university of groningen

faculty of mathematics and natural sciences

Developing the FLIM algorithm for the LIFA software platform

Internship report of Lambert Instruments BV

May 2017

Student: R. B. Christoffers

Internal supervisor: J. Wehmeijer & W. Hellinga

External RUG supervisor: Prof. R. W. C. P. Verstappen

Contents

1 Introduction

1.1 The internship

For the past four months have I done an internship at Lambert Instruments in Groningen. I found their company through the Beta Business Days, where they would have a stand to recruit students for internships and possibilities for a career. I read into their work and products and decides to approach them directly before the Beta Business Days. After a quick response I was invited for an interview and the week after that I started my internship. My internship was divided into three main components, which were:

- Reading about the theory of Fluorescence Lifetime Imaging Microscopy, (FLIM).
- Working on a noise filtering method which would improve the visualization of the results of an experiment.
- Work on a software module for the new FLIM camera which is being developed.

The second point did not have overlap with the other two points, that is why it is explained separately in the appendix, while the first and third point will be explained thoroughly in this report.

1.2 Lambert Instruments

Lambert Instruments is a small company in the sense of number of employees, but has customers all around the world and has close relations to major research institutes. As is stated on their website [1], it is part of their mission to develop and produce high-end products in the field of scientific imaging on a worldwide scale. For this, they listen closely to the feedback of their customers to make sure that the product fits the customers needs. Besides that, they follow the latest

developments to improve their existing products and to apply new techniques in their devices. One of these products which is being updated with new techniques is the FLIM camera.

1.3 About FLIM

Fluorescence.
FLIM is a method which studies properties of molecules by observing how they react when they are exposed to light. Different molecules and different environmental factors influence the reaction of a FLIM sample. When photons from a light source reach a single molecule, it can react in one of the following ways:

- The molecule does not react to the photons.
- The photons bring the molecule to an excited state, after which it goes almost immediately back to the ground state by re-emitting photons. This process is called fluorescence and is a process which occurs in a few nanoseconds. A compound which shows this behavior is called a fluorophore.
- The photons bring the molecule to an excited state, in which it stays longer before re-emitting photons. This process is called phosphorescence and the duration of ranges from a few milliseconds up to a couple of hours.
- The photons damage or alter the molecule after which it can no longer directly go back to its normal ground state. An example of such a process is photo bleaching, which can influence the quality of the FLIM measurement.

All of these situations can happen with a single molecule, but an overall behavior can be found by looking at a set of molecules. In FLIM multiple photons are emitted by a light source and a camera measures the re-emitted light from the sample. One of the variants of FLIM is Time-Domain FLIM (TD-FLIM), in which the emitted light is a short pulse. This pulse increases the emitted intensity of the compound to the level I_0, as can be seen in Figure 1. During and after the pulse each excited molecule in the compound has a chance to re-emit a photon. The overall time for a compound to reduce its emitted intensity by $\frac{1}{e}$ is called the lifetime τ. This lifetime is characteristic for molecule and its environment.

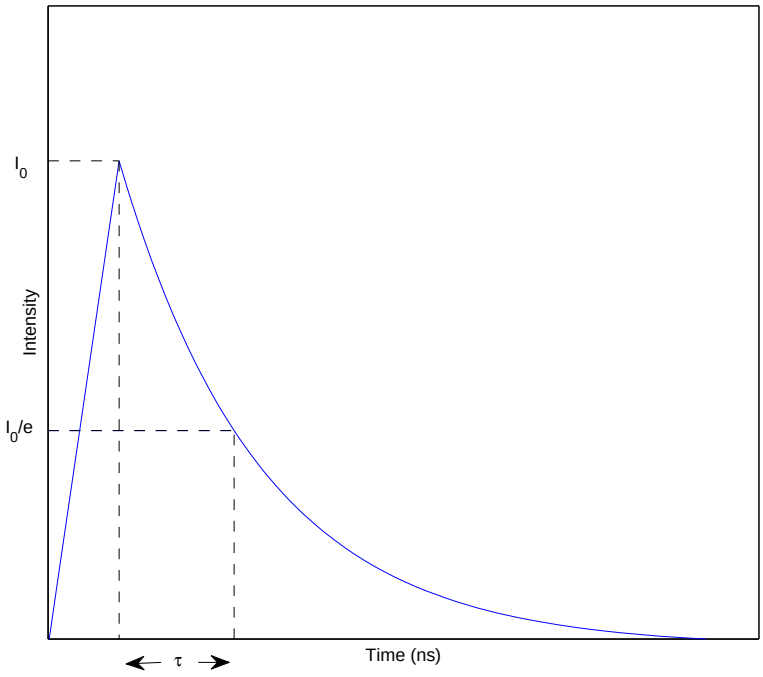
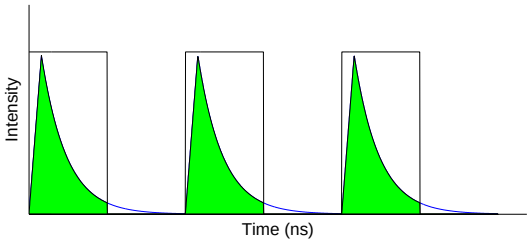
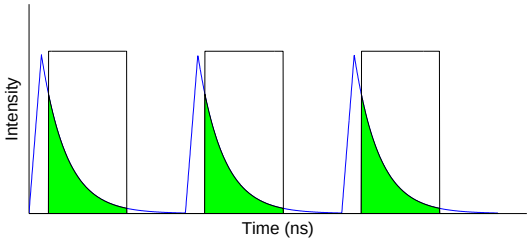


