Time Delay Estimation in Gravitationally Lensed Photon Stream Pairs

Michał Staniaszek
Supervisor: Peter Tiňo

March 21, 2013

Abstract

In this report, we present a system for estimating the time delay Δ between multiple realisations of a Poisson process with the underlying function $\lambda(t)$, with particular application to gravitationally lensed photon streams. We develop a linear estimator based on weighted least squares, and a kernel density estimator which we use to estimate $\lambda(t)$. We then introduce two methods for estimating the value of Δ using the function estimates, one using inter-function area, and another using probability density functions. Finally, we compare the performance of the two function estimation methods and time delay estimation methods on simulated data, and show that there is not a significant difference between the approaches.

Keywords: Poisson process, gravitational lensing, machine learning, linear estimation

Typeset in Linux Libertine using XeLaTeX.

Contents

1 Introduction

With continued advances in computing and sensing technologies, the amount of data that can be gathered from both everyday objects and scientific experiments has increased rapidly. However, more data is not always a blessing—it must be stored and analysed for it to have any use, and this is not an easy task when one has terabytes of data to deal with. The Large Hadron Collider at CERN is one perhaps extreme example, producing on the order of five terabytes of data each second. Storing this amount of data, let along analysing it is impossible, and so multiple stages of intelligent filtering are applied, reducing the throughput to 100 gigabytes per second, and then further to around 200 megabytes per second, where it is finally stored, producing almost two CDs each second [15]. This project focuses on creating the foundations for a system to do such intelligent filtering, but in the context of astronomical data. The volume of data produced by modern telescopes, while not on the same scale as the LHC, is nonetheless overwhelming. Image sizes of one to two gigabytes are not uncommon, and deciding what data is actually relevant is not a trivial task [19]. Using intelligent filtering algorithms, it should be possible to flag up interesting-looking data for further study. While there are many areas in which such capabilities would be useful, we are particularly interested in finding candidates for images of gravitationally lensed objects. In order to do this, it is necessary to find pairs of observations of photon flux which appear to have the same underlying function. More precisely, given a set of data containing the time of arrival of photons from a particular source, henceforth called a *stream*, we wish to find another stream which, when shifted in time by some value Δ, has similar numbers of photons arriving in a given interval as the first stream. We call Δ the *delay* between the two streams. In this project, we develop a system which can generate simulated photon streams using Poisson processes, use linear regression to estimate the underlying function of a given stream, and, given the function estimates of two streams, estimate the time delay between them. Knowing the value of the time delay has many applications in astrophysics, and with more precise estimates, more accurate calculations can be made to increase our understanding of the universe we live in.

- strong lensing has delays on the order of hundreds of days, but weak lensing is more like on the order of hours - no longer sufficient to calculate flux for a single day, must do it in a different way, by measuring individual photon arrival times.

In section ?? we discuss the concepts underpinning the project in more detail, with a more in-depth explanation of the issues surrounding the calculation of the time delay and its uses. In section ?? we introduce our method of generating photon streams from Poisson processes. Section ?? shows our approach to estimating the underlying function of a given stream of photons. Our methods of calculating the time delays between multiple photon streams are explained in

section ??. Section ?? gives detailed information on the design and development of the system, including the software and project management aspects. Finally, in section ?? we present experimental data from both simulated and real data and discuss the relative effectiveness of our methods.

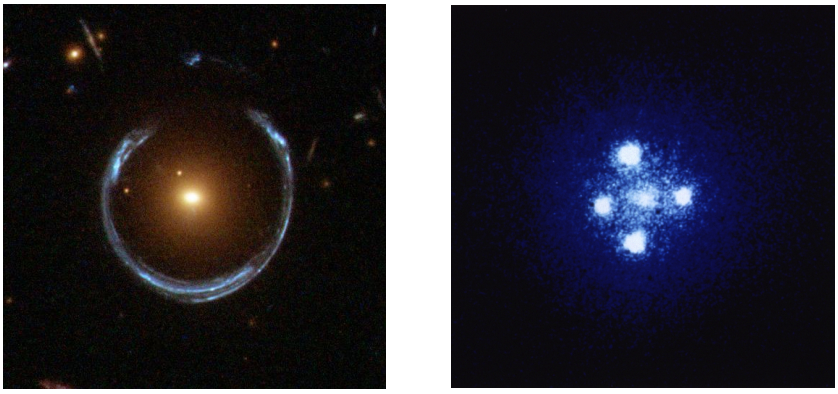
2 Background

2.1 Gravitational Lensing

In an eight-year period starting in 1907 and ending in 1915 with the publication of a paper on field equations of gravitation [6], Albert Einstein wrote many papers developing a new theory of gravitation, his general theory of relativity. This generalisation of special relativity and Newton's law of universal gravitation led to a revolution in the field of physics, and remains one of the most important scientific discoveries to date. The theory describes how spacetime is affected by the presence of matter and radiation, and this idea has many important consequences, but one of the effects in particular is important in the context of this report.

