Timing based source separation

A DISSERTATION SUBMITTED TO THE GRADUATE DIVISION OF THE
UNIVERSITY OF HAWAIʻI AT MĀNOA IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY
IN
ELECTRICAL ENGINEERING

December 2019

By

Jeremy Young

Dissertation Committee
Anders Høst-Madsen, Chairperson
Anthony Kuh
James Yee
June Zhang
Eva-Marie Nosal

Contact: jbwmyoung[at]gmail[dot]com

Acknowledgments

First, I would like to thank my advisor Prof. Anders Høst-Madsen for his support, guidance, and patience through my many years of doctoral studies. Your continuing motivation was instrumental into me finishing this degree. I also want to extend my gratitude to Prof. Eva-Marie Nosal who provided the opportunity to work on her project for passive acoustic monitoring of marine mammals. Thank you for opening a new part of the world to me. Your mentorship helped me to grow as a researcher and a person. And your student, Brendan, his companionship during the early years of my studies was invaluable.

Thank you also to Prof. Anthony Kuh, Prof. James Yee, and Prof. June Zhang, as well for taking the time out of their busy schedules to serve on my dissertation committee and provide feedback to help me improve my work. I also wish to express my appreciation my graduate chair, Prof. Gurdal Arslan, and the staff at the Graduate Division for allowing me to finish my degree on an extended timeline.

It was a pleasure to work with Yutaro from Tokyo University of Agriculture and Technology, and Tomohiro from Tokyo National College of Technology, who while visiting UH over a summer helped in my research. Thank you also to Andy, Philip, Christian, and Jonathan who also participated in this project as part of their EE496 work.

Uzair, though you often came to me seeking advice, I learned a lot more from you. Your constant drive as you strove towards your goals inspired me. To all my lab mates in HH490: Sharif, Kareem, Arif, Seyeed, Navid, Trevor, Cirl, Minqi, and now Meenakshi, though we all worked on different projects I could not have done this without your friendship, encouragement, and support. Additionally, I want to thank all of the faculty members and EE office staff of past and present for all the help and assistance throughout my time here. I am eternally grateful for all opportunities and experiences the department gave me.

Finally, I would be remiss if I did not thank all of my family and friends who supported me not only through graduate school, but for my entire life. Thank you for always believing in me.

This research was supported in part by NSF grant 1017775 and in part by ONR grant N000141612598 and by Shenzhen Peacock Plan under Grant No. KQTD2015033114415450.

Abstract

The motivation of this work is to use statistical signal processing to help to separate mixtures of marine mammal vocalizations, in particular sperm whale click trains. It is observed that clicks from a single whale are spaced at regular intervals which we exploit to do the separation.

To begin, we consider an idealized problem: each source $k \in \{1, 2, \ldots, K\}$ emits at impulses at iid intervals according to some distribution $\mathcal{T}_k$. That is, the impulse times for each source constitute a renewal process. The $\mathcal{T}_k$ induce a likelihood function for impulse times given some assignment. We present an algorithm inspired by the Viterbi algorithm to give assignments that maximizes the likelihood. Additionally, we provide more computationally feasible approximations of this algorithm. Under the assumption that $\mathcal{T}_k$ are Gaussians truncated at 0, we develop a lower bound on the classification error rate. We verify the bound through simulations and show that $c_v = \sigma/\mu$ is directly proportional to the error rate. That is, if known, c_v can be used to tell whether or not separation based on timing only is viable.

Additionally, we provide methods to estimate the parameters of $\mathcal{T}_k$ using alternating maximization (AM) and the expectation maximization Viterbi algorithm (EMV). The solution space is not convex, so the outcome of AM or EMV are sensitive to their initialization point. In this case, it means we need to provide timing parameters that are at least in the neighborhood of the actual parameters for $\mathcal{T}_k$. We tested a number of methods but found that with $\delta\tau$-histogram based methods we were able to approach the same classification error rates when the actual parameters were given. Armed with classification and parameter estimation for any given number of sources, we were then able to use the minimum description length principle to determine the actual number of sources. We used simulated data to verify that we can generally achieve low source number estimation error and low classification error rates.

Next we attempted to use timing to help improve detection methods for impulses. Using a Bayesian framework we came up with an objective function to maximize that consisted of the previous timing likelihood function and a likelihood function based on the shapes of impulses, but it was impractical to actually optimize to determine the detection times. We explored some feasibility approximations, but we still needed to assume too much to get reasonable results. So detection using timing was abandoned. However, the concepts were used for classification using timing information and impulse shape information. This is one way in which we begin to move away from the idealized problem and towards the actual problem of separating sperm whale click trains. The shapes or features of clicks have long been used for classification.

A sperm whale's interclick interval (ICI) does not have a stationary $\mathcal{T}_k$, but it can be approximated as stationary over short periods of times. Over these short periods, the methods developed for the idealized problem can be applied. To determine these short periods, we explored the use of the PRI map which is to the $\delta\tau$-histogram as spectrograms are to Fourier transforms. We propose a parameter estimation method for the map values and then segment by grouping adjacent times with similar parameters together. We tested this on a recording of sperm whale clicks and the initial results are promising. A pressing issue is the handling of missing clicks: a whale may skip one or many clicks for various reasons or the detection method might miss certain clicks. We simulated removal of clicks and compared our method with a sequential search method designed to handle missing clicks and found that we met or exceeded its performance without any modifications. However, there is still room for improvement. In total, we have provided a basis for the practical application, but it needs to be expanded upon.

Contents

Nomenclature

Abbreviations

A#	shorthand for Algorithm #; used to refer to numbered algoritghms
AM	Alternating Maximization optimization method
EM	Expectation Maximization algorithm
EMV	Expectation Maximization Viterbi algorithm (a Viterbi inspired approximation of the EM algorithm)
i.f.f.	if and only if
ICI	Inter-Click-Interval
iid	Independent and Identically Distributed
LHS	Left Hand Side of an equation
MDL	Minimum Description Length
MI	Maximum number of Iterations in reference to unknown parameter algorithms
ML	Maximum Likelihood (estimation/estimate)
PAM	Passive Acoustic Monitoring
PRI	Pulse Reptition Interval
RHS	Right Hand Side of an equation
RMS	Root Mean Square (of a signal)
SDT	Single Detection Threshold
SNR	Signal to Noise Ratio
SS	Sequential Search algorithm
STFT	Short Time Fourier Transform
STO	Source TimeOut
TDOA	Time Difference of Arrival
TK	Taeger-Kaiser measure

TOA	Time Of Arrival

Symbols

$(d_1(\mathcal{P}_i), d_2(\mathcal{P}_i), \ldots, d_K(\mathcal{P}_i))$	the K-tuple that defines the tail of a path $\mathcal{P}_i$
$\bar{P}_e$	the average error rate over a range of parameter sets
$\boldsymbol{\Theta}$	the collection of parameters $\{\theta_1, \theta_2, \ldots, \theta_K\}$ for the timing distributions $\mathcal{T}_k$
$\mathcal{M}$	a model for the data including the number of sources, distribution type, parameters, and assigment scheme
$\mathcal{Q}$	the objective function for the EM algorithm (not to be confused with Q)
ϵ or ε	an arbitrarily small number
$\mathcal{N}_0(\mu, \sigma^2)$	a Gaussian distribution with mean μ and variance σ truncated at 0
$\mathcal{P}_i$	a set of impulse decisions/assigments up to the i-th impulse (inclusive)
$\mathcal{P}_{i,k}$	a set of impulse decisions/assigments up to the i-th impulse with the i-th impulse assigned to the k-th source
$\mathcal{T}_k(t)$	the distribution for interimpulse spacing, t, on the k-th source
μ	the mean of some distribution
σ^2	the variance of some distribution
τ_i	the i-th impulse time
$\tau_{k,i}$	the time of the i-th impulse from the k-th source
c_v	the cofficient of variation
$c_{v,i}$	the cofficient of variation for the i-th source
D_i	the source of the i-th impulse
$d_k(\mathcal{P}_i)$	the number of impulses sincethe last impulse was to source k in path $\mathcal{P}_i$
K	the total number of sources
L	log-likelihood
M	the total number of impulses in the signal given
P_e	the total assignment error probability
P_e^0	the therotical lower bound for the error rate P_e
$P_{e,k}$	the assignment error probability for the k-th source
$P_{e,k}^0$	the theoretical lower bound for the error rate $P_{e,k}$
$r_k(\mathcal{P}_i)$	the time/spacing sincethe last impulse was to source k in path $\mathcal{P}_i$

$S_k(i)$	the index corresponding to the i-th impulse assigned to source k
T_k	the timeout for source k; spacings larger than this value will have a small likelihood

Mathematical Operators

$[\cdot]$	indicates a sequence (i.e. order of the elements matters), vector, or matrix
$[a,b]$	indicates the range of values $x : a \leq x \leq b$; if a parentheses is used instead of a square bracket, the range does not include the end point(s)
$[a,b]^c$	indicates the range in the c-th dimension: $[a,b] \times [a,b] \times \cdots \times [a,b]$
$\cdot^*$	the complex conjugate of $\cdot$
$\cdot^T$	indicates the transpose of matrix of vector $\cdot$
$\hat{\cdot}$	indicates an estimate for $\cdot$
$\in$	argrument to the left is in or an element of the right argument
$\mathbb{R}$	the set of real numbers
$\mathbf{A}$	a boldfaced varaible indicates a collection/vector of the corresponding non-boldfaced (sometimes lowercase) variable
$\propto$	proportional to
$\Pr \cdot$	probablility of the event $\cdot$
$\lvert \cdot \rvert$	magnitdue of $\cdot$
$\{\cdot\}$	indicates a set (i.e. order of the elements is arbitrary) of items between the braces
$I(\cdot)$	the indicator function evaluates to 1 if $\cdot$ is true and 0 otherwise
$O(\cdot)$	order of the function, typically an upper bound for the number of operations in an algorithm

1 Introduction

Sounds are everywhere, from crickets chirping to airplanes flying overhead; if we listen to them carefully, they can tell us a lot about the world we live in. For example, the vocalizations of marine mammals are an invaluable resource for those who wish to study them. Most marine mammals spend their lives far out of human sight, but not out of the reach of our ears/hydrophones. Underwater, sound travels quite far, and depending on the conditions, whale songs can be heard all the way on the other side of the world [1]. This makes marine mammals ideal candidates for passive acoustic monitoring (PAM). In PAM, fixed or mobile hydrophones (e.g. anchored to the seafloor or towed behind a boat) are used to record any sounds underwater over some duration of time (usually long periods). Though sometimes PAM of marine mammals is accompanied by visual observations, generally, what is being recorded is unknown [2]. Thus the following is needed:

- Automatic detection: The recordings are usually long (possibly months) and mostly noise; it is infeasible to listen to everything and manually mark the areas of interest (e.g. where there are vocalizations).
- Automatic classification: When we detect something, what is it? A ship? Whales? Wind or waves? Depending on what it is, we can process it differently.
- Source separation: If there are multiple animals vocalizing at the same time, who said what? (see Figure 1.0.1)
  - Density estimation: If there are multiple animals, how many of them are there?
  - Localization: Where are the animals in relation to each other/the hydrophone?

Though each aspect is important, the focus of this paper is on source separation. There are many different types of vocalizations marine mammals make, but we focus our efforts on separating a particular type– *clicks*. Clicks are quasi-impulsive signals that are produced by odontocetes; they often occur in sequences called *click trains* during which clicks are regularly spaced in time [3, 4]. That is, the time between successive clicks from a single animal, called the *inter-click-interval* (ICI) among biologists, *pulse repetition interval* (PRI) among the ELINT community, or *interimpulse spacing* here, is randomly distributed about some mean. Animals can and do vary their mean

impulse spacing over time, but a constant-mean can be a good approximation over short time periods. We will exploit this fact to classify each click in a mixture[1].

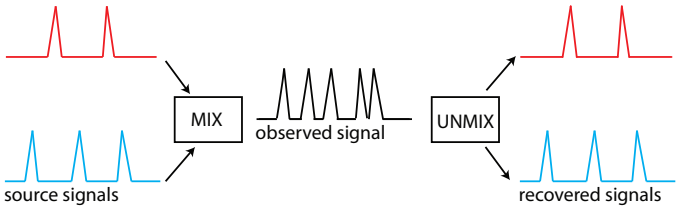


Figure 1.0.1: This is an illustration of the problem considered in this paper for two arbitrary sources. The original signals are mixed when recorded by the sensor, and the goal is to recover the original input signals from the mixture using timing.

Specifically, we consider the following problem. A source $k \in \{1, \ldots, K\}$ generates events at times $\tau_{k,i} \in [0, T]$, where subscript k, i denotes the i-th event from the k-th source. The events could be transmission of packets, or emission of an impulsive signal. An observer is given the total set of event times $\{\tau_{k,i}\}$ without any labels. Can the observer determine which event time belongs to which source solely from the timing information?

Though our main application is for marine mammals, this type of problem occurs in many other applications, for example:

- **Computer networks**: Can the timing of packets reveal who transmitted it?
- **Radar and sonar**: Can the sources of radar pulses be identified only using each pulse's time of arrival (TOA)? (this is called *deinterleaving* [5, 6, 7, 8, 9])
- **Heart beats**: In a mixture of heartbeats, e.g. in fetal ECG [10], is it possible to say which heartbeat belongs to the mother and which to each fetus (with multiple pregnancy) based solely on timing?
- **Nervous system**: In recordings of neural spike trains, can spike trains from different sources be separated [11]?

In many of these applications, there are secondary characteristics that would allow separations of events/impulses. However, we believe it is of value to understand the timing separation problem by itself as a fundamental problem. This is a classical approach: to extract and understand a simple mathematical problem from a practical problem (e.g. information theory [12]). Furthermore, there might be situations where we just have the timing information, for example due to privacy/security. There might also be situations where it is unknown if a source has secondary characteristics, in which case separating based on timing might make it easier to analyze common secondary characteristics.

Several approaches appear in prior literature to separate marine mammal click trains from recordings on single sensors. For example, [13] used click cross-correlation to establish relationships between clicks and to separate click trains. The paper [14] separated sperm whale click trains based on correlation between clicks, click structure detail (inter-pulse-intervals), and frequency characteristics (but ultimately relied on time delay estimates between two hydrophones to separate click trains). [15] used a cluster analysis approach on features extracted from detected clicks; [16] sequentially matched the frequency spectra of successive clicks; and [17] separated click trains based on spectral dissimilarity. These methods rely on the slowly-varying nature of click pulse shapes within a click train. Only a few click train separation methods have made use of another robust and important piece of information: click timing based on the slowly-varying nature of ICI within a click train. [18] tracked slow variations in click amplitude and ICI to separate two click trains. Baggenstoss' algorithms ([19, 20, 21]) define statistical measures

[1] To clarify, "classify each click in a mixture" means assign each click to its source

of click similarity based on features, including inter-click-interval, extracted from click pairs and that maximize a global measure of similarity between associated clicks. As does our paper, [22] analyzed click trains based on timing *only*. It used the PRI transform of [23] to estimate ICIs and extended this to time varying ICIs, but it did not separate (classify) each pulse.

Deinterleaving of radar signal has also been considered before, e.g. [6, 7, 8, 9]. Radar signals usually are either periodic, staggered, or jittered. In jittered PRI radar, a random time is added to each PRI by the transmitter, which makes the signal a renewal process (see Section 2.1 below). Deinterleaving radar signals (in the sense of classifying each impulse) has been considered before in [8, 9], but as far as we know, specifically deinterleaving jittered PRI radar as a renewal process has not been considered. [8] considered noisy time of arrival (TOA) and [9] used a random walk model for PRI, while [23] specifically considered jittered PRI, but only to estimate the PRIs.

The main contribution of this dissertation is to develop algorithms specifically for separation/deinterleaving of renewal processes, which we believe is a new problem.

In Chapter 2 we present solutions for separation based on interimpulse spacing only for an ideal system model. Section 2.1 describes the initial problem which is then extended to account for unknown parameters in Section 2.3 and an unknown number of sources in Section 2.4. The problem of parameter estimation/initialization itself is interesting, so Chapter 3 is dedicated to it. Then in Chapter 4 we explore the use of timing to detect (and classify) impulses. Chapter 5 goes over the basics of producing labeled datasets from actual recordings of sperm whales and some associated analysis. This analysis motivates the methods in Chapter 6 where we try to modify our previous solutions to be more practical for actual marine mammal click trains. Labeled datasets are used for testing and verification. Finally, we conclude in Chapter 7.

Various parts of and versions of this work have been disseminated in [24, 25, 26], and [27]. More specifically, [24, 25] were early versions of Sections 2.1, 2.2, 2.2.2, and 2.3.1. In [26], more current versions of the same sections were presented in addition to Sections 2.3.2, 2.4, 3.1, and 6.3. Finally, the contents of [27] are contained in Sections 2.1, 2.2, 2.5, 2.3.3, 3.2, and 6.2.

2

Ideal timing-only based separation

The general model for source separations models is that there are N source signals

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) & x_2(t) & \cdots & x_N(t) \end{bmatrix}^T \in \mathbb{R}^N$$

that we want to recover from the P mixed signals

$$\mathbf{s}(t) = \begin{bmatrix} s_1(t) & s_2(t) & \cdots & s_P(t) \end{bmatrix}^T \in \mathbb{R}^P$$

defined by

$$\mathbf{s}(t) = \mathcal{A}(\mathbf{x}(t)),$$

where $\mathcal{A} : \mathbb{R}^N \mapsto \mathbb{R}^P$ is usually unknown. Typical source separation methods require that there at least as many mixtures as there are sources (i.e. $P \geq N$) in order for the recovery to be possible [28], but here we consider $P = 1$ for $N \geq 1$. Setting up an array of sensors may be physically and/or cost prohibitive, so it is beneficial if the problem can be solved using a single receiver. Additionally, the results could be used to improve the more general multiple sensor case.

2.1 System model

We consider a system model with K sources that each generate events or emit impulse-like signals. Each source k transmits a sequence of impulses at times $\tau_{k,i}$. We assume that the support of the impulse response is much shorter than the pulse spacing, so there is no overlap. To be precise: We assume that the probability of overlap is low enough that it can be ignored. Without loss of generality, we assume that the first impulse time occurs at $t = 0$.

We want to decide which pulse belongs to which source. Our aim is to classify based only on the (unclassified) impulse times $\{\tau_i\}$. We assume the impulses have been found by some detection algorithm with low error probability.

To be able to classify the $\{\tau_i\}$ based only on timing information, we assume the following:

- The interimpulse spacing $\tau_{k,i} - \tau_{k,i-1} > 0$ is distributed according to some parametric prior distribution $\mathcal{T}_k(t)$ whose parameters may be known or unknown.

- The interimpulse spacings $\tau_{k,i} - \tau_{k,i-1}$ and $\tau_{l,j} - \tau_{l,j-1}$ are independent for $k \neq l$ or $i \neq j$.

With these assumptions, the set of event times for a single source constitutes a renewal or counting process [29, Section 8.3]; the most well-know example is the Poisson process. Renewal processes are widely used to model many processes, which is one reason to use this model. In particular, it is probably a good model for biological signals. At a purely speculative level, biological systems likely do not have an internal *metronome* to which they tie event (e.g. heartbeat, click) generation; rather they have a *timer* to count down to the next event. To generate nearly periodic signals, the timing distribution can then be made very narrow around a mean. The timing distribution might not be independent, but the first order approximation is usually to assume independence, which results in a renewal process. While a good model for click trains is not known, heartbeat is often modeled in this way [30].

For many applications we might not know the *type* of prior distribution $\mathcal{T}_k(t)$. Without further knowledge, a truncated Gaussian seems a reasonable assumption. One can argue for the truncated Gaussian distribution in terms of central limit theory or maximum entropy: given mean μ and variance σ^2, the positive distribution with largest entropy is exactly a truncated Gaussian distribution [31]. It is necessary to truncate the distribution at 0 to enforce the constraint of non-negative impulse spacings. We denote a Gaussian distribution truncated at 0 with mean μ and variance σ^2 by $\mathcal{N}_0(\mu, \sigma^2)$; the probability density function is $f(x) = \frac{1}{\Phi\left(\frac{\mu}{\sigma}\right)\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ for $x > 0$, where Φ is the normal cumulative distribution function [29]. In many cases we consider, $\mu \gg 0$ and $\mu \gg \sigma$. Therefore the correction $\Phi\left(\frac{\mu}{\sigma}\right)$ due to truncation is close to one and can be ignored in many calculations. (See Appendix B.1)

We assess the performance of our algorithms by the error probability, that is the probability that an impulse is wrongly classified. Let $D_i \in \{1, \dots K\}$ denote the source that originated the impulse at time τ_i, and $\hat{D}_i$ an estimate. We then define the error probability for source k as

$$P_{e,k} = \lim_{T\to\infty} P(\hat{D}_j \neq D_j | D_j = k)$$

and the total error probability as

$$P_e = \lim_{T\to\infty} P(\hat{D}_j \neq D_j). \tag{2.1.1}$$

Note, P_e and $P_{e,k}$ are directly related by

$$P_e = \sum_k P_{e,k}\rho_k \tag{2.1.2}$$

where ρ_k is the likelihood of source k. That is

$$\rho_k = \lim_{T\to\infty} \frac{\sum_j I(D_j = k)}{\sum_{n=1}^{K}\sum_i I(D_i = n)}.$$

Note, (2.1.2) is simply a consequence of the Law of Total Probability

$$P(A) = \sum_k P(A|B_k)P(B_k).$$

In this case, $A \equiv \hat{D}_j \neq D_j$ and $B_k \equiv D_j = k$. The general law of total probability does not have limits, but it is trivial to add

$$\lim_{T\to\infty} P(A) = \lim_{T\to\infty} \sum_k P(A|B_k)P(B_k).$$

2.2 Timing-based separation with known parameters

In the basic approach, we assume that the number of sources and prior distributions are completely known. The problem is therefore to classify the impulses based on the impulse times $\{\tau_i\}$ only using the known prior distributions $\mathcal{T}_k(t)$. We consider the maximum likelihood (ML) solution[1]

$$\max_{\hat{\mathbf{D}}} L; \quad L(\hat{\mathbf{D}}) = \sum_{k=1}^{K} \sum_{i} \log \mathcal{T}_k(\tau_{S_k(i)} - \tau_{S_k(i-1)}) \tag{2.2.1}$$

with respect to $S_k(i)$. Here $S_k(i)$ denotes the index corresponding to the i-th impulse *assigned* to the k-th source, and is a function of $\hat{\mathbf{D}} = [D_1, D_2, \ldots]$. For example, if for two sources $\hat{\mathbf{D}} = [1,1,2,1,2,2]$ then $\mathbf{S}_1 = [1,2,4]$ and $\mathbf{S}_2 = [3,5,6]$.