Figure 1: Emitted intensity of a compound.

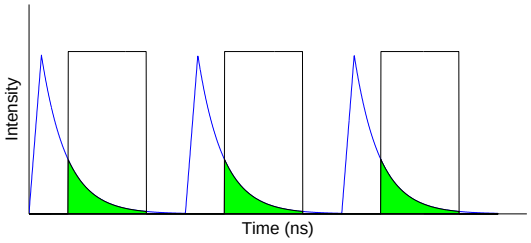
Frequency-Domain FLIM.
Another variant of FLIM measurements is Frequency-domain FLIM (FD-FLIM). In FD-FLIM is the emission of the photons from the light source a sequence of pulses at a fixed frequency, instead of one single pulse. The emitted intensity of the sample will follow the sequence and its frequency, but with a lower intensity and with a phase-shift with respect to the signal of the light source. The change in modulation and phase is directly dependent on the lifetime of the sample. The intensity emitted by the sample is measured by a detector, which has a gate signal which can be switched on and off at a fixed frequency. The frequency of the detector is set to the same frequency as that of the light source. The detector measures all light intensity emitted over the time interval in which the detector is active. The measured intensities are approximately equal over all pulses because the re-emitted intensity of the sample and the signal of the detector are in phase. By taking equidistantly spread values between 0 and 2π for the phase-shift of the detector it is possible to measure different intensities, as can be seen in Figure 2.



(a) $\phi = 0$



(b) $\phi = \pi/4$



(c) $\phi = \pi/2$

Figure 2: The emission of the sample (blue), the gate signal of the detector with a phase-shift ϕ (black) and the measured intensities (green).

There is a trigonometric relation between the measured intensities as function of the phase-shift, $I(\phi_i)$. The following section explains the formulas which are used to derive the lifetime based on these intensities.

Derivation of the formulas. The derivation of the formulas in based on the work in [3]. The intensities over time emitted by the modulating photon source is given by,

$$L(t) = a + b\cos(\omega t) \tag{1}$$

where ω is the modulation frequency. The fluorescent sample emits the photons with the same frequency ω and should have a similar intensity function, but with a lower fluctuation in the intensity. The lower fluctuation is the result of the additional time in which the photons are kept in the molecule, which smears the peaks of the function. The phase of the emitted signal is shifted by θ with respect to the emission of the source. This results in,

$$N(t) = A + B\cos(\omega t - \theta) \tag{2}$$

with $A < a$ and $B < b$. The overall decay of an intensity I_0, with a lifetime τ is given by the following exponential relation,

$$D(t) = I_0 e^{-\frac{t}{\tau}} \tag{3}$$

Because the fluorophore is sent from its ground state to an excited state and back to its ground state the intensity at every time be can computed with the convolution operator between the modulation signal from Equation (1) and the decay function from Equation (3).

$$\begin{aligned} I(t) &= L * D = \int_0^\infty L(t')D(t-t')dt' \\ &= I_0\tau\left(a + \frac{b}{\sqrt{1+\omega^2\tau^2}}\left(\frac{\cos(\omega t)}{\sqrt{1+\omega^2\tau^2}} + \frac{\omega\tau\cos(\omega t)}{\sqrt{1+\omega^2\tau^2}}\right)\right) \end{aligned}$$

Since both amplitudes of the trigonometric functions are positive values between 0 and 1 it holds that a right-angled triangle exists with, $\sin\theta = \frac{1}{\sqrt{1+(\tau\omega)^2}}$ and $\cos\theta = \frac{(\omega\tau)^2}{\sqrt{1+(\tau\omega)^2}}$ for $0 \leq \theta \leq \frac{\pi}{2}$. From which follows that the emitted photon intensity of the fluorophore is described by the following lifetime dependent function,

$$I(t) = I_0\tau\left(a + \frac{b}{\sqrt{1+\omega^2\tau^2}}\cos(\omega t - \theta)\right) \quad \text{with} \quad \theta = \arctan(\omega\tau) \tag{4}$$

When θ is measured and ω is known, the lifetime τ can be computed by the second part of Equation (4).