According to the theory, objects with mass, or massive objects, cause spacetime to curve around them. A simple way to visualise this effect is to imagine dropping a ball onto a sheet of cloth which has been pulled taut. The ball will eventually come to a stop in the centre of the cloth, and cause it to sag. Here, the sheet represents spacetime, and the ball represents anything from planets, to stars, or even entire galaxies. Depending on the weight of the ball, the shape of the cloth will be affected to different degrees—a ping pong ball will have hardly any effect at all, but if we drop a bowling ball onto the sheet, the effect will be significant. In a similar way, the amount that spacetime curves around a massive object depends on its mass. An object with high mass will cause a large amount of curvature, whereas a lower mass object will cause less. If a second ball, lighter than the first, is introduced to the system, what happens? With no initial velocity, it will roll in a straight line towards the first ball sitting at the centre of the sheet. This is one way of thinking about gravity and its relationship with spacetime—an object's gravitational attraction is a result of its mass curving spacetime, and the strength of the attraction is proportional to the mass. While objects with no mass, such as photons, cannot be affected by gravity directly, they *are* affected by the curvature of spacetime. This bending of light rays is known as *gravitational lensing*.

The first person to study the effects of gravitational lensing was Orest Chvolson, publishing a short note to *Astronomische Nachrichten* in 1924 [4]. However, the concept was largely unknown until a short calculation by Einstein was published in *Science* in 1936 [7]. Interestingly, Chvolson's note appears directly above a note from Einstein[1], but there appears to be no evidence that Einstein had ever seen it [16]. The first gravitationally lensed object to be identified was the twin quasar SBS 0957+561, in 1979, and since then, over a hundred such objects have been discovered [20, 9]. The effect of gravitational lensing is, as the name suggests,



(a) An Einstein ring (b) Einstein's cross

Figure 1: Two examples of strong lensing effects. a) shows light from a distant blue galaxy being distorted by the central galaxy LRG 3-757 [5]. b) shows four images of a distant quasar being lensed by a foreground galaxy [11].

similar to that of a lens, such as that of a camera. Unlike a camera lens, however, gravitational lenses do not have a focal point, but instead a focal line, resulting in images such as that shown in Figure **??** if the source (the object being lensed), the lensing object (the massive object around which the light is being bent) and the observer lie on a straight line. This effect is relatively rare, however, and in general rather than a ring, multiple images of the source can be observed. In these so called *strong* lensing effects, the distortion is very clearly visible. However, two other classes of lensing exist—*weak lensing* and *microlensing*. The effects of weak lensing cannot easily be observed visually, but statistical techniques can show the distortion produced. Microlensing works on even smaller scales than the other two classes, and can be used to detect planets and stars. It has also been proposed as a method to find objects such as black holes and brown dwarfs, which are otherwise difficult to detect [18].

2.1.1 Importance of the Time Delay

In gravitationally lensed systems, there is a delay between photon streams coming from images of the source due to the bending of light. Light from one source may have had to travel a slightly longer distance than that from the other, and while photons travel extremely fast, over astronomical distances the delay can become quite large.

- strong lensing p86
- Talk generally about the problem of time delay estimation
- refer to physics papers attempting to make estimates of the delay

- talk about time delay estimation in particular, refer to kundic et al, many others
- talk about how better estimates benefit the scientific community
- refer to peter's paper about the efficacy of kernel regression
- better estimators are necessary to increase the accuracy of estimates
- this is an experiment to see whether this method has any use
- build on technique introduced in massey et al

2.2 Poisson Processes

In certain situations, there are many benefits of having good models of the numbers of events that occur in a given period. For example, being able to estimate the number of incoming requests to a server, the number of calls made to emergency services, and the rate of radioactive decay at any given time are all useful in different applications. Poisson processes are *stochastic processes* that can be used to do just that. A stochastic process is a way of representing the evolution of a random value or system over time by using collections of random variables. Most such processes do not evolve in a *deterministic* way. That is, the way they change as time passes is not predictable.

A Poisson process is one such process which counts the number of events and the time at which they occur in a given time interval, and have been used to model all of the above examples [10, 3, 2]. In their basic form, Poisson processes have the following important properties [17]:

1. $N(0) = 0$.
    - $N(t)$ represents the total number of events that occurred up until time t. Thus, if $N(0) = 0$, it follows that the process begins at $t = 0$.
2. The numbers of events occurring in disjoint time intervals are independent.
    - The *independent increment* assumption. This states that $N(t)$, the number of events that occur up to time t is *independent* of the number $N(t + s) - N(t)$, i.e. the number of events in the time interval between t and s. In other words, the number of events that occur in one interval does not have an effect on the number of events in any other time interval.
3. The probability distribution of the number of events that occur in a given interval is dependent only on the length of the interval.

