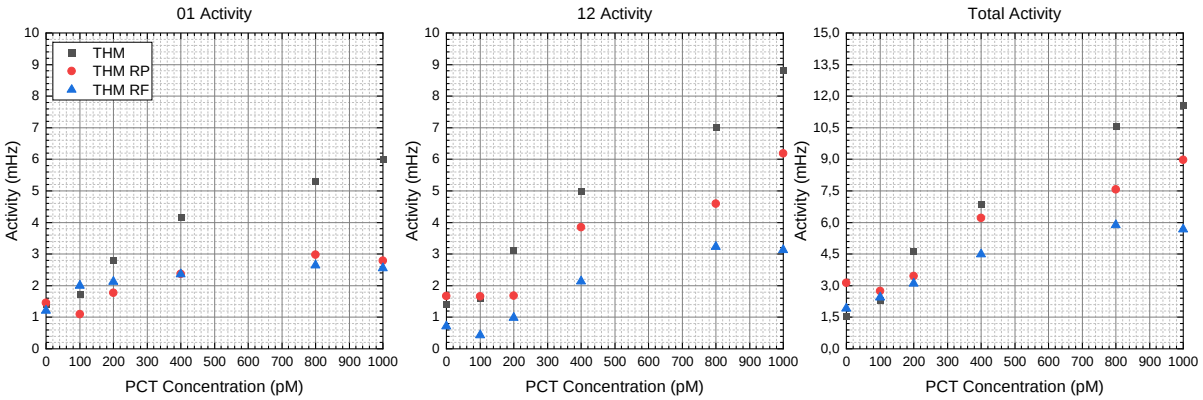

Figure 5.5: A comparison of THM values of the 01, 12 and total activity (for the PPC3-B27A3 antibody pair), for the original data compared to that of filtering out frames (RF), and of removing particles (RP) listed as agglomerated.

What springs to the eye is that editing decreases both the 01 and the 12 activity significantly. Both the frame and particle editing seem to saturate the 01 activity around 800 pM, the frame editing even more so. This is in contrast with the original data, which linearly increases. But in line with the predicted behavior. The 01 activity for THM RF is slightly higher than that of RP for low concentration, but is no longer so at higher concentration. This is in contrast with expectation, based on the fact that the 'saved' frames make up mostly unbound and single bound states. It seems these frames do not contain many events.

The decrease in activity after editing THM data proves that agglomeration induces false events, like predicted earlier. Combined with the increase in bound fraction indicates that many of these false events are short-lived and that the "true" bound states have a longer longevity. This is true for both types of editing. The reason the RF editing has lower activity than RP editing might be that the frames saved, of otherwise removed particles, have a below average activity.

5.3 Editing for DLM

The DLM method has a much more complex structure and is thus less easily changed. The main advantage for editing by removing particles compared to removing frames, is that it works exactly the same as with original data. The only difference is in xy-lists. For lack of time, there was no attempt made to check the effectivity of editing by removing frames for DLM, all edited DLM data shown is done by removing frames. To be able to compare this, also the THM and edited THM data is inserted. Note that this is also edited by particle removal, to be able to compare its impact on processing style.

5.3.1 Bound Fraction

In Figure 5.6 the unbound, single and double bound fraction are displayed for both original and edited data. Like before (Figure 4.1), there is hardly any difference in unbound fraction, this too holds for the edited data. What is noticeable is that the removing particles lowers the unbound fraction, but this happens for both THM and DLM in equal amounts.

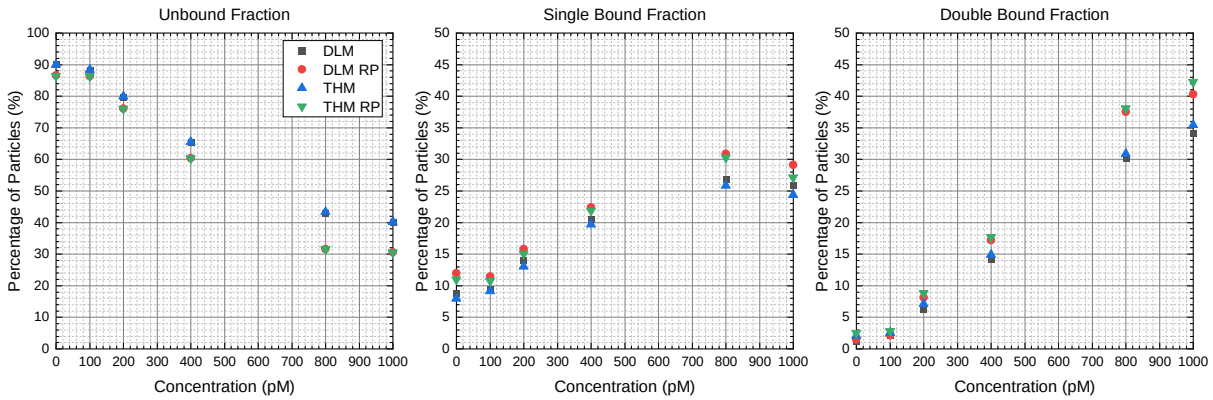

Figure 5.6: Unbound, single bound and double bound fraction for PPC3-B27A3 for DLM and THM both before and after editing.

This trend is continued in both the single and the double bound fraction. There is an increase in both bound fractions after editing, but this is similar for both methods. There is an increase up to 5% to either bound fraction. There is no real difference in increase per method, apart from the initial difference mentioned earlier.

5.3.2 Activity

Looking at the activities in Figure 5.7, it can be seen that the total activity is hardly changed by removing particles for DLM, in contrast with THM. This result is due to significant decrease in 01 activity counterbalanced by a small increase in 12 activity. This indicates that the removed particles played a larger role in 01 activity than in 12 activity. Note that an increase in activity after editing is possible due to the fact that activity is inversely proportional to the number of particles. The result of the 13B9-B27A3 antibody pair (Figure A.8) is a lot messier but shows a similar pattern.

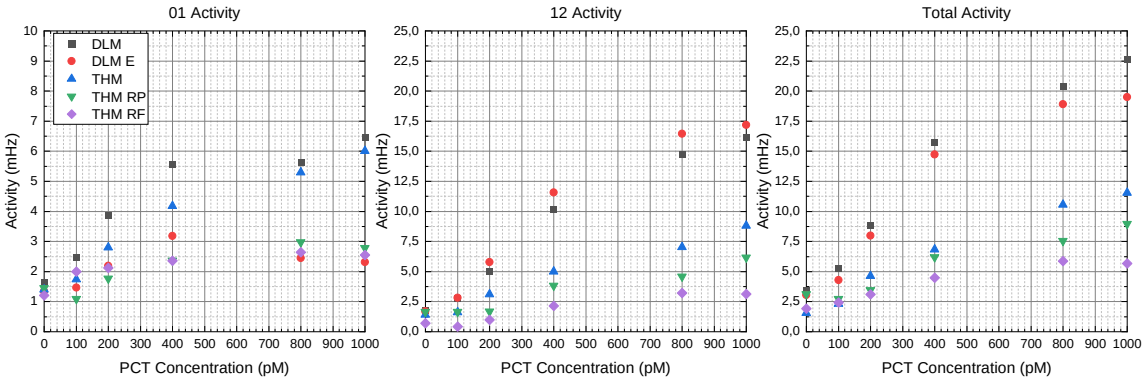

Figure 5.7: A comparison of DLM and THM values of the 01, 12 and total activity (for the PPC3-B27A3 antibody pair), for the original data compared to that of the RP and RF edited data.

Once again, a saturation can be seen in the 01 activity around 400 pM, much like the pattern seen in edited THM data. The values of the edited DLM 01 activity are very similar to those of the edited THM. However, unlike the edited THM data, 01 activity here eventually turns into a decline. This is indicative of the predicted shift from 01 activity to 12 activity as the majority of the particles are in a bound state.

We also see that the edited DLM data has an increased 12 activity, meaning that the 'false' events are mostly 01 events in the DLM output. By removing 'faulty' particles the 12 activity increase, this can only be the case if the number of particles decreases faster than the number of 12 events. To back up this hypothesis we can look at the results or the 13B9-B27A3 pair. Because it was measured multiple times, the conclusions drawn from this pair has more decisiveness.

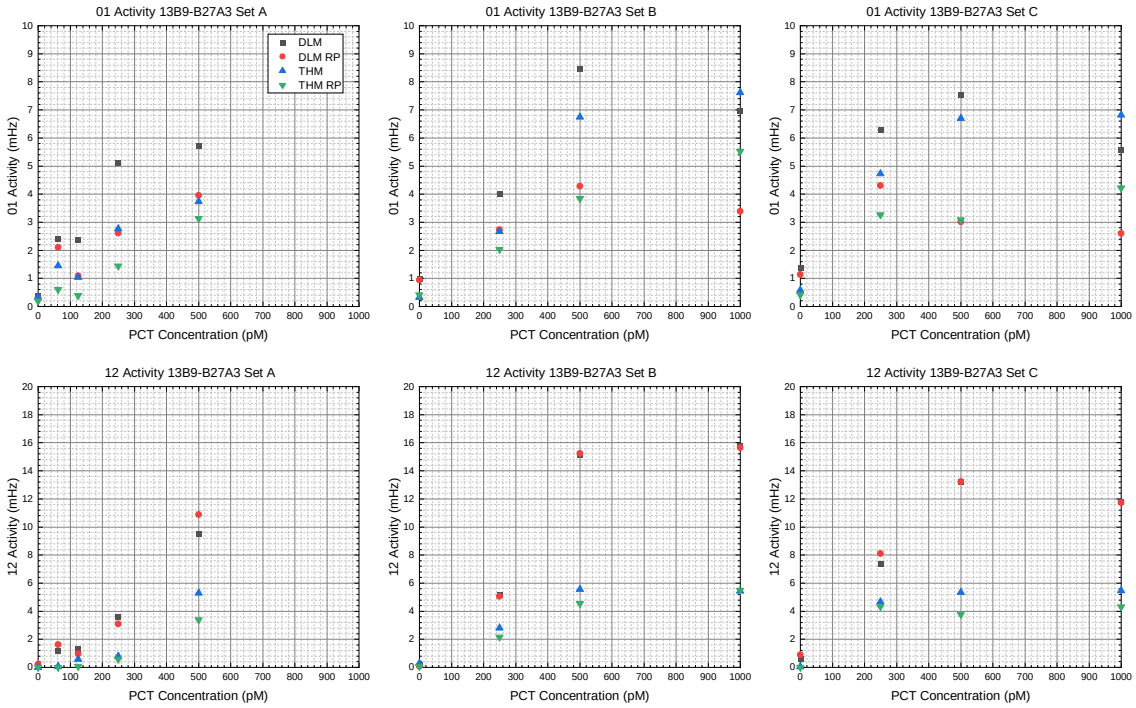

Figure 5.8: A comparison of DLM and THM values of the 01 and 12 activity (for the 13B9-B27A3 antibody pair), for the original data compared to that of removing particles (RP) listed as agglomerated.

Omitting the THM editing based on removing frames for now, the following conclusions can be drawn. In general, removing particles has much more impact on 01 activity than on 12 activity. Removing particles significantly reduces 01 activity at higher concentrations, this holds for both THM and DLM. Editing DLM does not lead to an increase in 12 activity, it rather stays unaltered. Editing THM does slightly decrease the 12 activity. Looking specifically at the effect of editing on DLM activity, the hypothesis is confirmed. The fact that the 12 activity slightly decreases after editing can be discredited because it is only a minor decrease, and that we have previously established the DLM to be better at double bound state recognition. Concluding that indeed, most of the false events are 01 events. Neglecting to remove agglomeration would artificially increase the 01 activity.

To check the accuracy of these measurements, some of the sub-figures from Figure 5.8 have been merged into Figure 5.9. From this result it is clear that, like with the unedited data, the 12 DLM activity has high standard deviation. Whereas the 12 THM activity does not, neither do both 01 activities. It could be that the standard deviation of the 12 activity is higher for DLM RP, because the signal is higher. The deviation could simply scale with the amplitude of the signal. Alternatively, the increase in standard deviation might be attributed to the low mobility filter, mentioned before. The difference in the number of stuck particles between different batches would influence the value of the 12 activity significantly.

