Time Delay Estimation in Gravitationally Lensed Photon Stream Pairs

Michał Staniaszek
Supervisor: Peter Tiňo

July 26, 2013

Abstract

Due to a phenomenon called gravitational lensing, under certain conditions we can see multiple images of the same objects in space. The light from each image takes a different amount of time to get to us. Our system estimates this time difference by looking at individual photons coming from each image, and reconstructing a function which represents the image. We then find the time shift where these functions match up best, either by looking at the area between the functions, or by creating an "average" function and calculating the likelihood of the functions for individual images being created from it.

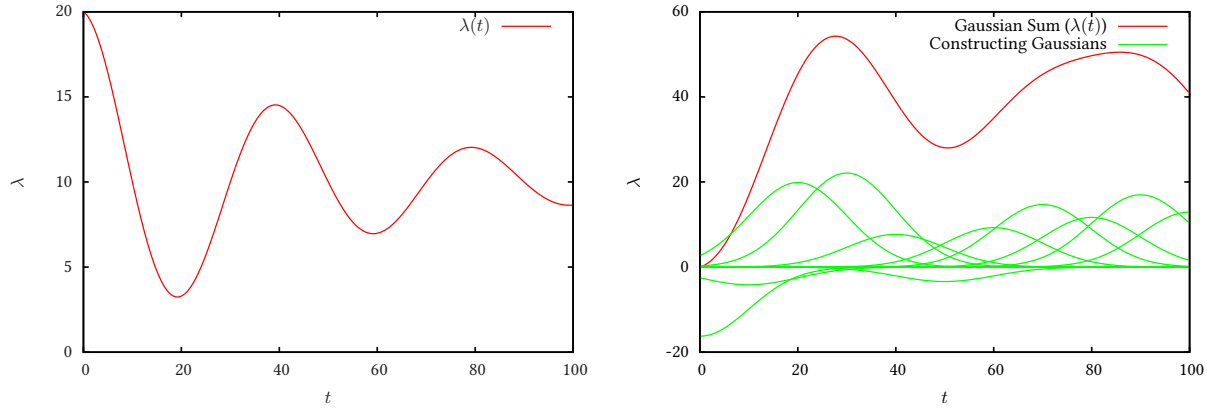
1 Introduction

2 Photon Stream Simulation

First, we developed a subsystem which could be used to generate simulated photon stream data to use for the development and testing of the rest of the system. The only property of the photons which we are interested in is their arrival time at our capture device, so the simulator should produce some event vector $\Phi = [\phi_0, \ldots, \phi_N], \phi_n \in \mathbb{R}$, where ϕ_n is the arrival time of the nth photon. In order to generate arrival times, we represent the source as some random variable X, which defines the average number of photons per unit time that arrive at the capture device. This varies according to the characteristic function of the source object. The characteristic function of X is modelled as a non-homogeneous Poisson process (NHPP) as a continuous function of time, $\lambda(t)$, known as the rate function. The system allows the rate function to be specified either by providing an expression which is a function of t, or by sampling from a randomly generated function. Random functions are constructed by uniformly distributing M Gaussians across the interval $[t_0, T]$ in which arrival times are to be generated. Each Gaussian g_i is defined by its mean μ_i, its width σ_i, and its weight w_i, which determines its height. The means of successive Gaussians are separated by some distance Δt, such that $\mu_{m+1} = \mu_m + \Delta t$, where $\mu_0 = 0$. Greater variation in the functions is introduced by sampling the weights w_i from a uniform distribution $U(-1, 1)$ and scaling them by some multiplier. The value of the randomly generated function at some time t is computed by a weighted sum of Gaussians.

$$\lambda(t) = \sum_{i=0}^{M} w_i \cdot e^{-(t-\mu_i)^2/2\sigma_i^2} \tag{1}$$

Having defined or constructed $\lambda(t)$, photon arrival times are generated from a homogeneous Poisson process (HPP) with constant rate λ, using inverse transform sampling. The waiting time to the



(a) (b)