- The *stationary increment* assumption. The implication of this is that the probability distribution of $N(t+s) - N(t)$ is the same for all values of t. That is, the likelihood of a number of events n occurring in the above time interval does not change, regardless of the value of t.

4. No counted occurrences are simultaneous.

    - For all events that occur in the duration of the process, no two events will occur at the same time.

The most important thing about Poisson processes is the *rate parameter*, λ. This value represents the number of events that occur in each time interval. As we are counting events, it is clear that the rate parameter can never go below zero—there cannot be a negative number of occurrences in a given time interval. There are two types of Poisson processes, *homogeneous* and *non-homogeneous*. In a homogeneous Poisson process (HPP), the rate parameter is constant throughout the running of the process. This means that in every interval, the same number of events are likely to occur. In contrast, a non-homogeneous Poisson process (NHPP) has a rate parameter which varies. This means that the rate at which events occur varies during the running time of the process.

2.3 Function Estimation

2.3.1 Linear Regression

Linear regression is a statistical technique used to fit lines or curves to data points in order to find some sort of relationship between them. The number of variables in the data is important. One of the variables is called a *dependent* variable. We want to find the relationship between this variable and the other variables, called *independent* variables. What makes one variable dependent and another independent? Consider the expression $y = f(x)$. If $f(x)$ is some function of the variable x, then we know that the value of y depends on the value of x. This is where the names come from. In this simple example, x is the independent variable, and y is the dependent variable. There can be multiple independent variables.

Linear regression is used in many different fields to find the trend between variables. It is heavily used in economics to make predictions about what happens in many economical situations. Finding trends in data is useful to many people in different ways.

2.3.2 Kernel Density Estimation

This is another method which can be used to estimate functions, but which applies specifically to the probability density function of random variables. This technique uses *kernels* to estimate the function densities. A kernel is a function which has some parameters. To estimate functions, kernels are centred at certain

points along the axis which is being estimated. The spread can be either at uniform intervals, each sample value, etc. Kernels may have a weight assigned to them. Varying the parameters of the kernels results in different properties of the estimate. There are many different kernels that can be used. Different kernels are used in different applications.

- Show some examples of different kernels

3 Simulation of Photon Streams

The first step in building the system was the development of a photon stream simulator. The ability to simulate photon streams means that the system can be tested on many different stream types, so that we are able to determine where its strengths and weaknesses lie. While many simulation tools are very complex, our system does not require simulation of the source objects or the movement of photons, as we are only interested in their arrival time. A source can be represented by some random variable X, which indicates the variability of the source with time. Different types of sources will have different types of characteristic functions—the variation in a quasar will be very different to that of an individual star, for example. A NHPP is an ideal way to represent this type of system. The function $\lambda(t)$ will represent the random variable, and the values output from the process will represent the arrival times of the photons. $\lambda(t)$ describes the variability of the source in time. In other words, it provides a rate parameter at each time t for the duration of the simulation. To be able to simulate a wide variety of functions, it is necessary to have the capacity to generate functions with different characteristics.

3.1 Function Generation

To evaluate the performance of the function and time delay estimators, it will be necessary to test the accuracy of the estimates on different types of functions. To this end, the capability of generating random functions will be very useful. To generate random functions, we make use of Gaussian kernels or just Gaussians?. The generation process involves four simple steps:

1. Pick some value Δt which represents the distance between the mean μ of successive Gaussians.
2. Define some value α, where the standard deviation σ of each Gaussian is determined by $\alpha \cdot \Delta t$.
3. For each Gaussian, choose some weight w_i, from a uniform distribution between -1 and 1.
4. Using some step s, sum all the Gaussians at each point on the x-axis which we get from these s values.

The first step defines how spread out the Gaussians should be in the interval $[t_0, T]$ in which the function is to be generated. If the spread is large, then depending on the standard deviation of the Gaussians there will be many points on the interval where the value of the function is zero. On the other hand, with a low value of Δt, most points on the line should have some non-zero value.

The α parameter determines the standard deviation σ of all the Gaussians used to generate the function. The value of σ is the one that affects the final function the most. Low values will result in each Gaussian covering only a small interval, so if the Gaussians are sufficiently spread out, the variation in the function will be much larger than if higher values of σ are used.

With just the above two steps, the functions generated would be very homogeneous, because each Gaussian has the same weight. With uniform Gaussians, there would be hills at each point where a Gaussian is centred, and very little to speak of in between, and the height of the function would never exceed a certain value. To introduce more variation, a weight w_i is sampled uniformly from ($w_i \sim U(-1, 1)$). Uniform sampling simply means that each value between -1 and 1 has an equal probability of being chosen. To further increase the variation in the functions that can be generated, some multiplier can be used, which scales the values of the weights, meaning that the function will have larger values over the whole interval.