Brute force maximization of (2.2.1) has complexity $O(K^M)$, where M is the total number of impulses in the received signal; adopting ideas from the Viterbi algorithm [33, 34], algorithms with lower complexity can be found. In order to develop these algorithms, we first describe the Viterbi algorithm in a way that fits the current context. Suppose that we have to make decisions between K options at times $1,2,3,\ldots$, and that process is Markov. A path to the the i-th time is a path or set of decisions $\mathcal{P}_i = \{\hat{D}_j; j = 1 \ldots i\}$. Let $\mathcal{P}_{i,k}$ denote the set of paths leading to $\hat{D}_i = k$. With respect to decisions at times $i+1, i+2, \ldots$ all of these paths are equivalent, that is, the optimum decisions at times $i+1, i+2, \ldots$ does not depend on which path led to $\hat{D}_i = k$ because of the Markov property. It is therefore sufficient to only memorize the *optimum* path leading to $\hat{D}_i = k$ for each k. At each stage there are K optimum paths. For the optimum path leading to $\hat{D}_i = k$ we now consider the K possible extensions to time $i+1$, and we do so for all $k \le K$. We then again choose the K optimum paths for each of $\hat{D}_{i+1} = k$. The complexity is therefore $O(MK)$.

For timing based separations, the issue is that while each source's timing process is assumed to be Markov (i.e., a renewal process), this is in general not true for the mixture. Decisions at times $i+1, i+2, \ldots$ do not only depend on decisions at time i. However, it is still possible to discard many paths, as outlined in the following. For a path $\mathcal{P}_i$ let $d_k(\mathcal{P}_i)$ be number of pulses since the last pulse assigned to source k. For example, if $\mathcal{P}_6 = \hat{\mathbf{D}} = [1,1,2,1,2,2]$ we would have $d_1(\mathcal{P}_6) = 2$, $d_2(\mathcal{P}_6) = 0$, and $d_3(\mathcal{P}_6) = \infty$. A key observation is that paths with the same K-tuple $(d_1(\mathcal{P}_i), d_2(\mathcal{P}_i), \ldots, d_K(\mathcal{P}_i))$ are equivalent with respect to future decisions, as in the Viterbi case. This is easiest seen through an example. Let's assume $\mathcal{P}_{100} = [\ldots, 3,1,1,2,1,2,2]$, then $(d_1(\mathcal{P}_{100}), d_2(\mathcal{P}_{100}), d_3(\mathcal{P}_{100})) = (2,0,6)$. When impulse 101 is assigned to each of the three sources, one of the following terms is added to the log-likelihood

$$\begin{aligned} \log \mathcal{T}_1(\tau_{101} - \tau_{98}) &\quad \text{if } D_{101} = 1 \\ \log \mathcal{T}_2(\tau_{101} - \tau_{100}) &\quad \text{if } D_{101} = 2 \\ \log \mathcal{T}_3(\tau_{101} - \tau_{94}) &\quad \text{if } D_{101} = 3 \end{aligned}$$

Now consider $\mathcal{P}_{100} = [\ldots, 3,2,2,1,1,2,2]$, the K-tuple is still $(2,0,6)$, and terms potentially added are still the same

$$\begin{aligned} \log \mathcal{T}_1(\tau_{101} - \tau_{98}) &\quad \text{if } D_{101} = 1 \\ \log \mathcal{T}_2(\tau_{101} - \tau_{100}) &\quad \text{if } D_{101} = 2 \\ \log \mathcal{T}_3(\tau_{101} - \tau_{94}) &\quad \text{if } D_{101} = 3 \end{aligned}$$

With regards to the assignment of the 101st impulse these two paths are identical. On the other hand if $\mathcal{P}_{100} =$

[1] This is not directly an optimum solution to (2.1.1); but in general ML is considered near-optimum ([32, 33])

$[\ldots,3,2,1,1,2,1,2,2]$, the K-tuple is $(2,0,7)$], and we get the following modified terms

$$\begin{aligned} \log \mathcal{T}_1(\tau_{101}-\tau_{98}) &\quad \text{if } D_{101}=1 \\ \log \mathcal{T}_2(\tau_{101}-\tau_{100}) &\quad \text{if } D_{101}=2 \\ \log \mathcal{T}_3(\tau_{101}-\tau_{93}) &\quad \text{if } D_{101}=3 \end{aligned}$$

It is clear that these terms do not depend on what happened in the part of the path before the 3, indicated by $(\ldots)$. Thus, the likelihood for the decision at time $i+1$ only depends on the tail of the path, and this is exactly characterized by $(d_1(\mathcal{P}_i), d_2(\mathcal{P}_i), \ldots, d_K(\mathcal{P}_i))$. Therefore, among paths with the same K-tuple $(d_1(\mathcal{P}_i), d_2(\mathcal{P}_i), \ldots, d_K(\mathcal{P}_i))$ we only need to memorize the *optimum* one. By definition, $d_k(\mathcal{P}_i)$ can take on $i+1$ values (i.e. $0,1,2,\ldots,i-1,\infty$), and $d_k(\mathcal{P}_i)$ must be different for different values of k. Therefore, at a time i there are $(i+1)(i)(i-1)\cdots(i+1-(K-1)) \sim i^K$ unique paths. The overall complexity therefore is $O(M^{K+1})$, which is feasible to implement for moderate values of M and K. This ML algorithm is outlined in Algorithm 2.1; we will refer to it as A2.1.

For large M the complexity can still be prohibitive. We therefore consider a number of simplifications of the algorithm. First we notice that many of the paths considered for the optimum solution have very low likelihood of emerging as the final optimum solution. To see this, for a path $\mathcal{P}_i$ in addition to $d_k(\mathcal{P}_i)$ define $r_k(\mathcal{P}_i)$ as the actual time since the last pulse assigned to source k. Let T_k be the time so that

$$\int_{T_K}^{\infty} \mathcal{T}_k(t)dt < \epsilon$$

for some choice of a small ϵ (see Theorem E.1). If $r_k(\mathcal{P}_i) > T_k$ this means that if we extend path $\mathcal{P}_i$ with $\hat{D}_j = k$ for $j > i$, a term of at least $\log\epsilon$ will be added to the likelihood. If ϵ is small, this term is a large negative number which makes any extensions of this path unlikely to come out as the maximum likelihood solution; we call this a *source timeout* (STO). Thus, such paths can be eliminated from the the set of paths to keep with very little loss compared to the optimum solution – if ϵ is chosen small. This STO algorithm is outlined in Algorithm 2.2; we will refer to it as A2.2.

The complexity, that is, the number of paths to memorize, of the simplified algorithm is stochastic. Suppose that at time i there are $N(T_k)$ impulses in the interval $(\tau_i - T_k, \tau_i)$. Then the algorithm only considers paths with $d_k(\mathcal{P}_i) \le N(T_k)$. The complexity therefore is $N(T_1)N(T_2)\cdots N(T_K)$, which is independent of M. The exact expected complexity is hard to calculate (and probably not that important), but we can do a "back of the envelope" estimate as follows. Suppose that the average impulse spacing is ρ – this is a generalization of all the distributions $\mathcal{T}_k$. The number of impulses from source k during a time period of T_k then is approximately $\frac{T_k}{\rho}$, so the approximate total number of paths to keep is $\frac{T_1}{\rho}\cdot\frac{T_2}{\rho}\cdots\frac{T_K}{\rho}$. Assume further that $T_k \approx T$ for all k. Then the approximate complexity is $O\left(M\left(\frac{T}{\rho}\right)^K\right)$, which for large M is much less than the $O(M^{K+1})$ of the optimum algorithm.

A further simplified algorithm is to keep a fixed number (or maximum number) of paths, call this number a. At each step, we maintain the paths with the largest likelihood. However, we still want to avoid any errors caused by source timeouts, so we still eliminate paths with $r_k(\mathcal{P}_i) > T_k$. This simplified STO algorithm is outlined in Algorithm 2.3; we will refer to it as A2.3.

Alternatively, instead of eliminating paths, we restart the algorithm whenever a source timeout is detected in the most-likely path. Another way to think of this algorithm is that we are exploring the solution space incrementally. When a source timeout error is encountered, then we have converged towards a poor solution. To get out of this mode, we forget where we came from and restart the exploration. The argument for only considering the most-likely path rests on the premise that if a source timeout occurs in a lesser path, it would probably be eliminated in further iterations. This STO restart algorithm is outlined in Algorithm 2.4; we will refer to it as A2.4. Since we only maintain a fixed number of paths, A2.3 and A2.4 both have complexity $O(aM)$. However, in terms of total

runtime, A2.4 is generally faster than A2.3 because it does not have to check for timeouts in every path or manage path deletion. After some experimentation, we found that $a = K^3$ works well for both versions (see Section 2.2.2).

Algorithm 2.1 (A2.1 or A1 in plots) Exact maximum likelihood solution for separation based solely on timing with known distributions

Given K sources and $\mathcal{T}_k$ with observations $\tau_i, i = 1, \ldots, M$

1. Initialize the assignment of τ_1 such that in the first K paths τ_1 is labeled $1, \ldots, K$. Set $i = 1$.
2. For each path consider the extensions where τ_{i+1} is assigned to source $k = 1, \ldots, K$
   (a) Update $(d_1(\mathcal{P}_i), \ldots, d_K(\mathcal{P}_i))$ for each extended path as follows: if the extension is $\hat{D}_{i+1} = k$ then
   $$d_k(\mathcal{P}_i) = 0$$
   $$d_j(\mathcal{P}_i) = d_j(\mathcal{P}_i) + 1$$
   (b) Update the likelihood of each path.
3. Put the paths with same $(d_1(\mathcal{P}_i), \ldots, d_K(\mathcal{P}_i))$ into a group.
4. In each group, eliminate all paths except the one with largest likelihood.
5. If $i = M$, done. Choose the most likely path as the assignment. Otherwise, increment i and go to step 2.

Algorithm 2.2 (A2.2 or A2 in plots) Maximum likelihood algorithm for separation based solely on timing with known distributions with STO pruning

Given K sources and $\mathcal{T}_k$ with observations $\tau_i, i = 1, \ldots, M$

1. Initialize the assignment of τ_1 such that in the first K paths τ_1 is labeled $1, \ldots, K$. Set $i = 1$.
2. For each path consider the extensions where τ_{i+1} is assigned to source $k = 1, \ldots, K$
   (a) Update $(d_1(\mathcal{P}_i), \ldots, d_K(\mathcal{P}_i))$ and $(r_1(\mathcal{P}_i), \ldots, r_K(\mathcal{P}_i))$ for each extended path as follows: if the extension is $\hat{D}_{i+1} = k$ then
   $$\begin{aligned} d_k(\mathcal{P}_i) &= 0 & r_k(\mathcal{P}_i) &= 0 \\ d_j(\mathcal{P}_i) &= d_j(P) + 1 & r_j(\mathcal{P}_i) &= r_j(\mathcal{P}_i) + \tau_{i+1} - \tau_i \end{aligned}$$
   (b) Update the likelihood of each path.
3. Put the paths with same $(d_1(\mathcal{P}_i), \ldots, d_K(\mathcal{P}_i))$ into a group.
4. Eliminate groups where for at least one k: $r_k(\mathcal{P}_i) > T_k$.
5. In each remaining group, eliminate all paths except the one with largest likelihood.
6. If $i = M$, done. Choose the most likely path as the assignment. Otherwise, increment i and go to step 2.

Algorithm 2.3 (A2.3 or A3 in plots) Simplified algorithm for separation based solely on timing with known distributions with STO pruning and a static number of states

Given K sources and $\mathcal{T}_k$ with observations $\tau_i, i = 1, \ldots, M$, and a paths in memory

1. Initialize the assignment of τ_1 such that in the first K paths τ_1 is labeled $1, \ldots, K$. The assignment of the remaining $a - K$ paths is arbitrary. Set $i = 1$.
2. For each path consider the extensions where τ_{i+1} is assigned to source $k = 1, \ldots, K$
   (a) Update $(r_1(\mathcal{P}_i), \ldots, r_K(\mathcal{P}_i))$ for each extended path as follows: if the extension is $\hat{D}_{i+1} = k$ then
   $$\begin{aligned} r_k(\mathcal{P}_i) &= 0 \\ r_j(\mathcal{P}_i) &= r_j(\mathcal{P}_i) + \tau_{i+1} - \tau_i \end{aligned}$$
   (b) Update the likelihood of each path.
3. Eliminate paths where for at least one k: $r_k(\mathcal{P}_i) > T_k$.
4. For the remaining paths, keep the a paths with highest likelihood.
5. If $i = M$, done. Choose the most likely path as the assignment. Otherwise, increment i and go to step 2.

Algorithm 2.4 (A2.4 or A4 in plots) Simplified algorithm for separation based solely on timing with known distributions with a static number of states and STO restarts

Given K sources and $\mathcal{T}_k$ with observations $\tau_i, i = 1, \ldots, M$, and a paths in memory

1. Initialize the assignment of τ_1 such that in the first K paths τ_1 is labeled $1, \ldots, K$. The assignment of the remaining $a - K$ paths is arbitrary. Set $i = 1$.
2. For each path consider the extensions where τ_{i+1} is assigned to source $k = 1, \ldots, K$
   (a) Update $(r_1(\mathcal{P}_i), \ldots, r_K(\mathcal{P}_i))$ for each extended path as follows: if the extension is $\hat{D}_{i+1} = k$ then
   $$r_k(\mathcal{P}_i) = 0$$
   $$r_j(\mathcal{P}_i) = r_j(\mathcal{P}_i) + \tau_{i+1} - \tau_i$$
   (b) Update the likelihood of each path.
3. If in the most likely path for at least one k: $r_k(\mathcal{P}_i) > T_k$, start at step 1 for the next impulse. If not, proceed to the next step.
4. For the remaining paths, keep the a paths with highest likelihood.
5. If $i = M$, done. Choose the most likely path as the assignment, and if you have restarted, concatenate all the segments together in order. Otherwise, increment i and go to step 2.

2.2.1 Theoretical error analysis

In this section we develop a lower bound on the error probability for separating two impulsive sources using timing only. The bound is unrelated to any specific algorithm, and can therefore be used to evaluate how close an algorithm is to optimum. It also gives insight into what kind of distributions can be separated, and how the parameters of those distributions influence separability. We derive the bound for two sources only, but the bound can be generalized to more sources.

The bound is based on a rather crude analysis. In general, there are many ways the two sources can be confused, but to be able to evaluate the bound, we focus on one specific type of error event. Consider the i-th impulse of source 1, and let $j(i)$ be the nearest impulse of source 2. Let $\mathcal{A}_i$ be the event that impulses i and $j(i)$ are nearest neighbors and that the classifications of the i-th impulse and the $j(i)$-th impulse are swapped (i.e. i is assigned to 2 and j is assigned to 1). If $\mathcal{A}_i$ happens, one error for source 1 happens. Since i and $j(i)$ are nearest neighbors, the pair is disjoint from any other nearest neighbor pair. We discount all other possible errors because we are developing a lower bound. A lower bound on the error probability of source 1 therefore is

$$P_{e,1} \geq \lim_{n\to\infty} E\left[\frac{1}{n}\sum_{i=1}^{n} I(\mathcal{A}_i)\right] = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} P(\mathcal{A}_i) \tag{2.2.2}$$

$$= P(\mathcal{A}_i), \tag{2.2.3}$$

where n is the number of impulses from source 1 and $I(\cdot)$ is the indicator function for event $\mathcal{A}_i$. We arrive at (2.2.3) since for large n, the $\mathcal{A}_i$ have the same probability (though they are not independent).

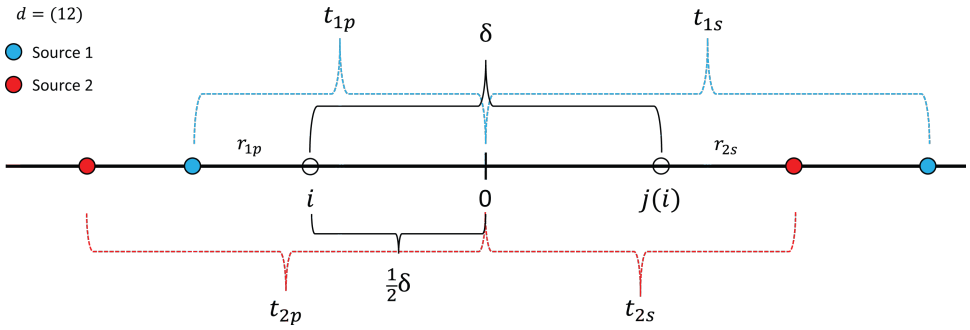


Figure 2.2.1: Timeline illustrating the various quantities and their relations introduced for the error bound calculation for a specific source assignment. The assumption of $d = (12)$ only matters for the r_{kp}, r_{ks} distinctions. Drawing not to scale.

The probability $P(\mathcal{A}_i)$ is still difficult to calculate. However, we can find a lower bound for $P(\mathcal{A}_i)$ by letting a *genie* provide the correct decision for all impulses except i and $j(i)$. The genie also tells us that impulses i and $j(i)$ are from different sources. In the following refer to Figure 2.2.1 for an illustration of notation. Denote $\delta = \tau_{j(i)} - \tau_i$, t_{kp} as the time-difference from the midpoint between i and $j(i)$ to the nearest prior impulse from source k, and t_{ks} as the time-difference to the nearest following impulse. Let $\mathbf{t} = (t_{1p}, t_{2p}, t_{1s}, t_{2s})$ and let $d \in \{(12), (21)\}$, where $d = (12)$ means that the first impulse is from source 1, the second from source 2. The optimum decision is now

$$\hat{d} = \operatorname{argmax} P(d|\mathbf{t}) = \operatorname{argmax} \frac{f(\mathbf{t}|d)P(d)}{f(\mathbf{t})}$$

$$= \operatorname{argmax} f(\mathbf{t}|d),$$

where $f(\mathbf{t})$ is the joint pdf of $\mathbf{t}$ obtained from the $\mathcal{T}_k$. Here we have used that $P(d) = \frac{1}{2}$ (due to symmetry). We normalize time so that the midpoint between pulses i and $j(i)$ is zero. We have

$$\begin{aligned}
f(\mathbf{t}|d=(12)) =& \mathcal{T}_1(t_{1p}-\tfrac{1}{2}\delta|t_{1p}>\tfrac{3}{2}\delta)\mathcal{T}_1(t_{1s}+\tfrac{1}{2}\delta|t_{1s}>\tfrac{3}{2}\delta)\mathcal{T}_2(t_{2p}+\tfrac{1}{2}\delta|t_{2p}>\tfrac{3}{2}\delta)\mathcal{T}_2(t_{2s}-\tfrac{1}{2}\delta|t_{2s}>\tfrac{3}{2}\delta)\\
=& \frac{\mathcal{T}_1(t_{1p}-\frac{1}{2}\delta)\mathcal{T}_1(t_{1s}+\frac{1}{2}\delta)\mathcal{T}_2(t_{2p}+\frac{1}{2}\delta)\mathcal{T}_2(t_{2s}-\frac{1}{2}\delta)}{P_1(t>\frac{3}{2}\delta)^2P_2(t>\frac{3}{2}\delta)^2}\\
f(\mathbf{t}|d=(21)) =& \frac{\mathcal{T}_1(t_{1p}+\frac{1}{2}\delta)\mathcal{T}_1(t_{1s}-\frac{1}{2}\delta)\mathcal{T}_2(t_{2p}-\frac{1}{2}\delta)\mathcal{T}_2(t_{2s}+\frac{1}{2}\delta),}{P_1(t>\frac{3}{2}\delta)^2P_2(t>\frac{3}{2}\delta)^2}
\end{aligned}$$

where P_k is the probability under distribution $\mathcal{T}_k$.

If the timing process is a Poisson process, $\mathcal{T}_1$ and $\mathcal{T}_2$ are exponential, i.e., $\mathcal{T}_i(t)=\lambda_i e^{-\lambda_i t}$, so that

$$\mathcal{T}_1(t_{1p}-\tfrac{1}{2}\delta)\mathcal{T}_1(t_{1s}+\tfrac{1}{2}\delta)=\lambda_1^2 e^{-\lambda_1(t_{1p}+t_{1s})}=\mathcal{T}_1(t_{1p}+\tfrac{1}{2}\delta)\mathcal{T}_1(t_{1s}-\tfrac{1}{2}\delta)$$

and similarly for $\mathcal{T}_2$. Therefore these two probabilities are equal and the optimum decision is achieved by randomly choosing one of the hypotheses. Since this is a lower bound, it shows

Proposition 2.1. *Poisson processes cannot be separated purely based on timing.*

This is a direct result of the memoryless property of exponential distributions that make up a Poisson process. For more information, see Theorem E.3 in Appendix E.

Now assuming that the process is not Poisson, we can find the error probability by calculating the probability that $f(\mathbf{t}|d)$ is larger for the incorrect decision than for the the correct decision. This can be done for truncated Gaussians:

Theorem 2.1. *Suppose $\mathcal{T}_i\sim\mathcal{N}_0(\mu_i,\sigma_i^2)$. A lower bound on the error probability for source 1 then is*

$$P_{e,1}\geq\int_0^\infty\left(\int_x^\infty g_{X_2}(x,t)dt\right)\left(\int_{\sigma_2 x/\sigma_1}^\infty g_{X_1}(\sigma_2 x/\sigma_1,t)dt\right)\frac{\sigma_2}{\mu_2}Q\left(x-\frac{\mu_2}{\sigma_2}\right)dx. \tag{2.2.4}$$

where

$$g_{X_i}(x,t)=\frac{Q\left(\sqrt{2}(-\frac{\mu_i}{\sigma_i}+x-\frac{t}{2})\right)+Q\left(\sqrt{2}(-\frac{\mu_i}{\sigma_i}+2x-\frac{t}{2})\right)-1}{2\sqrt{\pi}Q\left(\frac{-\mu_i}{\sigma_i}+x\right)Q\left(\frac{-\mu_i}{\sigma_i}+2x\right)}e^{-\frac{t^2}{4}}$$

and $Q(x)=1-\Phi(x)$. Note, the same holds for $P_{e,2}$; in (2.2.4) just change all $\cdot_1$ for $\cdot_2$ and vice versa.