Figure 1: Two examples of function generation capabilities. (a) is generated from a damped sine function of the form $e^{-t} \cdot \cos(2\pi t)$. (b) shows a randomly generated function where the red function is constructed from the green Gaussians with $\Delta t = 10$, $\mu = 10$ and shifted so that all points are ≥ 0.

next event in a Poisson process is [1]

$$t = -\frac{1}{\lambda} \log(U) \tag{2}$$

where $U \sim U(0, 1)$. Knowing this, it is possible to generate successive events of a HPP for any finite interval, from which events for the NHPP can then be extracted by thinning, using Algorithm 1. The number of events added to the event vector Φ in any given interval is proportional to the value of $\lambda(t)$ in that interval; the probability of adding an event is low when $\lambda(t)$ is small, and increases with the value of the rate function.

Algorithm 1 Generating event times for a NHPP by thinning

Require: $\lambda \geq \lambda(t), t_0 \leq t \leq T$

$\Phi = \emptyset$, $t = t_0$, $T =$ interval length
while $t < T$ **do**
  Generate $U_1 \sim U(0, 1)$
  $t = t - \frac{1}{\lambda} \ln(U_1)$
  Generate $U_2 \sim U(0, 1)$, independent of U_1
  if $U_2 \leq \frac{\lambda(t)}{\lambda}$ **then**
    $\Phi \leftarrow t$
  end if
end while
return Φ

3 Function Estimation

The function estimator subsystem receives input of the event vector Φ, and attempts to reconstruct the rate function. As the photons are emitted by a truly random process, it is only possible to obtain an estimate of the true rate function. In the project, we used two different methods to obtain an estimate.

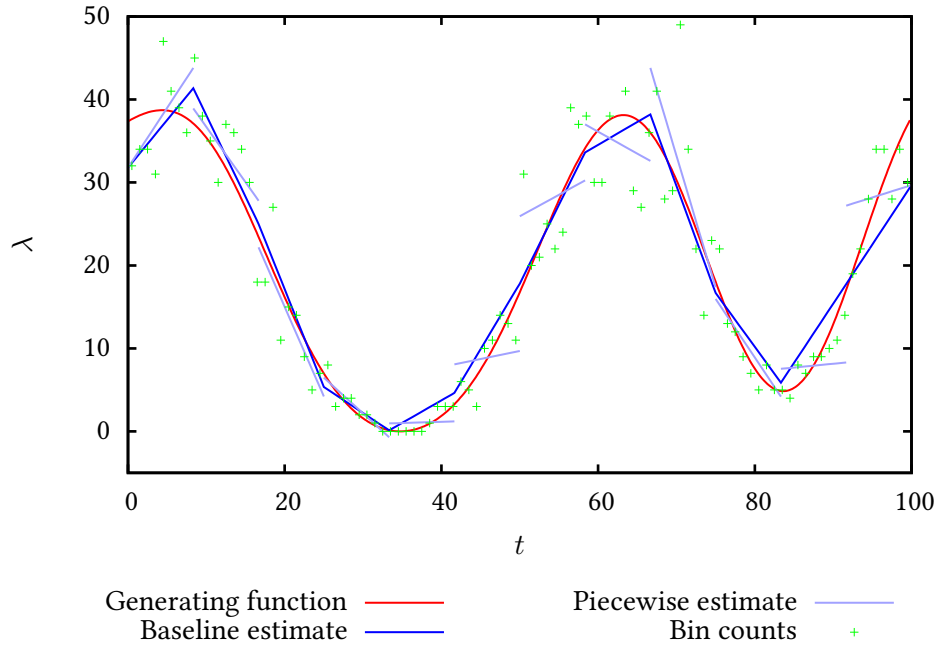


Figure 2: A comparison of the baseline and piecewise estimates on the same function. Note how the baseline estimate passes through the midpoint of the disjoint piecewise estimates at the breakpoints. The estimators used an upper limit of 12 sub-intervals, and bins were 1 time unit in length.