The final step is to calculate the values which will make up the function. Starting at the beginning of the interval t_0, we sum the values of all the Gaussians at points along the line until the end of the interval T is reached. The points that are sampled are defined by $t_i = t_{i-1} + s$, where s is some sample step. The use of smaller sample steps results in a higher resolution. The sum of the Gaussians at time t can be calculated by

$$f(t) = \sum_{g \in G} w_g \cdot e^{-(t-\mu_g)^2/2\sigma_g^2} \tag{1}$$

Where G is the set of Gaussians which make up the function, and w_g, μ_g and σ_g are the weight, mean and standard deviation respectively of the current Gaussian being processed.

- random functions using gaussian sums
- explain how the gaussians are placed, how the standard deviation is calculated, what effect this has on the function shape ($\sigma = \alpha \cdot \Delta t$)
- examples at various alpha and delta t values?
- $w_t \cdot e^{\frac{x-\mu^2}{2\sigma^2}}$

In addition to the random function generation, it may sometimes be useful to generate a function from a known expression, and the system includes this functionality as well, which will be described below.

3.2 Generating Streams from Functions

Once the function has been generated, we can use it to generate values for the random variable X which governs a NHPP. To generate a NHPP, it is necessary to build on the generation of a HPP. It is well known that probability of an event occurring follows an exponential distribution. The rate parameter λ determines how many events occur in a given time interval. Knowing this, we can calculate the time of the next event by sampling from this distribution. Generate a random value $U \sim U(0, 1)$. The time t to the next event is defined by

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

Using this calculation, it is possible to generate a realisation of a HPP for any length of time. This provides a base which can be extended to generate events from NHPPs. To generate events from the NHPP, we use a technique called thinning. The basic concept behind thinning is to generate a large amount of values, and then remove them based on some method. In the case of the NHPP, we generate events with a rate parameter λ, where $\lambda > \lambda(t)$ for $0 <= t <= T$. In other words, the homogeneous lambda value must be larger than the value of the function we are generating from at any point. First, two random values are independently sampled from a uniform distribution between 0 and 1, $U_1, U_2 \sim U(0, 1)$. U_1 is used in (**??**) to find the next event time from the homogeneous process governed by λ. Using the time t generated from that, the value of $\lambda(t)$ is found. Depending on the ratio between $\lambda(t)$ and λ, the event is kept or discarded. When the value of $\lambda(t)$ is close to that of λ, more events are kept because $U_2 \leq \frac{\lambda(t)}{\lambda}$ will be true more of the time. The variation of $\lambda(t)$ in time means that events are generated proportional to the value of lambda.

- Need to generate event times - use Poisson process
- start with homogeneous
- extend homogeneous to non-homogeneous (explain math)
- Issues with the implementation - must have $\lambda > \lambda(t)$ for all $0 <= t <= T$.
- Diagram showing HPP and NHPP

- http://preshing.com/20111007/how-to-generate-random-timings-for-a-poisson-process

3.3 Implementation

The implementation of the random and expression based function generators form the first part of the *generator* sub-system. This part of the system deals with the generation of functions, and the generation of photon streams from these functions.

- gaussians as structs
- gauss vector structs
- functions into math library
- separation of functions so that they can be called externally and internally
- what can be generated
- How expr and gauss version differ

3.3.1 Generating from Expressions

- muparser
- enter expression and define variables in parameter file
- parsed in and calculated automatically

3.3.2 Generating from Random Functions

-

4 Function Estimation

4.1 Baseline Estimation

4.1.1 Ordinary Least Squares

The Ordinary least squares (OLS) estimator forms the core of the baseline estimator. This estimator will form an estimate by minimising the sum of squared residuals. It is important to note the difference between errors and residuals. In statistical terms, an *error* is "The difference between the observed value of an index and its "true" value" [14], and a *residual* is "The difference between the observed value of a response variable and the

value predicted by some model of interest" [8]. The "true" in the definition of error is in inverted commas due to the fact that the true value of the function is unobservable—it is only possible to obtain a statistical sample. The residual, on the other hand, is the difference of the observation from some *estimate* of the function. This first estimator estimates a linear function of the form $y = ax + b$, or a straight line. While this is not directly useful for estimating characteristic functions, it was developed in order to gain a deeper understanding of the ideas behind regression, and in order to construct a simple estimator which could then be extended.

In order to estimate the function, the stream of events must first be converted into a form which is suitable for processing. To do this, we first pick a time interval $(0, T]$, and divide it into N sub-intervals, or *bins*. According to [13], the kth bin I_k is calculated by

$$I_k = \left(\frac{(k-1)T}{N}, \frac{kT}{N} \right], \ 1 \leq k \leq N \tag{3}$$

and the midpoint x_k of each bin is

$$x_k = \left(k - \frac{1}{2} \right) \frac{T}{N}, \ 1 \leq k \leq N \tag{4}$$

Due to the independent increments property of Poisson processes, splitting the interval leaves us with N bins, and according to Massey et al. [13], each is defined by an independent Poisson random variable Y_k with mean