Besides that, it is possible to derive the lifetime, τ, based on the modulation, m, between the source and the fluorophore. The modulation of a single signal is the ratio between the amplitude and the equilibrium. The modulation between two signals is the ratio between both their modulations. Based on Equations (1), (2) and (4) the modulation is computed by,

$$m = \frac{\frac{B}{A}}{\frac{b}{a}} = \frac{\frac{I_0\tau b}{I_0\tau\ a\sqrt{1+\omega^2\tau^2}}}{\frac{b}{a}} = (1+\omega^2\tau^2)^{-\frac{1}{2}} \tag{5}$$

The value of m can be calculated from the intensities and ω is known. This means Equation (5) can be used to determine the lifetime τ.

Implementation of the FLIM algorithm.
The usual implementation of the previous algorithm is more complicated, because the method so far is very sensitive to noise. For the derivation of the lifetime of a sample a reference material is necessary, of which the lifetime is already known. Even though the lifetime of the reference material is known, both the sample and the reference are measured with the FLIM method with the same modulation frequency ω. This is necessary to determine the phase and modulation of the system. For both the sample and the reference measurements the light intensity, I, is measured at each phase ϕ_i. The intensity is fit with a trigonometric function, which gives the least square error by fitting with the zeroth and first Fourier harmonics. From both the fits of the sample and reference material the phase-shift, θ, and modulation, m, are derived.

The frequency modulation, ω, of the source is known and can be used to derive a theoretical phase, θ_{th}, and theoretical modulation, m_{th}. Under perfect conditions the theoretical values would be equal to the measurements from the reference material, which is never the case due to environmental variables and measurement noise. The results of the theoretical values and the measurement of the reference material lead to a phase-shift and modulation from the system.

$$\theta_{sys} = \theta_{ref} - \theta_{th} \quad m_{sys} = \frac{m_{ref}}{m_{th}}$$

These system values are characteristic for the current conditions of the measurements and follow from the phase induced by delay in the system. These factors should also be accounted for in the measurement of the sample. Removing these system terms from the phase and modulation of the sample will improve the quality of the determined lifetime of the sample.

$$\theta_{clean} = \theta_{sam} - \theta_{sys} \quad m_{clean} = \frac{m_{sam}}{m_{sys}}$$

This phase and modulation can be used to calculate the lifetime with $\tan(\theta_{clean}) = \omega\tau$ and $m_{clean} = (1 + \omega^2\tau^2)^{-\frac{1}{2}}$. If the reference values and theoretical values are the same then the modulation of the system would be 1 and the phase 0, which will mean that the $\theta_{clean} = \theta_{sam}$ and $m_{clean} = m_{sam}$. This is equivalent to the derivation of the lifetime in the previous subsection about FD-FLIM.

1.4 Usage of FLIM

The method of FLIM has a wide range of applications in microbiology. A list of some publications of customers of Lambert Instruments is shared on their site [2]. Besides that, Lambert Instruments investigates other possible uses for their products to broaden their market.

One of their customers is the Netherlands Cancer Institute (NKI) in Amsterdam. Lambert Instruments has a close relation with the NKI, to help them with their hardware and software. Besides, they highly value the feedback of their customers to help and improve their own products. The NKI uses the FLIM products of Lambert Instruments to track activity in living cells. They have high requirements for the quality and speed at which the data could be acquired, which is delivered by the products of Lambert Instruments.

1.5 Assignment

My main assignment was to construct a software module which could do the computational part of the single lifetime calculation in the Frequency-Domain FLIM. The algorithm is described in Section 1.3. In addition the implementation should satisfy the following properties:

- The algorithm should be fast. The previous software worked with one sample and one reference recording. However, an idea for the new software was to be able to work with multiple sample and reference recording at the same time and compute different lifetimes based on these recordings. A user of the software should be able to recompute these lifetimes fast for

a user friendlier workflow. In the new software speed could be an issue when multiple timestamps are recorded. So the implementation should be fast enough to prevent this.

- The algorithm should be fail safe, such that any error which may arise is caught by the software.The software module is called from a `C#` frame developed by Lambert Instruments, while the module itself is written in `C++`. For a smooth interaction between software modules should any possible crashes resulting from the new module be prevented.

- Just like in the current software there should be an estimate for the error of the calculated parameters. The previous creator of the software did not document the derivation of the error estimates well and some customers had questions about the error goodness of fit value for the lifetimes. The new software module should include a well documented error estimation method.