3.1 Baseline Estimation

Development of the baseline estimator went through several stages. Based on the work of Massey et al.[2], we implemented a system to estimate the rate function of a set of events using iteratively weighted least squares (IWLS). The interval $[t_0, T]$ is split into several bins, each represented by the number of events which occur within it. IWLS produces a linear estimate of the rate function by an iterative process which minimises the sum of squared residuals from an initial estimate of the function.

Linear estimates are not sufficient for representing rate functions, so we extended the technique by estimating the rate function in several sub-intervals and combining these estimates into a single estimate, rather than using a single estimate from the whole interval. Once an estimate for the sub-interval has been computed, attempts are made to extend the estimate into a short interval after the initial sub-interval. The Poisson probability density function (PDF) in Equation 3 is used to determine the likelihood of obtaining the count Y_k for each bin in the extension interval. The likelihood of each bin is required to be above a certain threshold. If it is not, the estimate is not extended.

$$P(Y_k = x) = \frac{\lambda^x e^{-\lambda}}{x!} \tag{3}$$

This extension of IWLS produces piecewise disjoint estimates of the rate function. In order to produce the piecewise continuous functions that we require, we adjust the estimate in each sub-interval. We define breakpoints as the point in time where one sub-interval ends and another begins. There are $R = L - 1$ breakpoints r, where L is the number of sub-intervals. At each breakpoint, the values of the two function estimates f before adjustment are computed, and the midpoint m is calculated.

$$m_i = \frac{f_i(r_i) + f_{i+1}(r_i)}{2}, \quad 0 \leq i < R \tag{4}$$

At the start of the first and end of the last sub-intervals the original function value is used as the midpoint. Each sub-interval is now represented by a point p at the start and q at the end, each with

an x and y coordinate. With these points, we can recalculate each sub-interval estimate f of the form $y = \hat{a} + \hat{b}x$ by replacing y with p_y and x with p_x, and recalculating the gradient $\hat{b}$ and intercept $\hat{a}$ with

$$\hat{b} = \frac{q_y - p_y}{q_x - p_x} \tag{5}$$

$$\hat{a} = p_y - \hat{b} \cdot p_x \tag{6}$$

This adjustment produces the desired piecewise continuous function by having the values of the estimates from the previous and next sub-interval have the same value at the breakpoint.