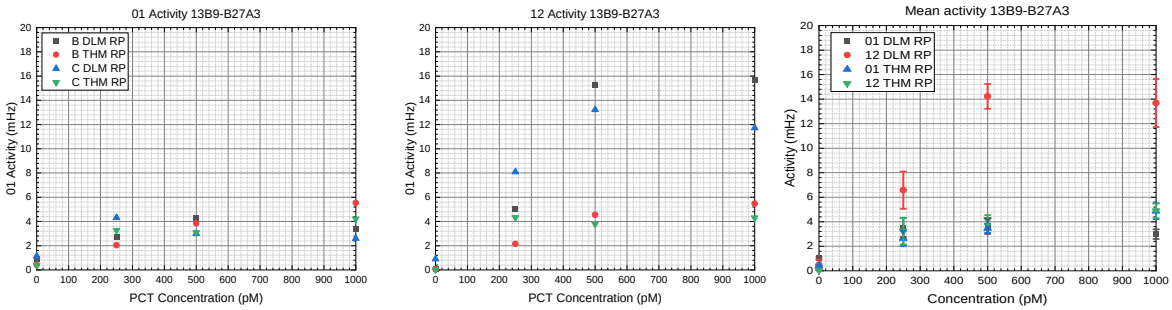

Figure 5.9: The respective 01, 12 and mean activities for the 13B9-B27A3 antibody pair, comparing original and particle edited (RP) DLM and THM results of set B and C.

5.3.3 State Lifetimes

In Figure 5.10 the state lifetimes of the PPC3-B27A3 pair have been plotted. The edited unbound state lifetimes are very similar to the original unbound state lifetimes, only a bit higher. This is also reflected in the slightly higher EC_{50}, in Table 5.1. The lifetimes of THM are a bit higher than DLM, the same holds for the edited data. We can see that removing particles increases all τ_2's slightly, meaning bonds live longer. This is logical as we have just seen in that the 01 activity decreases.

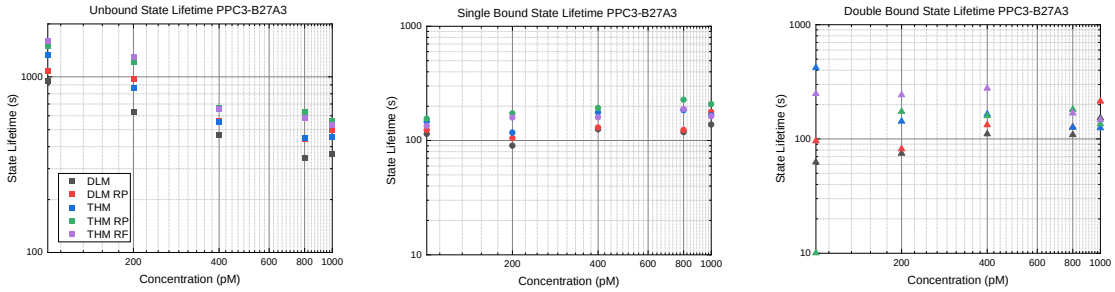

Figure 5.10: A comparison of DLM and THM values of the state lifetimes (for the PPC3-B27A3 antibody pair), for the original data compared to edited by removing particles (RP) and by removing frames (RF).

The single bound state lifetime does not change much, removing particles slightly increases it. The same holds for the double bound state lifetime, but the spread here is a bit larger. The double bound state lifetime only becomes reliable for THM after 200 pM, since that method finds it hard to get enough data points before then. The DLM however, does have enough datapoints at 100 pM and can therefor properly predict the double bound state lifetime. This goes to show once more that the double bound state detection of DLM is superior to that of THM.

One thing to note is that edited DLM still displays a small increase in double bound state lifetime, with increasing concentration. This is once again due to the immobilized particles. These are not filtered out by the cropping script. This can be included quite simply, but the effect of this is left as a future investigation.

The increase in state lifetimes (both bound and unbound) after editing is a direct consequence of the decrease in activity. Less state switches means the time spend in said state is on average longer. This proves that without filtering out agglomerations, the lifetime is underestimated.

5.3.4 EC_{50}

To conclude the results, the EC_{50}'s for all responses, for all methods have been collected in Table 5.1. The standard deviation here is that of the Hill fit.

Table 5.1: The EC50's of the state fractions, activities and unbound state lifetime for all analysis methods

EC50's	DLM	DLM RP	THM	THM RP	THM RF
SB Fraction	314 ± 39,6	357 ± 62,4	304 ± 42,5	333 ± 71,3	345 ± 60,5
DB Fraction	651 ± 109	569 ± 158	695 ± 157	630 ± 186	683 ± 242
UB Fraction	481 ± 71,0	467 ± 100	482 ± 69,7	467 ± 100	470 ± 102
01 Activity	204 ± 37,8	196 ± 2,43E6	338 ± 73,4	300 ± 83,8	349 ± 1460
12 Activity	400 ± 27,6	346 ± 4,05	465 ± 328	523 ± 700	361 ± 39,4
Total Activity	350 ± 44,1	295 ± 0,581	403 ± 179	382 ± 78,9	328 ± 58,8
UB State Lifetime	173 ± 15,9	262 ± 26,6	119 ± 2,71	145 ± 33,3	129 ± 33,1

In general, the EC_{50} values for the RF edited THM lie closer to the DLM RP edited, the THM RP edited. Indicating that this method might be more accurate than THM RF, but since even the DLM data is experimental, this cannot be said with certainty.

We see that the EC_{50} values for the state fractions of THM and DLM are very similar, even after editing. The moment the unbound state lifetime starts to decrease is the same moment the 01 activity starts. This is approximately true for the DLM and to a lesser extend the DLM RP as well. This is less so the case for the THM values, as these UB state lifetimes saturate much earlier than the 01 activity. When the 01 activity starts to increase, the SB fraction should too. This does fit for the THM, THM RP and THM RF values, but is less so the case for the DLM and DLM RP. This is likely due to the immobilized particles deflating the DLM 01 activity.

The extremely high standard deviations for the edited DLM 01 activities mean that that data does not behave like a normal dose response curve, and can therefor not be fitted with the Hill equation. This is understandable, because of the drop in 01 activity after editing, seen with RP edited DLM. This is probably due to the lack of the LMF. The high deviation in 12 activity for THM and THM indicate that these do not accurately predict the activity, but we have already concluded that the DLM is better in distinguishing double bound states. Fortunately the DLM RP can still accurately the 12 activity, so this problem is not only because of RP editing. What is interesting is that the THM RF can predict the 12 activity very similar to that of DLM RP, while the THM RP is way off.

These half maximal effective concentrations are only a guideline off course. Small changes in response values deeply influence the Hill curve and implicitly the EC_{50}. It does give insight to the general response behavior, so should only be used as a guideline.

5.4 Conclusions Editing

The fBPM immunoassay system is prone to agglomeration, an artefact where two particles merge together and are registered as one. When they fuse together, their lifetimes exceed the measurement time. The imaging software is unable to distinguish the particles from each other, so the location of the particles intermittently switches to that of the other particle. This generates a lot of false events. These events are mainly 01 events, as seen in a reduction in 01 activity and increase in unbound state lifetime after editing. Agglomeration reduces the bound fraction, because it affects otherwise active particles.

This can be circumvented by simply removing particles which display this behavior from the analysis. Another option is to remove frames in which agglomeration occurs, this is more sophisticated and leads to more natural activities, but is yet to be implemented in the DLM.

The combination of a decrease in activity and an while the bound fraction also increase must mean that the particles stay longer in the bound state once they bind. This can also be seen in the increase of bound state lifetime after editing. This means that without editing the state lifetime is underestimated. On the one hand this is a negative result, as the number of binding events is even lower than expected, but this also means that the chance to miss events is lower, as particles stay in their state for longer, making the activity derivation more reliable.

6 Discussion

The PPC3-B27A3 antibody pair was tested and found to have a low concentration response. However, this pair was only successfully measured once. To confirm its reproducibility and test its precision, multiple measurements need to be done. A lot of conclusions are drawn from a single measurement, which might be unrepresentative for this antibody pair. It would also be best to measure at more different PCT concentrations. Adding more low concentrations and higher concentrations will make it easier to tell trends from a logarithmic plot, such as state lifetime. For linear plots, the gap between 400 and 800 pM is quite high, especially since most of the change in bound fraction and activity happens there. Adding for instance 600 pM would to more accurate bound fraction and activity derivations.

The unbound state lifetime is extrapolated from a double exponential fit over the survival curve of the length of unbound lifetimes. For low concentrations, this exceeds the measurement times. For more reliable results, it might be better to extend the measurement time to 1200 seconds, or 20 minutes. Maybe the unbound state lifetime is even higher, but is cut of because of the measurement duration.

Due to lack of time and lack of knowledge on the coding language Python, the DLM data could only be edited such that agglomerated particles are removed from the input. As mentioned before, removing frames gives better results than removing entire particles. In order to compare the THM edited data with the DLM edited data, the frame editing way could not be used. This would not give a fair comparison.

The comparison of the DLM versus the THM is warped anyway, since the THM does include a low mobility filter and the DLM does not. This low mobility filter allows the THM to have a better grasp at the number of activite particles. This improves the accurate state fraction and activity determination. To be able to look objectively at the quality of state determination, this has to be taken in account. The next step is to either include this in the DLM or to remove the low mobility particles from the xy-list.

7 Summary

As we have seen, the fBPM immunoassay is a viable option to measure the concentration of procalcitonin. The PPC3-B27A3 antibody pair is a good choice, since it shows response at low concentration. The linear regime of unbound fraction and activity starts around 100 pM, possibly even lower. This is the concentration where local infections become more severe, and therefore a crucial concentration to start treating the infection. This antibody pairing needs some further research to test its accuracy and precision but shows great prospect.

The analysis of of experimental data can be done either with the thresholding model or with the deep-learning model. This paper shows that the DLM is better at distinguishing single bound from double bound states. It also registers higher activity, increasing the signal to noise ratio on this response. This all leads to an increased robustness to heterogeneity in particles. The one advantage the THM has over the DLM at the moment is the inclusion of a low mobility filter, removing immobile particles and improving the precision of the activity determination. This would improve the performance of the DLM even further.

During experimenting, we stumbled upon an artefact in the imaging software due to the agglomeration of particles in fBPM immunoassays. This agglomeration leads to false events, artificially raising the activity. It also reduces the bound fraction, reducing the signal to noise ratio here and therefor the quality of the calibration curve. The real state lifetimes are underestimated without taking in account the agglomeration. Because the amount of agglomerated particles per measurement is somewhat random, this drastically reduces the precision of the measurement and this of concentration derivation.

This is circumvented by filtering out the particles which contribute to this artefact. This is done for both DLM and THM. The results of which reduce the activity and a increase in bound fraction, particles stay bound longer. This confirms the hypothesis that agglomeration introduces false events. The problem with removing particles is that this removes useful data too. It has been found that once agglomeration takes place, it does not unbind within the measurement time. This means that all the frames before the collision are still useful. Less frames mean lower signal to noise. To alleviate this problem the editing script for THM was altered to include pre-collision frames and only remove the agglomerated frames. Unfortunately, this was not done for DLM, due to a lack of time. But this is a way the performance of the DLM can be increased even further.

Bibliography

[1] Hyuck Lee. Procalcitonin as a biomarker of infectious diseases. *The Korean Journal of Internal Medicine*, 28 (3):285, 2013. doi: 10.3904/kjim.2013.28.3.285.

[2] Sun Ju Kim, Sumg Oh Hwang, Yong Won Kim, Jun Hyeok Lee, and Kyoung-Chul Cha. Procalcitonin as a diagnostic marker for sepsis/septic shock in the emergency department; a study based on sepsis-3 definition. *The American Journal of Emergency Medicine*, 37(2):272–276, February 2019. doi: 10.1016/j.ajem.2018.05.047.

[3] Chris Ann Dague. Procalcitonin measurement for effective antibiotic stewardship, Sep 2017. URL `https://www.technologynetworks.com/diagnostics/articles/procalcitonin-measurement-for-effective-antibiotic-stewardship-307670`.

[4] Emiel W. A. Visser, Junhong Yan, Leo J. van IJzendoorn, and Menno W. J. Prins. Continuous biomarker monitoring by particle mobility sensing with single molecule resolution. *Nature Communications*, 9(1), June 2018. doi: 10.1038/s41467-018-04802-8.

[5] Clinical and Inflammation Research Area. *Procalcitonin (PCT) Technotes*. HyTest Ltd, Turku, Finland, 2019.

[6] Anna Marie Chang and Morgan Oakland. Biomarkers in shortness of breath. In *Biomarkers in Cardiovascular Disease*, pages 129–137. Elsevier, 2019. doi: 10.1016/b978-0-323-54835-9.00012-0.

[7] Immunoglobulin structure and classes, 2021. URL `https://www.thermofisher.com/nl/en/home/life-science/antibodies/antibodies-learning-center/antibodies-resource-library/antibody-methods/immunoglobulin-structure-classes.html`.

[8] Gestur Vidarsson, Gillian Dekkers, and Theo Rispens. IgG subclasses and allotypes: From structure to effector functions. *Frontiers in Immunology*, 5, October 2014. doi: 10.3389/fimmu.2014.00520.