Proof. Let us assume $d=(12)$. An error happens if

$$\mathcal{T}_1(t_{1p}+\tfrac{1}{2}\delta)\mathcal{T}_1(t_{1s}-\tfrac{1}{2}\delta)\mathcal{T}_2(t_{2p}-\tfrac{1}{2}\delta)\mathcal{T}_2(t_{2s}+\tfrac{1}{2}\delta)>\mathcal{T}_1(t_{1p}-\tfrac{1}{2}\delta)\mathcal{T}_1(t_{1s}+\tfrac{1}{2}\delta)\mathcal{T}_2(t_{2p}+\tfrac{1}{2}\delta)\mathcal{T}_2(t_{2s}-\tfrac{1}{2}\delta).$$

In particular if we let $\mathcal{T}_i\sim\mathcal{N}_0(\mu_i,\sigma_i^2)$ an error happens if[2]

$$\begin{aligned}
&\tfrac{1}{\sigma_1^2}(t_{1p}+\tfrac{1}{2}\delta-\mu_1)^2+\tfrac{1}{\sigma_1^2}(t_{1s}-\tfrac{1}{2}\delta-\mu_1)^2\\
&+\tfrac{1}{\sigma_2^2}(t_{2p}-\tfrac{1}{2}\delta-\mu_2)^2+\tfrac{1}{\sigma_2^2}(t_{2s}+\tfrac{1}{2}\delta-\mu_2)^2
\end{aligned}\quad<\quad\begin{aligned}
&\tfrac{1}{\sigma_1^2}(t_{1p}-\tfrac{1}{2}\delta-\mu_1)^2+\tfrac{1}{\sigma_1^2}(t_{1s}+\tfrac{1}{2}\delta-\mu_1)^2\\
&+\tfrac{1}{\sigma_2^2}(t_{2p}+\tfrac{1}{2}\delta-\mu_2)^2+\tfrac{1}{\sigma_2^2}(t_{2s}-\tfrac{1}{2}\delta-\mu_2)^2.
\end{aligned}$$

Equivalently,

$$0<-\frac{1}{\sigma_1^2}(t_{1p}-\mu_1)+\frac{1}{\sigma_1^2}(t_{1s}-\mu_1)-\frac{1}{\sigma_2^2}(t_{2s}-\mu_2)+\frac{1}{\sigma_2^2}(t_{2p}-\mu_2).$$

[2]Here the truncated Gaussian gives the same result as Gaussian, as the scaling constants cancel out.

Rewritten in terms of the inter-impulse spacing, $r_{1p} = t_{1p} - \frac{1}{2}\delta$, $r_{1s} = t_{1s} + \frac{1}{2}\delta$ etc.,

$$\left(\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}\right)\delta < -\frac{1}{\sigma_1^2}(r_{1p} - \mu_1) + \frac{1}{\sigma_1^2}(r_{1s} - \mu_1) - \frac{1}{\sigma_2^2}(r_{2s} - \mu_2) + \frac{1}{\sigma_2^2}(r_{2p} - \mu_2).$$

Since $X > a, Y > b \Rightarrow X + Y > a + b$, for independent random variables X, Y

$$\Pr(X + Y > a + b) \geq \Pr(X > a)P(Y > b),$$

and we can therefore lower bound

$$P_{e,1} \geq \Pr\left(-\frac{1}{\sigma_1}(r_{1p} - \mu_1) + \frac{1}{\sigma_1}(r_{1s} - \mu_1) > \frac{1}{\sigma_1}\delta\right)\Pr\left(-\frac{1}{\sigma_2}(r_{2p} - \mu_2) + \frac{1}{\sigma_2}(r_{2s} - \mu_2) > \frac{1}{\sigma_2}\delta\right). \quad (2.2.5)$$

Here $r_{1p} > \delta$ by the setup, and the random variable $\frac{1}{\sigma_1}(r_{1p} - \mu_1)$ therefore has pdf (for $t > \frac{-\mu_1+\delta}{\sigma_1}$)

$$f(t) = \frac{1}{Q\left(\frac{-\mu_1+\delta}{\sigma_1}\right)\sqrt{2\pi}}\exp\left(-\frac{t^2}{2}\right).$$

Also $r_{1s} > 2\delta$ and the random variable $\frac{1}{\sigma_1}(r_{1s} - \mu_1)$ therefore has pdf (for $t > \frac{-\mu_1+2\delta}{\sigma_1}$)

$$\tilde{f}(t) = \frac{1}{Q\left(\frac{-\mu_1+2\delta}{\sigma_1}\right)\sqrt{2\pi}}\exp\left(-\frac{t^2}{2}\right).$$

The random variable $X_1 = -\frac{1}{\sigma_1}(r_{1p} - \mu_1) + \frac{1}{\sigma_1}(r_{1s} - \mu_1)$ is difference, and the pdf can be calculated using a convolution-type integral of these two pdfs, which gives[3]

$$\begin{aligned} f_{X_1}(t|\delta) &= -\frac{e^{-\frac{t^2}{4}}}{4\sqrt{\pi}Q\left(\frac{-\mu_1+\delta}{\sigma_1}\right)Q\left(\frac{-\mu_1+2\delta}{\sigma_1}\right)}\left(\text{Erf}\left[\frac{-\mu_1+\delta}{\sigma_1} - \frac{t}{2}\right] + \text{Erf}\left[\frac{-\mu_1+2\delta}{\sigma_1} - \frac{t}{2}\right]\right) \\ &= -\frac{e^{-\frac{t^2}{4}}}{2\sqrt{\pi}Q\left(\frac{-\mu_1+\delta}{\sigma_1}\right)Q\left(\frac{-\mu_1+2\delta}{\sigma_1}\right)}\left(Q\left[\sqrt{2}\left(\frac{-\mu_1+\delta}{\sigma_1} - \frac{t}{2}\right)\right] + Q\left[\sqrt{2}\left(\frac{-\mu_1+2\delta}{\sigma_1} - \frac{t}{2}\right)\right] - 1\right), \end{aligned}$$

where $-2\mu_1 + 3\delta \leq t$. A similar calculation can be done for the quantities related to source 2. Let

$$g_{X_i}(x,t) == -\frac{\left(Q\left[\sqrt{2}\left(\frac{-\mu_i}{\sigma_i} + x - \frac{t}{2}\right)\right] + Q\left[\sqrt{2}\left(\frac{-\mu_i}{\sigma_i} + 2x - \frac{t}{2}\right)\right] - 1\right)e^{-\frac{t^2}{4}}}{2\sqrt{\pi}Q\left(\frac{-\mu_i}{\sigma_i} + x\right)Q\left(\frac{-\mu_i}{\sigma_i} + 2x\right)}.$$

Given $D = \frac{\delta}{\sigma_2}$, we now write the second factor in (2.2.5) as

$$\int_D^\infty g_{x_2}(D,t)dt,$$

and therefore

$$P_{e,1} \geq \int_{\frac{\sigma_2}{\sigma_1}D}^\infty g_{x_1}(\frac{\sigma_2}{\sigma_1}D,t)dt\int_D^\infty g_{x_2}(D,t)dt.$$

According to renewal theory [29, Theorem 10.3.5], D has distribution

$$f_D(d) = \frac{\sigma_2}{\mu_2}Q\left(d - \frac{\mu_2}{\sigma_2}\right).$$

We can then lower bound the error probability by (2.2.4). □

[3] The integral was done in Mathematica.

2.2.2 Performance of algorithms

In this section, under the assumption that $\mathcal{T}_k$ are completely known, we compare the different algorithms from Section 2.2, as well as the lower bound of Theorem 2.1.

In order to verify the performance of the different algorithms we simulate impulse times according to Algorithm 2.5. Algorithm 2.5 generates data according to the generic system model (Section 2.1) for specified parameters and number of sources. We will often refer to it as the *data generation algorithm* within this paper.

Algorithm 2.5 Method to generate a mixture of impulse times for given $\boldsymbol{\Theta}$ and K

Given the number of sources K and the timing distribution type and parameters $\boldsymbol{\Theta}$ (e.g. $\{(\mu_1, \sigma_1^2), (\mu_2, \sigma_2^2), \ldots\}$), output M impulse times of mixture corresponding to the model in Section 2.1.

1. Generate $\tilde{M}$ interimpulse times for each source according to $\boldsymbol{\Theta}$, where $\tilde{M}$ is some constant such that $\tilde{M} \gg M$
   (a) See Appendix A for more details on generating samples from a specified distribution
   (b) Let $\boldsymbol{t}_k = \{t_{k,1}, t_{k,2}, \ldots, t_{k,\tilde{M}}\}$ where $t_{k,i}$ is the i-th interimpulse spacing sample generated for the k-th source
2. To convert $\boldsymbol{t}_k$ to impulse times $\boldsymbol{\tau}_k = \{\tau_{k,1}, \tau_{k,2}, \ldots, \tau_{k,\tilde{M}-1}\}$, take the cumulative sum
   (a) $\tau_{k,1} = t_{k,1}$
   (b) For $i > 1$, $\tau_{k,i} = \tau_{k,i-1} + t_{k,i}$
3. Remove the trailing ends
   (a) Find $\tilde{T} = \min\{\tau_{1,\tilde{M}}, \tau_{2,\tilde{M}}, \ldots, \tau_{K,\tilde{M}}\}$
   (b) For each $\boldsymbol{\tau}_k$, delete all times $> \tilde{T}$
4. Merge and sort the $\boldsymbol{\tau}_k$ in to $\boldsymbol{\tau}$
   (a) $\boldsymbol{\tau} = \text{sort}\{\boldsymbol{\tau}_1, \boldsymbol{\tau}_2, \ldots, \boldsymbol{\tau}_K\}$, where "sort" is a function that reorders elements in ascending order (e.g. $\text{sort}\{a, b, c\} = [c, a, b]$ if $c \leq a \leq b$)
   (b) Take special care to also keep record (possibly in another vector) of which source each impulse time belongs to
   (c) Inspect $\boldsymbol{\tau}$ for duplicate values. Usually there are none, but if one or more is found restart the process.
5. Get the out impulse train times
   (a) Delete all but the *last* M times from $\boldsymbol{\tau}$(this is how we avoid any initial transient behavior)
   (b) Set first time to 0 (this step is optional): $\boldsymbol{\tau} = \boldsymbol{\tau} - \tau_1$

First, we test the lower bound. We use the data generation algorithm to generate 500 impulses from two sources with $\mathcal{T}_1 \sim \mathcal{N}_0(1, \sigma^2)$ and $\mathcal{T}_2 \sim \mathcal{N}_0(\pi, \sigma^2)$. Then A2.1, A2.2, A2.3, and A2.4 are applied to the times, and we compute the source error on the outputs as $P_{e,k}$ defined in Section 2.1 evaluated for finite time/number of impulses

$$P_{e,k} = \frac{\sum_i I(\hat{D}_i \neq D_i) I(D_i = k)}{\sum_i I(D_i = k)},$$

where $I(\cdot)$ is the indicator function

$$I(X) = \begin{cases} 1 & \text{if } X \\ 0 & \text{otherwise} \end{cases}.$$

Note, $I(\hat{D}_i \neq D_i)I(D_i = k) = 1$ i.f.f. there is an error *and* the correct assignment is source k; it is 0 otherwise. We adjust σ between 10^{-3} and 10^{-1}, and repeat this process 500 times for each σ. The average of the source errors over the 500 trials are computed and are compared with the lower bound as shown in Figure 2.2.2. All the proposed algorithms perform similarly; there is a constant factor of about 5 in difference between the lower bound and the algorithms' error rates, although, importantly, they have the same slope. We do not know if the gap is due to the bound being loose, or the fact that even the exact maximum likelihood algorithm A2.1 is not a direct optimum solution. As σ increases, the simplified algorithms (A2.3 & A2.4) show a slight degradation in performance in comparison with the "optimal" ones (A2.1 & A2.2). Thus with some confidence we can substitute these simplified algorithms for the optimal ones for large problems where the optimum algorithms become too complex. In this test, there is no significant difference between the two simplified algorithms.

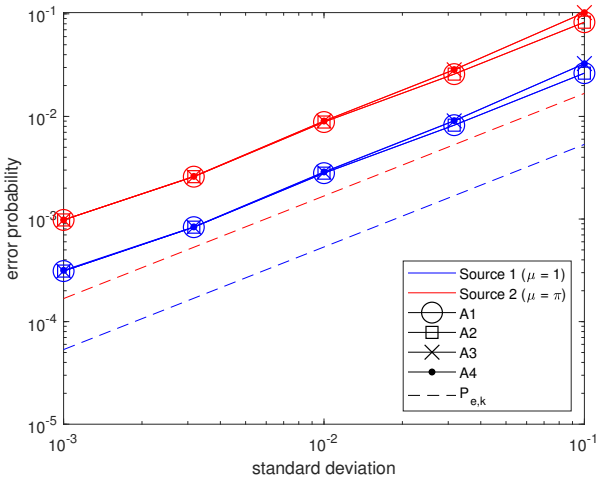


Figure 2.2.2: For two synthetic sources with fixed mean impulse spacings and identical standard deviations, the source error probability of each source from A2.1, A2.2, A2.3, and A2.4 with $T_k = \mu_k + 3\sigma_k$ and $a = K^3$ where needed was averaged over 500 trials. The theoretical lower bounds (2.2.4) are shown as a dashed lines.

Then to give some idea of how the algorithm behaves with respect to different means, we consider two synthetic sources with mean impulse spacings 1-100 (at 50 evenly spaced values) and standard deviations of 0.1. We generate a total of 100 impulses using the data generation algorithm and then apply a timing separation algorithm using the actual timing parameters (specifically we show results from A2.2 here, but for these parameters, all methods have virtually identical performance; differences are on the order of 10^{-5}) . As a performance metric we will use P_e given by (2.1.1) evaluated for finite M. When means are equal, source numbering is arbitrary, so we use the minimum P_e over all permutations of labelings. For each dataset (i.e. a set of means), we compute P_e and then compute the average over 500 trials. The results are shown in Figure 2.2.3. In general, we see that the probability of error is low except when the source means are low (< 10) and similar. This is not unexpected: when means are equal there are situations where sources emit at almost simultaneously for a length of time which makes separation is difficult. To see why separation is difficult, consider Figure 2.2.4. When there is a near simultaneous emission, the distance to the next set of impulses is approximately the same from either. Thus the resulting assignment is essentially a coin flip (e.g. 50% error for $K = 2$). Near simultaneous emission can happen in any dataset,

but if the means are the same, near simultaneous emissions are very likely to continue to occur after a single occurrence. There is always some chance that a near simultaneous emission occurs at the first impulse from each source, but another possibility is that one source "drifts" into the other. When then means are lower, the distance a source has to "drift" is smaller. For example, consider we have two sources with $(\mu_k, \sigma_k) = (0.3, 0.1)$ and $t_{1,1} = 0, t_{1,2} = 0.3$. (Here we are using the notation of the data generation algorithm) If $t_{2,1} = 0.1$, it is not very unlikely that $t_{2,2} \approx 0.3 = t_{1,2}$. However, if instead $(\mu_k, \sigma_k) = (25, 0.1)$ and $t_{1,1} = 0, t_{1,2} = 25$ and $t_{2,1} = 8.3$ (again $\approx 1/3$ between $t_{1,1}$ and $t_{1,2}$), it would be quite unusual for $t_{2,2} \approx t_{1,2}$. This explains why we do not see as much error when means are equal but large. Furthermore, when the means are equal but low we do not see the random assignment error rate 0.50 exactly since although simultaneous emissions are occurring often, there are still times where it is not.

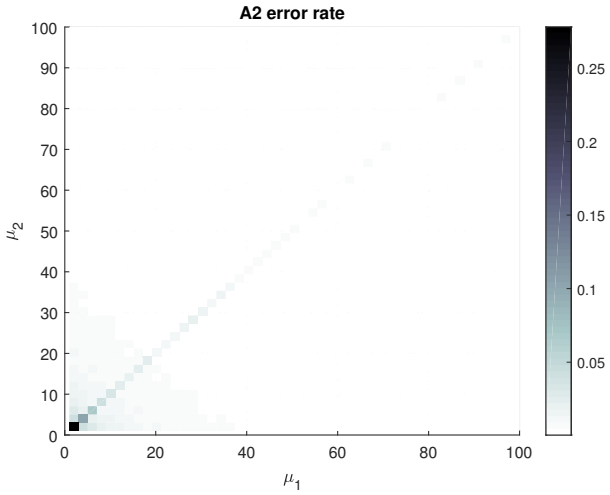


Figure 2.2.3: Total error probability of A2.2 for two sources with $\sigma = 0.1$s and μ_1, μ_2 as shown.

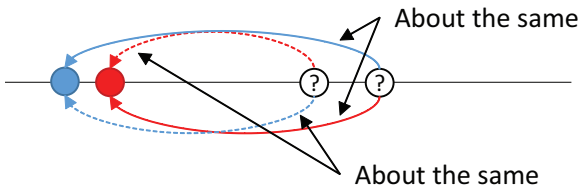


Figure 2.2.4: When two sources have the same mean, it is possible that their impulses from different sources will line up close together as shown. The dots indicate impulses and their position on the line indicates their relative time of occurrence. Consider the red dot as an impulse assigned source 1, the blue dot as an impulse assigned to source 2, and the "?" labeled dots as impulses to be assigned. Note, that the distance to the next two impulses from either the blue or red dot is roughly the same.

We explore this concept a more by considering the coefficient of variation, c_v. For a dataset, it is defined as its standard deviation of some dataset divided by its mean;

$$c_v \triangleq \frac{\sigma}{\mu}.$$

In Figure 2.2.3, we changed the means of sources, but not the standard deviation, so there is a one-to-one correspondence between c_v and a μ_i (e.g. when $\mu_i = 1 \Rightarrow c_v = 10^{-1}$ and when $\mu_i = 100 \Rightarrow c_v = 10^{-3}$). Since the mean shows up in the denominator, we say c_v is inversely proportional to μ_i. That is, when μ_i increases, c_v decreases, and vice versa; see Figure 2.2.10. Based on this we suppose then that when c_v is small, the error is also small. We can confirm this by reconsidering Figure 2.2.2. For this simulation, we changed the standard deviation, but not the mean, so again there is a one-to-one direct correspondence. Though it is a simply transformation, we choose to re-plot the result with regard to c_v in Figure 2.2.5 as it emphasizes that our theoretical lower bound (Theorem 2.1) also behaves in the same fashion: Error rate increases proportionally with c_v (in what appears to be a linear fashion). We do not have a closed form for the integral (2.2.4), but we do recognize that c_v and c_v^{-1} show up in various places in the expression. Most notably, we see c_v for the second source σ_2/μ_2 shows up as a scale factor; perhaps this is the dominant term that determines the trend.

To further support the notion that error rate is tied to c_v, see Figure 2.2.6b. For this figure we consider two identical sources with mean impulse spacings 1-100 (at 50 evenly spaced values) and $c_v \in [10^{-3}, 1]$ (at 50 logarithmically spaced values). Akin to Figure 2.2.3, 100 total impulses are generated we apply the same algorithms and average the error rates over 500 trials. In Figure 2.2.6b, notice that across all μ_i the behavior are virtually the same; the differences are well within normal variation for the number of trials. Looking at Figure 2.2.6c and Figure 2.2.7, the trend appears to be linear[4] up to $c_v \approx 10^{-1}$. After this it begins to roll-off because there is a maximum error rate (0.5 for two sources when optimized over labeling).

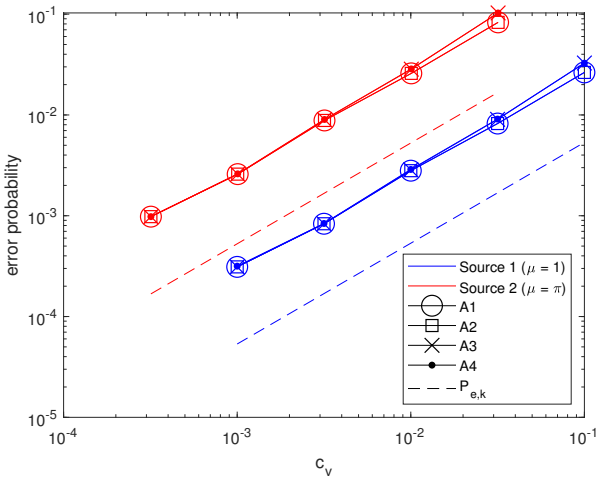


Figure 2.2.5: This is Figure 2.2.2 re-plotted using c_v instead of σ.

According to Figure 2.2.7, $c_v \leq 10^{-2}$ yield error rates less than 0.05. Having a $c_v \leq 10^{-2}$ is a little restrictive, but for this, we assumed the sources have the same c_v. It is also interesting to consider how error rate changes when sources have different c_v. So we consider another simulation where we set the mean of the two sources arbitrarily to 50 (we also repeated the experiment with $\mu_i = 10$ and see the exact same results) and then vary the c_v between 10^{-3} and 10^{0}. For each c_v pair, we apply the same algorithms and compute the error rate and report the average over 500 trials; the result is shown in Figure 2.2.8. Let $c_{v,i}$ be the c_v for the i-th source. When the pair of $c_{v,1} = c_{v,2}$, we get the same result as 2.2.6. Interestingly, when the $c_{v,1} \neq c_{v,2}$, we can achieve the same error rate for larger values of c_v. That is, when at least one $c_{v,i} \leq 10^{-1.5}$, the error rate is less than 0.05 (when c_v are not equal). Note, this means that you can still get a good error rate with large a c_v (e.g. $c_v = 1$) as long as the other

[4] In a semi-log plot, linear data looks like an exponential.

source has a c_v small enough– this is much less restrictive.