2 Methods

2.1 Error Estimation

In FLIM measurements the intensities are measured at a fixed number of equidistantly spread phases (ϕ_i) at each pixel of the domain. For each pixel this intensity is fit by a sine function, which means that the amplitude α, equilibrium β and the phase θ are the fitting parameters. The intensity $I(\phi_i)$ is influenced by noise resulting from different sources, meaning that the fit will not be perfect. The fitting parameters can be shifted by rewriting the fitting function as,

$$I(\phi_i) = \alpha \cos(\phi_i + \theta) + \beta \quad \Leftrightarrow \quad I(\phi_i) = a + b\cos\phi_i + c\cos\phi_i$$

for $\alpha = \sqrt{b^2 + c^2}$, $\beta = c$ and $\theta = \arctan\frac{c}{b}$. This way the fit equation is linear in its parameters, which means that it can be written as a system of linear equations at each pixel as $A\vec{x} = \vec{b}$. Matrix A is a Vandermonde like equation, where each row is given by $A_i = [1 \;\; \cos\phi_i \;\; \cos\phi_i]$. The vector with the fitting parameters is given by $\vec{x} = [a \;\; b \;\; c]^T$ and the right hand-side is given by $\vec{b}_i = I(\phi_i)$. The linear system is assumed to be overdetermined, which is the case when there are more data points than fitting parameters. This is also necessary for a unique least square error solution. This solution can be found by projecting the measured points to the span A, which is given by $A^TA\vec{x} = A^T\vec{b}$. The matrix A^TA is a square nonsingular matrix, therefore the inverse exists. This means that the least square solution is given by $\vec{x} = (A^TA)^{-1}A^T\vec{b}$. A similar derivation is given in [4], which uses a statistical approach for the fit, which is similar to the derivation so far but which can also be used for the error estimations of the fitting parameters. The statistical derivation assumes that all the noise is of Gaussian nature and that the standard error at each phase ϕ_i is the same. Computing the maximum likelihood of the fit is then equivalent to minimizing the error in the least square sense. An overall measure for the error is given by,

$$\sigma = \sqrt{\sum_{i=1}^{N} \frac{1}{\nu}(I(\phi_i) - A\vec{x})^2} \tag{6}$$

Where $\nu = N - 2K - 1$ is the degree of freedom, with N the number of phases ϕ_i and K the number of included harmonics. This measure for the error satisfies the following two properties.

- Increasing the number of phases decreases the error in the fit, while trying to compute multiple lifetimes increases the number of elements included in the fit and so increases the overall error.
- The method of fitting assumes that the zeroth and first harmonics are not influenced by noise, while all other harmonics are the result of noise. So the difference between the fit and the data is a measure for the noise and for the uncertainty of the error, which is also covered by the goodness of fit equation in Equation (6).

The goodness of fit equation can be used to determine a measure for the goodness of the estimations of the parameters of $\vec{x}$. These estimations can be derived from the inverse of $A^T A$, which is also called covariance matrix between the data and fitting parameters. Each diagonal element from this matrix, $(A^T A)^{-1}_{ii}$, multiplied by the overall goodness of fit equation, Equation (6), is a measure for the error in $\vec{x}_i$. This means that the error estimations for the fitting parameters are,

$$\begin{aligned} \sigma_a &= \tfrac{\sigma}{\sqrt{N}} \\ \sigma_b = \sigma_c &= \sigma\sqrt{\tfrac{2}{N}} \end{aligned} \tag{7}$$

Similar to how the parameters a, b and c can be used to determine the parameters θ and α, can σ_b an σ_c be used to determine σ_θ and σ_α. This follows from error propagation arithmetic. For the phase follows that,

$$\theta = \arctan\frac{c}{b} \quad \Leftrightarrow \quad \sigma_\theta = \frac{\sigma}{\alpha}\sqrt{\frac{2}{N}} \tag{8}$$

With FLIM two lifetimes are computed, one following from the phase, τ_θ, and one from the modulation, τ_m. To determine the error in the estimation of τ_θ the previous estimates could be used. For the error of the other lifetime, τ_m, also the error of the modulation is necessary which follows from the amplitude and equilibrium.

$$\begin{aligned} \alpha = \sqrt{b^2 + c^2} \quad &\Leftrightarrow \quad \sigma_\alpha = \sigma\sqrt{\tfrac{2}{N}} \\ m = \tfrac{\alpha}{\beta} \quad &\Leftrightarrow \quad \sigma_m = \tfrac{\sigma}{|\beta|\sqrt{N}}\sqrt{2 + m^2} \end{aligned} \tag{9}$$

Using these error estimates from the fitting parameters allows us to determine an error estimation in the computed lifetimes. It is assumed that there are no uncertainties in the modulation frequency and lifetime of the reference material.