$$\lambda_k = \frac{T}{N}(a + bx_k) \tag{5}$$

T/N is used to normalise the value of λ_k. The value of Y_k in our case is the number of photon arrival times for each bin. In order to perform regression on the data, we need a model of the data. At this stage, we make the assumption that the data is represented by a linear function, so the model is linear. The model is used to connect the random variables and the parameters, and describes how they are related. Our model becomes $Y = \alpha + \beta x + \epsilon$, or $Y_k = \alpha + \beta x_k + \epsilon_k$ [13]. The values α and β are the two regression parameters which we use to estimate the values of a and b of the characteristic function. What is a regression parameter? ϵ represents the Poisson error that is present in the data that we are trying to model. As mentioned before, this technique works by minimising the sum of squared residuals. The value of a residual can be computed by [12]

$$d_k = Y_k - (a + bx_k) \tag{6}$$

However, since we cannot know the real values of a and b, we must instead use the regression parameters α and β. Substituting these into (??) we get

$$d_k = Y_k - (\alpha + \beta x_k) \tag{7}$$

With this, we can calculate residuals for our function estimate. This calculation by itself is not sufficient, though, as summing the residuals necessarily results in a value of zero **FIND A CITATION FOR THIS!**. To get a useful value from the residuals, we square the value of each residual.

$$d_k^2 = (Y_k - [\alpha + \beta x_k])^2 \tag{8}$$

Until now, we have been ignoring ϵ_k, the Poisson noise associated with the random variable. In order to compensate for this, it is necessary to introduce a weight w_k for each interval, initialised to 1 as we are using the OLS technique [13]. Introducing this weight into (??) and summing over all bins, we have

$$\sum_{k=1}^{N} w_k (Y_k - [\alpha + \beta x_k])^2 \tag{9}$$

Which is known as the residual sum of squares (RSS). We want to find the values of α and β for which the RSS is minimised, and so the final expression becomes

$$\min_{\alpha,\beta} \sum_{k=1}^{N} w_k (Y_k - [\alpha + \beta x_k])^2 \tag{10}$$

Now that we know what we are looking for, we define estimators $\hat{\alpha}$ and $\hat{\beta}$, which we will use to estimate values of α and β to find the minimum.

$$\hat{\beta} = \frac{\sum_{k=1}^{N} w_k (x_k - \bar{x})(Y_k - \bar{Y})}{\sum_{k=1}^{N} w_k (x_k - \bar{x})^2} = \frac{\sum_{k=1}^{N} w_k (x_k - \bar{x}) Y_k}{\sum_{k=1}^{N} w_k (x_k - \bar{x})^2} \tag{11}$$

$$\hat{\alpha} = \bar{Y} - \hat{\beta}\bar{x} \tag{12}$$

$$\bar{x} = \frac{1}{N}\sum_{k=1}^{N} w_k x_k \quad \text{and} \quad \bar{Y} = \frac{1}{N}\sum_{k=1}^{N} w_k Y_k \tag{13}$$

$$\hat{a} = \frac{N}{T}\hat{\alpha} \quad \text{and} \quad \hat{b} = \frac{N}{T}\hat{\beta} \tag{14}$$

Impose a constraint on the values of $\hat{a}$ and $\hat{b}$ which states that the rate function must be non-negative throughout the entire interval $[0, T]$ [13]

$$\hat{a} \geq 0 \quad \text{and} \quad \hat{b} \geq -\hat{a}/T \tag{15}$$

There are two cases in which this constraint can be violated; when $a < 0$ or $b < -\hat{a}/T$ [13]. In the first case, we set

$$\hat{a} = 0 \tag{16}$$

$$\hat{b} = \frac{N}{T} \frac{\sum_{k=1}^{N} w_k x_k Y_k}{\sum_{k=1}^{N} w_k x_k^2} \tag{17}$$

and in the second,

$$\hat{a} = -\hat{b}T \tag{18}$$

$$\hat{b} = \frac{N}{T} \frac{\sum_{k=1}^{N} (T - x_k) Y_k}{\sum_{k=1}^{N} w_k (T - x_k)^2} \tag{19}$$

With this set of equations, the structure of the OLS estimator is complete.

- Explain main parameters, the main things that are important to the workings, and exactly how it works. Number of intervals, the total time, see massey paper
- Example of residuals
- Example function estimate on a linear function, use Poisson to generate
- Talk about this being the first step, both to learn about regression and to get a simple estimator working before moving on to more complex ones

4.1.2 Iterative Weighted Least Squares

The iterative weighted least squares (IWLS) builds upon the OLS estimator. As the name suggests, the extension is to include an iterative part. The OLS estimator performs a single estimate of the function and leaves it at that.