So far, the experiments designed to test c_v have always considered $\mu_1 = \mu_2$. This was inspired by the fact that in Figure 2.2.3 the highest error was found when the means were equal. That is, equal means will give us a worst case scenario, so any other combination of means will be bounded by it. In order to confirm this we replicated the same test in Figure 2.2.8, but for unequal mean pairs as shown in Figure 2.2.9. Clearly, this is not an exhaustive set, but we tried to pick pairs to showcase small ($\mu_1 = 1, \mu_2 = 3$), moderate ($\mu_1 = 50, \mu_2 = 75$), and large ($\mu_1 = 10, \mu_2 = 100$) differences in the means. Across all pairs, we see largely the same behavior: Higher error when both sources have large c_v. Perhaps unsurprisingly, Figure 2.2.9a, the moderate difference, is the same as Figure 2.2.8 but without high error along the diagonal (where $c_{v,1} = c_{v,2}$). Then in Figure 2.2.9b, the small difference, the area of high error is slightly lower and *squashed* a bit more to the right than Figure 2.2.9a. This effect is even more exaggerated with the large difference, $\mu_1 = 10, \mu_2 = 100$. Let us define this effect more clearly. For $\mu_1 = \mu_2$, we gave a threshold below which the error rate was less than 0.05 (i.e. $c_v \leq 10^{-1.5}$); *squashing* operates on this threshold. Specifically, squashing means that the $c_{v,1}$ threshold increases but the $c_{v,2}$ threshold decreases. A higher threshold means that we can handle sources with larger variations, whereas a lowered threshold can be viewed as a decrease in performance. We have a combination of an increase and a decrease, but also the maximum error rate is lowered in a squash. So though we only have a few examples, we say that increased squash is to be desired. Squashing appears increase as the ratio between the means (i.e. μ_1/μ_2) decreases rather than the size of the difference between the means. In this experiment μ_1/μ_2 is also the approximate ratio of number of impulses from source 2 to number of impulses from source 1. This seems to imply that c_v is more important when there are less samples, and similarly when there are more samples, c_v is less important. In other words, if there is a lot of data points, we can find the pattern even if the amount of variation is high. But if there is not a lot of data points, then even small amounts of variation can cause issues in detecting the source.

Note with the moderate and small differences, A2.1, A2.2, and A2.3 all have essentially the same outcome, so it suffices to show just one of the results. However, with the $\mu_1 = 10, \mu_2 = 100$, there is noticeable degradation as the more approximate methods are used. Figure 2.2.9d, the A2.2 result, is shown to highlight the difference. The loss in performance can be attributed to the trade off between accuracy and speed.

These results and conclusions about c_v are not in conflict with previous results. Again consider Figure 2.2.10, these are the c_v that are generated in Figure 2.2.3. With exception of the first two means, all are well below $c_v = 10^{-1.5}$, the threshold for less than 0.05 error rate for unequal means. This is why we see very low error rates for most of the mean pairs in Figure 2.2.3. Further, only the first five means are above $c_v = 10^{-2}$, the threshold for less than 0.05 error rate for equal means. This is consistent with Figure 2.2.3 when $\mu_1 = \mu_2$, error rates are only highly elevated for the first few data points.

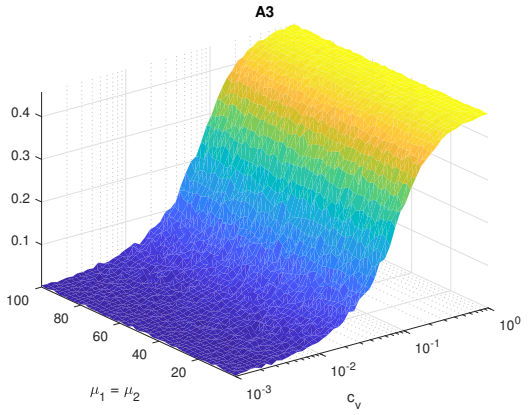
The take away with c_v is that even in the best case, our algorithms will perform poorly in the case of high c_v values. This makes sense; a large c_v indicates a large amount of jitter (relative to the mean interimpulse spacing). While our algorithms are designed to handle jitter directly via the optimization of the likelihood function, at some point too much jitter can obfuscate any pattern. Though we see that c_v directly influences error rate, in future simulations, we do not always consider c_v as an independent variable. This is because as seen in Figure 2.2.9, individual parameters (e.g. the μ_i) also impact performance, and c_v is a composite parameter computed from the individual parameters. Further, the way c_v is computed, there is not a one-to-one mapping between c_v and the individual parameters, so it is not sufficient to only consider c_v.

Finally, for the approximate algorithms A2.3 and A2.4, we mentioned that K^3 seemed to be a sufficient number of paths; now we will provide some evidence. We repeat the previous simulation for $a = K, K^2, K^3, K^4, K^5$. As performance measure we use $\bar{P}_e$, the average P_e over all the $(\mu_1, \mu_2) \in [1, 100]^2$ generated. Results are given in Table 2.2.1. As expected, the performance improves with increasing a. However, the improvement diminishes. Going to K^4 from K^3, the improvement in performance is on the order of 10^{-5}. For this reason, we decide that added overhead is not worth the reduction in error rate, and we settle on K^3 as the best value for a to use. From

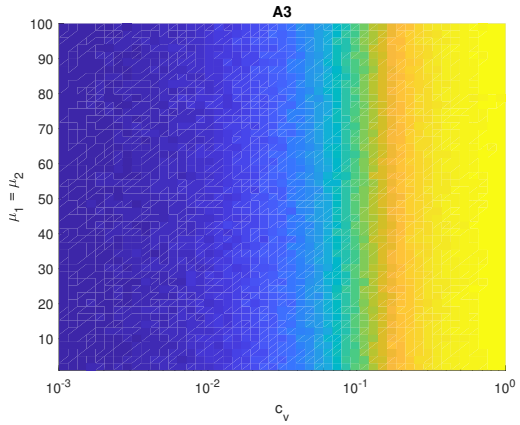
this point on, all algorithms/results that require a use K^3 unless otherwise stated. While both algorithms perform similarly, it is interesting to note that A2.4 is consistently better than A2.3 but only by about 1.91×10^{-5}.

Table 2.2.1: Error rate for different values of a and the improvement in rate from increasing a

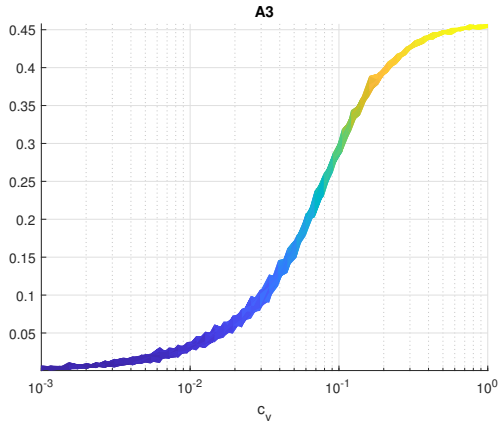
a	$\bar{P}_e$ (A2.3)	Improvement	$\bar{P}_e$ (A2.4)	Improvement
K	0.0083198	-	0.008832	-
K^2	0.002573	0.0057468	0.0025527	0.0062790
K^3	0.0025368	3.62E-5	0.0025197	3.3E-5
K^4	0.0025283	8.5E-6	0.0025078	1.19E-5
K^5	0.0025225	5.8E-6	0.002504	3.8E-6



(a) This is a 3D-plot of the error rate. The c_v axis is a log scale.



(b) X-Y plane view of the same result in Figure 2.2.6a to emphasize its behavior is constant regardless of choice of mean.



(c) X-Z plane view of the same result in Figure 2.2.6a so the trend as a function of c_v can better be revealed.

Figure 2.2.6: Total error probability averaged over 500 trials of A2.3 for two sources with $\mu_1 = \mu_2$ as shown, with $\sigma_1 = \sigma_2$ computed for the c_v as shown. A2.1 and A2.2 are not shown here, but have essentially the same result.

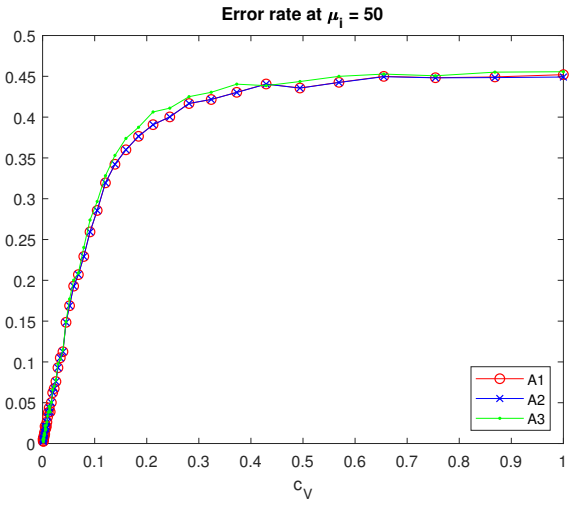


Figure 2.2.7: We take the result shown in Figure 2.2.6 and only plot the data points for $\mu_i = 50$. Additionally, we compare the data for A2.1 and A2.2. Note, in this plot, we used a linear scale for c_v instead of the log-scale.

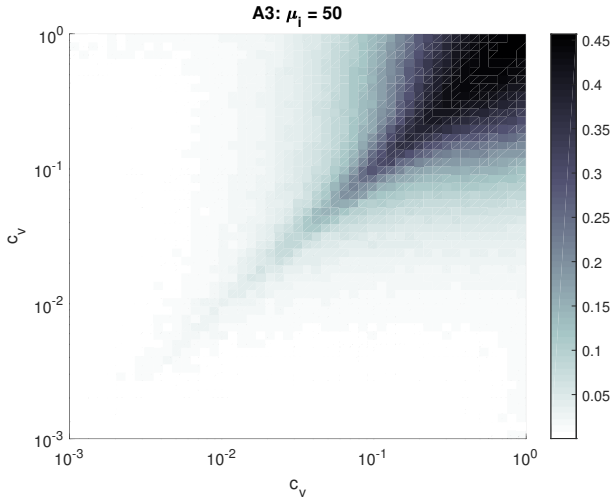
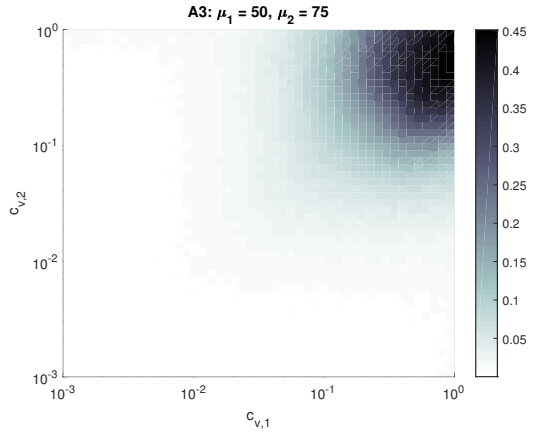
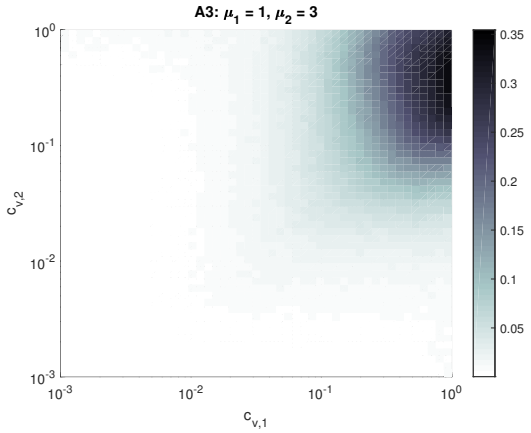


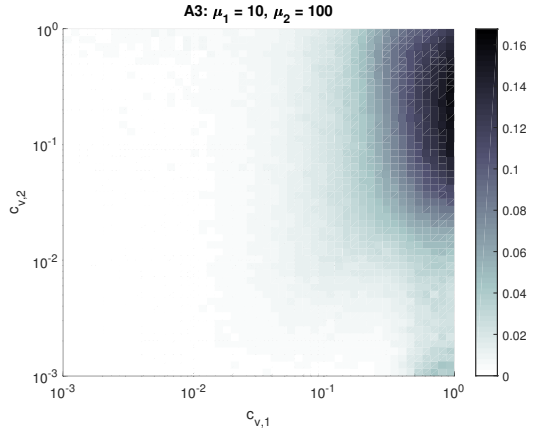
Figure 2.2.8: Total error probability averaged over 500 trials of A2.3 for two sources with $\mu_1 = \mu_2 = 50$ and with $\sigma_1 = \sigma_2$ computed for the c_v as shown. A2.1 and A2.2 are not shown here, but have essentially the same result.



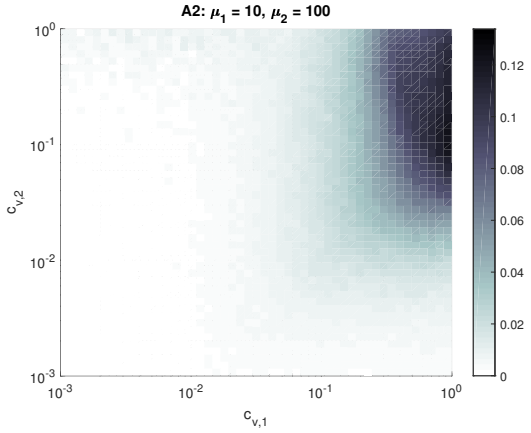
(a) $\mu_1 = 50, \mu_2 = 75$



(b) $\mu_1 = 1, \mu_2 = 3$



(c) $\mu_1 = 10, \mu_2 = 100$



(d) $\mu_1 = 10, \mu_2 = 100$ for A2.2

Figure 2.2.9: The is the same experiment as in Figure 2.2.8 but repeated for different mean pairs as shown. Since $\mu_1 \neq \mu_2$, we specify that the x-axis denotes $c_{v,1}$ and the y-axis denotes $c_{v,2}$. Again A2.1 and A2.2 are not shown here, but have essentially the same result as A2.3 except for the $\mu_1 = 10, \mu_2 = 100$ pair. So to compare with Figure 2.2.9c, we also show Figure 2.2.9d which uses A2.2 for $\mu_1 = 10, \mu_2 = 100$. (A2.1 and A2.2 are nearly identical for this pair)

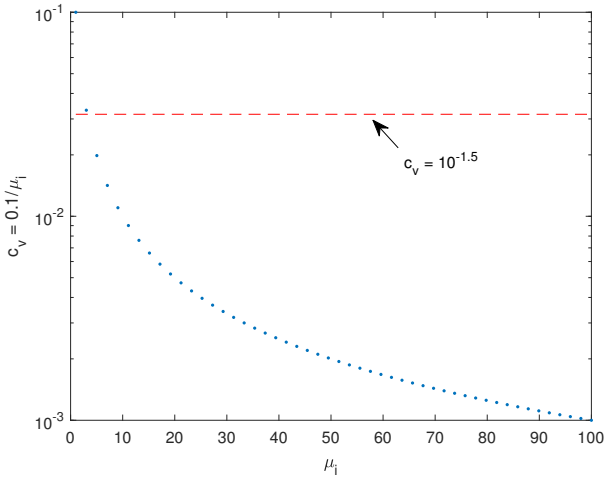


Figure 2.2.10: This is a semi-log plot of the c_v for the μ_i used in Figure 2.2.3 where $\sigma_i = 0.1$.

2.2.3 Effect of signal length

Except in Section 2.2.1 where we assume $T \to \infty$, it was always understood that we would be working with finite values for T. Note a finite value of T implies a finite value of M. The development of the algorithm(s) were done under the assumption that there would be enough impulses, but we never addressed: How much is *enough*? Or: How much is *not* enough? For A2.2, we did discuss elimination of low likelihood solution paths (i.e. paths with STO) which is somewhat related but does not directly tell us how many impulses are needed for our algorithm to work *well* (it will operate on any amount of clicks).

Decisions are made on an impulse-by-impulse basis with some memory of the past. This dependence on the past gives some indication that there is some dependence on M with regards to performance. For example, in the case where $M = 1$, the problem is trivial. We can achieve an error rate of 0– it is arbitrary to which source we say the click belongs. Additionally, we do not even need to use any algorithm. However when $M = 2$, then more ambiguity is introduced. Intuition tells us that we should have at least 1 interimpulse spacing from each source, or 2 impulses from each source. This means at least we should have $M \geq 2K$, but let us study this more carefully through some simulations.

First consider $K = 2$. We generate 500 pairs of means randomly selected from the range of 0.5 to 2 (i.e. the neighborhood of sperm whale ICI). With the standard deviation of each source set to 0.1, we generate $M = 4000$ impulses. For each set of impulses, we consider the subset of impulses listed in Table 2.2.2. A2.2 and A2.3 are applied to each subset and we compute the error rate P_e. For each M_e, effective M value, P_e is averaged for the 500 parameter sets; the result is shown in Figure 2.2.11a and Figure 2.2.12a. Looking at the results for A2.2, for $M < 20$, the error rate decreases, then for $M > 20$ it increases slightly and then settles to a near constant error rate of approximately 0.092 beginning where $M_e = 200$. The approximate algorithm mimics this behavior, but it does not appear to approach a limit; it continues to increase, but the rate of increase does slow. As M grows larger, the number of possible assignments also grows. However, in A2.3, we only maintain a set number of paths (independent of M). Thus the ratio of the number of paths maintained to the total number of possible assignments decreases as M increases, so it follows that the quality of the approximation also degrades.

Additionally, let P_e^0 be the lower bound for $P_{e,}$ computed using (2.1.2) substituting our expression for the lower bound (2.2.4) for $P_{e,k}$. We compute the ρ_k based off of the total $M = 4000$ impulses rather than separately for each of the subset. Averaging over the parameter sets we get $\bar{P}_e^0 = 0.0147$; this is shown as a green dotted line in the Figure 2.2.11a. Further, for each sub-click-train length we compute the ratio

$$\frac{P_e}{P_e^0}$$

and report the average shown in Figure 2.2.11b. Note, this is not the same as taking the points in Figure 2.2.11a and dividing by $\bar{P}_e^0$. Here we see that the empirical error of A2.2 settles at ≈ 6.3 of the theoretical lower bound (A2.3 is higher at ≈ 8.3 times). In Figure 2.2.2 the empirical error rate was about 5 times that of the theoretical lower bound. We explore the reason for this difference more in Section 2.2.4, but without going into detail, it is because when $\mu_1 = \mu_2$, the ratio is higher.

Finally, we repeat the same test for $K = 3$, but we do not compare with the lower bound (since we do not have an expression for it). This result is shown in Figure 2.2.11c and Figure 2.2.12b. The results are nearly the same as for $K = 2$; just the rates are slightly higher. That is instead of settling to an error rate of 0.092, A2.2 settles to about 0.17. Also for small M, unlike $K = 2$, it does not jump to a high rate after $M = 1$, it increases more slowly till $M = 6$ and then there is a small decrease. The approximate algorithm again mirrors the behavior but at higher error rates.

The effect of the number of impulses on performance can be summarized:

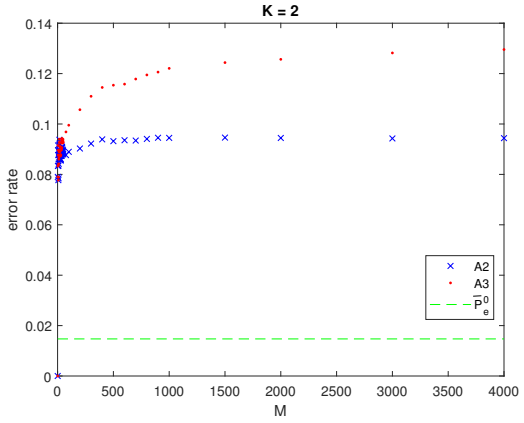
- There is a transient regime for approximately $M < 50$. Error rate is erratic, but the minimal error rate may occur here.

- Stable regime, when $M > 50$, the error rate slowly increases approaching some maximum rate. A2.3 reaches its maximal rate later than A2.2.

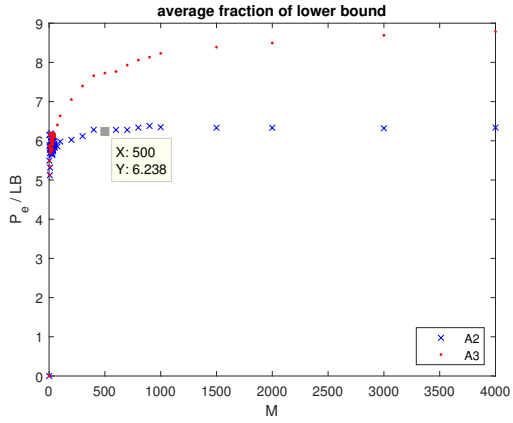
(Note these values are assuming small values for K, e.g. $K = 1, 2, \ldots, 10$, we expect that if K is large, e.g. $K = 200$, the boundaries for the transient and stable regimes will shift upwards appropriately) Though the best error rate may occur within the transient, as seen in Figure 2.2.11c, the maximum rate may also occur. Additionally, it is less meaningful to classify short segments especially if there is more error and/or complexity introduced associating adjacent short segments. Therefore, based on these results we recommend $M \in [50, 200]$ for our algorithms. This is a very general ball park suggestion, interested users should check the range using their specific problem parameters.

Table 2.2.2: Subset indexes

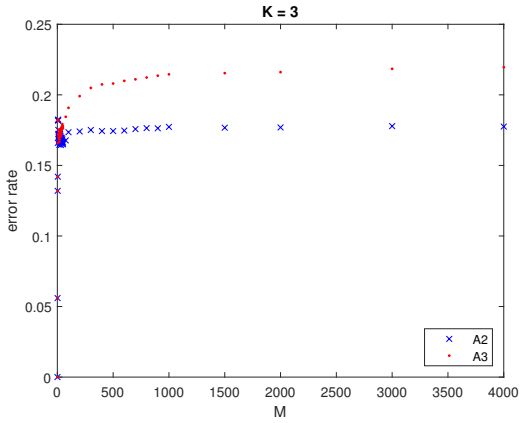
First impulse	1	1	1	$\cdots$	1	1	1	1	1	$\cdots$	1	1	1	1	1
Last impulse	1	2	3	$\cdots$	50	75	100	200	300	$\cdots$	1000	1500	2000	3000	4000
M_e	1	2	3	$\cdots$	50	75	100	200	300	$\cdots$	1000	1500	2000	3000	4000



(a) $K = 2$

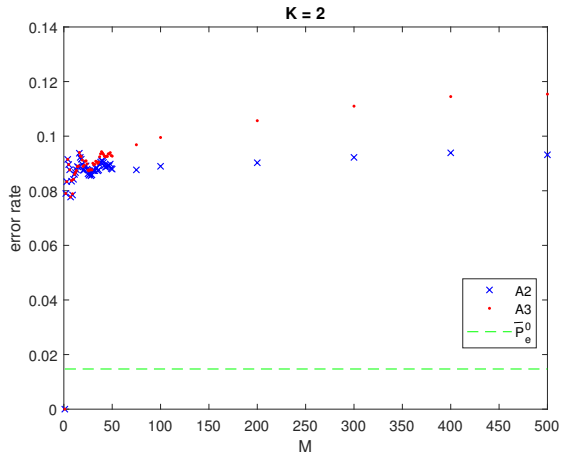


(b) Ratio of error to theoretical lower bound for $K = 2$.