$$\begin{aligned} \tau_\theta &= \frac{\tan\left(\theta_{ref}-\theta_{sam}+\theta_{th}\right)}{\omega} \quad \Leftrightarrow \quad \sigma_{\tau_\theta} = \frac{\sqrt{\sigma^2_{\theta_{ref}}+\sigma^2_{\theta_{sam}}}}{\omega\cos^2\left(\theta_{ref}-\theta_{sam}+\theta_{th}\right)} \\ \tau_m &= \frac{\sqrt{\left(\frac{m_{ref}}{m_{th}m_{sam}}\right)^2-1}}{\omega} \quad \Leftrightarrow \quad \sigma_{\tau_m} = \frac{\sqrt{\left(\frac{\sigma m_{ref}}{m_{ref}}\right)^2+\left(\frac{\sigma m_{sam}}{m_{sam}}\right)^2}}{\sqrt{1-\left(\frac{m_{sam}m_{th}}{m_{ref}}\right)^2}\frac{m_{sam}m_{th}}{m_{ref}}} \end{aligned} \tag{10}$$

Notes on software implementation.
The error estimates should be real and positive quantities. However, for the error estimate of τ_m this is not always the case. The inner value of the square root in the denominator can be negative, which would result in a complex error estimate. Besides that, the method can also fail in the case where the modulation of the sample or reference is zero, which could happen in dark regions in the recording or by overexposure of the camera. In both the cases where the error estimation may fail the error estimate is set to -1. By doing this `NaN`-values are omitted, which makes it easier to exchange the data with other sources. One should know about the choice of setting these quantities to -1 in the case of error averaging over a set of multiple pixels. The computation of the error happens in two different functions, `calcEstimateErrorFit` and `calcErrorLifetime`. All the data of a recording are stored in a class, which is called a flimage during development. Flimages contain at least the recorded intensities and the modulation frequency. Other variables can be stored in the flimage, like the amplitude, phase and equilibrium from the cosine fit on the data. By calling `calcEstimateErrorFit` the error estimates for the phase and modulation of the cosine fit are stored in the flimage. It was decided to keep the number of included elements in the flimage low, so instead of an error estimate for the phase, amplitude and equilibrium of the fit only the error estimates of the phase and modulation are stored. These two error estimates of both a sample flimage and reference flimage are input for `calcErrorLifetime`. This function stores the error estimate for the phase lifetime and modulation lifetime in the flimage of the sample recording.

Increasing the number of harmonics.

One single pixel consists of multiple molecules and each of these molecules has its own expected lifetime. When these molecules are similar then the observed lifetime could be assumed for all molecules of the pixel. However, when there is a varied mixture of molecules with different lifetimes then a single lifetime is not sufficient. By including an additional harmonic it is possible to derive a second lifetime, but the method is no longer a single lifetime computation. For the error estimations the previous FLIM software uses one harmonic if the number of phases is less than seven, two if it is less than nine and three otherwise. Two reasons for this could be:

- The creator did not thought that the error estimate of the lifetime based on the first harmonic was influenced by the number of included harmonics.
- Increasing the number of harmonics decreases the error between the fit and the data.

Besides single lifetime computations, the previous software is able to also compute multiple lifetimes per pixel. The first lifetime is strictly based on the first harmonic, meaning that the value of the first lifetime is the same for single and multiple lifetime computations. However, from the definition in Equation (6) can be concluded that the overall goodness of the fit does depend on the number of included harmonics. Since the error estimate of the lifetime is proportional to the goodness of the fit follows that the error in the lifetimes depends on the number of included harmonics. It is possible that this difference was an oversight by the previous developer, which explains why the same function is used for single and multiple lifetimes computations.

It is also possible that the implementation was intentional. Increasing the number of harmonics increases the size of matrix A with two rows and columns per harmonic, because each included harmonic introduces two new fitting parameters. In general does more fitting parameters improve the approximation of the data $I(\phi_i)$, but reduces the degree of freedom. When the relative reduction of the error of the fit is bigger than the relative reduction of the fitting parameters, then the goodness of fit from Equation (6) will be smaller. This would mean that the error estimate would be lower when more fitting parameters are included. This leads to a smaller error estimate for the parameter, which increase the certainty of the computed fitting parameter.

It is decided to not use the method of the previous error estimation with multiple lifetimes for a few reasons. At first, fitting a trigonometric function to a dataset is a form of regression. A general rule of thump for regression is that there should not be more than one fitting parameter per ten data points. Since the fitting functions start with three fitting parameters is it advised to not increase this number even more to preserve the quality of the overall estimation method. The second point is that error estimate is also a measure for the noise in the recording. When more harmonics are used for the fit is the quality of the approximation of the noise reduced. A good estimation for the noise is important to compensate for the assumption that the zeroth and first harmonic are free from any noise, because they are not. The last argument is that a large error estimation could indicate an additional dominant lifetime in a pixel. The user should investigate the second lifetime to improve his overall results. Deriving the other lifetimes will directly improve the previous estimate of the error of the first lifetimes.