The IWLS estimator, on the other hand, repeats the process multiple times, updating its estimates. Perhaps the most important update to the estimator is the use of unequal weights, which change depending on the variances of the random variable which defines the bin which the weight is being applied to. A Poisson random variable has a variance that is equal to its mean—this means that a higher value of λ_k results in a larger variance. To compensate for this, we give higher weights to bins which have lower values of λ, as the variances will be lower. As shown in equation (??), the value of λ is easy to calculate, but the values of a and b must be known. In order to modify weights appropriately, we must be able to obtain estimates of λ, which can be done using [13]

$$\hat{\lambda}_k = \frac{T}{N}(\hat{a} + \hat{b}x_k) \tag{20}$$

The weights can then be updated using

$$\hat{w_k} = \frac{\dfrac{N}{\hat{\lambda}_k}}{\displaystyle\sum_{k-1}^{N}\left(\frac{1}{\hat{\lambda}_k}\right)} \tag{21}$$

which has some desirable properties [13]. Minimum variance estimator among linear functions of the observations Y_k that are unbiased. Each iteration of the estimator updates these estimates of λ and the weight for each bin, and the process is stopped when the change in the estimates becomes negligible, which consistently happens in between two and five iterations [13]. With this estimator, we have something which can improve upon the estimates from OLS with only a small amount of additional calculation. However, for our purposes this is not sufficient. The characteristic function of stellar objects are not linear functions, so we must extend this linear approach to give us some reasonable estimates of functions which are not straight lines.

- Extension of the OLS technique which iterates it multiple times, updating weights
- show weight update equation
- talk about how varying the number of iterations affects it (massey)
- Show it estimating a line
- Explain how line estimation is not useful since lensed photon streams are always some function which is not a line.

4.1.3 Piecewise Iterative Weighted Least Squares

- explain intuition behind the technique. Split the whole interval into some finite number of sub-intervals and estimate the function of each interval in turn using IWLS.
- give reasoning behind moving to this technique. Some parts of functions look like they are pretty much linear - maybe it is a nice way to solve them. mention that this was developed on my own interest in seeing how it worked
- Explain the not-so-good parts - each subsection estimate is disjoint from the next, but the stream must be a continuous function.
- Talk about line extension and the minimum length issue

Initially, we thought that it may be possible to decide whether to extend the line or not based on the difference in slope between the estimate from the previous time interval and the estimate of the next. If the previous estimate was positive, and the next negative, and vice versa, clearly the line should not be continued. The intercept parameter was considered to be much less important. However, this assumption was highly flawed. Due to the nature of poisson processes, it is perfectly possible that although the function has changed significantly after the end of the previous interval, the estimate for the interval that we are trying to extend the line into could return very similar values to that of the previous interval. Because of this, we extend the line when we should not be doing so. There are several potential solutions to this problem. First, rather than forming a new estimate, we extend the line and then check how much the error has increased. If it goes over a certain threshold, then we reject the extension attempt and try again, this time with a shorter extension. Another potential way of improving the piecewise estimation in general would be to require the estimate for the next time period to start from the end point of the last time period. This would prevent the intercept parameter from changing, and would force the estimator to find the best estimate given a specific starting point, rather than giving it free reign to find the estimate which actually minimises the error.

- coding issues
  what to do with the issue of minimum length of intervals? Sometimes not extending the original gives a better estimate of the line than re-estimating the interval extended, or adding the short interval onto the end of the previous one and using its estimate. See data in the $\text{min}_{\text{intervallength}}$ folder in data. The better fitting line is the baseline estimate of that with no minimum, and the other set is the estimate with minimum interval length applied

4.1.4 Baseline

- improvement on the piecewise method by making sure that the function is continuous, i.e. the start of the function at each interval is the end of the one in the previous
- How we calculate the points at which to join the functions - do it at the midpoint on y between the start and end
- show a baseline and piecewise estimate on the same function

4.2 Kernel Density Estimation

- explain how kernels are calculated, and how they are centred
- explain the effect of the standard deviation on the estimate
- needs normalisation to get the proper estimate
- how the normalisation constant is found, using pmf stuff

4.3 Implementation

5 Time Delay Estimation

- basic explanation of what we want to do with this part of the system, referring to the introduction a little?

5.1 Area Method

- use the area of the space between two functions, find the time delay which minimises the value
- show integral formula and then show the simplified discrete formula

5.2 Probability Mass Function Method

- calculate probability mass function at each point on the function and choose the time delay which maximises the value