[9] Heino Prinz. Hill coefficients, dose–response curves and allosteric mechanisms. *Journal of Chemical Biology*, 3(1):37–44, September 2009. doi: 10.1007/s12154-009-0029-3.

[10] Polyclonal vs. monoclonal antibodies, Dec 2017. URL `https://www.ptglab.com/news/blog/polyclonal-vs-monoclonal-antibodies/`.

[11] Jipke Snellen van Vollenhoven. Antibody affinity screening for procalcitonin detection using bpm. Master's thesis, Eindhoven University of Technology, Eindhoven, Netherlands, August 2020.

[12] M.H.Bergkamp. Real-time data processing in continuous biomarker monitoring by particle mobility sensing (bpm). Master's thesis, TU Eindhoven, 2019.

[13] Koen J. A. Martens, Arjen N. Bader, Sander Baas, Bernd Rieger, and Johannes Hohlbein. Phasor based single-molecule localization microscopy in 3d (pSMLM-3d): An algorithm for MHz localization rates using standard CPUs. *The Journal of Chemical Physics*, 148(12):123311, March 2018. doi: 10.1063/1.5005899.

[14] Moving mean, 2021. URL `https://nl.mathworks.com/help/matlab/ref/movmean.html`.

[15] John Paul Mueller and Luca Massaron. *Machine Learning For Dummies*. For Dummies, 1st edition, 2016. ISBN 1119245516.

[16] P.W.T. Bult. Design and feasibility of deep learning algorithms for biosensing by particle mobility. Master's thesis, Eindhoven University of Technology, 2021.

[17] Christopher Olah. Understanding lstm networks, Aug 2015. URL https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[18] Christopher Olah. Conv nets: A modular perspective, July 2014. URL https://colah.github.io/posts/2014-07-Conv-Nets-Modular/.

[19] E.W.A. Visser. *Biosensing based on tethered particle motion.* PhD thesis, Eindhoven University of Technology, 2017.

A Appendix

A.1 Bound fraction other pairs

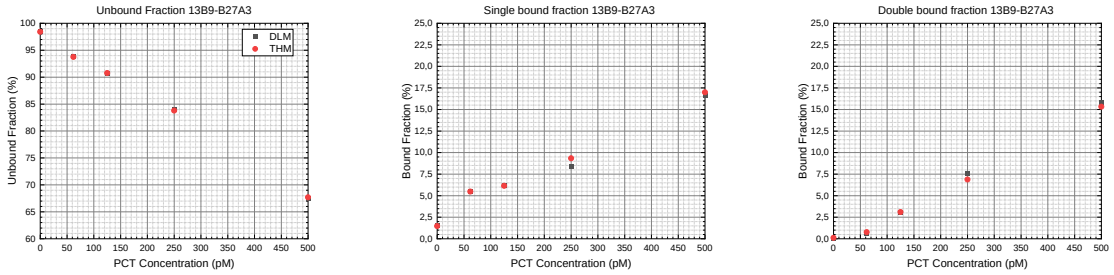

Figure A.1: The unbound, single bound and double for the 13B9-B27A3 antibody pair, comparing THM results versus DLM results.

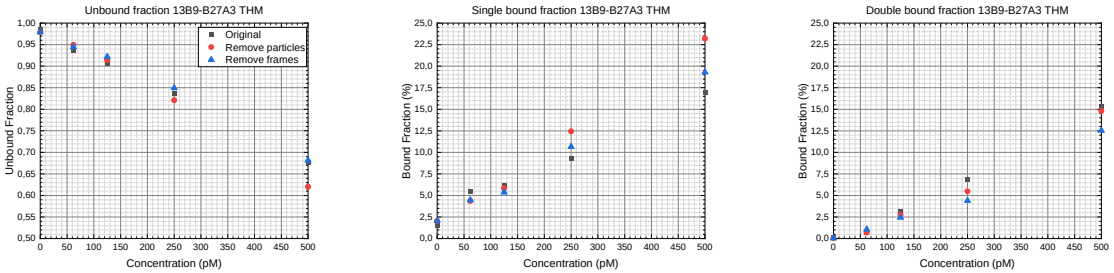

Figure A.2: The unbound, single bound and double for the 13B9-B27A3 antibody pair, comparing THM results versus edited THM results by removal of particles (RP) and frames (RF).

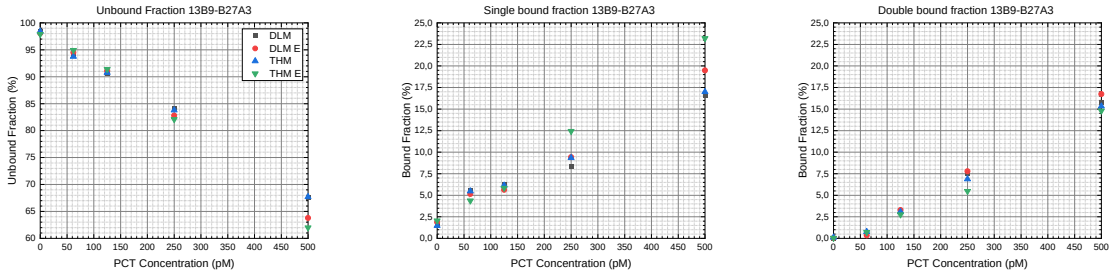

Figure A.3: The unbound, single bound and double for the 13B9-B27A3 antibody pair, comparing DLM and THM results versus edited DLM and THM results by removal of particles.

A.2 Low mobility filter

Below the effects of using the low mobility filter are displayed on the diffusion coefficient distribution.

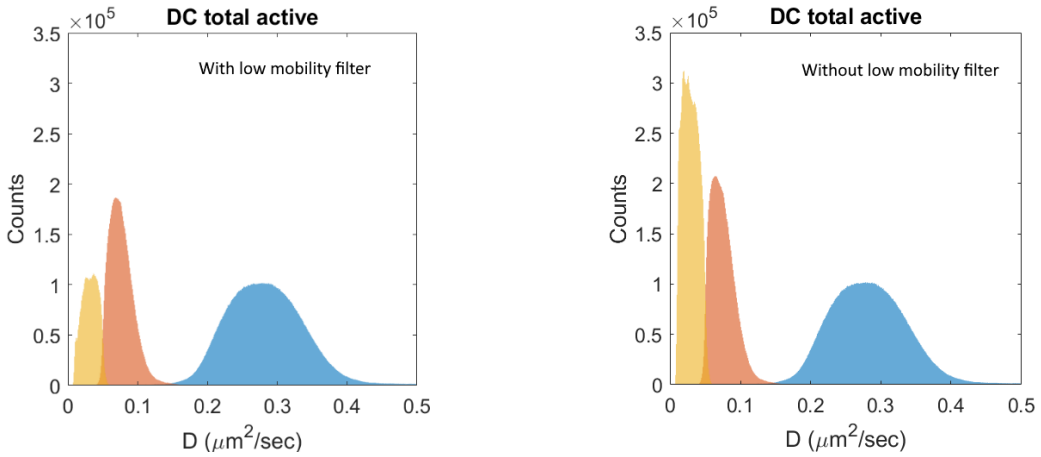

Figure A.4: The influence of filtering low mobility particles on the diffusion coefficient distribution per state for the THM processed 800 pM PPC3-B27A3

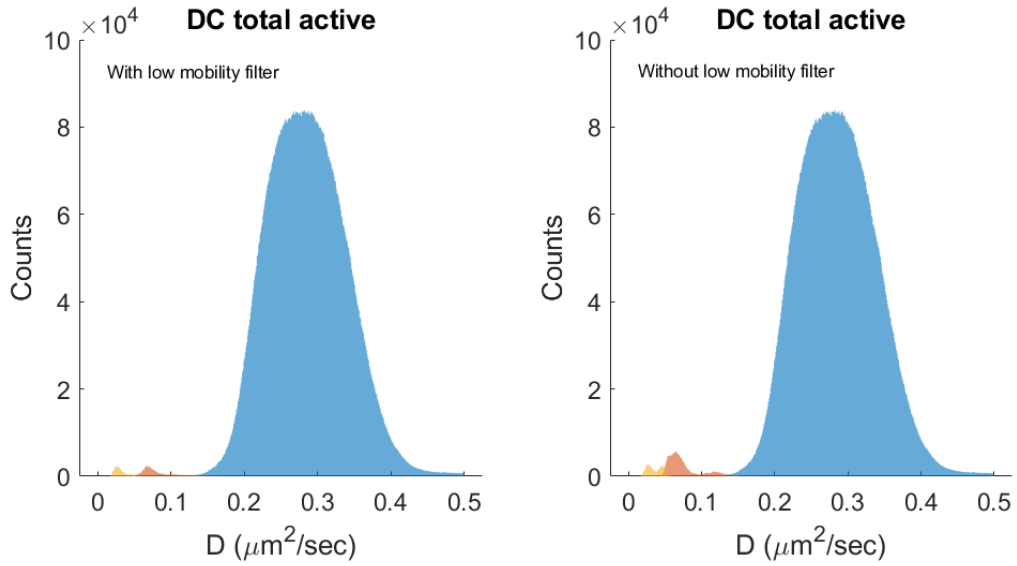

Figure A.5: The influence of filtering low mobility particles on the diffusion coefficient distribution per state for the THM processed 800 pM PPC3-B27A3 control measurement

A.3 Activities other pairs

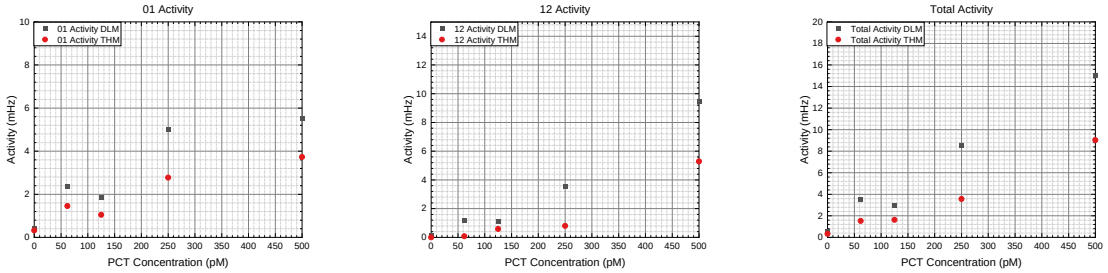

Figure A.6: The respective 01, 12 and cumulative activities for the 13B9-B27A3 antibody pair, comparing DLM results versus THM results.

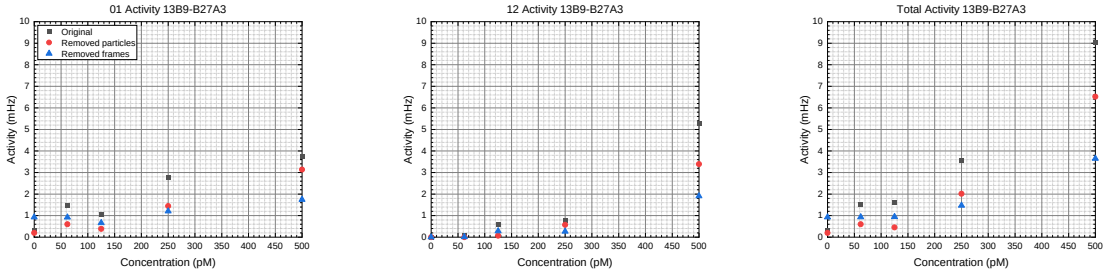

Figure A.7: The respective 01, 12 and cumulative activities for the 13B9-B27A3 antibody pair, comparing THM results versus particle (RP) and frame (RF) edited THM results.

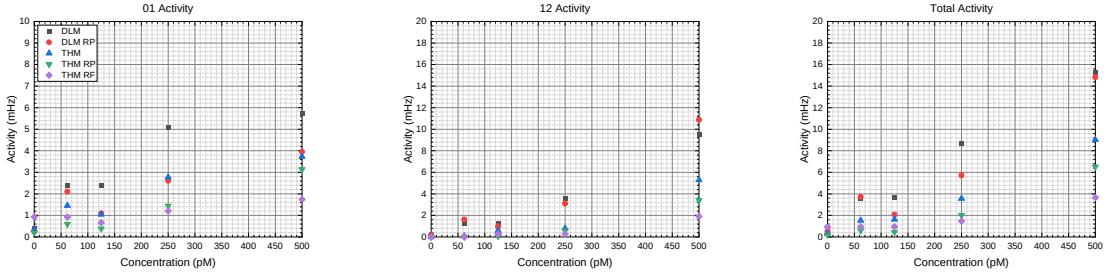

Figure A.8: The respective 01, 12 and cumulative activities for the 13B9-B27A3 antibody pair, comparing original and particle edited (RP) DLM results versus the original, particle edited and frame edited (RF) THM results.