3.2 Kernel Density Estimation

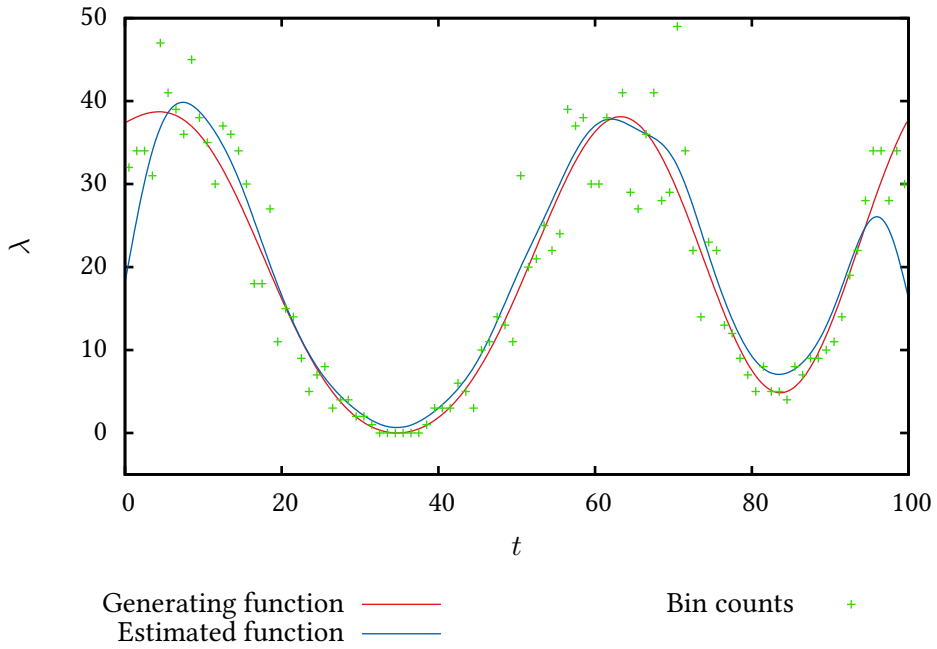


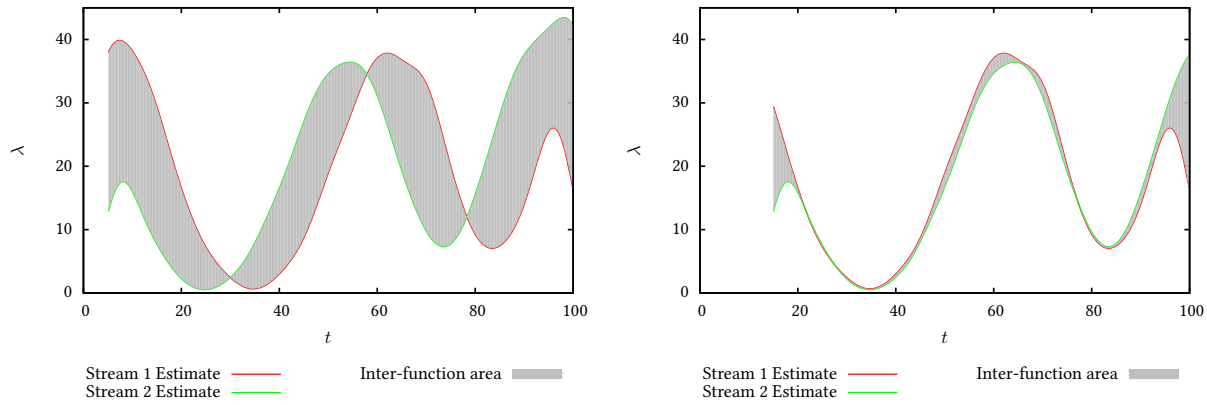
Figure 3: Kernel density estimate of the function from Figure 3 showing an example of the smoother functions produced. Note the drop-off of the function at the start and end of the interval caused by a lack of samples in those areas to allow the result of the transform to give an accurate estimate.

The second function estimation method implemented was a kernel density estimator, which uses *kernels* to estimate the probability density of a random variable. Since the photon stream data is generated by a source whose variability is defined by some random variable, the event times are a sample drawn from the PDF of that variable. We use a Gaussian kernel

$$K(t, \mu) = e^{-(t-\mu)^2/2\sigma^2} \tag{7}$$

to estimate the PDF, centring a kernel at each photon arrival time ϕ_n by setting $\mu = \phi_n$. The width of the kernel depends on some fixed value σ. We perform a Gauss transform on the N kernels, finding the contribution of all the kernels at M points in time, from which we get an estimate $\hat{\lambda}(t)$ of the characteristic function.

$$\hat{\lambda}(t_i) = \sum_{j=1}^{N} K(t_i, \mu_j), \quad i = 1, \ldots, M \tag{8}$$



(a) High area ($\Delta = 5$)

(b) Low area ($\Delta = 15.1$)

Figure 4: An illustration of the idea behind the area method. In (a), the applied shift results in a large area between the functions. In (b), the shift is very close to the actual time delay ($\Delta = 15$), and the resulting area is much smaller than (a), indicating that the shift in (b) is a better estimate.

Using a larger M gives a higher resolution. Depending on the value of σ used, $\hat{\lambda}(t)$ will be some multiple of the actual function $\lambda(t)$. Thus, the final step is to normalise $\hat{\lambda}(t)$. We split the stream data into B bins with midpoints b and calculate the bin count x for each. We start with the normalisation constant η at a low value, and gradually increase it to some threshold, finding

$$\sum_{i=1}^{B} \log\left(\frac{\phi^x e^{-\phi}}{x!}\right), \quad \phi = \eta \cdot \hat{\lambda}(b_i) \tag{9}$$

for each value of η. The value of η which maximises this sum of log Poisson PDFs is used to normalise $\hat{\lambda}(t)$ in subsequent computations.