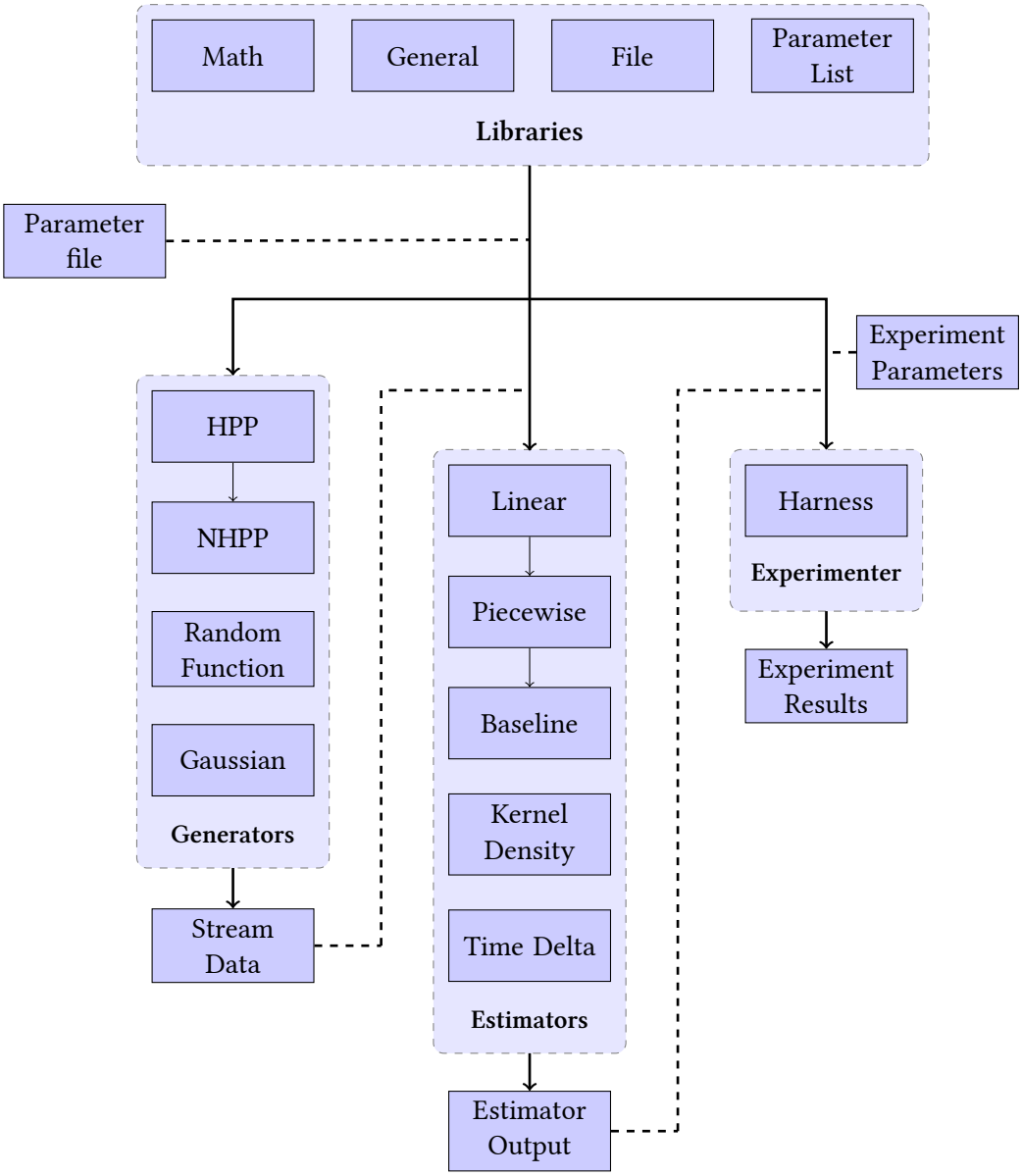


Figure 2: System structure

6 System

6.1 System Structure

6.1.1 Overall Structure

6.1.2 Libraries

- say what the function of each library is and what sort of functions it contains.

6.1.3 Estimators

Maybe this stuff should be in each individual section rather than here? talk about how the system is interlinked in those sections - makes more sense with the flow of things?

- Function estimators
  - talk about each estimator and where it is used, and how they are related.
  - OLS->IWLS->Piecewise->Baseline
  - gaussian + normalisation
- Time delay estimators

6.1.4 Generators

- muparser generator

6.1.5 Interface

6.2 Development

6.2.1 Development Process

- first draft up code in notebook to get down the concept
- write a basic code skeleton and add tests to make sure that it works as intended - particularly for mathematics and the like
- flesh out the code and integrate it with the system
- make code as modular as possible to make it easy to add stuff in later

6.2.2 Development methodologies

- could be merged with previous section, but mention some of the unix rules of thumb, like the rule of least surprise and so on.

6.2.3 Testing

- talk about check, which functions were tested (mostly library functions, hard to test estimators in a reliable way)
- give examples of the tests

6.2.4 Version Control

- branching strategy
- commit frequency
- using issues on github
- storing backups of tags on svn

6.2.5 Project Management

- keep changelog
- show examples of changelogs and commit messages from the same time period
- writing up and planning layout in notebook

7 Evaluation

7.1 experimentation on simulated data

Talk about why there is a window on the effective values.