2.2 Floats versus doubles

One of the differences between the old and the new software is that the old software stored all of its non-integer data with double precision. The benefit is the high accuracy, which results in a smaller error in the computed lifetimes. However, there are some advantages for working with floats instead. The first benefit is memory storage, which is half the size of a double. The second point is processing speed. In general memory storage is only a limitation in situations with a lot of measurements but the second benefit is more useful. Most processors can process float arithmetic faster than doubles. A recording contains multiple images of 516×504 pixels. Since each pixel needs multiple operations the total number of operations is of the order of 10^6, so the processing speed will be significantly faster for floats than for doubles.

The downside of using floats during computations is the loss of accuracy.

Using a float instead of a double reduces the number of significant digits from around 16 to 8. This is in general not a problem since lifetimes do not have more than two significant digits, due to measurement errors and other uncertainties. However, there are some noticeable situations where the error is exceptionally large. When the error estimate of the lifetime is computed there are some regions which have an error of 10^3 nanoseconds, which extraordinary because lifetimes are usually of order 10^0 nanoseconds. Inspection of the values at those points show that the phase, θ is close to $\pm\frac{1}{2}\pi$ and $\pm 1\frac{1}{2}\pi$. This explains the large error in the lifetime, because of the asymptote of $\tan(\theta)$. Around these values for θ a small perturbation lead to very different values. In general, lifetimes this big are excluded from the data, since they do not describe the real lifetime. A usual method is to only accept 98% of the data and to remove the top and bottom 1%. The remaining lifetimes should not differ too much from the lifetimes derived using doubles.

Multiple test cases of real FLIM recordings were run to check the difference between using float precision and double precision. From the measured intensities the lifetimes are computed by using the previous software with double precision, and the new software with float precision. The difference between the two at each pixel is then normalized by the computed lifetime with double precision. The results can be seen in Figure 3. From Figure 3 it can be seen that lifetimes given in floats and doubles match at least in first significant decimal. The lifetimes corresponding to the largest difference between the float and double lifetimes are the lifetimes with the smallest and largest values. Since the lifetime is usually given up to two significant digits, does it hold that the usage of floats instead of doubles will not significantly affect the data.

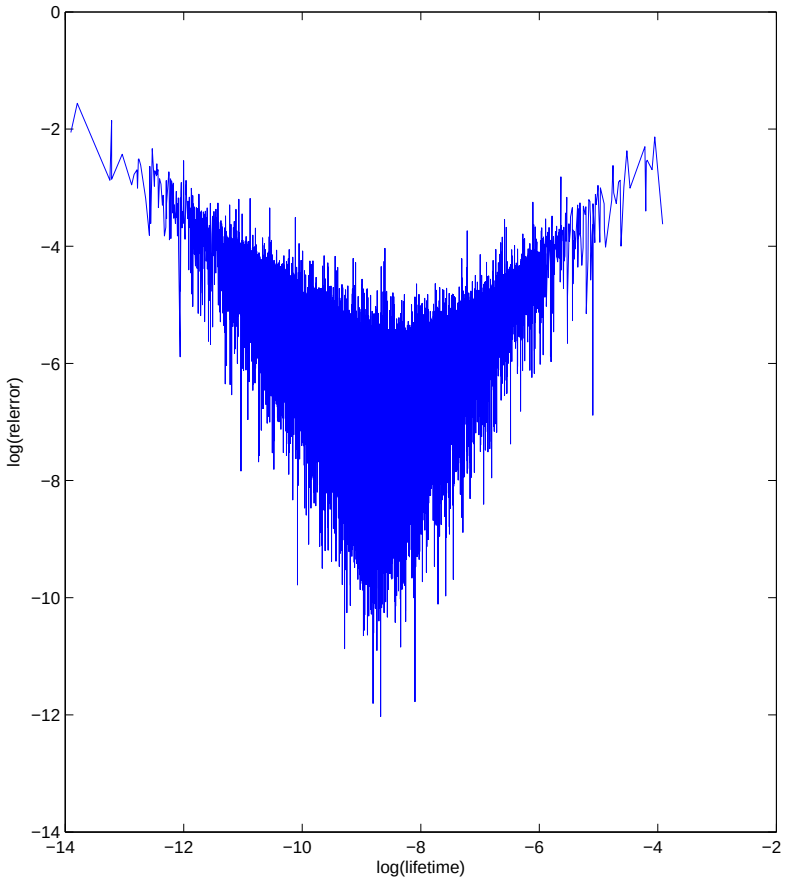


Figure 3: The relative difference between using floats and doubles for the lifetime derived from the phase in seconds

The largest error in the lifetime based on the phase happened because the phase was close to $\pm\frac{1}{2}\pi$ and $\pm 1\frac{1}{2}\pi$. Close to those points the method for the lifetimes based on the phase gives inaccurate results. This problem does not happen for the lifetime based on the modulation. This explains why the overall error based on this method is much lower, as can be seen in Figure 4. It can be concluded that the usage of floats instead of doubles does also not affect the lifetimes based on the modulation much. Moreover, relevant lifetimes are in the range from 10^{-2} to 10^{2} nanoseconds. For both lifetimes in this range is the relative difference between both precisions less than 10^{-4}. From which can be concluded that using floats or doubles will not change the results significantly.