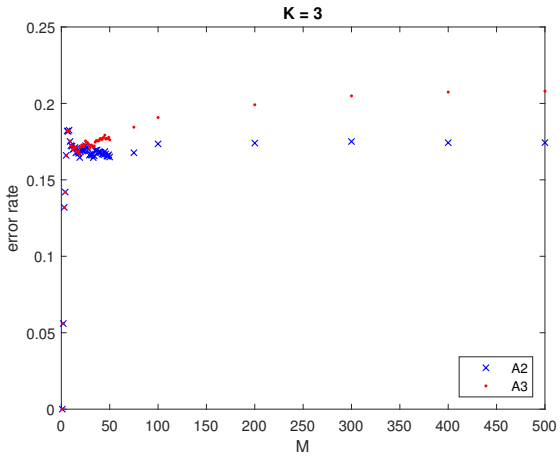


(c) $K = 3$, note we do not have an expression for the lower bound here

Figure 2.2.11: The performance with respect to M is studied by looking at the average error rates of impulse sets of varying lengths.



(a) Figure 2.2.11a zoomed in.



(b) Figure 2.2.11c zoomed in.

Figure 2.2.12: Figure 2.2.11 zoomed in to $M \in [0, 500]$.

2.2.4 Limitations of the lower bound

In Figure 2.2.2 the actual error rate was about 5 times more than the predicted lower bound, but in Figure 2.2.11b, we saw that as $T \to \infty$ the actual error rate settled to 6.3 times the lower bound. To see why this is the case, we will explore the limitations of the lower bound derived in Section 2.2.1. This extends results regarding the lower bound presented/published previously.

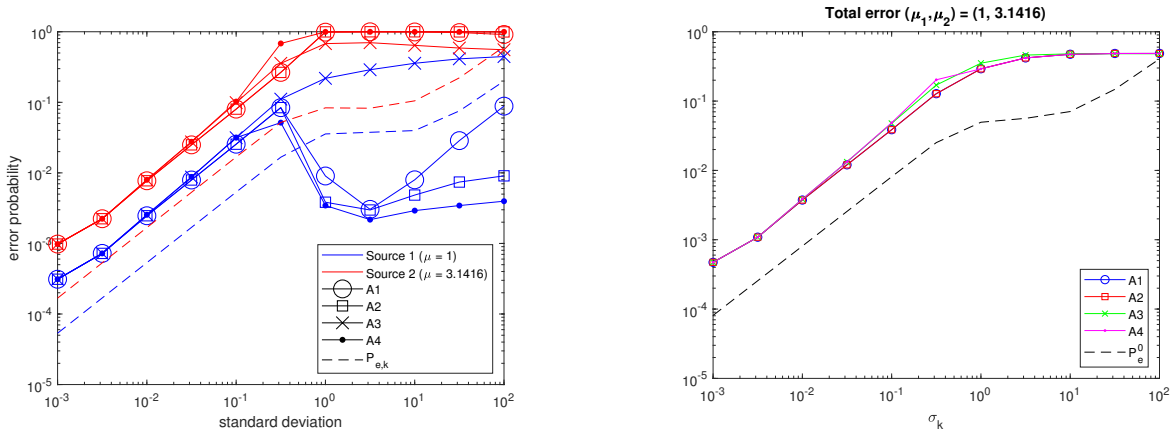
The relationship seen in Figure 2.2.2 is that error rate and lower bound is linear with respect to σ_k. And in other results we showed this translated to error rate being proportional to c_v as illustrated in Figure 2.2.6c and Figure 2.2.7. Importantly, we see that while the error rate is linear for roughly $c_v \in [0.01, 0.20]$, the behavior is non-linear on either end of the range. This makes sense, the linear trend cannot continue indefinitely. On the lower end, error rate is non-negative, so we cannot surpass 0. Thus c_v decreases, P_e asymptotically approaches 0. On the upper end, error rate cannot surpass 1. Thus as c_v increases, P_e asymptotically approaches 1. This is seen for when $\mu_1 = \mu_2$, let us show a few more examples when $\mu_1 \neq \mu_2$ to confirm this result.

We repeat the simulation used to generate Figure 2.2.2, but we extend the range of σ_i to $[10^{-3}, 10^2]$ in order to see what happens on the upper end. The result is shown in Figure 2.2.13. Additionally, we record the number of impulses for each source per each σ_k to use to estimate ρ_k and then compute P_e^0. We compare P_e^0 with the actual total error rate shown in Figure 2.2.13b. The total error rate here mirrors what we saw in Figure 2.2.6c and Figure 2.2.7. However, if we look at P_e^0, we note that at $\sigma_k = 10^0$ its linear trend is loss; it levels out. At the same time in P_e is close to its maximum value and has begun its asymptotic approach. For two sources, the maximum error is 0.5 (optimizing over labelings), but here $P_e \to 0.486$ roughly. We argue that this is close enough.

More interestingly, looking at the per source errors, Figure 2.2.13a, we see the lower bounds, denoted as $P_{e,k}^0$, behave in the same manner as P_e^0. The error rate corresponding to the larger mean, $P_{e,2}$, follows the response of the total error rate. At $\sigma_k = 10^0$ it reaches a maximum error of 1 (for $P_{e,k}$ we do not optimize over labelings). Whereas $P_{e,1}$, at the same point, suddenly decreases, even below the $P_{e,1}^0$. Note $P_{e,2} = 1$ implies that all impulses assigned to source 2 have instead been assigned to source 1. This is a problem since the development of $P_{e.k}^0$ assumed a *genie* would provide the correct decision for all impulses except where we would make the error. When $P_{e,2} = 1$, this is clearly not the case. For this situation, another model would need to be developed to accurately predict the behavior. Based on the intuition, it seems like when $P_{e,2} =$1, we end up assigning everything to source 1, and this is why $P_{e,1}$ is low. However, this high error regime is not of practical interest, so we do not pursue further analysis. Though interestingly, with A2.4, $P_{e,1}$ does track the asymptotic behavior of $P_{e,2}$ unlike the other algorithms. A2.4 was designed to restart when it detects an error. Perhaps it detects the situation of assigning everything to only source 1 and "recovers" to the predicted behavior. It is difficult to say whether or not following the lower bound desirable when the error rate is high, but maybe it is a moot point. Practically, we do not want results with high error; it is enough to know when that will happen. Also the total error rate behavior is the same across all algorithms.

To address the point of knowing when high error rates will happen, we take another example to see if we can identify any commonalities. We consider $\mu_1 = \mu_2 = 1$ and repeat the same test; the results are shown in Figure 2.2.14. Though there is variation at higher σ_k, $P_{e,k}$ starts off nearly constant at 0.5 not following the $P_{e,k}^0$ trend at all. This is because labeling is arbitrary since $\mu_1 = \mu_2$, but when computing $P_{e,k}$ we do not optimize over the different labeling schemes. What is happening is that half of the time we get the labeling correct (i.e. $P_{e,k} \approx 0$) and half the time we get the labels flipped (i.e. $P_{e,k} \approx 1$); the average of this is $P_{e,k} \approx 0.5$. We confirm this by looking at P_e, where we do optimize over labelings, and P_e^0 (computed from $P_{e,k}$ with $\rho_i = 0.5$); we see that is similar to Figure 2.2.13b. In the lower bound there is a plateau, or in this case a slight dip before the increase which corresponds to reaching the maximum P_e of 0.5 and erratic $P_{e,k}$. Specifically, the plateau starts at $\sigma_k = 10^0$, but there is a slight reduction in the exactly linear trend of the actual error rate at $\sigma_k = 10^{-0.5}$.

In Figure 2.2.14, $\mu_i = 1$, so $\sigma_k \equiv c_v$. Like with Figure 2.2.6, the result is the same with respect to c_v when we use $\mu_k = 10$, shown in Figure 2.2.15. Though we plot versus c_v, we consider the same values of σ_k, so the c_v range is different. With $\mu_i = 10$, we shift one power of 10 down in c_v giving us further insight on the lower end. Similar



(a) Per source error rate

(b) Total error rate

Figure 2.2.13: Empirical and theoretical error rates for $\mu_1 = 1$, $\mu_2 = \pi$ and σ_i as shown. Both per source and total error rate are shown.

to the error "maxing out" on the upper end, the error also appears to "bottom out" on the lower end. Specifically we see that $c_v < 10^{-3}$ bottoms out around $P_e \approx 0.005$. Theoretically, we should be able to achieve 0 error rate, but practically one set of errors (i.e. a single swap of $1 \leftrightarrow 2$ will count as 2 errors) is a reasonable amount. It may have to do with the difficulty of separating identical sources.

Thus far, the results seem to imply that due to practical constraints of error rate and other sources of error, the lower bound (2.2.4) is roughly linear and is a good descriptor of the error rate for $c_v \in [10^{-3}, 10^{0}]$. We confirm this with a few other examples shown in Figure 2.2.16. In call cases, when $c_v > 1$, the source with the larger mean will have reached maximum error and the other source will have departed from the trend. However, the threshold does not appear to be exactly at 10^0 , and perhaps in general it would be safer to say $10^{-0.5}$ is where this divergent behavior begins. Additionally, we do have a few points below 10^{-3}, that do not appear to be bottoming out, perhaps that level was only for $\mu_1 = \mu_2$.

Perhaps the this result is best captured in Figure 2.2.17 where we compare the ratio of error rates with their lower bound. In the case of $\mu_1 \neq \mu_2$ we consider $P_{e,k}/P^0_{e,k}$, and when $\mu_1 = \mu_2$ we look at P_e/P^0_e. Though total error an per source error are fundamentally different, their ratio with their corresponding lower bound are directly comparable. Consider the following:

Claim 2.1. If

$$\alpha = \frac{P_{e,k}}{P^0_{e,k}},$$

where α is some number, then

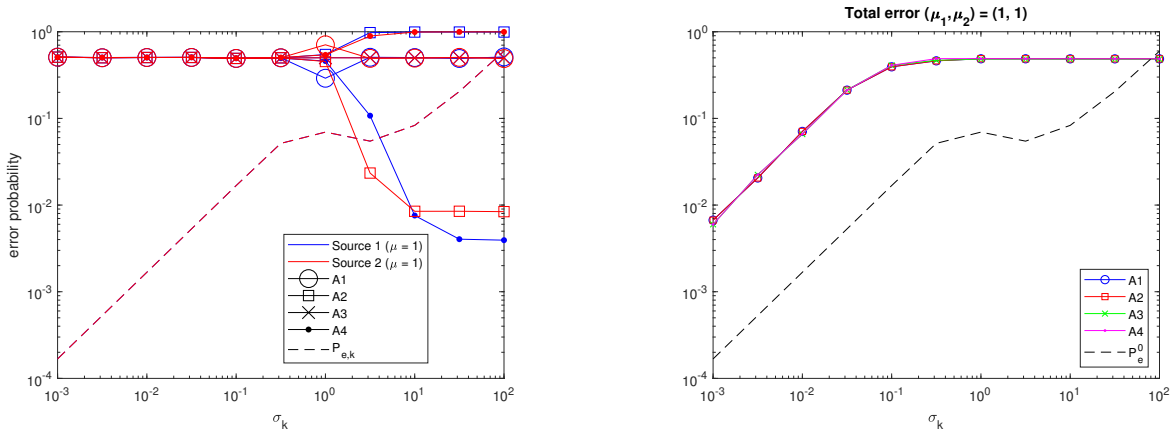
$$\frac{P_e}{P^0_e} = \alpha.$$

Proof. This is a direct result of (2.1.2) which states

$$P_e = \sum \rho_k P_{e,k}.$$

We have a similar expression for the lower bounds

$$P^0_e = \sum \rho_k P^0_{e,k}.$$



(a) Per source error rate

(b) Total error rate

Figure 2.2.14: Empirical and theoretical error rates for $\mu_1 = \mu_2 = 1$ and σ_i as shown. Both per source and total error rate are shown.

If we rearrange the claim, we have

$$P_{e,k} = \alpha P^0_{e,k}.$$

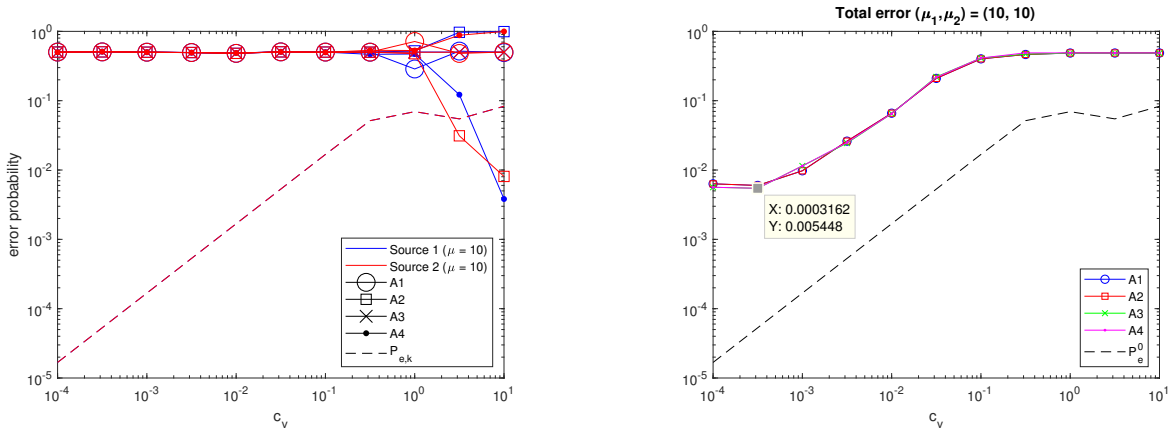
Substituting this into (2.1.2)

$$\begin{aligned} P_e &= \sum \rho_k \alpha P^0_{e,k} \\ &= \alpha \underbrace{\sum \rho_k P^0_{e,k}}_{P^0_e} \\ &= \alpha P^0_e \Rightarrow \frac{P_e}{P^0_e} = \alpha. \end{aligned}$$

□

So what we see in Figure 2.2.17 is that error rate is indeed approximately 5 times the lower bound when $\mu_1 \neq \mu_2$ when $c_v < 10^{-0.5}$ roughly. Above this c_v, ratio jumps for the source with the larger mean and dips for other one. This corresponds to the divergent behavior we saw once a plateau was reached. When $\mu_1 = \mu_2$, the ratio is much higher than when the sources are not identical before $c_v \approx 10^{-0.5}$. After this c_v, the behavior follows the trend of the larger mean when $\mu_1 \neq \mu_2$. In short, this implies is that our bound is too conservative when the sources are identical. Other sources of error need to be modeled to improve accuracy. This further supports the notion that sources with identical timing distributions are more difficult to separate.

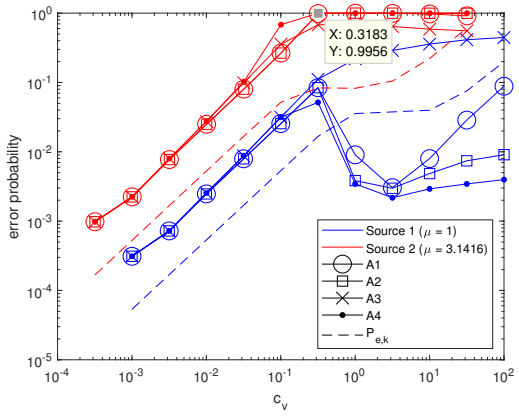
Additionally, this result finally illuminates why we do not see ratio of 5 times the lower bound in Figure 2.2.11b. For the simulation, we take mean pairs at random; the specific means used were recorded and are shown in Figure 2.2.18. Though there are many points where $\mu_1 \neq \mu_2$, there are still few points along the diagonal $\mu_1 = \mu_2$. While the average P_e/P^0_e for majority of the mean pairs is ≈ 5, the few along diagonal are also included in the final average reported which skews the number higher.



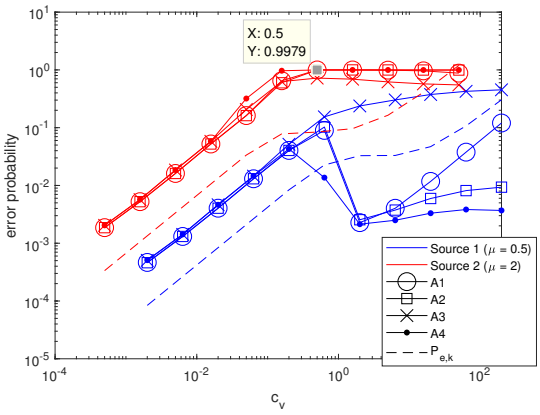
(a) Per source error rate

(b) Total error rate

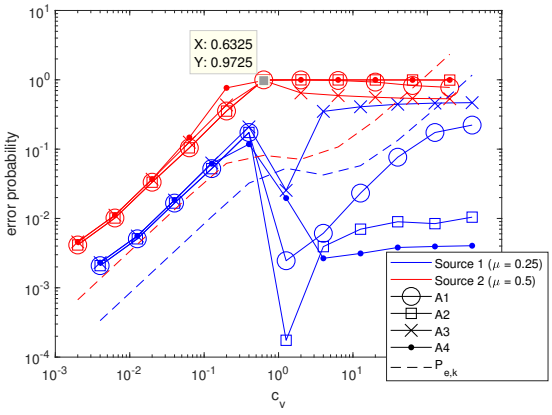
Figure 2.2.15: Empirical and theoretical error rates for $\mu_1 = \mu_2 = 10$ and c_v as shown. Both per source and total error rate are shown.



(a) $\mu_1 = 1, \mu_2 = \pi$; Note this is Figure 2.2.13a just re-plotted for c_v instead of σ_i



(b) $\mu_1 = 0.5, \mu_2 = 2$



(c) $\mu_1 = 0.25, \mu_2 = 0.5$

Figure 2.2.16: These are more examples of $P_{e,i}$ plotted for different mean pairs.

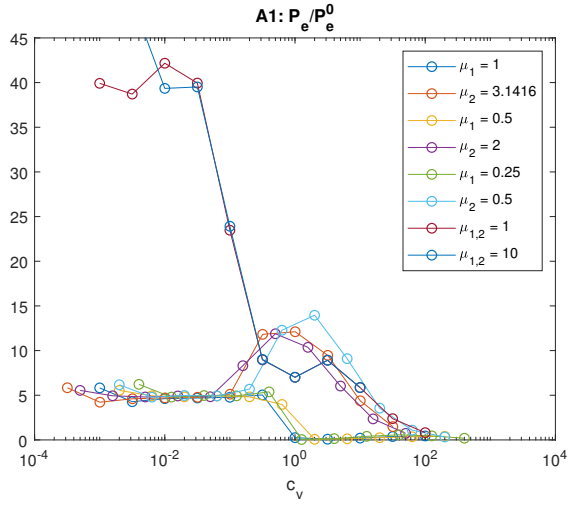


Figure 2.2.17: This is the ratio of actual error rate to the theoretical lower bound for A2.1 for mean pairs and c_v as shown.

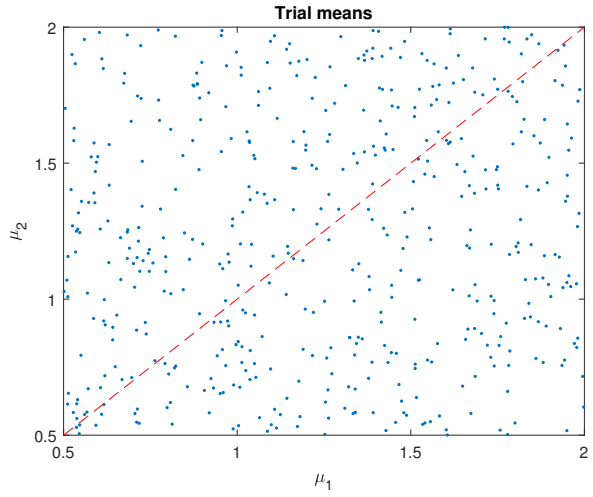


Figure 2.2.18: These are the 500 mean pairs randomly selected to generate the impulse trains used to get the results in Figure 2.2.11 for $K = 2$. The red dotted line indicates $\mu_1 = \mu_2$.

2.2.5 Practical normalization for likelihood or log-likelihood

At each step of A2.1,2.2,2.3,2.4 it is necessary to compute each path's or extension's likelihood or log-likelihood so we can compare and decide which to keep. However, if there are many points, then there is a high possibility of running into numerical representation problems during implementation. To this end, in practice, it is prudent to normalize at each step. To not distract from the theoretical aspects of the method, we omitted this discussion in the algorithm descriptions as this is only an implementation issue.

Let $x_1, x_2, \ldots$ denote events (e.g. interimpulse spacings) which we want to test. The joint likelihood of all the events, which we will denote here with a $p(\cdot)$, under our assumption of independence is

$$p(\{x_1, x_2, x_3, \ldots\}) = p(x_1)p(x_2)p(x_3)\cdots .$$

At each step we compare the running likelihood of each path. That is we start comparing

$$\begin{matrix} p(x_1^1) \\ p(x_1^2) \\ p(x_1^3) \\ \vdots \end{matrix}$$

where x_i^j indicates the j-th possibility for the i-th event (e.g. assign the i-th impulse to the j-th source). In the next step we have

$$\begin{matrix} p(x_1^1)p(x_2^1) \\ p(x_1^2)p(x_2^2) \\ p(x_1^3)p(x_3^3) \\ \vdots \end{matrix}$$

and so on. In the end we have something like

$$\begin{matrix} \ell_1 = \prod_{i=1}^{M} p(x_i^1) \\ \ell_2 = \prod_{i=1}^{M} p(x_i^2) \\ \ell_3 = \prod_{i=1}^{M} p(x_i^3) \\ \vdots \end{matrix}$$

Note, the relationship between items will not change if we divide each by some constant. For example, if

$$\ell_1 < \ell_2,$$

then

$$\frac{\ell_1}{C} < \frac{\ell_2}{C}$$

for any positive constant C. The same is true at each step. That is, we can divide by c_1 in the first step, c_2 in the second step, and so on. Then in the end $C = \prod c_j$. To avoid numerical representation trouble, we choose $c_j = \max_k p(x_j^k)$. This normalizes the best path to 1.0 at each step, this prevents the values from converging towards 0 or ∞ as the number of data points increase.