4 Time Delay Estimation

Once we are able to estimate the characteristic function of photon streams, we can use these estimates to compute an estimate of the time delay between two streams. If the two streams come from the same source, then they should have the same characteristic function, but delayed by some value Δ. Our estimates of the characteristic function will differ for both streams due to the fact that the number of photon arrivals in each bin will be different for each stream, but each should look relatively similar. In this section we present two methods for estimating the time delay between a pair of streams based on their function estimates.

Both of the estimators work by starting Δ at $-\Delta_{\text{max}}$, and increment it by some step until $+\Delta_{\text{max}}$ is reached, using a metric to evaluate how good the estimate is with that value. Estimates are computed based on data in the interval $T_{\text{est}} = [t_0 + \Delta_{\text{max}}, T - \Delta_{\text{max}}]$, where $\Delta_{\text{max}} < T$. This constraint on T_{est} means that there is no need to normalise the likelihood of matches due to differing interval lengths.

In the approximation, t is incremented by some finite step for each successive value. As with the previous estimator, the estimate is made in two stages, first with a coarse pass over the values of delta to compute an initial estimate, and then a finer second pass around the first estimated value in order to refine the estimate.

4.1 Area Method

The first of the two methods uses a very simple metric to estimate the time delay. By taking the two function estimates, we can attempt to match up the two functions so that they "fit together" best. The goodness of fit between two functions $\hat{\lambda}_1$ and $\hat{\lambda}_2$ is defined by the area between them, calculated by

$$\begin{aligned} d(\hat{\lambda}_1, \hat{\lambda}_2) &= \int (\hat{\lambda}_1(t) - \hat{\lambda}_2(t+\Delta))^2 \, dt \\ &\approx \frac{1}{N} \sum_{i=1}^{N} (\hat{\lambda}_1(t) - \hat{\lambda}_2(t+\Delta))^2 \end{aligned} \tag{10}$$

for each value of Δ. Our estimate of Δ is set to the value at which $d(\hat{\lambda}_1, \hat{\lambda}_2)$ is minimised. Rather than using an integral to get the exact area between the functions, we use a less computationally expensive discrete approximation.

4.2 PDF Method

The second method of estimation is using probability density functions. As before, we guess a value of Δ between $-\Delta_{\max}$ and $+\Delta_{\max}$ and shift $\hat{\lambda}_2$ by that amount. However, we know that there must be a single characteristic function, and we want to see how well our estimate of that matches the bin counts in each stream. We make an "average" function $\bar{\lambda}$ by combining the two function estimates we have, $\hat{\lambda}_1$ and $\hat{\lambda}_2$ (which is shifted by Δ).

$$\bar{\lambda}(t) = \frac{\hat{\lambda}_1(t) + \hat{\lambda}_2(t+\Delta)}{2} \tag{11}$$

The point on $\bar{\lambda}$ at time t is the midpoint between the values of the two estimates at that time. Once we have $\bar{\lambda}$, we can assign some score to the current estimate of the value of Δ.

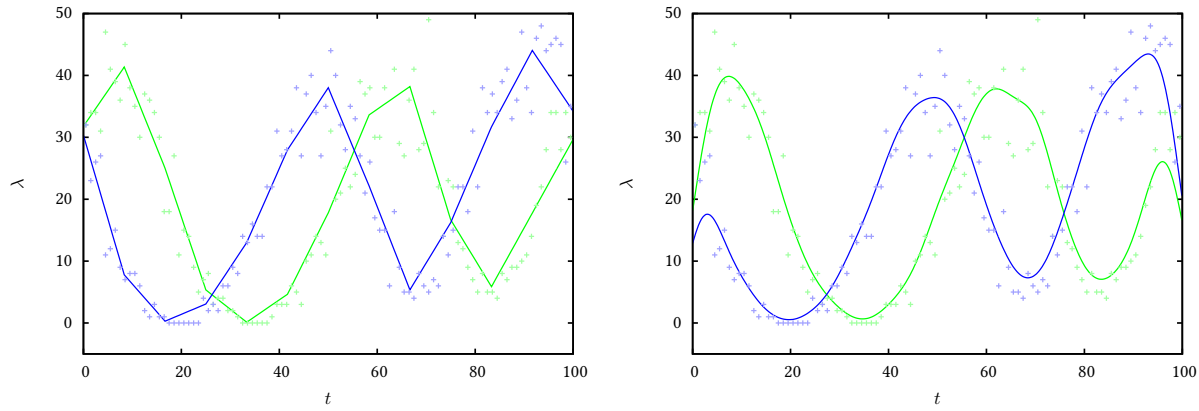
$$\begin{aligned} \log P(S_A, S_B \mid \bar{\lambda}(t)) = \sum_{t=\Delta_{\max}}^{T-\Delta_{\max}} & \log P(S_A(t) \mid \bar{\lambda}(t)) \\ & + \log P(S_B(t+\Delta) \mid \bar{\lambda}(t)) \end{aligned} \tag{12}$$