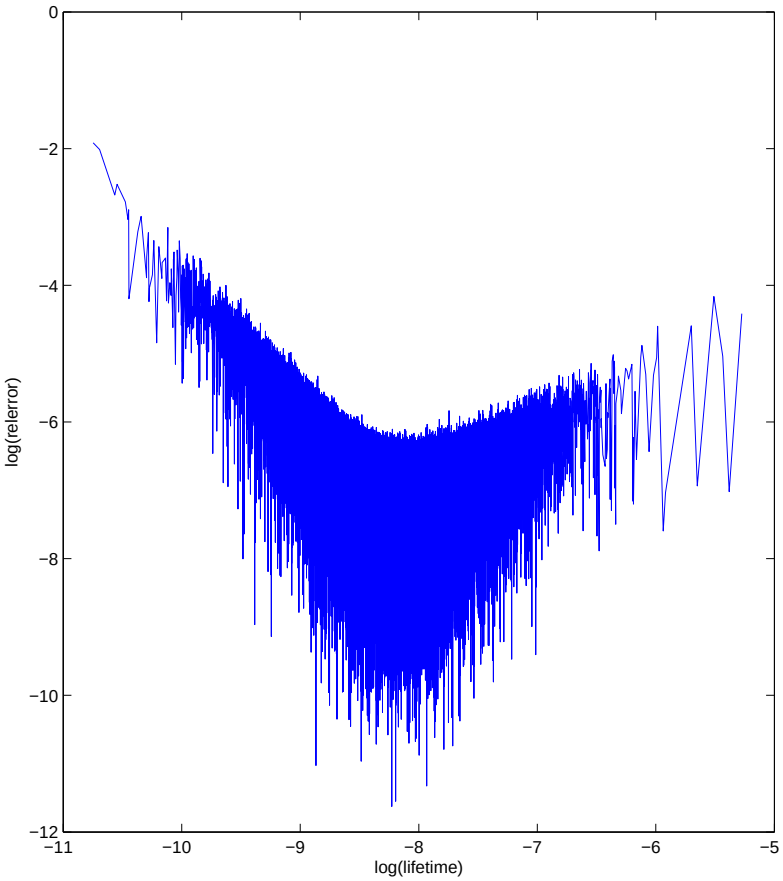


Figure 4: The relative difference between using floats and doubles for the lifetime derived from the modulation in seconds

2.3 Implementation overview and Failsafe

The designed software should be failsafe. This means that it may fail in its task, but should not crash while doing so. Besides that, everything should be put back to normal when it fails in its run. To explain how I did this I will first show the layout of the software module. The software module consists of four main functions.

- `fitSineToRecording(dataToFit, timeStamp)` This function removes the dark image from the intensities and computes the best cosine fit through these points. The dark image is a recording of a black screen, which should be subtracted from each recording to reduce dark noise.
- `calcLifetimeImage(sample, reference, timeStamp, minIntensity, maxIntensity)` This function takes the fitting parameters from the sample

and reference so it can compute the modulation and phase based lifetimes.

- `calcEstimateErrorFit(data, timeStamp)` This function computes the error estimate of the fitting parameters of the data.
- `calcErrorLifetime(sample, reference, timeStamp)` This function uses the error estimates and fitting parameters of both the sample and reference recordings to derive an estimate for the error in the lifetimes computation.

`fitSineToRecording(dataToFit, timeStamp)`:
This function processes the data, so the first check is if the data is set. This is the case when the width, height and number of frames of the recording are not zero. Besides that, the number of phases cannot be less than 3, because this would result in an undefined phase-shift and amplitude of the fit. The software was mainly tested for 16 bit unsigned data, but also supports 8 bit input. The first element of the recording set determines in what format the other elements are treated. The dark image is by default a zero matrix, with the same size as the first image from the recoding. An error is thrown if the bit depth or size are not the same as those of the recording. One of the first actions of the function is to lock the data from the recording. This way no other process could change the data on which the function determines the fit. The function halts if it cannot lock the data, which could happen if another process uses it or the mutex failed. A mutex is the right for a function to use a variable, which cannot be acquired if something else has the mutex, but also in some rare cases where the mutex is available. The data is stored in a vector of matrices, where the matrices could differ from size from each other. That is why the dimensions of each recording are checked. The next step is to reserve memory for the output and lock it so no other functions can use it. The program halts if any of those steps fails. At the arithmetic part the code can not fail, since all elements are checked and there are no zero divisions. The only function which could fail is `atan2`, but this has some internal methods to ensure a floating point output. After this all mutexes are freed. If this function or any of the other functions fail after the mutexes are locked, then its first priority is to release all of its locked mutexes. This way a variable cannot be locked, while a function is not using it.