Another common way to prevent numerical problems is to instead consider the logarithm of the above since for small numbers the range is widened, e.g. $(0, 1] \mapsto (-\infty, 0]$.[5] Additionally, the log is a monotonic increasing

[5] It has the opposite effect for larger numbers (i.e. > 1)– it compresses the range. This is clearly seen in a plot, but you can also see it analytically in the derivative. Consider $f(x) = x$ and $g(x) = \log(x)$. $f(x)$ is linear, and $f'(x) = 1$. In comparison, $g'(x) = \frac{1}{x}$. When $x < 1$, $g'(x) > 1$; the slope is large, so the spread between numbers is widened. On the other hand when $x > 1$, $g'(x) < 1$; the slope is small, so the difference between numbers is lessened. This second fact is why the logarithm is used to visualize data when there is large disparity between magnitudes of data points.

function comparative relationships will be preserved. However, sometimes taking the log by itself is still insufficient to prevent all numerical issues. In this case, rather than a precision error, we are dealing with an overflow or underflow. As the log of multiplication/division is addition/subtraction, the exact same argument can be used: adding the same arbitrary constant at each step to every path will not change the comparative result. Any constant can be used, but one choice would be to shift the minimum/maximum value to 0 at each step ($\log(1) = 0$).

Even with these modifications, the likelihood of every path can still turn out to be 0 rendering any future decisions random. This may occur either because we are using one of the approximate algorithms and they fail, or the $\mathcal{T}_k$ do no match the data (e.g. in Section 2.3, the parameters estimated are wrong).

2.3 Unknown parameters

So far we have assumed that the distributions $\mathcal{T}_k$ are known. We now extend this to the case where we still know the total number of sources K, but $\mathcal{T}_k$ depends on an unknown parameter vector $\boldsymbol{\theta}_k$ (e.g. $\boldsymbol{\theta}_k = [\mu_k, \sigma_k^2]$ where μ_k and σ_k^2 are the mean and variance of the distribution). We update (2.2.1) to reflect this change

$$L(\mathbf{D}, \boldsymbol{\Theta}) = \sum_{k=1}^{K} \sum_{i} \log \mathcal{T}_{k,\hat{\boldsymbol{\theta}}_k}(\tau_{S_k(i)} - \tau_{S_k(i-1)}). \tag{2.3.1}$$

This is a mixed optimization problem where are trying to find the assignment $\mathbf{D}$ which is discrete and parameters $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_K]$ which are continuous. It is straightforward to optimize for $\boldsymbol{\Theta}$ if given the timing assignment $\mathbf{D}$ using standard ML estimation:

$$\hat{\boldsymbol{\theta}}_k = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{i} \log \mathcal{T}_{k,\boldsymbol{\theta}}(\tau_{S_k(i)} - \tau_{S_k(i-1)}), \tag{2.3.2}$$

is computed separately for each distribution $\mathcal{T}_k$. For example, for $\mathcal{T}_k \sim \mathcal{N}(\mu_k, \sigma_k^2)$ we have

$$\begin{aligned} \hat{\mu}_k &= \frac{1}{N_k} \sum_{i=1}^{N_k} x_i \\ \hat{\sigma}_k^2 &= \frac{1}{N_k} \sum_{i} (x_i - \hat{\mu}_k)^2, \end{aligned} \tag{2.3.3}$$

where $x_i = \tau_{S_k(i)} - \tau_{S_k(i-1)}$ (the interimpulse spacing) and N_k is the number of x_i for source k. However, for $\mathcal{T}_k \sim \mathcal{N}_0(\mu_k, \sigma_k^2)$, $E[X] \neq \mu_k$ and $E[(X - E[X])^2] \neq \sigma_k^2$. There is no closed from solution for the ML estimate of (μ_k, σ_k^2) for truncated Gaussians $\mathcal{N}_0(\mu_k, \sigma_k^2)$, but for $\sigma_k \ll \mu_k$ the solution (2.3.3) is close to the exact ML estimate. For more discussion of truncated Gaussians, see [35] and Appendix B.1.

In light of this, one might consider a brute-force approach: For each of the K^M possible assignment schemes, compute $\hat{\boldsymbol{\Theta}}$, and then compute L with the parameters. Then $\hat{\mathbf{D}}, \hat{\boldsymbol{\Theta}} = \operatorname{argmax} L$. However, as before, this is infeasible due to computational complexity; $O(K^M)$ is too much.

Another approach is to cleverly explore a subset of the K^M solutions by alternating optimization of $\mathbf{D}$ and $\boldsymbol{\Theta}$. This is the general idea of two methods presented in the following sections.

2.3.1 Alternating maximization

The idea behind alternating maximization is to alternate between between a discrete ML and a continuous ML. Starting with some $\boldsymbol{\Theta}^{[0]}$, run an algorithm from Section 2.2 on the data and get an assignment $\mathbf{D}^{[0]}$ which corresponds to $L(\boldsymbol{\Theta}^{[0]}, \mathbf{D}^{[0]})$. Next use (2.3.2) to get a new solution $\boldsymbol{\Theta}^{[1]} = \hat{\boldsymbol{\Theta}}$, and then again an algorithm from

Section 2.2 to get $\mathbf{D}^{[1]}$ and so on. For each step we have $L(\mathbf{\Theta}^{[i+1]}, \mathbf{D}^{[i+1]}) \geq L(\mathbf{\Theta}^{[i]}, \mathbf{D}^{[i]})$, and in principle we can keep repeating this process until it converges (i.e. the assignments/parameters no longer change). This process is outlined in Algorithm 2.6, which we will refer to as AM.

Algorithm 2.6 (AM) Alternating maximization timing separation algorithm

Given the observed impulse times $\tau_i, i = 1, \ldots, M$ that we want to assign to K sources

1. Choose some initial parameters $\mathbf{\Theta}^{[0]}$. Set $t = 0$.
2. Assignment: For iteration t, assign each impulse to maximize overall likelihood by using an algorithm from Section 2.2 with $\mathbf{\Theta}^{[t]}$ (i.e. find $\mathbf{D}^{[t]}$)
3. Update: Given the new assignment, compute $\mathbf{\Theta}^{[t+1]}$ using (approximate) ML estimation on $\mathbf{D}^{[t]}$.
4. If $\mathbf{D}^{[t]} = \mathbf{D}^{[t-1]}$ (i.e. assignment does not change) or maximum number of iterations reached, then done.

An issue with AM is that we are only guaranteed convergence to some local maximum. This is because the incremental improvements made are only within the neighborhood of the current solution. The process is similar to gradient ascent/descent methods where steps are taken in what looks like the best direction from current location; the global optimum can only be guaranteed if there is only one extremum and no local extrema to get stuck on. To avoid converging to a poor solution, care must to be taken with respect to the choice of $\mathbf{\Theta}^{[0]}$ which is discussed in Sec. 2.3.3.

In the ideal case we always have $L(\mathbf{\Theta}^{[i+1]}, \mathbf{D}^{[i+1]}) \geq L(\mathbf{\Theta}^{[i]}, \mathbf{D}^{[i]})$. However, in many applications we might use an approximate algorithms, A2.3 or A2.4, due to the high computational cost of the alternatives. There are times when these algorithms do not maximize the likelihood because the corresponding path was pruned at some point. Furthermore, at the parameter estimation step, we might also use an approximation such as (2.3.3) for a truncated Gaussian. As a result, the likelihood may not always increase in each step. Though a decrease may occur, the end result is typically an increase in L, but other times it *cycling* happens. That is, the assignments/parameters get repeated. For example, consider $\mathbf{D}^{[i]}$ generates $\mathbf{\Theta}^{[i+1]}$, and $\mathbf{\Theta}^{[i+1]} \rightarrow \mathbf{D}^{[i+1]}$. Then $\mathbf{D}^{[i+1]}$ generates $\mathbf{\Theta}^{[i+2]}$, and $\mathbf{\Theta}^{[i+2]} \rightarrow \mathbf{D}^{[i+2]}$. This becomes a two iteration *cycle* if $\mathbf{D}^{[i+2]} \equiv \mathbf{D}^{[i]}$. Longer cycles are possible as well. As far as we can tell, it is not possible to predict or avoid these cycles (when we are using approximations). Therefore, to avoid an infinite loop, we enforce a maximum number of iterations (MI) constraint. For the application to sperm whale clicks, a MI of 50 appears to suffice. We opt for this method rather than simply terminating when the objective function starts to decrease to allow better exploration of the solution space. Also, we keep track of the assignment that yielded the highest likelihood, so the result is no worse than if we had terminated.

2.3.2 Expectation maximization approach

A more sophisticated version of alternating optimization is Expectation Maximization (EM). Specifically, we modify the Expectation Maximization Viterbi (EMV) algorithm presented in [36, 37] to suit the timing separation problem. The following discussion requires some familiarity with the EM algorithm, e.g. [38]. In this section, we use ℓ for likelihood and L for its logarithm.

To start, instead of (2.3.1), consider the complete log-likelihood function $L_{\mathbf{x}}(\mathbf{x}|\mathbf{\Theta})$. Here $\mathbf{x} = (\mathbf{y}, \mathbf{D})$ is the complete data set, where $\mathbf{y} = \{\tau_i\}_{i=1}^{M}$, the set of impulse times, are the observations, and $\mathbf{D}$, the assignment of the impulse times, is the unobservable/missing data (as defined in the EM approach). In this section, we make it a point to denote exactly what random quantity each (log-)likelihood function is for via a subscript to avoid confusion. It is not possible to evaluate and maximize $L_{\mathbf{X}}(\mathbf{x}|\mathbf{\Theta})$ directly since it involves the unobservable $\mathbf{D}$, so instead we maximize the conditional expectation given $\mathbf{y}$ and some parameters $\mathbf{\Theta}^{[t]}$

$$\mathcal{Q}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{[t]}) \triangleq E[L_{\mathbf{X}}(\mathbf{x}|\boldsymbol{\Theta})|\mathbf{y},\boldsymbol{\Theta}^{[t]}], \tag{2.3.4}$$

where $.^{[t]}$ denotes the value at the t-th iteration of the algorithm which will be discussed later. As discussed in [38], improvements in (2.3.4) correspond to improvements in $L_{\mathbf{X}}(\mathbf{x}|\boldsymbol{\Theta})$. Thus if we make iterative improvements to (2.3.4), like in the AM approach, we will at least reach a locally jointly optimum solution for the parameters and assignment. In terms of the timing problem, (2.3.4) becomes

$$\begin{aligned}\mathcal{Q}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{[t]}) =& E[L_{\mathbf{y}|\mathbf{D}}(\mathbf{y}|\mathbf{D},\boldsymbol{\Theta})|\mathbf{y},\boldsymbol{\Theta}^{[t]}] + \underbrace{E[L_{\mathbf{D}}(\mathbf{D}|\boldsymbol{\Theta})|\mathbf{y},\boldsymbol{\Theta}^{[t]}]}_{c} \\ =& \sum_{u=1}^{K^M} \Pr\{\mathbf{D}=\mathbf{D}^u|\mathbf{y},\boldsymbol{\Theta}^{[t]}\} L_{\mathbf{y}|\mathbf{D}}(\mathbf{y}|\mathbf{D}^u,\boldsymbol{\Theta}^{[t]}) + c\end{aligned} \tag{2.3.5}$$

where $\mathbf{D}^u$ denotes the u-th assignment scheme.[6] We replace $E[L_{\mathbf{D}}(\mathbf{D}|\boldsymbol{\Theta})|\mathbf{y},\boldsymbol{\Theta}^{[t]}]$ with a constant c because the assignment $\mathbf{D}$ is independent from the parameter values $\boldsymbol{\Theta}$ when there are no observations. That is, let $\Pr\{\mathbf{D}^u\} = p_u$ then

$$\begin{aligned}E[\log \ell_{\mathbf{D}}(\mathbf{D}|\boldsymbol{\Theta})|\mathbf{y},\boldsymbol{\Theta}^{[t]}] &= E[\log \Pr\{\mathbf{D}\}] \\ &= \sum_u p_u \log p_u \triangleq c\end{aligned}$$

is still a constant in the respect that it does not depend on $\boldsymbol{\Theta}$ or $\mathbf{y}$ (in fact it is the entropy of $\Pr\{\mathbf{D}^u\}$). As such it can be ignored it in subsequent optimization steps. Immediately, we notice a problem with (2.3.5); it requires the computation of $\Pr\{\mathbf{D}=\mathbf{D}^u|\mathbf{y},\boldsymbol{\Theta}^{[t]}\}$ for every possible assignment which is generally computationally infeasible. In order to have any hope of actually computing this value, we make an approximation based on the observation that if $\mathbf{D}^u$ are *labeled in order of descending probability* (given $\mathbf{y},\boldsymbol{\Theta}$), then for some value $U \ll K^M$

$$\sum_{u=1}^{U} \Pr\{\mathbf{D}=\mathbf{D}^u|\mathbf{y},\boldsymbol{\Theta}^{[t]}\} \approx 1.$$

That is, any contribution to the expectation from the $\mathbf{D}^{U+1},\ldots,\mathbf{D}^{K^M}$ is negligible, so we ignore them and just calculate (2.3.5) for the top U assignments (later we will discuss how to determine them). These probabilities do not exactly sum to 1, so we correct this by normalizing over the U paths. This is the Viterbi adjustment to the typical EM algorithm [36, 37]. For assignment D^u use

$$P_u = \frac{\Pr\{\mathbf{D}=\mathbf{D}^u|\mathbf{y},\boldsymbol{\Theta}^{[t]}\}}{\sum_{v=1}^{U}\Pr\{\mathbf{D}=\mathbf{D}^v|\mathbf{y},\boldsymbol{\Theta}^{[t]}\}} \tag{2.3.6}$$

$$= \frac{\ell_{\mathbf{y}|\mathbf{D}}(\mathbf{y}|\mathbf{D}^u,\boldsymbol{\Theta}^{[t]})}{\sum_{v=1}^{U}\ell_{\mathbf{y}|\mathbf{D}}(\mathbf{y}|\mathbf{D}^v,\boldsymbol{\Theta}^{[t]})} \tag{2.3.7}$$

as a substitute for $\Pr\{\mathbf{D}=\mathbf{D}^u|\mathbf{y},\boldsymbol{\Theta}^{[t]}\}$; (2.3.6) follows from (2.3.7) using Bayes rule. Note, if $U=K^M$, then (2.3.7) is exact. The adjusted objective function is

$$\mathcal{Q}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{[t]}) = \left(\sum_{u=1}^{U} P_u L_{\mathbf{y}|\mathbf{D}}(\mathbf{y}|\mathbf{D},\boldsymbol{\Theta})|\mathbf{y},\boldsymbol{\Theta}^{[t]})\right).$$

[6] Recall $E[f(X)] = \sum_x f(x)\Pr\{x\}$ where X is a discrete random variable.

This is the *expectation* step or *E-step* of the EM algorithm.

In the *maximization* part or *M-step* of the algorithm, we want to find a new parameter set $\mathbf{\Theta}$ that improves the above $\mathcal{Q}$. That is, find

$$\mathbf{\Theta}^{[t+1]} = \arg\max_{\mathbf{\Theta}} \mathcal{Q}(\mathbf{\Theta}|\mathbf{\Theta}^{[t]}). \tag{2.3.8}$$

We will next solve (2.3.8) for truncated Gaussians. First, ignore the truncation correction; specifically, assume $\mathcal{T}_k \sim \mathcal{N}(\mu_k, \sigma_k^2)$. Then solving $\nabla\mathcal{Q} = 0$ gives

$$\hat{\mu}_k = \frac{\sum_{u=1}^{U} \left(P_u \sum_i d_{k(i)}(u)\right)}{\sum_{u=1}^{U} P_u n_k(u)} \tag{2.3.9}$$

$$\hat{\sigma}_k^2 = \frac{\sum_{u=1}^{U} P_u \sum_i (d_{k(i)}(u) - \hat{\mu}_k)^2}{\sum_{u=1}^{U} P_u n_k(u)}, \tag{2.3.10}$$

where $d_{k(i)}(u)$ denotes the i-th interimpulse interval for source k in the u-th solution path and $n_k(u)$ denotes the number of interimpulse intervals for the k-th source in the u-th solution path. This is the typical M-step for a collection of Gaussian sources [38]. The truncated distribution can be derived from the non-truncated distribution. That is, if $\mathcal{T}_k \sim \mathcal{N}_0(\mu_k, \sigma_k^2)$, then

$$\mathcal{T}_k(t) = \frac{f_k(t)}{\Pr(t > 0)} = \frac{f_k(t)}{Q(-\frac{\mu_k}{\sigma_k})},$$

where $f_k \sim \mathcal{N}(\mu_k, \sigma_k^2)$, the distribution with no truncation, and Q (not to be confused with $\mathcal{Q}$) is the is the complementary CDF the standard Gaussian. Then

$$\begin{aligned}
\mathcal{Q}(\mathbf{\Theta}|\mathbf{\Theta}^{[t]}) &= \sum_{u=1}^{U} P_u \sum_{k=1}^{K} \sum_i \log \mathcal{T}_k(\underbrace{\tau_{S_k(i)} - \tau_{S_k(i-1)}}_{d_{k(i)}}) \\
&= \sum_u P_u \sum_{k=1}^{K} \sum_i \log \frac{f_k(d_{k(i)}(u))}{Q(\frac{-\mu_k}{\sigma_k})} \\
&= -\sum_u P_u \sum_{k=1}^{K} n_k(u) \log Q(-\frac{\mu_k}{\sigma_k}) + \sum_u P_u \sum_{k=1}^{K} \sum_i \log f_k(d_{k(i)}(u); \mu_k, \sigma_k).
\end{aligned} \tag{2.3.11}$$

Note, the second term is the $\mathcal{Q}$ if we did not consider any truncation correction. And if $\mu_k \gg \sigma_k$ or μ_k is sufficiently large, then $-\frac{\mu_k}{\sigma_k}$ is a large negative number (assuming $\mu_k > 0$; it would be odd if this were not the case) which makes $Q(-\frac{\mu_k}{\sigma_k}) \approx 1$ (Q of a large negative number is essentially asking, "what is the probability of the entire distribution?"). So when $\mu_k \gg \sigma_k$ or μ_k is large for all sources, any contribution from the first term is negligible since $\log(1) = 0$. Thus in many applications using (2.3.9) and (2.3.10) will suffice. But if this is not the case, a numerical solution can be found since the objective function is differentiable: for each source start at (2.3.9) and (2.3.10) and then use gradient ascent on (2.3.11).

Choosing the top U paths is an important part of this process. The top U paths should be the ones with the largest P_u. In (2.3.7) the denominator is a nominal scaling factor; the relative size of each $\mathbf{D}^u$ is determined by the the numerator

$$\ell_{\mathbf{y}|\mathbf{D}}(\mathbf{y}|\mathbf{D}^u, \mathbf{\Theta}^{[t]}) = e^L$$

where L here refers to (2.3.1). In each of the algorithms discussed in Section 2.2, this is value we use to pick the output path out of the final collection of surviving paths. It is natural to use this final collection of surviving paths as the top U paths. In the case of the approximate algorithms, only the top a paths are maintained, so $U = a$. For A2.1 and A2.2, there will be more paths, but a considerable amount will have low likelihood. Since we multiply the

path variables (e.g. $n_k(u)$) by P_u (and $\ell_{\mathbf{y}|\mathbf{D}} \propto P_u$), the algorithm naturally disregards this information. However, we can save on few computations by ignoring those paths with P_u under some threshold in the subsequent M-step (e.g. computation of (2.3.9) and (2.3.10)).

Similar to AM, there is still the problem of cycling which arises from the approximations made, so again an MI is enforced (50 also seems to work here). The entire EMV methodology for the timing separation problem is outlined in Algorithm 2.7; in other parts of the paper EMV will refer to this (rather than the generic EMV algorithm).

Algorithm 2.7 EMV timing separation algorithm

Given the observed impulse times $\tau_i, i = 1, \ldots, M$ that we want to assign to K sources

1. Choose some initial parameters $\mathbf{\Theta}^{[0]}$. Set $t = 0$.
2. E-step/Assignment: Use a timing separation algorithm from Section 2.2 with $\mathbf{\Theta}^{[t]}$. The top U paths are collection (or subset) of paths the output of timing separation algorithm picks the final output assignment. Compute P_u for each using (2.3.7).
3. M-step/Update: Given the survivor paths and their P_u, compute $\mathbf{\Theta}^{[t+1]}$ using (2.3.8).
4. If $\mathbf{D}^{[t]} = \mathbf{D}^{[t-1]}$ or maximum number of iterations reached, then done. Otherwise, go to step 2.
   (a) Choose the final output $\mathbf{D}^{[t]}$ to be the assignment scheme from the top survivor path (i.e. the one with the largest P_u).

The difference between the AM algorithm and the EMV algorithm is the use of (2.3.2) versus (2.3.8). In the AM algorithm, the parameter assignment in the next steps is based on only the most likely path, whereas in the EMV algorithm, it is determined by the U most likely paths. As a consequence, each step in the EMV algorithm is slightly more complex which may provide more accurate and/or faster convergence.

2.3.3 Initialization

Both the EMV and AM algorithms are quite dependent on a good initial guess of the parameters $\mathbf{\Theta}^{[0]}$. Our intuition is that if we choose $\mathbf{\Theta}^{[0]}$ close to the actual $\mathbf{\Theta}$ values, then the unknown parameter algorithms are likely to converge to it. So now the problem is to get an estimate for the actual $\mathbf{\Theta}$ from the data. This is actually, a large topic so we discuss it in Chapter 3.

However, a key point is that instead of finding a single best estimate for $\mathbf{\Theta}$, we aim at generating multiple estimates of $\mathbf{\Theta}$. Then for each estimate apply AM or EMV and choose the assignment that results in the largest (2.2.1). In this manner, at the cost of running the algorithm multiple times, we increase our chances of coming up with a better result. This is easily paralleled, the result from each $\mathbf{\Theta}$ can be computed independently.