Matlab Scripts

A.4 Cropping Script

This script was designed to find agglomerations. The outputs are a list of all particles agglomerating, a graph of how many particles are removed as a function of time, a new xylist without the removed particles and a new xylist without the frames in which agglomeration takes place.

```
clear all;close all;
tic
%% Open data
[FileName, PathName] = uigetfile({'*.txt'}, 'Select the data files you want to
    analyze')
FilePath = strcat(PathName,FileName);
xylist=importdata(FilePath);
NameExtension=erase(FileName,'.txt');
disp(NameExtension)
flaglist=[];
k=25;
mislocplot=20;
%% Parameters
PixelSize=0.588;
N_particles=size(xylist,1)
Np_before=N_particles;
xylist=xylist*PixelSize;

%%% Conversion frames to seconds
TotalFrames=numel(xylist(1,3:2:end));
FrameStep=1;
FrameRate=60; %Hz
TimeStep=FrameStep/FrameRate; %seconds
Time=[1:TotalFrames]*(FrameStep/FrameRate);

trajectory_x=xylist(:,1)+xylist(:,(3:2:end)); %convert differential coordinates
trajectory_y=xylist(:,2)+xylist(:,(4:2:end));

%%
 binlist=[];
for j=1:size(xylist,1)
startloclist=[xylist(:,1) xylist(:,2)]; %determine starting locations
startlocpart=[xylist(j,1) xylist(j,2)];

Idx=knnsearch(startloclist,startlocpart,'K',k+1);
nearnb=Idx(2:1:k+1).'; %pick 15 nearest neighbours

    for jj=1:k
    diff_x=trajectory_x(nearnb(jj),:)-trajectory_x(j,:); %compare locations
    diff_y=trajectory_y(nearnb(jj),:)-trajectory_y(j,:);
    bin_diff_x=abs(diff_x)<10^(-10); %check if there is a moment where coordinates
        overlap
```

```
    bin_diff_y=abs(diff_y)<10^(-10);
    doublebin=bin_diff_x+bin_diff_y;
        t_1=find(doublebin==2,1,'first'); %start of overlap moment
        t_2=find(doublebin==2,1,'last');
        totaldoublesum=doublebin>1;
    if t_2-t_1+1==sum(totaldoublesum) & t_2-t_1+1>6000
        binlength=t_2-t_1+1;
        binlist=[binlist; j nearnb(jj) t_1 t_2];
    elseif t_1>30000 & t_2-t_1+1==sum(totaldoublesum) & t_2-t_1+1>600 %check for
        different bond lengths
        binlength=t_2-t_1+1;
        binlist=[binlist; j nearnb(jj) t_1 t_2];
    elseif sum(totaldoublesum(t_1:1:t_2))>60 & t_2~=TotalFrames
        disp(['t_2 is not end particle: ',num2str(j)])
    elseif t_2-t_1+1>600 & sum(totaldoublesum(t_1:1:t_2))>60
        if t_2==TotalFrames & find(doublebin~=2,1,'last')<TotalFrames-6000
            t_2=find(doublebin~=2,1,'last')+1;
            binlength=t_2-t_1+1;
            binlist=[binlist; j nearnb(jj) t_1 t_2];
        else
        disp(['not continious problem: particle ',num2str(j),' and ',num2str(nearnb
            (jj))])
        end
    end
    end
end

%% write flaglist
flaglist=unique(binlist(:,1)); %particles that agglomerate: for DLM
size(flaglist)
DLMLIST=flaglist'-1;
% dlmwrite([NameExtension,'DLMLIST.txt'],DLMLIST)

%% mislocalization as function of measurement time
mlplot=[];
for ii=1:mislocplot
A=ii*TotalFrames/mislocplot;
mlplot=[mlplot; A*(FrameStep/FrameRate)];
B(ii)=numel(find(binlist(:,3)<A));
end

figure(1)
scatter(mlplot,B,15,'r','filled')
ax=gca;
ax.Title.String= 'Np removed as function of time';
ax.XLabel.String='Measurement time (s)';
ax.YLabel.String='Number of particles removed';
legend(['Np\_before=',num2str(Np_before)],'Location','northwest');
grid minor;
%% revert back to pixels
xylist=xylist/PixelSize;
```

```
%% save precolision data
xylist_special=xylist;
for j=1:size(binlist)
    rownr=binlist(j,1);
    x_1=2+2*binlist(j,3);
    x_2=2+2*binlist(j,4);
    xylist_special(rownr,x_1:1:x_2)=NaN;
end
%% write new xylists
    xylist_new=xylist;
 for j=1:size(flaglist)
     rownr=flaglist(j);
     xylist_new(rownr,:)=NaN;
 end
xylist_new(any(isnan(xylist_new), 2), :) = []; %removes NaN rows
xylist_edited=xylist_new;
%% save new xylists
Np_after=size(xylist_edited,1);
dlmwrite([NameExtension,'_EDITED.txt'],xylist_edited) %removed particles
dlmwrite([NameExtension,'_Special.txt'],xylist_special) %removed frames
  toc
```