Using (12), we calculate the probability that the function $\bar{\lambda}$ is the characteristic function of the two streams S_A and S_B. The streams are split into bins, and the log probability of the number of events in each bin given the value of λ calculated for that bin is computed and summed over all bins, as in Equation (9).

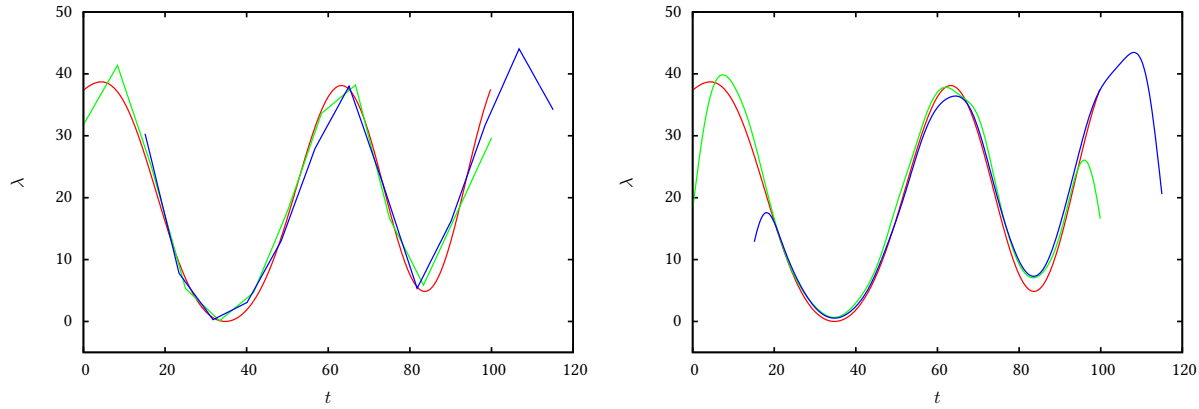
The calculation of λ is slightly more complicated than just taking its value at the midpoint of each bin. Since we are considering a number of events occurring in a given interval, we must consider the value of λ for the same interval. In order to do this, we use a discrete approximation of integrating $\lambda(t)$ over the interval.

$$\lambda_{a,b} = \int_a^b \lambda(t) \, dt \tag{13}$$

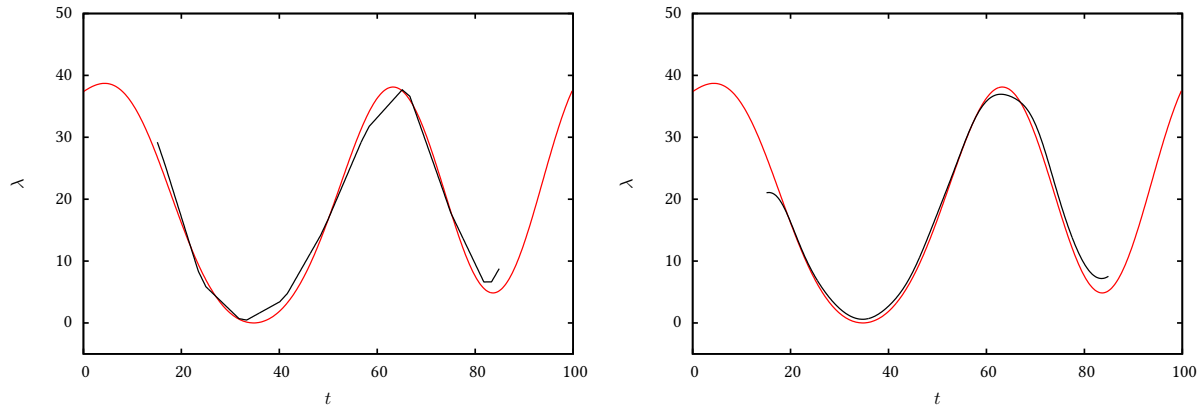
5 illustrates the whole estimation process.