7.1.1 Sine Functions

- $y = a - b\sin(\alpha t)$
- five values of α: 0.05, 0.1, 0.15, 0.3, 0.6
- 50 pairs of realisations for each value of alpha
- first find optimal parameter setting to estimate the functions by using model selection
- Take the parameters from that and run the time delay estimator on the functions to see what results they give for the time delay.
- Compare the means and standard deviations etc of the gaussian and baseline estimators to find out how good they are by using t-tests.

7.1.2 Random Functions

- Streams in reality are not likely to follow sine curves, so additional experiments on randomly generated functions were also performed.

7.2 experimentation on real-world data

8 Conclusion

8.1 Future Work and Improvements

- Poisson generator - must set lambda to be greater than the value of the function at all points, otherwise it breaks. There are some methods of fixing this
- baseline estimator - better way of joining functions - use the pmf to calculate the best possible point at which to join all points on the estimates. Use a hierarchical method to scan across the points between the points on the y axis and find the best values for all breakpoints

References

[1] SAO/NASA Astrophysics Data System (ADS). *Über eine mögliche Form fiktiver Doppelsterne*. URL: http://articles.adsabs.harvard.edu/full/1924AN....221..329C.

[2] Martin F Arlitt and Carey L Williamson. "Internet web servers: Workload characterization and performance implications". In: *IEEE/ACM Transactions on Networking (ToN)* 5.5 (1997), pp. 631–645.

[3] F Cannizzaro et al. "Results of the measurements carried out in order to verify the validity of the poisson-exponential distribution in radioactive decay events". In: *The International Journal of Applied Radiation and Isotopes* 29.11 (1978), 649–IN1.

[4] Orest Chwolson. "Über eine mögliche Form fiktiver Doppelsterne". In: *Astronomische Nachrichten* 221 (June 1924), p. 329.

[5] NASA Astronomy Picture of the Day. *A Horseshoe Einstein Ring from Hubble*. URL: http://apod.nasa.gov/apod/ap111221.html.

[6] Albert Einstein. "Die Feldgleichungen der Gravitation". In: *Preußische Akademie der Wissenschaften, Sitzungsberichte*. 1915.

[7] Albert Einstein. "Lens-Like Action of a Star by the Deviation of Light in the Gravitational Field". In: *Science* 84 (Dec. 1936), pp. 506–507.

[8] B.S. Everitt and A. Skrondal. *The Cambridge Dictionary of Statistics*. Cambridge University Press, 2010.

[9] CFA-Arizona Space Telescope Lens Survey of Gravitational Lenses. *CASTLeS Survey*. URL: http://www.cfa.harvard.edu/castles/.

[10] Amir Hajjam, J-C Creput, and Abder Koukam. "Approach for optimized and dynamic medical emergency management". In: *Service Systems and Service Management, 2008 International Conference on*. IEEE. 2008, pp. 1–6.

[11] HubbleSite. *First ESA Faint Object Camera Science Images the Gravitational Lens G2237 + 0305*. URL: http://hubblesite.org/newscenter/archive/releases/1990/20/image/a/.

[12] J.F. Kenney. *Mathematics of statistics*. v. 1. D. Van Nostrand company, inc., 1939.

[13] 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.

[14] *OECD Glossary of Statistical Terms*. OECD glossaries. OECD Publishing, 2008.

[15] Worldwide LHC Computing Grid Project. *What is WLCG: Data Processing*. 2011. URL: http://lcg-archive.web.cern.ch/lcg-archive/public/data-processing.htm.

[16] Jürgen Renn and Tilman A Sauer. *Eclipses of the stars*. Max-Planck-Inst. für Wissenschaftsgeschichte, 2000.

[17] Sheldon M Ross. *Simulation*. Academic Press, 1997.

[18] Peter Schneider, Christopher Kochanek, and Joachim Wambsganss. *Gravitational Lensing: Strong, Weak and Micro: Saas-Fee Advanced Course 33*. Vol. 33. Springer, 2006.

[19] Jean-Luc Starck and Fionn Murtagh. *Handbook of Astronomical Data Analysis*. 2002.

[20] D Walsh, RF Carswell, and RJ Weymann. "0957+ 561 A, B- Twin quasistellar objects or gravitational lens?" In: *Nature* 279.5712 (1979), pp. 381–384.

Appendices

Appendix A Installation

A.1 MuParser

download package
run `./configure --prefix=/usr`, followed by `make && make install` (may require sudo) this installs muparser so that headers can be found in `/usr/include`

```
sudo apt-get install libgsl0-dev check
```

Appendix B Usage

B.1 Creating functions for experimentation

Generate 10 random functions using gaussians, and output the transforms as well so they can be plotted

```
./launcher -g ../data/params.txt -r -t 2 -c 10
```

Generate two streams from each of these generated functions using the gaussian generator

```
./launcher -g ../data/params.txt -f rand -n 2 -c 10
```

Generate stuttered streams from files in this directory, so that you can perform model fitting to find the best parameters to use on the generated set.

```
./launcher -x ../data/exp_params.txt -p ../data/params.txt -c 10 -n 2 -s -i .
```