A.5 THM Post-processing Script

This is the thesholding-model, this script transforms xy-lists into results by calculating the diffusion coefficient, and assigning particles to states according to their diffusion values. The output is amongst others: bound fraction, activities and state lifetimes.

```
%%%Standalone f-BPM analysis script for analysing single data files
clear all;close all;
tic
%% Load xy-lists
[FileName, PathName] = uigetfile({'*.txt'}, 'Select the data files you want to
    analyze')
FilePath = strcat(PathName,FileName);
xylist=importdata(FilePath);
NameExtension=erase(FileName,'.txt');
disp(NameExtension)
%% Parameters
PixelSize=0.588;
N_particles=size(xylist,1);
xylist=xylist*PixelSize;


%%% Conversion frames to seconds
FrameRate=60; %Hz
TotalFrames=numel(xylist(1,3:2:end));
FrameStep=1;
TimeStep=FrameStep/FrameRate; %seconds
Time=[1:TotalFrames]*(FrameStep/FrameRate);
Time_seconds=TotalFrames/FrameRate;%total measurement time in seconds

%%%Specify the axis limits for plotting to make data easily comparable.
DC_plotlimit=0.7; % um^2/sec


%%% Set thresholds for average DC based event detection:
Unbound_threshold=0.12; %um^2/sec; any average DC above this value will be assigned
     as unbound
MVbound_threshold=0.04; %um^2/sec; any average DC below this value will be assigned
     as multivalent bound
Stuck_threshold=0.02;

%%%Values used for plotting the events signal over the DC time trace
unbound_value=0.6;
sbound_value=0.55;
mbound_value=0.5;
%%%Specify a minimum length for an event
Min_eventlength=120; %frames


%%% Specify the type of fits that are used for fitting events and lifetimes
Events_fittype='normal';
```

```
Lifetimes_fittype='exp2';

%% Split xy-list into two lists if frames have been removed
xylistedited=[];
xylistnew=[];
for i_Trajectory=1:size(xylist,1)

        if sum(isnan(xylist(i_Trajectory,:)))>1
     xylistedited=[xylistedited; xylist(i_Trajectory,:)];
    else
    xylistnew=[xylistnew; xylist(i_Trajectory,:)];
        end
end
xylist=xylistnew;

%% Diffusion constant complete trajectory
%%% Calculate the average diffusion coefficient for each particle using the
%%% mean squared displacement
for ii=1:size(xylist,1)
    x=(xylist(ii,3:2:end));
    y=(xylist(ii,4:2:end));
    n_frames=numel(x);

    maxdt=10;
    MSD=zeros(1,maxdt+1);
    MSD(1)=0;
    for dt=1:maxdt
        SD=zeros(1,n_frames-dt);
        for l=1:n_frames-dt
            SD(l)=(x(l+dt)-x(l)).^2+(y(l+dt)-y(l)).^2;
        end
        MSD(dt+1)=mean(SD);
    end
    dtlist=0:maxdt;
    try
    myfit=fit(dtlist',MSD','poly1');
    Dtrajectory(ii)=myfit.p1*FrameRate/4;
    catch
    end
end

%% Diffusion coefficient as function of time
%%% calculate the diffusion coefficient over time for each particle
%%% Setting some parameters:
MeasurementWindow=120; %sliding window length in number of frames
DiffmeanWindow=60; %sliding window for mean DC calculation in number of frames
plot_number=1;
plot_max=11; %maximum number of particles plotted in the for loop
events_avDC=zeros(size(xylistnew,1),TotalFrames)*NaN;
events_avDC2=zeros(size(xylistedited,1),TotalFrames)*NaN;%initialize events matrix
unbound_lifetimes=[];
```

```
sbound_lifetimes=[];
mvbound_lifetimes=[];
zero_one_events1=[];
unbound_lifetimes2=[];
sbound_lifetimes2=[];
mvbound_lifetimes2=[];
zero_one_events2=[];
one_two_events1=[];
one_two_events2=[];

Dtotalspecial=[];
for j=1:size(xylist,1) %loop over each particle
    x=(xylist(j,3:2:end));
    y=(xylist(j,4:2:end));
    checkval=0;

    maxdt=10;
    DtraceWindow=zeros(maxdt,numel(x)-MeasurementWindow+1);
    weight=zeros(1,maxdt);

    for dt=1:maxdt
        SDtrace=(x(1+dt:end)-x(1:end-dt)).^2+(y(1+dt:end)-y(1:end-dt)).^2; %squared
            displacement

        SDtraceWindow=movmean(SDtrace,MeasurementWindow-dt,'Endpoints','discard');
            %mean squared displacement

        DtraceWindow(dt,:)=SDtraceWindow*FrameRate/(4*dt); %diffusion coeff for
            certain dt

        Vrel=dt*(2*dt^2+1)/(MeasurementWindow-dt+1); %calculate relative variance
        weight(dt)=1/Vrel; %assign weights to data points (rows) DtraceWindow
    end
    sumweight=sum(weight);
    weight=weight/sumweight;
    weight=weight';

    Dtotal(j,:)=sum(DtraceWindow.*weight); %calculate diffusion coefficient

    %% State assignment based on DC
    if Dtrajectory(j)>Stuck_threshold    %low mobility filter
        for nf=1:size(Dtotal,2)-DiffmeanWindow %number of frames to slide the
            window over
            if mean(Dtotal(j,nf:(nf+DiffmeanWindow-1)))>Unbound_threshold
                events_avDC(j,nf+DiffmeanWindow/2)=unbound_value;
                %assign value to position in middle of sliding window to match DC
                    time trace
            elseif mean(Dtotal(j,nf:(nf+DiffmeanWindow-1)))<MVbound_threshold
                events_avDC(j,nf+DiffmeanWindow/2)=mbound_value;
            else
                events_avDC(j,nf+DiffmeanWindow/2)=sbound_value;
```

```
140             end
141         end
142
143         %Remove states shorter than a specified number of frames
144         %Unbound events
145         Adiff = diff(events_avDC(j,:)==unbound_value);
146         nnz(Adiff==1)+nnz(Adiff==-1);
147         ind_start = find(Adiff==1);
148         ind_stop = find(Adiff==-1);
149         ubblock_length = ind_stop-ind_start; % list of consecutive section lengths
150         blocks_ind = find(ubblock_length<Min_eventlength);% list of blocks below
                min length in frames
151         for ii = 1:numel(blocks_ind) % loops through each block
152             events_avDC(j,(ind_start(blocks_ind(ii))+1:ind_stop(blocks_ind(ii))))=
                    events_avDC(j,(ind_start(blocks_ind(ii))));
153         end
154         nnz(Adiff==1)+nnz(Adiff==-1);
155
156         %Single bound events
157         Adiff = diff(events_avDC(j,:)==sbound_value);
158         ind_start = find(Adiff==1);
159         ind_stop = find(Adiff==-1);
160         sbblock_length = ind_stop-ind_start; % list of consecutive section lengths
161         blocks_ind = find(sbblock_length<Min_eventlength);% list of blocks below
                min length
162         for ii = 1:numel(blocks_ind) % loops through each block
163             events_avDC(j,(ind_start(blocks_ind(ii))+1:ind_stop(blocks_ind(ii))))=
                    events_avDC(j,(ind_start(blocks_ind(ii))));
164         end
165
166         %Double bound events
167         Adiff = diff(events_avDC(j,:)==mbound_value);
168         ind_start = find(Adiff==1);
169         ind_stop = find(Adiff==-1);
170         mvblock_length = ind_stop-ind_start; % list of consecutive section lengths
171         blocks_ind = find(mvblock_length<Min_eventlength);% list of blocks below
                min length
172         for ii = 1:numel(blocks_ind) % loops through each block
173             events_avDC(j,(ind_start(blocks_ind(ii))+1:ind_stop(blocks_ind(ii))))=
                    events_avDC(j,(ind_start(blocks_ind(ii))));
174         end
175
176         %Find 01 and 12 events
177         Adiff = diff(events_avDC(j,(DiffmeanWindow+1):(nf+DiffmeanWindow/2))==
                unbound_value);
178         zero_one_events1(j)=nnz(Adiff==1)+nnz(Adiff==-1);
179
180         Bdiff = diff(events_avDC(j,(DiffmeanWindow+1):(nf+DiffmeanWindow/2))==
                mbound_value);
181         one_two_events1(j)=nnz(Bdiff==1)+nnz(Bdiff==-1);
182
```

```
183         %%% Extract state lifetimes:
184         events_vector=events_avDC(j,:);
185         events_vector=rmmissing(events_vector);
186 
187         %%% unbound lifetimes:
188         unbound_vector=(events_vector==unbound_value);
189         measurements=regionprops(unbound_vector,'Area');
190         m_area=[measurements.Area];
191         unbound_lifetimes=[unbound_lifetimes, m_area];
192         unbound_lifetimes_corr{j}=m_area;
193 
194         %%% single bound lifetimes == multivalent unbound lifetimes:
195         sbound_vector=(events_vector==sbound_value);
196         measurements=regionprops(sbound_vector,'Area');
197         m_area=[measurements.Area];
198         sbound_lifetimes=[sbound_lifetimes, m_area];
199         sbound_lifetimes_corr{j}=m_area;
200 
201         %%% multivalent bound lifetimes:
202         mvbound_vector=(events_vector==mbound_value);
203         measurements=regionprops(mvbound_vector,'Area');
204         m_area=[measurements.Area];
205         mvbound_lifetimes=[mvbound_lifetimes, m_area];
206         mvbound_lifetimes_corr{j}=m_area;
207 
208     else
209         events_avDC(j,:)=NaN;
210         zero_one_events1(j)=NaN;
211         one_two_events1(j)=NaN;
212         unbound_lifetimes_corr{j}=NaN;
213         sbound_lifetimes_corr{j}=NaN;
214         Dtotal_stuck=Dtotal(j,:);
215     end
216 end
217 %Separate loop 1 results from loop 2 results
218 unbound_lifetimes1=unbound_lifetimes;
219 sbound_lifetimes1=sbound_lifetimes;
220 mvbound_lifetimes1=mvbound_lifetimes;
221 
222 Dtotal_unbound=events_avDC(:,1:size(Dtotal,2))==unbound_value;
223 Dtotal_unbound=Dtotal_unbound.*Dtotal;
224 Dtotal_unbound_vector=Dtotal_unbound(Dtotal_unbound~=0);
225 
226 Dtotal_sbound=events_avDC(:,1:size(Dtotal,2))==sbound_value;
227 Dtotal_sbound=Dtotal_sbound.*Dtotal;
228 Dtotal_sbound_vector=Dtotal_sbound(Dtotal_sbound~=0);
229 
230 Dtotal_mvbound=events_avDC(:,1:size(Dtotal,2))==mbound_value;
231 Dtotal_mvbound=Dtotal_mvbound.*Dtotal;
232 Dtotal_mvbound_vector=Dtotal_mvbound(Dtotal_mvbound~=0);
233 
```

```
%% special loop
%Do the same for the other xy-list, but do this line-by-line,
%since different particles have different number of frames
%To work around NaN inputs translate NaN to -1 and then filter >0
Dtotal2=-1*ones(size(xylistedited,1),numel(x)-MeasurementWindow+1);

for j=1:size(xylistedited,1) %loop over each particle
    xylist2=xylistedited(j,1:1:find(isnan(xylistedited(j,:)),1,'first')-2);
    %shorten xy input to just before input becomes NaN
    x=(xylist2(1,3:2:end));
    y=(xylist2(1,4:2:end));