Furthermore, the methods discussed in Chapter 3 can be very accurate, so sometimes it is unnecessary to even use AM or EMV. Therefore, another algorithm is to consider the multiple estimates of $\mathbf{\Theta}$ as described above, but instead of running AM or EMV, just run a timing separation algorithm (e.g. A2.3). As a subtle point, for the comparison at the end, (2.2.1) is computed using the ML $\mathbf{\Theta}$ from the resulting assignment (e.g. (2.3.3)) rather than the $\mathbf{\Theta}$ used to generate the assignment. We will refer to this method as the *best guess* or $\mathbf{\Theta}$ *guessed*.

The general process for getting an output from multiple $\mathbf{\Theta}$ is outlined in Algorithm 2.8.

Algorithm 2.8 Timing separation starting with multiple parameter sets

Given the observed impulse times $\tau_i, i = 1, \ldots, M$ that we want to assign

1. Based on the data generate $\mathcal{K}$ parameter sets: $\{\mathbf{\Theta}_1^{[0]}, \mathbf{\Theta}_2^{[0]}, \ldots, \mathbf{\Theta}_{\mathcal{K}}^{[0]}\}$ using some method (e.g. Algorithm 3.2)
2. For each $\mathbf{\Theta}_i^{[0]}$, run a timing separation algorithm (this can be paralleled):
   (a) Best guess: A2.1, A2.2, A2.3, or A2.4
   (b) AM use any of (a) internally
   (c) EMV use any of (a) internally
3. For the $\mathcal{K}$ parameters, record the output assignment and compute (2.2.1). Pick the final output as the assignment that corresponds to the largest (2.2.1).

2.3.4 Performance with unknown parameters

In the same fashion as Section 2.2.2, we check the algorithms for unknown parameters. Specifically, we will look at use A2.2 in AM and EMV with parameter initialization through Algorithm 3.2. As in the second half of Section 2.2.2, we generate a total of 100 impulses from two synthetic sources with mean impulse spacings 1-100 and standard deviations of 0.1 and report the error averaged over 100 trials. Again, we use P_e as a performance metric. However, here source numbering is arbitrary for all outputs, so we use the minimum P_e over all permutations of labelings. We compare the error rates when the parameters are estimated (AM, EMV, and best guess) with the error rates when the actual parameters are used (Section 2.2.2).

The results are summarized in Figure 2.3.1. Figure 2.3.1a directly compares all of the different cases by mapping $(\mu_1, \mu_2) \mapsto \mu_1 - \mu_2$ and averaging the error rate along like indices for ease of interpretation. To be specific, A2.2 with known parameters is the best at $\bar{P}_e = 0.0025067$, while for AM and it is $\bar{P}_e = 0.0093941$ and $\bar{P}_e = 0.0089238$ respectively, which are both slightly better than best guess which has $\bar{P}_e = 0.011988$. As should be expected, the algorithm that uses the actual $\mathbf{\Theta}$ does better than when we try to estimate $\mathbf{\Theta}$ from the data. Fortunately, the average error rate when it is estimated is still low. Also, as we hoped, AM and EMV show improvement over the best guess, but the amount is so small that it is not a convincing argument to use AM or EMV. However, this is partially because the parameter initialization is generally doing a very good job of estimating the actual parameters, so there is not much improvement to be made. And then when the initialization gives a poor estimate, it is too far off such that AM and EMV cannot recover. To understand this more, consider Figure 2.3.1b.

Figure 2.3.1b gives a more detailed view of the error rates for AM, but the ones for EMV and the best guess are very similar. In comparison with Figure 2.2.3, the main difference is elevated error on the edges. More specifically, the is error is higher approximately where $\mu_1 < 1/2\mu_2$ up to $\mu_1 \approx 40$ and tapering down from its peak till $\mu_1 \approx 80$. Source numbering is arbitrary, so we also see the same behavior reflected across the $\mu_1 = \mu_2$ line (i.e. $1 \leftrightarrow 2$). For simplicity, in the following we will assume $\mu_1 < \mu_2$, but the discussion also applies to $\mu_2 > \mu_1$. Upon closer inspection, the cause of this error is due suboptimal $\mathbf{\Theta}$ candidate generations. For example, in a simulation with $\mathbf{\Theta} = \{(9, 0.1^2), (29, 0.1^2)\}$, Algorithm 3.2 lets us pick two distributions from:

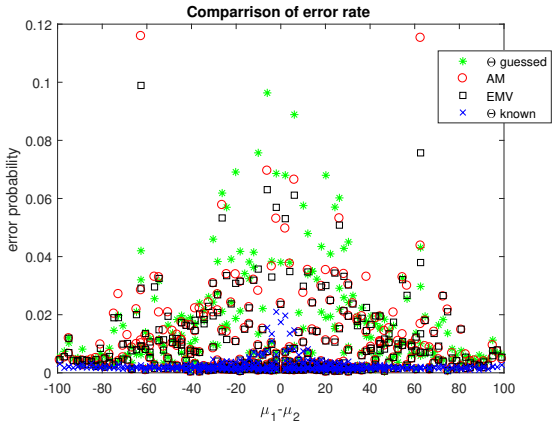
$$(90, 1.0991^2), (99, 1.1090^2), (67, 1.2500^2), (18, 0.6801^2), (9, 0.5865^2).$$

Only the last choice, $(9, 0.5865^2)$, is close to an underlying distribution. Thus, we do not converge to the proper parameters or assignments using AM, EMV, or the best guess. For all algorithms, the error rate is 0.11.

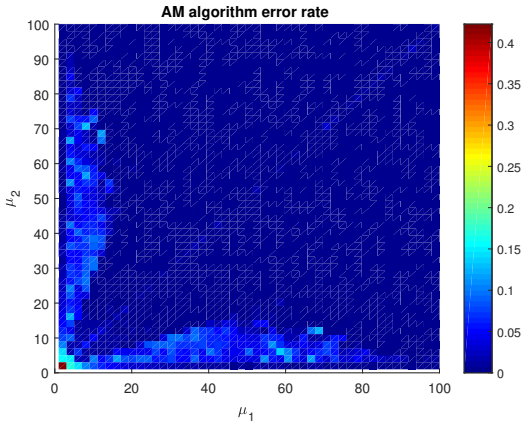
This outcome is based on the mechanics of Algorithm 3.2, so refer to Section 3.2 to understand the following discussion. When $\mu_i \approx 2\mu_j$ there will be the additive mixing of μ_i's peak in the histogram with the first subharmonic of μ_j. In efforts to still identify μ_i, we came up with the protocol of also picking the K largest peaks above

the threshold (3.2.3). However, it seems like this is not always good enough when $(\mu_1, \mu_2) < (40, 20)$. Potentially, this is because both sources will have multiple sub-harmonics before $T_{\max} = 100$. There will be 7 or more peaks of which we pick at most 6. The criteria we use may not be sufficient to discern among this many peaks. There is a drop off from this behavior for $\mu_1 > 40$ which can be attributed to a drop in the number of sub-harmonics from the first source.

This result could mean that either AM and EMV are not effective or we need a better initialization method. There are many ways to generate $\mathbf{\Theta}^{[0]}$ as discussed in Section 3. The results here are just from a single method, and though it proved better than sequential search, Algorithm 3.1, (results not shown) there is still room for improvement in the areas mentioned above. These improvements could be made with further heuristics, or we could consider a different method entirely. In particular, the PRI transform described in Section 3.3 is of interest.



(a) Parameter estimation algorithms compared with best parameter guess and actual parameter performance



(b) AM performance. The plot for EMV performance is very similar.

Figure 2.3.1: Total error probability of AM and EMV (using A2.2) for two sources with sperm whale like impulse timing.

With regards to the effectiveness of AM and EMV, consider another simulation removing the influence of parameter initialization. We consider a data set with 100 impulses and $\mathbf{\Theta} = \{(20, 0.1^2), (50, 0.1^2)\}$ and then check the outputs of A2.2, AM, and EMV when $\mathbf{\Theta}^{[0]} = \{(20, 1.0^2), (\tilde{\mu}_2, 1.0^2)\})$ for $\tilde{\mu}_2 \in [1, 100]$. For comparison, we also compute the error rate for A2.2 given $\mathbf{\Theta} = \{(20, 0.1^2), (50, 0.1^2)\}$. This is repeated 100 times for each choice of $\tilde{\mu}_2$ and the results are summarized in Figure 2.3.2. As we have seen multiple times already, when given the actual parameters, the performance is good at an average of 0.0024. There is negligible variation between the different $\tilde{\mu}_2$ since we give the actual parameters for every trial and choice of $\tilde{\mu}_2$. Looking at the other algorithms' results, we can see when the parameter estimation breaks down. As expected, all algorithms, do well when $\tilde{\mu}_2 = \mu_2$, but they also approach the known $\mathbf{\Theta}$ rate for some window around $\tilde{\mu}_2 = \mu_2$. Using the best guess, this window is about $-5 < \mu_2 - \tilde{\mu}_2 < 3$; outside of this region, the error is high. Using AM and EMV, we are still able to match known parameter performance the beyond this to $-15 < \mu_2 - \tilde{\mu}_2 < 12$. This window is more than $3\times$ as wide as the best guess. Though this analysis is just for a specific parameter set, it shows that when the initial parameters are incorrect, AM and EMV will converge to the right solution more often than the best guess.

In terms of error rate, as noted above, EMV has $\bar{P}_e = 0.0089238$ whereas AM has $\bar{P}_e = 0.0093941$, so we might conclude that we should always use EMV. However, there is also a difference in the number of iterations required for convergence. Recall that MI is a constraint enforced to avoid cycling. So for a more thorough investigation, we repeat the same simulation used to generate Figure 2.3.1, but instead of 50 MI only, we try MI of 20, 50, 100, 200, and

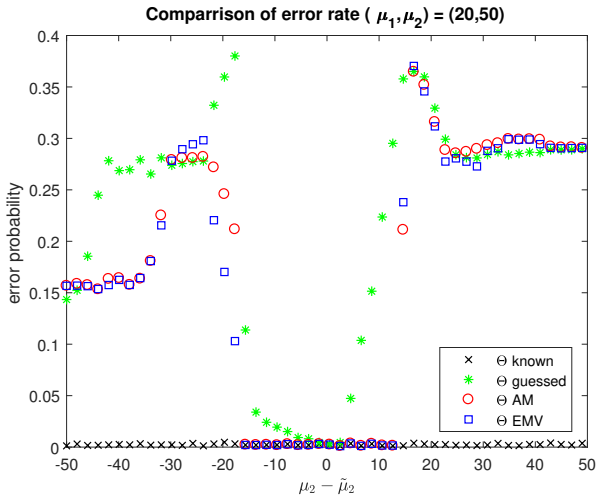


Figure 2.3.2: For data generated with $\mathbf{\Theta} = \{(20, 0.1^2), (50, 0.1^2)\}$, we choose $\mathbf{\Theta}^{[0]} = \{(20, 1.0), (\tilde{\mu}_2, 1.0)\})$ for $\tilde{\mu}_2 \in [1, 100]$a. With these $\mathbf{\Theta}^{[0]}$, error rates are for the parameter estimation algorithms compared with best guess and actual parameter performance and averaged over 100 trials.

500 and count the number of iterations required to converge for each trial. The average number of iterations per each MI over the entire simulation space are reported in Table 2.3.1. The fact that the average number of iterations increases (though very slightly in the case of AM) with increasing MI probably indicates that for some parameter values cycling occurs. AM consistently requires less iterations than EMV, so we recommend using AM when speed is of greatest importance. Interestingly, the error rates for AM and EMV do not change regardless of MI (the exact values are listed at the beginning of this paragraph). Thus, when accuracy is the priority, we recommend EMV. Additionally, this is further evidence that cycling is likely occurring, since although the number of iterations used is larger, the error rate remains unchanged. This would also indicate that 20 MI would be sufficient to converge to the best solution, however, we recommend 50 MI to err on the side of caution.

Table 2.3.1: Average number of iterations used for A2.2 over $(\mu_1, \mu_2) \in [1, 100]^2$, $\sigma = 0.1$

MI	AM	EMV
20	17.4241	18.4211
50	17.4266	19.5757
100	17.4308	21.4983
200	17.4392	25.3435
500	17.4644	36.8791

To show that these algorithms are not limited to separating only two interleaved pulse trains, we also consider $K \in \{2, 3, 4, 5\}$. The K means are picked at random form $[1, 100]$ and again $\sigma = 0.1$. As before, the interleaved pulse trains are generated, however, this time, $M = 1000$. The AM and EMV algorithms are run with A2.3, and P_e is calculated. For each K, this is repeated 100 times (new means generated each time), and the P_e are averaged and presented in Table 2.3.2. Note this time Algorithm 3.2* is used with $\sigma_k^{[0]} = 1.0$. For all algorithms, as K increases, the error rate also increases. In general, there is more opportunity for error when K is higher. For example, the only type mistake that can be made with $K = 2$ is $1 \leftrightarrow 2$ (i.e. putting a 1 where there should be a 2 or vice versa).

When $K = 3$, the types of mistakes are $1 \leftrightarrow 2$, $2 \leftrightarrow 3$, and $1 \leftrightarrow 3$. This is supported by the fact that when the actual $\mathbf{\Theta}$ is given, the error rate approximately doubles for every increment of K.

Table 2.3.2: Average error rates for A2.3 and A3.2* with different numbers of sources with $\mu_k \in [1, 100]$, $\sigma = 0.1$

K	2	3	4	5
Known	0.0022	0.0044	0.0085	0.0171
Guess	0.0325	0.2257	0.4121	0.4993
AM	0.0306	0.0566	0.1005	0.1355
EMV	0.0307	0.1427	0.3919	0.4538

A second key point here is that for $K > 2$, the AM greatly out performs the best guess. What this means is that A3.2* is not generating estimates close enough to the actual parameters for a straight best guess, but AM is able to converge to the right solution from the poor parameters. Like the results shown in Figure 2.3.2, this is an argument for using AM. EMV is better than the guess but only marginally so. EMV is suffering from a similar problem to the best guess, which is contradictory to what we saw in Figure 2.3.2.

To better understand the problem consider we have a single source with $\mathcal{T} \sim \mathcal{N}(12, 0.1^2)$, but we try to model it with $\hat{\mathcal{T}} \sim \mathcal{N}(0.5, 0.1^2)$. Suppose $\tau_2 - \tau_1 = 12$, then

$$\hat{\mathcal{T}}(12) = \frac{1}{\sqrt{2\pi(0.1)^2}} \exp(-\frac{(12-0.5)^2}{2(0.1)^2}) \approx 0 \Rightarrow \log(\hat{\mathcal{T}}(12)) = -\infty.$$

Since we either multiply likelihood or add log-likelihood of subsequent interimpulse intervals, the total computed likelihood would be 0. The same will hold for multiple sources if the given $\mathbf{\Theta}$ is a grave mismatch for every source. Likelihood of all survivor paths will also be 0, this poses a problem for the computation of P_u (2.3.7)

$$P_u = \frac{0}{\sum 0} =?$$

It is not well defined what the algorithm should do in this case, however we originally adopted the convention that it should stop trying to converge to a solution and just output the top path[7]. As seen in the results, this does not seem to be the best choice. For the results EMV results in Table 2.3.2, this situation arose 17, 59, and 64 times for $K = 3, 4, 5$ respectively which directly corresponds to the increased error. However, this situation is detectable and AM does not run into the same issue, so when we see that the likelihood of all survivor paths is 0, we can default to AM. Results for this modified EMV algorithm are shown in Table 2.3.3. The trend is much more similar to AM except that the rate for $K = 5$ is higher.

Table 2.3.3: Average error rates for modified EMV with the same setup as Table 2.3.2

K	2	3	4	5
EMV	0.0162	0.0471	0.1178	0.2147

Another alternative is to use a better initialization algorithm to avoid bad $\mathbf{\Theta}^{[0]}$. This is the basis for the more current A3.2. Table 2.3.4 shows some results using the newer method (the experimental setup is the same as before). Note, the known results are slightly different than the previous table since it is from a different dataset, however, they are similar enough. As intended, the best guess error rates are much better than the old initialization. However, AM even with the inferior A3.2* is still better than best guess with the updated version. We would expect that AM and EMV would be better as well with the new method.

[7] Since all paths have the same likelihood, the single top or best path could be any of them. We use the first path that is output from the standard `sort` function in MATLAB

Table 2.3.4: Average error rates for A2.4 and A3.2 with different numbers of sources with $\mu_k \in [1, 100]$, $\sigma = 0.1$

K	2	3	4	5
Known	0.0018	0.0046	0.0076	0.0310
Guess	0.0173	0.0611	0.1211	0.2405

In Section 3.3.4, we present a different initialization method in Algorithm 3.6. A key benefit of this method is that it also outputs the number of sources, so we delay discussion of its performance until Section 2.4.1.

2.4 Source number estimation

We next extend our methods to operate when both the number of sources K and $\mathbf{\Theta}$ is unknown. In fact, in some applications, knowing the number of sources is as useful as the classification of impulses. For example, PAM single sensor population density estimation methods, which currently rely on assumptions regarding vocalization rates, would benefit from a more direct way of counting animals [39].

A standard solution for this type of problem is minimum description length (MDL) [40, 41, 31, 42]. For the observations $\mathbf{y}$, define $\mathcal{M} = (K, \mathbf{\Theta}, \mathbf{D})$ as a model. The MDL is the minimum amount of information (e.g. bits) needed to describe the data using the model. There are many approaches to MDL [41], but in most cases an asymptotic expression in the number of data points M is given by [40]

$$MDL(\mathcal{M}) = -L(\mathbf{y}|\mathbf{D}, \mathbf{\Theta}) + \frac{1}{2}|\mathbf{\Theta}| \log M, \tag{2.4.1}$$

where the first term is (2.3.1) and $|\mathbf{\Theta}|$ denotes the number of parameters $\mathbf{\Theta}$ contains which is $2K$ for Gaussians. Note, $\mathbf{\Theta}$ and $\mathbf{D}$ should be the best estimates from one of the methods in Section 2.3 for a given K, so $\mathcal{M}$ is essentially determined just by the number of sources K. The model with the lowest MDL can be thought of as the most efficient or optimal in some sense. Thus we choose the model/number of sources corresponding to the lowest MDL, that is

$$\hat{\mathcal{M}} = (\hat{K}, \hat{\mathbf{\Theta}}, \hat{\mathbf{D}}) = \arg \min_{\mathcal{M} \in \mathbb{M}} MDL(\mathcal{M}),$$

where $\mathbb{M}$ is the set of all models. In theory, there can be models for $K = 1, 2, \ldots, M$, but sometimes, there is some a priori knowledge of the maximum number of sources $K_{\max} < M$. In such cases, to save on computation time, it makes sense to test only up to $K_{\max}$ (similarly if there is a minimum there is no need to test below it).

There are other alternatives, to MDL. For example, we can allow AM or EMV to decides how many sources it needs. That is, try to run AM/EMV with $K_{\max}$; the resulting assignment may or may not use all the sources allotted. The number of sources it actually uses can be taken as $\hat{K}$. We will call this the *cluster number* method. To see why this might work, consider if the true $\mathbf{\Theta} = \{(30.1^2), (70, 0.1^2)\}$, but we give $\mathbf{\Theta}^{[0]} = \{30, 0.1^2), (70, 0.1^2), (100, 0.1^2)\}$. It is likely then that the third source given will be ignored and unused. While such a $\mathbf{\Theta}$ and $\mathbf{\Theta}^{[0]}$ is a possibility, it should be noted that none of our algorithms were designed with this cluster number method in mind. A number of things have to go well for this to work out, namely: How does the initialization (Section 3) behave when given more than the actual number of sources? However, since we will compute $\mathcal{M}$ for $K_{\max}$ in the process of optimizing via MDL, it is easy to test cluster number, and for comparative purposes, the results may be interesting.

Additionally, section 3.3.4 describes a parameter estimation method that outputs a $\tilde{\Theta}$ from which we can draw (potentially) multiple $\mathbf{\Theta}^{[0]}$ from. Each $\mathbf{\Theta}^{[0]}$ drawn will imply a number of sources K. This K is based on the data, so it may be larger than the true of sources. Therefore, like with cluster number it is important to keep check how many sources are actually used in the final classification. Though $\tilde{\Theta}$ gives estimates for K directly, the MDL principle is still used to judge the models generated by the $\mathbf{\Theta}^{[0]}$ since each will have a different K. That is, for $\{\mathbf{\Theta}^{[0]}\}$ from $\tilde{\Theta}$ we apply Algorithm 2.8 but use the negative of (2.4.1) instead of likelihood in step 3.

2.4.1 Performance of MDL

To assess the performance of cluster number and MDL, we consider datasets with $K = 1, 2, 3, 4, 5$, and then try to find $\hat{K}$ with $K_{\max} = 5$. Specifically, for each of the values of K, we generate 100 different datasets that each have 1000 impulses. The means of each source are randomly selected from $[1, 100]$ and $\sigma = 0.1$ is set. (The setup is identical to what was used for Table 2.3.2 except for M) In addition to finding a $\hat{K}$ for each trial, we also compute the classification error rate at $\hat{K}$ and the actual K. Here we use sequential search (Algorithm 3.1) with $\sigma_k^{[0]} = 1.3$ for $\mathbf{\Theta}^{[0]}$ generation and A2.3 inside of AM and EMV. Results are summarized in Table 2.4.1 and Table 2.4.2.

The source error rate is defined as

$$E_K = \frac{\sum I(\hat{K} = K)}{R},$$

where R is the total number of trials. Table 2.4.1, reports this value for each K for each algorithm and the average across all values of K as well (indicated as AVG in the table). For cluster number, at $K = 5$, we get what we wanted: given the option to use $K_{\max}$ sources when there are indeed $K_{\max}$ sources and it always uses $K_{\max}$. However, with the exception of $K = 1$, the majority of the time $\hat{K} = K_{\max}$ which results in high error rates. For $K = 1$, $K_{\max}$ is not chosen most often, but neither does $\hat{K} = 1$ occur with great certainty. Due to its poor performance, we do not recommend cluster number for source number estimation. Except for $K = 5$, MDL is always better than cluster number. And within MDL, E_K generally increases with K, though for EMV at $K = 5$, it curiously decreases from $K = 4$. An increase with increasing K is expected. Recall for Table 2.3.2, we discussed how the possibility for error his higher when K is larger, and more errors typically correspond to a smaller (2.3.1) which reduces the effectiveness/correctness of MDL. It is difficult to say whether AM or EMV is better with MDL looking at each value of K, but considering the average error rate across all K, EMV has a slight edge.