`calcLifetimeImage(sample, reference, timeStamp, minIntensity, maxIntensity)`:
This function uses two flimages, one sample and one reference, and computes the lifetimes in the first of those two. The first check is if the modulation frequency is set for both flimages and if both have the same value. The next step is to check if a reference lifetime is set for the second flimage. It then checks if all fitting parameters of the sample are available. By the maximum of the fitted equilibrium of the sample an interval is determined by using `minIntensity` and `maxIntensity` in which the lifetimes are computed. Both the lifetimes of pixels with an equilibrium outside this range are not computed and set to zero. By default this thresholding is not active. The function `fitSineToRecording` is called if the fitting parameters are not computed and the threshold range is recalculated. The whole function stops if the fitting parameters are not correct after the recalculation. If the function did succeed all the fitting parameters of the sample are locked from this point. The same procedure follows for the

reference flimage, but without the computation of the threshold range. Now both the sample and reference are locked the function checks if they have the same length and width. A sample and reference are allowed to have a different number of phases in the recording or even be recorded with a different bit depth. The last step of the initialization is to create the output images in the sample flimage for the modulation lifetime and phase lifetime. The lifetime based on the phase will not give any errors, since it consist of some additions and a `tan` function. The fraction of π at which the `tan` is undefined cannot be represented on a computer, meaning that the `tan(x)` function will be defined for any float value input `x`. The lifetime derived from the modulation consists of many divisions, meaning that a zero division is possible. Besides that, the argument of `sqrt` could be negative. It is possible in `C++` to use a square root function which allows complex arithmetic, but this is of no value for the derivation of a real lifetime. All the lifetimes with `NaN` values are set to 0 to keep the output strictly numeric. This makes interaction with other software easier. After this all the mutexes are freed.

`calcEstimateErrorFit(data, timeStamp):`
This function is very similar in layout to `fitSineToRecording`. It checks the recording for the width, height, number of phases and bit depth. After which, the function locks the recording and checks if all the recordings have the same size. The dark image is checked for the same criteria as for the fitting function. In addition, it also locks and checks the fitting parameters. The function computes the fitting parameters by calling `fitSineToRecording` if one of the fitting parameters is not set. The final step of the initialization is the creation of the output parameters, the phase and modulation error estimates. The arithmetic part can only fail if the fitted amplitude or equilibrium is 0. In those cases the error estimate is set to -1, to indicate that the error estimate cannot be computed for that pixel. After the computation of the error estimates all mutexes are freed.

`calcErrorLifetime(sample, reference, timeStamp):`
This function needs both parameters from `fitSineToRecording` and `calcEstimateErrorFit`. Most of the initial checks are the same as for `calcLifetimeImage`, like checking if the frequency is set and if both frequencies are the same. After this, the fitting parameters from the sample and reference are checked, locked, and recalculated if necessary. The same procedure then follows for the error estimates for the phase and modulation for both the sample and the reference. The size of the error estimation matrices are checked and the output images are reserved and locked. After this, the errors in the lifetimes are determined for each pixel. If in a pixel the error estimate of the phase or modulation is -1, then the error in the lifetimes is also set to -1 to indicate that the error estimation function failed for this pixel. The same safeguard is also included for any possible divisions by 0. After the computation of the errors all mutexes are freed.

3 Conclusion

My assignment was to write a software module which could do the FLIM algorithm and had the following properties,

- The algorithm should be fast.
- The algorithm should be failsafe.
- It should include an error estimation method.

Compared to the previous software there are some improvements in terms of speed. However, the final product is far from complete and misses most of the functionality of the previous software. That is why this report does not contain any benchmark results. But based on the code implementation the new software should be faster.

In the current state the software is already very robust, but some unknown situations might still cause problems. Additional security checks might be needed and those can be added relatively simple due to the clear structure of the code.

Besides the implementation of a method for the estimation of the error, I did a clear documentation on the method. This is a big help at Lambert Instruments, because the previous code creator no longer works there and did not leave any documentation of his method. The previous error estimation and my version did differ in the inclusion of multiple harmonics, which was explained in the last paragraph of Section 2.1. My colleagues agreed with my decision to only use the first harmonic in the derivation of the error estimates, instead of also the second and third for recordings with more than seven phase, which was used in the previous software.

A Bibliography

References

[1] Lambert Instruments. https://www.lambertinstruments.com/about-us/.

[2] Lambert Instruments. https://www.lambertinstruments.com/publications/.

[3] J.R. Lakowicz. *Principles of Fluoescence Spectroscopy.* Kluwer Academic/Plenum Publishers, New York, Boston, Dordrecht, London, Moscow, 1991.

[4] Daphne S. Bindels Marten Postma and Dorus Gadella. Fitting the homodyne signal and parameter uncertainty analysis.