    n_frames=numel(x);

    %Calculate average DC
       MSD2=zeros(1,maxdt+1);
       MSD2(1)=0;
    for dt=1:maxdt
        SD=zeros(1,n_frames-dt);
        for l=1:n_frames-dt
            SD(l)=(x(l+dt)-x(l)).^2+(y(l+dt)-y(l)).^2;
        end
        MSD2(dt+1)=mean(SD);
    end
    dtlist=0:maxdt;
    try
    myfit=fit(dtlist',MSD2','poly1');
    Dtrajectory2(j)=myfit.p1*FrameRate/4;
    catch
    end

    %Calculate time dependant DC
    maxdt=10;
    DtraceWindow=zeros(maxdt,numel(x)-MeasurementWindow+1);
    weight=zeros(1,maxdt);

    for dt=1:maxdt
        SDtrace=(x(1+dt:end)-x(1:end-dt)).^2+(y(1+dt:end)-y(1:end-dt)).^2; %squared
            displacement

        SDtraceWindow=movmean(SDtrace,MeasurementWindow-dt,'Endpoints','discard');
           %mean squared displacement

        DtraceWindow(dt,:)=SDtraceWindow*FrameRate/(4*dt); %diffusion coeff for
           certain dt

        Vrel=dt*(2*dt^2+1)/(MeasurementWindow-dt+1); %calculate relative variance
        weight(dt)=1/Vrel; %assign weights to data points (rows) DtraceWindow
    end
    sumweight=sum(weight);
    weight=weight/sumweight;
    weight=weight';
```

```
282
283     %calculate diffusion coefficient
284     parameterD=sum(DtraceWindow.*weight);
285     for jj=1:length(sum(DtraceWindow.*weight))
286             Dtotal2(j,jj)=parameterD(jj);
287     end
288
289     %State assignment
290        if mean(sum(DtraceWindow.*weight))>Stuck_threshold    %only detect events
               for non-stuck particles, can change value
291         for nf=1:size(Dtotal2,2)-DiffmeanWindow %number of frames to slide the
               window over
292
293             if mean(Dtotal2(j,nf:(nf+DiffmeanWindow-1)))>Unbound_threshold
294                 events_avDC2(j,nf+DiffmeanWindow/2)=unbound_value; %assign value to
                        position in middle of sliding window to match DC time trace
295             elseif mean(Dtotal2(j,nf:(nf+DiffmeanWindow-1)))<MVbound_threshold &
                    mean(Dtotal2(j,nf:(nf+DiffmeanWindow-1)))>0
296                 events_avDC2(j,nf+DiffmeanWindow/2)=mbound_value;
297             elseif mean(Dtotal2(j,nf:(nf+DiffmeanWindow-1)))>MVbound_threshold &
                    mean(Dtotal2(j,nf:(nf+DiffmeanWindow-1)))<Unbound_threshold
298                 events_avDC2(j,nf+DiffmeanWindow/2)=sbound_value;
299             elseif Dtotal2(j,nf)<0
300                 events_avDC2(j,nf+DiffmeanWindow/2)=NaN;
301             end
302         end
303
304     %%%Remove states shorter than a specified number of frames
305     %Unbound state
306     Adiff = diff(events_avDC2(j,:)==unbound_value);
307     nnz(Adiff==1)+nnz(Adiff==-1);
308     ind_start = find(Adiff==1);
309     ind_stop = find(Adiff==-1);
310     ubblock_length = ind_stop-ind_start; % list of consecutive section lengths
311     blocks_ind = find(ubblock_length<Min_eventlength);% list of blocks below min
            length in frames
312     for ii = 1:numel(blocks_ind) % loops through each block
313         events_avDC2(j,(ind_start(blocks_ind(ii))+1:ind_stop(blocks_ind(ii))))=
                events_avDC2(j,(ind_start(blocks_ind(ii))));
314     end
315
316     %Single bound state
317     Adiff = diff(events_avDC2(j,:)==sbound_value);
318     ind_start = find(Adiff==1);
319     ind_stop = find(Adiff==-1);
320     sbblock_length = ind_stop-ind_start; % list of consecutive section lengths
321     blocks_ind = find(sbblock_length<Min_eventlength);% list of blocks below min
            length
322     for ii = 1:numel(blocks_ind) % loops through each block
323         events_avDC2(j,(ind_start(blocks_ind(ii))+1:ind_stop(blocks_ind(ii))))=
                events_avDC2(j,(ind_start(blocks_ind(ii))));
```

```
324     end
325
326     %Double bound state
327     Adiff = diff(events_avDC2(j,:)==mbound_value);
328     ind_stop = find(Adiff==-1);
329     ind_start = find(Adiff==1);
330     mvblock_length = ind_stop-ind_start; % list of consecutive section lengths
331     blocks_ind = find(mvblock_length<Min_eventlength);% list of blocks below min
            length
332     for ii = 1:numel(blocks_ind) % loops through each block
333         events_avDC2(j,(ind_start(blocks_ind(ii))+1:ind_stop(blocks_ind(ii))))=
                events_avDC2(j,(ind_start(blocks_ind(ii))));
334     end
335
336     % Check 01 and 12 events
337     Adiff = diff(events_avDC2(j,(DiffmeanWindow+1):(nf+DiffmeanWindow/2))==
            unbound_value);
338     zero_one_events2(j)=nnz(Adiff==1)+nnz(Adiff==-1);
339
340     Bdiff = diff(events_avDC2(j,(DiffmeanWindow+1):(nf+DiffmeanWindow/2))==
            mbound_value);
341     one_two_events2(j)=nnz(Bdiff==1)+nnz(Bdiff==-1);
342
343     %%% Extract state lifetimes:
344     events_vector2=events_avDC2(j,:);
345     events_vector2=rmmissing(events_vector2);
346
347     %%% unbound lifetimes:
348     unbound_vector2=(events_vector2==unbound_value);
349     measurements=regionprops(unbound_vector2,'Area');
350     m_area=[measurements.Area];
351     unbound_lifetimes2=[unbound_lifetimes2, m_area];
352     unbound_lifetimes_corr2{j}=m_area;
353
354     sbound_vector2=(events_vector2==sbound_value);
355     measurements=regionprops(sbound_vector2,'Area');
356     m_area=[measurements.Area];
357     sbound_lifetimes2=[sbound_lifetimes2, m_area];
358     sbound_lifetimes_corr2{j}=m_area;
359
360     %%% multivalent bound lifetimes:
361     mvbound_vector2=(events_vector2==mbound_value);
362     measurements=regionprops(mvbound_vector2,'Area');
363     m_area=[measurements.Area];
364     mvbound_lifetimes2=[mvbound_lifetimes2, m_area];
365     mvbound_lifetimes_corr2{j}=m_area;
366
367        else    %if particles have low mobility
368          events_avDC2(j,:)=NaN;
369         zero_one_events2(j)=NaN;
370         one_two_events2(j)=NaN;
```

```
371         unbound_lifetimes_corr2{j}=NaN;
372         sbound_lifetimes_corr2{j}=NaN;
373         Dtotal_stuck2=Dtotal2(j,:);
374     end
375 end
376 %Separate results loop 2 from loop 1
377 Dtotal_unbound2=events_avDC2(:,1:size(Dtotal2,2))==unbound_value;
378 Dtotal_unbound2=Dtotal_unbound2.*Dtotal2;
379 Dtotal_unbound_vector2=Dtotal_unbound2(Dtotal_unbound2~=0);
380
381 Dtotal_sbound2=events_avDC2(:,1:size(Dtotal2,2))==sbound_value;
382 Dtotal_sbound2=Dtotal_sbound2.*Dtotal2;
383 Dtotal_sbound_vector2=Dtotal_sbound2(Dtotal_sbound2~=0);
384
385 Dtotal_mvbound2=events_avDC2(:,1:size(Dtotal2,2))==mbound_value;
386 Dtotal_mvbound2=Dtotal_mvbound2.*Dtotal2;
387 Dtotal_mvbound_vector2=Dtotal_mvbound2(Dtotal_mvbound2~=0);
388
389 %Merge results loop 1 and 2
390 Dtotal_unbound=[Dtotal_unbound; Dtotal_unbound2];
391 Dtotal_sbound=[Dtotal_sbound; Dtotal_sbound2];
392 Dtotal_mvbound=[Dtotal_mvbound; Dtotal_mvbound2];
393 Dtotal_unbound_vector=[Dtotal_unbound_vector; Dtotal_unbound_vector2];
394 Dtotal_sbound_vector=[Dtotal_sbound_vector; Dtotal_sbound_vector2];
395 Dtotal_mvbound_vector=[Dtotal_mvbound_vector; Dtotal_mvbound_vector2];
396
397 try
398 unbound_lifetimes_corr=[unbound_lifetimes_corr unbound_lifetimes_corr2];
399 sbound_lifetimes_corr=[sbound_lifetimes_corr sbound_lifetimes_corr2];
400 mvbound_lifetimes_corr=[mvbound_lifetimes_corr mvbound_lifetimes_corr2];
401 end
402 save([NameExtension,'Dtotal.mat'],'Dtotal')
403
404 %Plot DC distribution
405 figure();
406 histogram(Dtotal_unbound_vector,0:0.001:DC_plotlimit,'FaceColor',[0 0.4470 0.7410],
      'EdgeColor','none'); %blue
407 hold on
408 histogram(Dtotal_sbound_vector,0:0.001:DC_plotlimit,'FaceColor',[0.8500 0.3250
      0.0980],'EdgeColor','none') %orange
409 hold on
410 histogram(Dtotal_mvbound_vector,0:0.001:DC_plotlimit,'FaceColor',[0.9290 0.6940
      0.1250],'EdgeColor','none'); %yellow
411 title('Total DC distribution');
412 xlabel('D (\mum^2/sec)');
413 ylabel('Counts');
414 set(gca,'FontSize',18);
415 ylim([0 3.5*10^5])
416 axis('square')
417 saveas(gcf,[NameExtension,' Total DC',num2str(Unbound_threshold),' Windowsize ',
      num2str(MeasurementWindow),'.tiff'])
```

```
close

%% Bound Fraction
 DCfractions=zeros(5,2);
DCfractions(1,1)=numel(Dtotal(:))+numel(find(Dtotal2(:)>0));
DCfractions(2,1)=numel(find(Dtotal(:)>Unbound_threshold))+numel(find(Dtotal2(:)>
    Unbound_threshold));
DCfractions(3,1)=numel(find(Dtotal(:)<Unbound_threshold&Dtotal(:)>MVbound_threshold
    ))+numel(find(Dtotal2(:)<Unbound_threshold&Dtotal2(:)>MVbound_threshold));
DCfractions(4,1)=numel(find(Dtotal(:)<MVbound_threshold))+numel(find(Dtotal2(:)<
    MVbound_threshold&Dtotal2(:)>0));
DCfractions(5,1)=DCfractions(3,1)+DCfractions(4,1);
for position=1:length(DCfractions)
    DCfractions(position,2)=DCfractions(position,1)/DCfractions(1,1);
end
DCfractions=array2table(DCfractions,'RowNames',{'total counts','free','single-
    molecule','multivalent','Total bound'});
writetable(DCfractions,[NameExtension,'DCfractions','.xlsx'],'WriteRowNames',true);

BoundFraction=(numel(find(Dtotal(:)<Unbound_threshold))+numel(find(Dtotal2(:)<
    Unbound_threshold&Dtotal2(:)>0)))/(numel(Dtotal(:))+numel(find(Dtotal2(:)>0)));

%% Events distributions
zero_one_events=[zero_one_events1 zero_one_events2];
one_two_events=[one_two_events1 one_two_events2];

all_events=rmmissing(zero_one_events)';
all12_events=rmmissing(one_two_events)';
all_events2=[rmmissing(zero_one_events) rmmissing(one_two_events)]';%01 + 12 events

N_particles_mobile=numel(all_events); %if particles are stuck, events will be NaN
    so all_events will skip this particle

unbound_lifetimes=[unbound_lifetimes1 unbound_lifetimes2];
sbound_lifetimes=[sbound_lifetimes1 sbound_lifetimes2];
mvbound_lifetimes=[mvbound_lifetimes1 mvbound_lifetimes2];


%% fit distribution before filtering
pd_all_events=fitdist(all_events,'normal');
pd_12_events=fitdist(all12_events,'normal');


all_events_mean=pd_all_events.mu;
all_events_std=pd_all_events.sigma;
all_events_SE=all_events_std/sqrt(numel(all_events));

all12_events_mean=pd_12_events.mu;
all12_events_std=pd_12_events.sigma;
all12_events_SE=all12_events_std/sqrt(numel(all12_events));