(a) Estimate $\lambda(t)$ for each stream based on bin data.



(b) Run the time delay estimator and shift the second (blue) function according to the estimate of Δ.



(c) Combine the two function estimates to create a final estimate (black).

Figure 5: An example of the estimation process using two different techniques. Left column shows baseline method, right column Gaussian. The estimated value of Δ was 15.1 for both estimators, found using the PDF method. The actual value of Δ was 15. Points in (a) represent bin counts for the function of the same colour. The red line in (b) and (c) indicates the actual function, black is the final estimate. Estimated functions are only combined in the interval in which they both have values.

5 Experimental Results

The four possible method combinations were compared in four sets experiments. 100 time units of Photon stream data was generated from sine functions of the form $y = a - b\sin(\alpha t)$ in the first set of two experiments, and from randomly generated functions in the second set. In both cases the α parameter defined the variability of the function. The experiments tested performance on functions generated with several different α values. Multiple photon streams were generated from each function to obtain a larger statistical sample. The sine function experiments used 25 independently generated stream pairs for the first, and 10 for the second. Random function experiments used 5 pairs for each of the 5 functions tested.

The first stage of each experiment found optimum parameter settings for each function in the experimental data set using model selection. The kernel density and baseline estimators were used to find $\hat{\lambda}(t)$ from stream data where 4 time units of stream data were withheld every 15 time units. The number of events in each bin in withheld sub-intervals was retrieved from the stream and compared to the value of $\hat{\lambda}(t)$ at the midpoint of that bin using the log Poisson PDF. The sum of log PDF values over all the sub-intervals was used to represent the parameter set's generalisation ability. The set with the highest value was used in the second stage of the experiment, where the time delay for a pair of streams was estimated using both the area and the PDF methods. This resulted in 4 estimates of the time delay, one from each combination of function estimator and time delay estimator.

Paired and single-sample t-tests were applied to the resulting estimates to check for any significant difference in the performance of the method combinations, but there was no indication of any significance.

6 System

The system was implemented in C, and uses the GMP and muParser libraries. The system is modelled after standard linux packages, using a similar directory structure compiling with GNU Automake. User interaction is through a command line interface, with modifiable parameter files. Bash scripts and Gnuplot provide graphing functionality. Over 60 unit tests implemented with the Check framework provide verification for critical functions. All source code is freely available on GitHub [3] under the GNU General Public License.

7 Conclusion

In this paper, we have provided a short overview of the concepts behind our methods of time delay estimation in paired photon streams. While the performance of the estimators is not optimal, we believe that it may indicate some directions for further work. A fast Gauss transform implementation would improve computation time for the kernel density estimator. Hierarchical search to find the maximum of the PDF at the baseline estimator breakpoints would improve the quality of estimates. Further improvements could come from an investigation into techniques to deal with highly symmetric or repeating functions such as sine waves.

References

[1] Donald E. Knuth. *The Art Of Computer Programming, Volume 2: Seminumerical Algorithms, 3/E.* 1998. Chap. 3.4.1.

[2] William A Massey, Geraldine A Parker, and Ward Whitt. “Estimating the parameters of a nonhomogeneous Poisson process with linear rate”. In: *Telecommunication Systems* 5.2 (1996), pp. 361–388.

[3] Michal Staniaszek. *Project Repository*. URL: `http://github.com/heuristicus/final-year-project`.