```

```
463 activity_all_events=(all_events_mean/Time_seconds)*1000;
464 std_activity_all_events=(all_events_std/Time_seconds)*1000;
465 SE_activity_all_events=(all_events_SE/Time_seconds)*1000;
466
467
468 activity_all12_events=(all12_events_mean/Time_seconds)*1000;
469 std_activity_all12_events=(all12_events_std/Time_seconds)*1000;
470 SE_activity_all12_events=(all12_events_SE/Time_seconds)*1000;
471
472 activity_01=(fitdist(rmmissing(zero_one_events'),'normal').mu/Time_seconds)*1000
473 activity_12=(fitdist(rmmissing(one_two_events'),'normal').mu/Time_seconds)*1000
474 %% iterative fitting of distribution to all events
475 Sigma = 4;
476 all_events=rmmissing(zero_one_events)';
477 x_scale=max(all_events);
478
479 %%% Fitting events distribution
480 for i = 1:20
481     if i == 1
482         pd_n = fitdist(all_events,Events_fittype);
483         pd_n12 = fitdist(all12_events,Events_fittype);
484         all_events_means(i)=pd_n.mu;
485         all_events_stds(i)=pd_n.sigma;
486         all_events_SEs(i)=pd_n.sigma/sqrt(numel(all_events));
487
488         all12_events_means(i)=pd_n12.mu;
489         all12_events_stds(i)=pd_n12.sigma;
490         all12_events_SEs(i)=pd_n12.sigma/sqrt(numel(all12_events));
491
492         all_events = all_events( all_events < (pd_n.mu+ Sigma*pd_n.sigma ));
493         pd_n = fitdist(all_events,Events_fittype);
494
495         if pd_n.mu+ Sigma*pd_n.sigma > 0
496         all12_events = all12_events( all12_events < (pd_n12.mu+ Sigma*pd_n12.sigma
                ));
497         pd_n12 = fitdist(all12_events,Events_fittype);
498         end
499         all_events_means(i+1)=pd_n.mu;
500         all_events_stds(i+1)=pd_n.sigma;
501         all_events_SEs(i+1)=pd_n.sigma/sqrt(numel(all_events));
502
503         all12_events_means(i+1)=pd_n12.mu;
504         all12_events_stds(i+1)=pd_n12.sigma;
505         all12_events_SEs(i+1)=pd_n12.sigma/sqrt(numel(all12_events));
506     end
507     pd_n = fitdist(all_events,Events_fittype);
508     all_events = all_events( all_events < (pd_n.mu+ Sigma*pd_n.sigma ));
509     pd_n = fitdist(all_events,Events_fittype);
510     all_events_means(i+1)=pd_n.mu;
511     all_events_stds(i+1)=pd_n.sigma;
512     all_events_SEs(i+1)=pd_n.sigma/sqrt(numel(all_events));
```

```
513
514     pd_n12 = fitdist(all12_events,Events_fittype);
515    if pd_n.mu+ Sigma*pd_n.sigma > min(all12_events(all12_events>0))
516     all12_events = all12_events( all12_events < (pd_n12.mu+ Sigma*pd_n12.sigma ));
517
518     pd_n12 = fitdist(all12_events,Events_fittype);
519    else
520        disp(i)
521    break
522     end
523     all12_events_means(i+1)=pd_n12.mu;
524     all12_events_stds(i+1)=pd_n12.sigma;
525     all12_events_SEs(i+1)=pd_n12.sigma/sqrt(numel(one_two_events));
526 end
527     remove_particles=find(zero_one_events>pd_n.mu+ Sigma*pd_n.sigma);
528     remove_particles2=find(one_two_events>pd_n12.mu+ Sigma*pd_n12.sigma);
529
530 all_events_filtered_n=all_events;
531 all_events=rmmissing(zero_one_events)';
532
533 all12_events_filtered_n=all12_events;
534 all12_events=rmmissing(one_two_events);
535
536
537 all_events_mean_n=all_events_means(end);
538 all_events_std_n=all_events_stds(end);
539 all_events_SE_n=all_events_SEs(end);
540
541 all12_events_mean_n=all12_events_means(end);
542 all12_events_std_n=all12_events_stds(end);
543 all12_events_SE_n=all12_events_SEs(end);
544
545 activity_all_events_n=(all_events_means(end)/Time_seconds)*1000; % (mHz)
546 std_activity_all_events_n=(all_events_stds(end)/Time_seconds)*1000; % (mHz)
547 SE_activity_all_events_n=(all_events_SEs(end)/Time_seconds)*1000; % (mHz)
548
549 activity_all12_events_n=(all12_events_means(end)/Time_seconds)*1000; % (mHz)
550 std_activity_all12_events_n=(all12_events_stds(end)/Time_seconds)*1000; % (mHz)
551 SE_activity_all12_events_n=(all12_events_SEs(end)/Time_seconds)*1000; % (mHz)
552
553
554
555 figure()
556 histfit(all_events_filtered_n,ceil(max(all_events_filtered_n)/2)+1,Events_fittype)
557 title('All 01 events 20it fitting')
558 xlabel('Events')
559 ylabel('Particles')
560 xlim([0 x_scale])
561 set(gca,'FontSize',18)
562 saveas(gcf,[NameExtension,' all 01 events 20it fitting ',Events_fittype,' ',num2str
    (Unbound_threshold),'.tiff'])
```

```
563 N_particles_eventsfiltered=numel(all_events_filtered_n);
564
565 figure()
566 histfit(all12_events_filtered_n,ceil(max(all12_events_filtered_n)/2)+1,
        Events_fittype)
567 title('All 12 events 20it fitting')
568 xlabel('Events')
569 ylabel('Particles')
570 xlim([0 x_scale])
571 set(gca,'FontSize',18)
572 saveas(gcf,[NameExtension,' all 12 events 20it fitting ',Events_fittype,' ',num2str
        (MVbound_threshold),'.tiff'])
573 N_particles_12eventsfiltered=numel(all12_events_filtered_n);
574
575 %% plot histogram of all diffusion coefficients after filtering by events:
576
577 Dtotal_unbound_filtered=Dtotal_unbound;
578 Dtotal_unbound_filtered(remove_particles,:)=[];
579 Dtotal_unbound_filtered_vector=Dtotal_unbound_filtered(Dtotal_unbound_filtered~=0);
580
581 Dtotal_sbound_filtered=Dtotal_sbound;
582 Dtotal_sbound_filtered(remove_particles,:)=[];
583 Dtotal_sbound_filtered_vector=Dtotal_sbound_filtered(Dtotal_sbound_filtered~=0);
584
585 Dtotal_mvbound_filtered=Dtotal_mvbound;
586 Dtotal_mvbound_filtered(remove_particles2,:)=[];
587 Dtotal_mvbound_filtered_vector=Dtotal_mvbound_filtered(Dtotal_mvbound_filtered~=0);
588
589 figure();
590 histogram(Dtotal_unbound_filtered_vector,0:0.001:DC_plotlimit,'FaceColor',[0 0.4470
        0.7410],'EdgeColor','none'); %blue
591 hold on
592 histogram(Dtotal_sbound_filtered_vector,0:0.001:DC_plotlimit,'FaceColor',[0.8500
        0.3250 0.0980],'EdgeColor','none') %orange
593 histogram(Dtotal_mvbound_filtered_vector,0:0.001:DC_plotlimit,'FaceColor',[0.9290
        0.6940 0.1250],'EdgeColor','none') %yellow
594
595 % histogram(Dtotal(:),0:0.001:DC_plotlimit,'EdgeColor','none');
596 title('DC total filtered');
597 xlabel('D (\mum^2/sec)');
598 ylabel('Counts');
599 set(gca,'FontSize',22);
600 ylim([0 3.5*10^5])
601 axis('square')
602 saveas(gcf,[NameExtension,'_Total DC_D',num2str(Unbound_threshold),' filtered ',
        num2str(Events_fittype),'.tiff'])
603 close
604
605 %% state lifetimes analysis
606 %%% ecdf of state lifetimes, fit log double exp
607
```

```
608 %%% determine which particles are filtered out before by normal fitting:
609 ub_lifetimes_filtered_events=unbound_lifetimes_corr;
610 ub_lifetimes_filtered_states=unbound_lifetimes_corr;
611 b_lifetimes_filtered_events=sbound_lifetimes_corr;
612 b_lifetimes_filtered_states=sbound_lifetimes_corr;
613 mv_lifetimes_filtered_states=mvbound_lifetimes_corr;
614
615 %%% for normal fitting of all events:
616 ActiveParticles_all_events=[zero_one_events<max(all_events_filtered_n) (
        one_two_events<max(all12_events_filtered_n))];
617 for ap=1:length(ActiveParticles_all_events)
618     if ActiveParticles_all_events(ap)==0
619         ub_lifetimes_filtered_events{ap}=[];
620         b_lifetimes_filtered_events{ap}=[];
621         mv_lifetimes_filtered_states{ap}=[];
622     end
623 end
624 ub_lifetimes_filtered_events=cell2mat(ub_lifetimes_filtered_events);
625 b_lifetimes_filtered_events=cell2mat(b_lifetimes_filtered_events);
626 mv_lifetimes_filtered_states=cell2mat(mv_lifetimes_filtered_states);
627
628
629 %% Unbound state lifetimes ecdf fitting - unfiltered
630 all_unbound_lifetimes=cell2mat(unbound_lifetimes_corr);
631 figure()
632 [f,x] = ecdf(all_unbound_lifetimes);
633 condition = f<1.0;
634 f = f(condition);
635 x = x(condition);
636 ft=fittype('a*exp(-b*x)+(1-a)*exp(-d*x)');
637
638 coeffnames(ft);
639 options=fitoptions(ft);
640 options.StartPoint = [0.1 0.1 0.001];
641 options.Lower      = [0 0 0];
642 options.Upper      = [1 1 1];
643
644
645 fit_lifetime = fit (x/FrameRate, (1-f), ft, options);
646 fraction_1 = fit_lifetime.a;
647 tau_1      = 1/fit_lifetime.b;
648 fraction_2 = 1-fit_lifetime.a;
649 tau_2      = 1/fit_lifetime.d;
650
651
652 plot(fit_lifetime, x/FrameRate, (1-f));
653 xlim([0 Time_seconds]);
654 ylim([0.01 1]);
655 set(gca,'YScale','log');
656 hold on
657 ci_bounds=predint(fit_lifetime,x/FrameRate);
```

```
plot(x/FrameRate,ci_bounds,'r--')
set(gca,'FontSize',14)
get_gca=gca;
legend(get_gca,'off');
title({
    ['Unbound state lifetimes  -  >0.01 \mum^2/s']
    ['\tau_{1}= ',num2str(tau_1,'%.1f'),'s (',num2str(fraction_1,'%.2f'),')','; \
        tau_{2}= ', num2str(tau_2,'%.1f'), 's',' (',num2str(fraction_2,'%.2f'),')']
    });xlabel('Lifetime (s)');
ylabel('Surviving fraction');
grid on;
saveas(gcf,[NameExtension,' ALL_unbound_ecdf_logfit','.tiff'])


y_value_fitting = fit_lifetime(x/FrameRate);
data_save_matrix = zeros(length(x) ,3);
data_save_matrix(:, 1) = x/FrameRate;
data_save_matrix(:, 2) = y_value_fitting;
data_save_matrix(:, 3) = (1-f);
data_save_matrix=array2table(data_save_matrix,'VariableNames',{'Time_s','
    y_value_fitting','data_1minf'});
writetable(data_save_matrix,([FileName,'ALL_unbound_lifetimes_plotdata','.xlsx']));


get_gca=gca;
legend(get_gca,'off');
title(['Unbound state lifetime  -  \tau_{1}= ',num2str(tau_1,'%.1f'), 's;  \tau_
    {2}= ', num2str(tau_2,'%.0f'), 's']);
xlabel('Lifetime (s)');
ylabel('log(1-cdf)');
grid on;
saveas(gcf,[NameExtension,' unbound_ecdf_logfit','.tiff'])
close

colNames={'tau_1','fraction_1','tau_2','fraction_2'};
UnboundStateResults(1,1)=tau_1;
UnboundStateResults(1,2)=fraction_1;
UnboundStateResults(1,3)=tau_2;
UnboundStateResults(1,4)=fraction_2;
UBStateLifetimes=array2table(UnboundStateResults,'VariableNames',colNames);
writetable(UBStateLifetimes,[NameExtension,' ALL_unbound_lifetimes_fitdata','.xlsx'
    ]);

CI_95pct=confint(fit_lifetime,0.95);
CI_68pct_SD=confint(fit_lifetime,0.68);
CI_95pct_table=array2table(CI_95pct,'VariableNames',{'fraction','tau_1','tau_2'});
writetable(CI_95pct_table,[NameExtension,' 95pct CI of ecdf fit unbound ALL','.xlsx
    ']);
CI_68pct_SD_table=array2table(CI_68pct_SD,'VariableNames',{'fraction','tau_1','
    tau_2'});
```

```
writetable(CI_68pct_SD_table,[NameExtension,' 68pct CI is SD of ecdf fit unbound ALL','.xlsx']);


%% Unbound state lifetimes ecdf fitting - events filtered

figure()
[f,x] = ecdf(ub_lifetimes_filtered_events);
condition = f<1.0;
f = f(condition);
x = x(condition);

ft=fittype('a*exp(-b*x)+(1-a)*exp(-d*x)');

coeffnames(ft);
options=fitoptions(ft);
options.StartPoint = [0.1 0.1 0.001];
options.Lower      = [0 0 0];
options.Upper      = [1 1 1];


fit_lifetime = fit (x/FrameRate, (1-f), ft, options);
fraction_1_ub_f = fit_lifetime.a;
tau_1_ub_f      = 1/fit_lifetime.b;
fraction_2_ub_f = 1-fit_lifetime.a;
tau_2_ub_f      = 1/fit_lifetime.d;
plot(fit_lifetime, x/FrameRate, (1-f));
xlim([0 Time_seconds]);
ylim([0.01 1])
set(gca,'YScale','log');
hold on
ci_bounds=predint(fit_lifetime,x/FrameRate);
plot(x/FrameRate,ci_bounds,'r--')
set(gca,'FontSize',14)
get_gca=gca;
legend(get_gca,'off');
title({
    ['Unbound state lifetimes  -  events filtered']
    ['\tau_{1}= ',num2str(tau_1_ub_f,'%.1f'),'s (',num2str(fraction_1_ub_f,'%.2f'),')','; \tau_{2}= ', num2str(tau_2_ub_f,'%.1f'), 's',' (',num2str(fraction_2_ub_f,'%.2f'),')']
    });xlabel('Lifetime (s)');
ylabel('Surviving fraction');
grid on;
saveas(gcf,[NameExtension,' events_filtered_unbound_ecdf_logfit','.tiff'])


y_value_fitting = fit_lifetime(x/FrameRate);
data_save_matrix = zeros(length(x) ,3);
data_save_matrix(:, 1) = x/FrameRate;
data_save_matrix(:, 2) = y_value_fitting;
```

```
750 data_save_matrix(:, 3) = (1-f);
751 data_save_matrix=array2table(data_save_matrix,'VariableNames',{'Time_s','
        y_value_fitting','data_1minf'});
752 writetable(data_save_matrix,([FileName,'events_filtered_ub_lifetimes_plotdata','.
        xlsx']));
753
754 colNames={'tau_1','fraction_1','tau_2','fraction_2'};
755 UnboundStateResults(1,1)=tau_1_ub_f;
756 UnboundStateResults(1,2)=fraction_1_ub_f;
757 UnboundStateResults(1,3)=tau_2_ub_f;
758 UnboundStateResults(1,4)=fraction_2_ub_f;
759 UBStateLifetimes=array2table(UnboundStateResults,'VariableNames',colNames);
760 writetable(UBStateLifetimes,[NameExtension,'
        events_filtered_unbound_lifetimes_fitdata','.xlsx']);
761
762 CI_95pct_ub_f=confint(fit_lifetime,0.95);
763 CI_68pct_SD_ub_f=confint(fit_lifetime,0.68);
764 CI_95pct_table=array2table(CI_95pct,'VariableNames',{'fraction','tau_1','tau_2'});
765 writetable(CI_95pct_table,[NameExtension,' 95pct CI of ecdf fit unbound events
        filterd','.xlsx']);
766 CI_68pct_SD_table=array2table(CI_68pct_SD,'VariableNames',{'fraction','tau_1','
        tau_2'});
767 writetable(CI_68pct_SD_table,[NameExtension,' 68pct CI is SD of ecdf fit unbound
        events filtered','.xlsx']);
768
769
770 %% Single Bound state lifetimes ecdf fitting - unfiltered
771 all_bound_lifetimes=cell2mat(sbound_lifetimes_corr);
772 figure(7)
773 [f,x] = ecdf(all_bound_lifetimes);
774 condition = f<1.0;
775 f = f(condition);
776 x = x(condition);
777
778 ft=fittype('a*exp(-b*x)+(1-a)*exp(-d*x)');
779
780 coeffnames(ft);
781 options=fitoptions(ft);
782 options.StartPoint = [0.1 0.1 0.001 ];
783 options.Lower      = [0 0 0];
784 options.Upper      = [1 1 1];
785
786
787 fit_lifetime = fit (x/FrameRate, (1-f), ft, options);
788 fraction_1 = fit_lifetime.a;
789 tau_1      = 1/fit_lifetime.b;
790 fraction_2 = 1-fit_lifetime.a;
791 tau_2      = 1/fit_lifetime.d;
792
793 plot(fit_lifetime, x/FrameRate, (1-f));
794 xlim([0 Time_seconds]);
```

```
795 ylim([0.01 1]);
796 set(gca,'YScale','log');
797 hold on
798 ci_bounds=predint(fit_lifetime,x/FrameRate);
799 plot(x/FrameRate,ci_bounds,'r--')
800 set(gca,'FontSize',14)
801 get_gca=gca;
802 legend(get_gca,'off');
803 title({
804     ['Bound state lifetimes  -  >0.01 \mum^2/s']
805     ['\tau_{1}= ',num2str(tau_1,'%.1f'),'s (',num2str(fraction_1,'%.2f'),')','; \
        tau_{2}= ', num2str(tau_2,'%.1f'), 's',' (',num2str(fraction_2,'%.2f'),')']
806     });xlabel('Lifetime (s)');
807 ylabel('Surviving fraction');
808 grid on;
809 saveas(gcf,[NameExtension,' ALL_bound_ecdf_logfit','.tiff'])
810
811
812 y_value_fitting = fit_lifetime(x/FrameRate);
813 data_save_matrix = zeros(length(x) ,3);
814 data_save_matrix(:, 1) = x/FrameRate;
815 data_save_matrix(:, 2) = y_value_fitting;
816 data_save_matrix(:, 3) = (1-f);
817 data_save_matrix=array2table(data_save_matrix,'VariableNames',{'Time_s','
    y_value_fitting','data_1minf'});
818 writetable(data_save_matrix,([FileName,'ALL_bound_lifetimes_plotdata','.xlsx']));
819
820
821 colNames={'tau_1','fraction_1','tau_2','fraction_2'};
822 UnboundStateResults(1,1)=tau_1;
823 UnboundStateResults(1,2)=fraction_1;
824 UnboundStateResults(1,3)=tau_2;
825 UnboundStateResults(1,4)=fraction_2;
826 UBStateLifetimes=array2table(UnboundStateResults,'VariableNames',colNames);
827 writetable(UBStateLifetimes,[NameExtension,' ALL_bound_lifetimes_fitdata','.xlsx'])
    ;
828
829 CI_95pct=confint(fit_lifetime,0.95);
830 CI_68pct_SD=confint(fit_lifetime,0.68);
831 CI_95pct_table=array2table(CI_95pct,'VariableNames',{'fraction','tau_1','tau_2'});
832 writetable(CI_95pct_table,[NameExtension,' 95pct CI of ecdf fit bound ALL','.xlsx'
    ]);
833 CI_68pct_SD_table=array2table(CI_68pct_SD,'VariableNames',{'fraction','tau_1','
    tau_2'});
834 writetable(CI_68pct_SD_table,[NameExtension,' 68pct CI is SD of ecdf fit bound ALL'
    ,'.xlsx']);
835
836
837 %% Single Bound state lifetimes ecdf fitting - events filtered
838 figure()
839 [f,x] = ecdf(b_lifetimes_filtered_events);
```

```
840 condition = f<1.0;
841 f = f(condition);
842 x = x(condition);
843
844
845 ft=fittype('a*exp(-b*x)+(1-a)*exp(-d*x)');
846
847 coeffnames(ft);
848 options=fitoptions(ft);
849 options.StartPoint = [0.1 0.1 0.001];
850 options.Lower      = [0 0 0];
851 options.Upper      = [1 1 1];
852
853
854 fit_lifetime = fit (x/FrameRate, (1-f), ft, options);
855 fraction_1_b_f = fit_lifetime.a;
856 tau_1_b_f      = 1/fit_lifetime.b;
857 fraction_2_b_f = 1-fit_lifetime.a;
858 tau_2_b_f      = 1/fit_lifetime.d;
859 plot(fit_lifetime, x/FrameRate, (1-f));
860 xlim([0 Time_seconds]);
861 ylim([0.01 1]);
862 set(gca,'YScale','log');
863 hold on
864 ci_bounds=predint(fit_lifetime,x/FrameRate);
865 plot(x/FrameRate,ci_bounds,'r--')
866 set(gca,'FontSize',14)
867 get_gca=gca;
868 legend(get_gca,'off');
869 title({
870     ['Bound state lifetimes  -  events filtered']
871     ['\tau_{1}= ',num2str(tau_1,'%.1f'),'s (',num2str(fraction_1,'%.2f'),')','; \
        tau_{2}= ', num2str(tau_2,'%.1f'), 's',' (',num2str(fraction_2,'%.2f'),')']
872     });xlabel('Lifetime (s)');
873 ylabel('Surviving fraction');
874 grid on;
875 saveas(gcf,[NameExtension,' events_filtered_bound_ecdf_logfit','.tiff'])
876
877
878 y_value_fitting = fit_lifetime(x/FrameRate);
879 data_save_matrix = zeros(length(x) ,3);
880 data_save_matrix(:, 1) = x/FrameRate;
881 data_save_matrix(:, 2) = y_value_fitting;
882 data_save_matrix(:, 3) = (1-f);
883 data_save_matrix=array2table(data_save_matrix,'VariableNames',{'Time_s','
    y_value_fitting','data_1minf'});
884 writetable(data_save_matrix,([FileName,' events_filtered_bound_lifetimes_plotdata',
    '.xlsx']));
885
886
887 colNames={'tau_1','fraction_1','tau_2','fraction_2'};
```

```
888 UnboundStateResults(1,1)=tau_1;
889 UnboundStateResults(1,2)=fraction_1;
890 UnboundStateResults(1,3)=tau_2;
891 UnboundStateResults(1,4)=fraction_2;
892 UBStateLifetimes=array2table(UnboundStateResults,'VariableNames',colNames);
893 writetable(UBStateLifetimes,[NameExtension,'
        events_filtered_bound_lifetimes_fitdata','.xlsx']);
894
895 CI_95pct=confint(fit_lifetime,0.95);
896 CI_68pct_SD_b_f=confint(fit_lifetime,0.68);
897 CI_95pct_table=array2table(CI_95pct,'VariableNames',{'fraction','tau_1','tau_2'});
898 writetable(CI_95pct_table,[NameExtension,' 95pct CI of ecdf fit bound events
        filtered','.xlsx']);
899 CI_68pct_SD_table=array2table(CI_68pct_SD_b_f,'VariableNames',{'fraction','tau_1','
        tau_2'});
900 writetable(CI_68pct_SD_table,[NameExtension,' 68pct CI is SD of ecdf fit bound
        events filtered','.xlsx']);
901
902 %% Double Bound unfiltered
903 all_bound_lifetimes=cell2mat(mvbound_lifetimes_corr);
904
905 figure()
906 [f,x] = ecdf(all_bound_lifetimes);
907 condition = f<1.0;
908 f = f(condition);
909 x = x(condition);
910
911 ft=fittype('a*exp(-b*x)+(1-a)*exp(-d*x)');
912
913 coeffnames(ft);
914 options=fitoptions(ft);
915 options.StartPoint = [0.1 0.1 0.001];
916 options.Lower      = [0 0 0];
917 options.Upper      = [1 1 1];
918
919
920 fit_lifetime = fit (x/FrameRate, (1-f), ft, options);
921 fraction_1 = fit_lifetime.a;
922 tau_1      = 1/fit_lifetime.b;
923 fraction_2 = 1-fit_lifetime.a;
924 tau_2      = 1/fit_lifetime.d;
925 plot(fit_lifetime, x/FrameRate, (1-f));
926 xlim([0 Time_seconds]);
927 ylim([0.01 1]);
928 set(gca,'YScale','log');
929 hold on
930 ci_bounds=predint(fit_lifetime,x/FrameRate);
931 plot(x/FrameRate,ci_bounds,'r--')
932 set(gca,'FontSize',14)
933 get_gca=gca;
934 legend(get_gca,'off');
```

```
935 title({
936     ['MVBound state lifetimes  -  >0.01 \mum^2/s']
937     ['\tau_{1}= ',num2str(tau_1,'%.1f'),'s (',num2str(fraction_1,'%.2f'),')','; \
            tau_{2}= ', num2str(tau_2,'%.1f'), 's',' (',num2str(fraction_2,'%.2f'),')']
938     });xlabel('Lifetime (s)');
939 ylabel('Surviving fraction');
940 grid on;
941 saveas(gcf,[NameExtension,' ALL_mvbound_ecdf_logfit','.tiff'])
942
943
944 y_value_fitting = fit_lifetime(x/FrameRate);
945 data_save_matrix = zeros(length(x) ,3);
946 data_save_matrix(:, 1) = x/FrameRate;
947 data_save_matrix(:, 2) = y_value_fitting;
948 data_save_matrix(:, 3) = (1-f);
949 data_save_matrix=array2table(data_save_matrix,'VariableNames',{'Time_s','
        y_value_fitting','data_1minf'});
950 writetable(data_save_matrix,([FileName,'ALL_mvbound_lifetimes_plotdata','.xlsx']));
951
952 colNames={'tau_1','fraction_1','tau_2','fraction_2'};
953 UnboundStateResults(1,1)=tau_1;
954 UnboundStateResults(1,2)=fraction_1;
955 UnboundStateResults(1,3)=tau_2;
956 UnboundStateResults(1,4)=fraction_2;
957 UBStateLifetimes=array2table(UnboundStateResults,'VariableNames',colNames);
958 writetable(UBStateLifetimes,[NameExtension,' ALL_mvbound_lifetimes_fitdata','.xlsx'
        ]);
959
960 CI_95pct=confint(fit_lifetime,0.95);
961 CI_68pct_SD=confint(fit_lifetime,0.68);
962 CI_95pct_table=array2table(CI_95pct,'VariableNames',{'fraction','tau_1','tau_2'});
963 writetable(CI_95pct_table,[NameExtension,' 95pct CI of ecdf fit mvbound ALL','.xlsx
        ']);
964 CI_68pct_SD_table=array2table(CI_68pct_SD,'VariableNames',{'fraction','tau_1','
        tau_2'});
965 writetable(CI_68pct_SD_table,[NameExtension,' 68pct CI is SD of ecdf fit mvbound
        ALL','.xlsx']);
966
967
968 %% Double bound state lifetimes ecdf fitting - events filtered
969
970 figure()
971 [f,x] = ecdf(mv_lifetimes_filtered_states);
972 condition = f<1.0;
973 f = f(condition);
974 x = x(condition);
975
976 ft=fittype('a*exp(-b*x)+(1-a)*exp(-d*x)');
977
978 coeffnames(ft);
979 options=fitoptions(ft);
```

```
options.StartPoint = [0.1 0.1 0.001];
options.Lower      = [0 0 0];
options.Upper      = [1 1 1];

fit_lifetime = fit (x/FrameRate, (1-f), ft, options);
fraction_1_b_f = fit_lifetime.a;
tau_1_b_f      = 1/fit_lifetime.b;
fraction_2_b_f = 1-fit_lifetime.a;
tau_2_b_f      = 1/fit_lifetime.d;
plot(fit_lifetime, x/FrameRate, (1-f));
xlim([0 Time_seconds]);
ylim([0.01 1]);
set(gca,'YScale','log');
hold on
ci_bounds=predint(fit_lifetime,x/FrameRate);
plot(x/FrameRate,ci_bounds,'r--')
set(gca,'FontSize',14)
get_gca=gca;
legend(get_gca,'off');
title({
    ['Mv Bound state lifetimes  -  events filtered']
    ['\tau_{1}= ',num2str(tau_1,'%.1f'),'s (',num2str(fraction_1,'%.2f'),')','; \
        tau_{2}= ', num2str(tau_2,'%.1f'), 's',' (',num2str(fraction_2,'%.2f'),')']
    });xlabel('Lifetime (s)');
ylabel('Surviving fraction');
grid on;
saveas(gcf,[NameExtension,' events_filtered_mvbound_ecdf_logfit','.tiff'])


y_value_fitting = fit_lifetime(x/FrameRate);
data_save_matrix = zeros(length(x) ,3);
data_save_matrix(:, 1) = x/FrameRate;
data_save_matrix(:, 2) = y_value_fitting;
data_save_matrix(:, 3) = (1-f);
data_save_matrix=array2table(data_save_matrix,'VariableNames',{'Time_s','
    y_value_fitting','data_1minf'});
writetable(data_save_matrix,([FileName,' events_filtered_mvbound_lifetimes_plotdata
    ','.xlsx']));


colNames={'tau_1','fraction_1','tau_2','fraction_2'};
UnboundStateResults(1,1)=tau_1;
UnboundStateResults(1,2)=fraction_1;
UnboundStateResults(1,3)=tau_2;
UnboundStateResults(1,4)=fraction_2;
UBStateLifetimes=array2table(UnboundStateResults,'VariableNames',colNames);
writetable(UBStateLifetimes,[NameExtension,'
    events_filtered_mvbound_lifetimes_fitdata','.xlsx']);

CI_95pct=confint(fit_lifetime,0.95);
CI_68pct_SD_mv_f=confint(fit_lifetime,0.68);
```

```
1027 CI_95pct_table=array2table(CI_95pct,'VariableNames',{'fraction','tau_1','tau_2'});
1028 writetable(CI_95pct_table,[NameExtension,' 95pct CI of ecdf fit mvbound events
         filtered','.xlsx']);
1029 CI_68pct_SD_table=array2table(CI_68pct_SD_mv_f,'VariableNames',{'fraction','tau_1',
         'tau_2'});
1030 writetable(CI_68pct_SD_table,[NameExtension,' 68pct CI is SD of ecdf fit mvbound
         events filtered','.xlsx']);
1031
1032 %% Make a results document
1033
1034 Results={N_particles,N_particles_mobile,N_particles_eventsfiltered,meanD,...
1035     BoundFraction,BoundFraction_filtered,activity_01,activity_12,
             std_activity_all_events,...
1036     activity_all_events_n,activity_all12_events_n,std_activity_all_events_n,...
1037     tau_1_ub_f, 1/CI_68pct_SD_ub_f(1,2), 1/CI_68pct_SD_ub_f(2,2),...
1038     fraction_1_ub_f, CI_68pct_SD_ub_f(1,1), CI_68pct_SD_ub_f(2,1),...
1039     tau_2_ub_f, 1/CI_68pct_SD_ub_f(1,3), 1/CI_68pct_SD_ub_f(2,3),...
1040     fraction_2_ub_f,1-CI_68pct_SD_ub_f(1,1),1-CI_68pct_SD_ub_f(2,1),...
1041     tau_1_b_f, 1/CI_68pct_SD_b_f(1,2), 1/CI_68pct_SD_b_f(2,2),...
1042     fraction_1_b_f, CI_68pct_SD_b_f(1,1), CI_68pct_SD_b_f(2,1),...
1043     tau_2_b_f, 1/CI_68pct_SD_b_f(1,3), 1/CI_68pct_SD_b_f(2,3),...
1044     fraction_2_b_f,1-CI_68pct_SD_b_f(1,1),1-CI_68pct_SD_b_f(2,1)};
1045
1046 Results_names={'N_particles','N_particles_mobile','N_particles_eventsfiltered','
         meanD',...
1047     'BoundFraction','BoundFraction_filtered','01_activity','12_activity','
             std_activity_01_events','activity_01_events_n','activity_12_events_n','
             std_activity_01_events_n',...
1048     'tau_1_ub_f', 'SD_t1_ub_lower', 'SD_t1__ub_upper',...
1049     'fraction_1_ub_f', 'SD_f1_ub_lower', 'SD_f1_ub_upper',...
1050     'tau_2_ub_f', 'SD_t2_ub_lower', 'SD_t2_ub_upper',...
1051     'fraction_2_ub_f','SD_f2_ub_lower','SD_f2_ub_upper',...
1052     'tau_1_bound_f', 'SD_t1_b_lower', 'SD_t1_b_upper',...
1053     'fraction_1_bound_f', 'SD_f1_b_lower', 'SD_f1_b_upper',...
1054     'tau_2_bound_f', 'SD_t2_b_lower', 'SD_t2_b_upper',...
1055     'fraction_2_bound_f','SD_f2_b_lower','SD_f2_b_upper'};
1056
1057 Results_table=cell2table(Results,'VariableNames',Results_names,'RowNames',{
         NameExtension});
1058 writetable(Results_table,[NameExtension,'_Results.xlsx'],'sheet',1,'WriteRowNames'
         ,1);
1059
1060 Settings={PixelSize,FrameRate,TotalFrames,Time_seconds,Unbound_threshold,
         Min_eventlength,DiffmeanWindow,maxdt,Events_fittype,Sigma,Lifetimes_fittype};
1061 Settings_names={'PixelSize','FrameRate','TotalFrames','TimeSeconds','D_threshold','
         Minimum_eventlength','WindowSize','maxdt','Events_fittype','sigma_cutoff','
         Lifetimes_fittype'};
1062 Settings_table=cell2table(Settings','RowNames',Settings_names);
1063 writetable(Settings_table,[NameExtension,'_Results.xlsx'],'sheet',2,'WriteRowNames'
         ,1);
1064
```

1065 toc