Table 2.4.1: Source number error rates E_K

$K =$	1	2	3	4	5	AVG
AM: cluster number	0.23	0.82	0.89	0.87	0	0.5620
EMV: cluster number	0.62	0.85	0.93	0.90	0	0.6600
AM: MDL	0	0.05	0.11	0.29	0.25	0.1400
EMV: MDL	0	0.05	0.13	0.24	0.16	0.1160

Table 2.4.2 reports average P_e at the K and $\hat{K}$ over the trials for the indicated K. Again, AVG indicates the average across all K. The results here are only for MDL, sinceE_K was very high for cluster number. Interestingly, the error at $\hat{K}$ is always lower than the error at the actual K. This would seem to indicate that even when we arrive at the wrong $\hat{K}$, the number of mistakes is lower than if we had used the proper K. One cause of this is that there may be a few outlier or problem impulse times. By assigning those impulses to auxiliary sources, the non-outlier impulses can be assigned without trouble. To support this theory, with the exception of $K = 5$, most of the $\hat{K}$ errors are overestimates. In the case of $K = K_{\max}$, the only possible error is an underestimate, but even in this case, we see the same behavior, classification error is lower at $\hat{K}$ than K. It may be that a source gets hidden within another. Consider $\mu_1 = \mu_2$ but the impulse times are at $\frac{1}{2}\mu_1 k$; the second source started emitting at exactly halfway between the first two impulses from the first source. Then the data can be modeled as a single source with $\mu = \frac{1}{2}\mu_1$. Another possibility is that there are very few of one or more sources, in which case, there is little to gain from actually modeling which is also reflected in a low error. For example, if first source has 100 impulses, but the second only has 3, the and drop in likelihood for not modeling the second source is low.

Furthermore, the assignment error rate is within a reasonable range for all values of K. Note, that the results are different than Table 2.3.2 because of the initialization method, but the trend is similar. More importantly, the error rate is low even when E_K is elevated (i.e. $K = 4, 5$). What this means is that in the case of over or under estimation, only a few impulse are falsely assigned. It is possible that upon closer inspection that these errors could be corrected, for example by eliminating sources with only a few impulses assigned.

We also repeated the same test using A2.4 instead of A2.3, but have omitted the specific results for simplicity since the general trends are the same. However, the error is consistently higher for both the $\hat{K}$ and $\hat{\mathbf{D}}$ using A2.4 (e.g. for AM, the average $E_K = 0.14$ and $P_e = 0.0410$ at $\hat{K}$ using A2.3, whereas using A2.4, the average $E_K = 0.24$ and $P_e = 0.0553$ at $\hat{K}$). Though A2.4 is much better than A2.3 in terms of run time; in our simulations, A2.4 finished the entire test about twice as fast as A2.3. Recall the difference between A2.3 and A2.4 is that when the best path seems to be converging to a low-likelihood solution (i.e. a path with a STO), A2.4 will stop at that impulse time and restart assigning sources from the next impulse, whereas A2.3 instead just prunes any paths with STO

Table 2.4.2: Classification error rates P_e for MDL

$K=$	1	2	3	4	5	AVG
AM at $\hat{K}$	0	0.0061	0.0156	0.0526	0.1305	0.0410
EM at $\hat{K}$	0	0.0086	0.0159	0.0579	0.1275	0.0420
AM at K	0	0.0142	0.0338	0.0907	0.1655	0.0608
EM at K	0	0.0114	0.0343	0.0818	0.1399	0.0535

and continues. To some degree, it is expected that A2.3 will take longer than A2.4 since it checks every path for STO instead of only the best one. The elevated error in A2.4 (which was not seen in Figure 2.2.2) is likely the trade-off it pays for the increased speed.

We test the method of Section 3.3.4 in a similar way: We consider datasets with $K = 1, 2, 3, 4, 5$ where the means of each source are randomly selected from the range $[0.5, 3]$ and $\sigma = 0.1$ is set. For this method's development, we were more interested in practical results for sperm whales, so here we test on appropriate parameters. Each dataset is generated to have $M = 100$ impulses, and for each K we consider 100 different trials. For computation of complex correlation $\rho(s)$ and thresholds we use:

- $\tau_j - \tau_{j-1} = 0.1$; the distance between centers of interimpulse bins
- $b = 0.1$; the width of the interimpulse bins
- $[0.01, 0.5]$ is the range allowed for estimates of σ
- $P_{fa} = 0.05$; the probability of falsely detecting noise
- $\beta = 0.15$; user defined scale factor for $r(s)$, regular correlation, threshold

We classify each using AM with A2.3 at $a = k^3$, where use k do denote the number of sources in $\mathbf{\Theta}^{[0]}$ pulled from $\tilde{\Theta}$. And as mentioned previously, MDL is used to determine which is the best resulting classification from $\{\mathbf{\Theta}^{[0]}\}$. Here we denote the number of sources used in the assignment output as $\hat{K}$, and we report the values in Table 2.4.3. A blank entry indicates 0, and the green entries highlight $\hat{K} = K$. Note when $\hat{K} = 0$ (denoted with red text), it means that our method decided that there were no sources with a distinguishable μ. As discussed in Algorithm 3.6, we only make this determination after exhausting the reduction of β. This happens almost only when $K = 1$ (it happens twice when $K = 2$). We are unsure as to why this happens, but readjustment of the other tuning parameters (e.g. P_{fa} or b) may help. Though examples we check at the same parameters, we note that the peak corresponding to μ in $r(s)$ is prominent. But considering when we do detect the presence of a source, we note that $\hat{K} = K$ is always the majority, though over all the E_K is not as good as the MDL method (Table 2.4.1). However, it is better that the cluster number method. There is room for improvement.

Like with the MDL results, we also look at the average classification error rate P_e as shown in Table 2.4.4. Since the source number estimation is not as good as MDL, it makes sense that P_e is also higher here than in Table 2.4.2. Even when we isolate $\hat{K} = K$ to remove the influence of MDL, the results here are still worse than before. However, we need to keep in mind the c_v for this simulation is higher based on the range of means considered. Another interesting way to view these results is shown in Figure 2.4.1, the red circles indicate when $\hat{K} = K$. Typically the red dots are at low P_e, and there are more of them when the number of sources are lower.

Overall, Section 3.3.4 initialization does not appear to be as good as MDL. As already mentioned, this is partly due to the c_v being higher. This method has the benefit that it outputs estimates for K, making it much faster than regular MDL. Perhaps the drop in performance is the trade-off between speed and accuracy. At least, these results show that the method has some potential to be used with further modifications.

Table 2.4.3: Source number estimates $\hat{K}$ for Section 3.3.4 initialization

$K =$	1	2	3	4	5
$\hat{K} = 0$	22	2			
1	**78**	19	5		
2		**72**	27	6	1
3		6	**50**	12	2
4			9	**38**	6
5		1	6	20	**31**
6			1	12	27
7			1	6	18
8			1	4	10
9				2	5
E_K	0.22	0.28	0.5	0.62	0.69

Table 2.4.4: Average classification error rate P_e for Section 3.3.4 initialization

$K =$	1	2	3	4	5	AVG
P_e	0	0.1663	0.2979	0.3479	0.3940	0.2412
P_e when $\hat{K} = K$	0	0.0839	0.1418	0.2168	0.2452	0.1375
$\lvert\{\hat{K} = K\}\rvert$	78	72	50	38	31	53.8

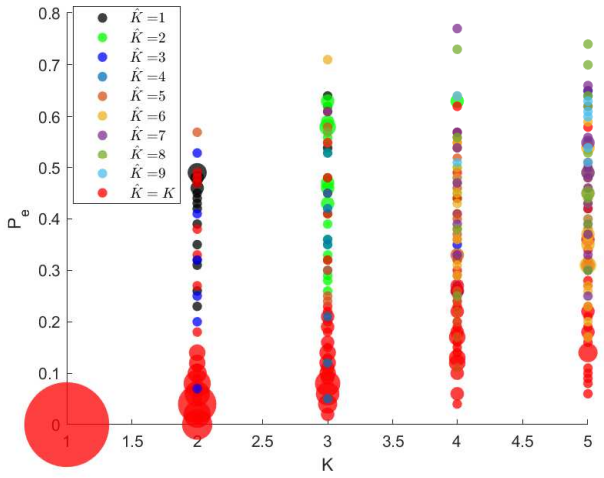


Figure 2.4.1: This is the error rate for the different trials using Section 3.3.4 initialization. The location of the circle indicates K and P_e for the trial. The size of the circle is proportional to the number of trials at the same location. For reference, a single trial is the size of the circle in the legend. Colors indicate $\hat{K}$ for the trial as indicated in the legend.

2.5 Applications

In the previous sections, we have only presented results from simulated data. Here we would like to provide a few examples (mentioned in Chapter 1) to demonstrate that the algorithms are effective on real world problems and data.

2.5.1 Neurons

It is often of interest to extract the activity from a single neuron. This is not possible without extracting a single neuron from the organism or using very specialized equipment. Even then, such recordings are often not practical or useful due to the specialized setup. Instead, extracellular methods can be (e.g. an EEG) in which a single electrode picks up all local activity from which individual spikes or action potentials can be identified. The process of identifying the activity from signal neurons out of the local mixture is referred to as *spike sorting* [11]. The firing rate of neurons is how information is communicated between cells. Thus, we hypothesize that the time between action potentials can be used to sort the spikes. To show that this is possible, we took two intracellular recordings (from [43]) and extracted the times of of the spikes using a Taeger-Kaiser-based threshold detector (see [44]). Then we combined the times to create a realistic mixture of extracellular recorded spike times. Specifically, the first neuron has $\theta = (0.17, 0.02^2)$, the second has $\theta = (0.45, 0.02^2)$. The timing mixture of 16 spikes spanning 2 seconds is shown in Figure 2.5.1. We apply A2.2 given these parameters and achieve the correct assignment. Note that had we used an actual extracellular measurement, we would have no way of knowing if the resulting classification was correct or not.

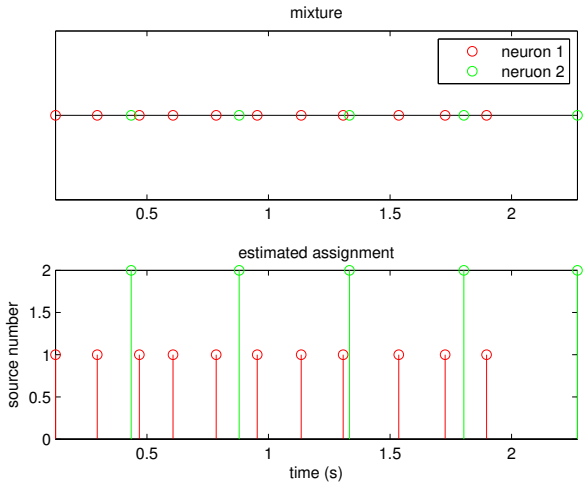


Figure 2.5.1: (top) A simulated mixture was created using recorded action potentials. (bottom) The assignment output from A2.2 is exact.

2.5.2 Sperm whales

Recall, separating odontocetes' click trains is motivation and intended application for this project. Click train ICI are relatively stable over short periods of time, so they are a good candidate for separation by timing. We will explore actual click train separation more thoroughly in Chapter 6, but here we present a small example.

To get a realistic data set for which we had ground truth (which is difficult to obtain in reality) we tested the algorithm on recordings of a single sperm whale (from [45]) where we mixed the timings (extracted from the waveform using techniques outlined in [46]) from three different segments. Specifically, the segments with $\mathbf{\Theta} = \{(1.3940, 0.1285^2), (0.8814, 0.0708^2), (0.9212, 0.0624^2)\}$ combined to form a mixture of 116 clicks spanning roughly 40s shown in Figure 2.5.2. We use the $\delta\tau$-histogram (Algorithm 3.2) with $T_{\max} = 2.5$ and a $\gamma = 0.1$; then we apply AM and EMV (the results are identical) using A2.3. We find there is 8.6% error in the resulting assignment (10 errors out of 116 clicks). Upon closer inspection, the errors are where two clicks very close to each other their assignments get swapped. This is a reasonable outcome, recall in Section 2.2.2 it was discussed how near simultaneous emissions can lead to errors.

As a follow up, we also tried A2.2 given the actual $\mathbf{\Theta}$. Again the resulting assignment had 8.6% error (10 errors out of 116 clicks); this is exactly the same error rate as for when we estimated the parameters. While the errors were not located at the same clicks, they were of the same type. That is, they were when two clicks were nearly simultaneous. This result is reassuring, it means our parameter estimation algorithm works.

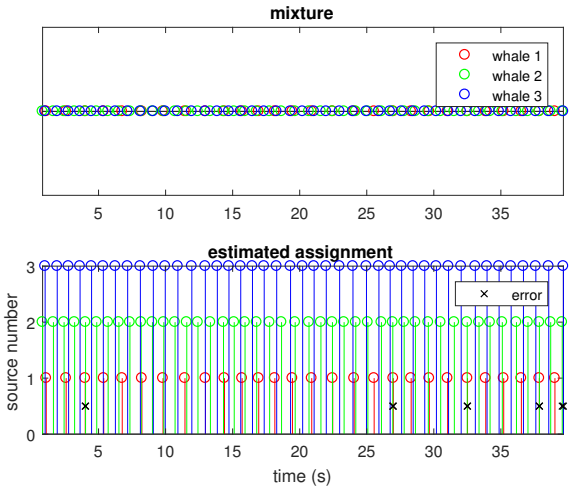


Figure 2.5.2: (top) A simulated mixture was created using recorded sperm whale clicks. (bottom) If the $\delta\tau$-histogram is used for initialization, then assignment output from with AM and EMV (the assignments are identical) using A2.3 has 10 errors which are indicated with a black "x".

2.5.3 Radar

Unlike the previous two examples, radar pulses and timings are completely generated by humans, therefore if the right parameters are chosen, any simulation is just as good as a real world recording. Though the choice of PRI and PRI modulation will depend on the application, PRI is reported in [9] on the order from 1 μs with jitter within 8% of the PRI. We emulate this by generating 3 sources with $\mathcal{T}_k \sim \mathcal{N}_0(\mu_k, \sigma_k)$ and

$$\mathbf{\Theta} = \{(50, 0.80^2), (72, 0.40^2), (84, 0.04^2)\},$$

where the values are in μs and μs^2. Enough of each source was generated so that when combined, a mixture of 1000 pulses was created.

To show the effectiveness of MDL (Section 2.4), we consider $\hat{K} = 1, 2, 3, 4, 5$ and generate models using the $\delta\tau$-histogram (Algorithm 3.2 with $T_{\max} = 100$ and a $\gamma = 1$) and AM with A2.3. We find that $\hat{K} = 3$ gives the best

MDL and the lowest error as shown in Table 2.5.1. More importantly, if we look at the source parameters at $\hat{K} = 3$ according to the output assignment at (i.e. calculating ML parameters from the assignment) we get

$$\hat{\mathbf{\Theta}} = \{(50.0243, 0.7742), (72.0138, 0.3858^2), (84.0002, 0.0399^2)\}.$$

So in addition to detecting the correct number of sources and getting a low classification error rate, we are able to estimate the parameters within a small margin which. In this sense, the algorithm performs well too.

Table 2.5.1: MDL results for radar example

$K =$	1	2	**3**	4	5
MDL	4111.9	2662.1	**209.8**	379.8	500.9
P_e	0.9990	0.5630	**0.0040**	0.2210	0.3710

3 Parameter estimation

As mentioned in Section 2.3.3, our timing separation algorithms require knowledge of $\boldsymbol{\Theta}$ (under the assumption that we at least know the type of $\mathcal{T}_k$) or at least some starting value. Parameter estimation for impulsive sources is of special interest in Electronic Intelligence (ELINT) where the goal is to secretly determine information of radar systems based only on the signals they output (like PAM) [5]. This is particularly useful during warfare when opposing parties would like to find out what the other is doing with their radars. For example, if the pulse repetition interval (i.e. interimpulse spacing), PRI, for a radar system is known, the maximum range, R_u, and velocity, V_u, that system can measure can be determined as

$$R_u = \frac{PRI}{2}c, \qquad V_u = \frac{PRI(RF)}{2}c,$$

where c is the speed of light (3×10^8 m/s), and RF is the radar carrier frequency in Hz [5].

In this chapter, we will highlight the methods that we have found to be useful for Algorithm 2.8. A difference from the original formulations is be that we will output multiple $\boldsymbol{\Theta}$ rather than just a single best guess. In particular, the $\boldsymbol{\Theta}$ we will extract will be $\{(\mu_k, \sigma_k)\}$ under the assumption of underlying Gaussian timing distributions, but with some modifications, other parameters can be extracted if needed. Additionally, in this chapter unless otherwise stated, we will assume that the number of sources K is known. However, some of the methods will not require K and instead output an estimate for K.

3.1 Sequential search

In the ELINT literature, a sequential search is, given some candidate PRI, going through the impulse times and removing/assigning all points that are associated with the candidate. This process is repeated for all candidate PRI or until all points are assigned. Often, this is the final step to validate the PRI candidates generated from some routine [5, 47, 6]. Our sequential search differs from the typical method in that we also generate the PRI candidates while we are doing the search. The theory of sequential search is based on assumption of constant PRI modulation; that is, the interimpulse spacings within each source is exactly μ_k. When this is not the case, then there are various heuristic adjustments that can be made. But as we will show, the more the variation, the poorer the performance.

Our sequential search is based on the assumption that c_v is low; that is, there is not a lot of variation, so the mean is still a very good descriptor of the data. When this is the case, each interimpulse spacing is close to the mean, and a reasonable estimate for a mean is an actual interimpulse spacing from the data. This is the basis of how we will generate the PRI candidates.

Without loss of generality assume that the first impulse τ_1 is from source 1, then potential interimpulse spacings/mean estimates are $\mu_{1,1} = \tau_2 - \tau_1, \mu_{1,2} = \tau_3 - \tau_1, \mu_{1,3} = \tau_4 - \tau_1, \ldots$ where $\mu_{k,j}$ denotes the j-th potential mean for source k. To narrow down the list of potential candidates, assume further that a maximum value for interimpulse spacings $T_{\max}$ is known, so only consider the interimpulse spacings $\mu_{1,j} \leq T_{\max}$. Following the previous notation in Section 2.2, $T_{\max} = \max(\{T_k\})$. Furthermore if $\mu_{1,j}$ is a good estimate, then we would expect that there would exist impulse times $s_n \approx \tau_1 + n\mu_{1,j}$ where $n = 2, 3, \ldots$. The theoretical impulse times s_n can be compared with the actual impulse times and extracted if they are within some tolerance. The tolerance should be related to the average variance, $\bar{\sigma}^2$. These extracted impulses can then be evaluated using a dummy likelihood function

$$L_j(\mu_{k,j}) = \sum_{\ell=1} \log \tilde{\mathcal{T}}_k(\tau_\ell - \tau_{\ell-1}),$$

where we will assume that $\tilde{\mathcal{T}}_k$ is $\mathcal{N}_0(\mu_{k,j}, \bar{\sigma}^2)$. However, this likelihood function only evaluates the fit of the impulses extracted, and the quality of the extraction should also be accounted for. As an example, suppose that we compute that there should be ten s_n for the length of the signal (recall, that if a source has μ_k, there is approximately T/μ_k impulses in time T), but only two are found/match. The value of $L_i(\mu_{k,j})$ may be high, but $\mu_{k,j}$ does not fit the data well, so we also consider an extraction completion ratio (ECR)

$$ECR_j = \frac{\text{number of matching impulses}}{\lfloor T/\mu_{1,j} \rfloor + 1},$$

where 1 is added to the denominator to count the impulse at time 0. The complete mean candidate score (MCS) is given by

$$MCS_j = \frac{L_j}{\max_i L_i} + ECR_j. \tag{3.1.1}$$

We normalize L_j by $\max L_i$ so that it is on the same scale as the ECR so the contributions are balanced (i.e. $L_j/\max L_j \in [0, 1]$ and $ECR \in [0, 1]$). The best candidate is the one with the largest MCS, so it should be chosen as the candidate μ_1. To find a candidate for μ_2, remove all the impulse times associated with the first candidate and repeat the process until K candidates are found. Alternatively, if K is unknown, the process can be repeated until all impulses have been assigned. It is hard to say which method is better, but ideally, both would occur at the same time (i.e. find K candidates with all the impulses assigned). Overall performance will depend on the data and the tuning of tolerance by the user. However, if K is known, using that information should lead to a better result.

To understand the complexity of this method, note, when we do a sequential search for $\mu_{1,j}$, we need to compare with the remaining $M - j$ impulses. If there is no $T_{\max}$, then there are $M - 1$ mean candidates to check. So it is an $O(M^2)$ operation to get a single mean candidate. We need to do this K times, and though the number of impulses we need to compare with will decrease with each mean candidate selection, we can still upper bound the complexity as $O(KM^2)$ in the case when K is unknown and we run until every impulse has been assigned). The take away is that this method is dependent on the number of impulses it has to search, so computational complexity can be reduced by only considering a portion of the signal. In particular, we choose the time period $[0, 3T_{\max}]$. This range guarantees that there are at least 3 interimpulse spacings per source (under the assumption that all sources start emitting within $[0, T_{\max}]$ and do not stop til the end of the recording). A pair of impulses is needed to form an interimpulse interval, and a third impulse will allow for validation of the interval. If the signal is not $3T_{\max}$ long, the sequential search can still be used, but you can only validate means up to $T/3$.

So far, we have only discussed how to find candidate means, but for a complete $\boldsymbol{\Theta}$ we also need estimates for σ_k. One option is that since we actually end up with impulses assigned to sources at the end, to use ML estimation

(2.3.3) for σ_k (or any other parameter you might need). However, since we only consider a small portion of the signal, there may only be a few samples making (2.3.3) inaccurate. Instead, recall we began with the assumption that $\sigma_k \ll \mu_k$ which means we have some notion of what σ_k is. For this Gaussian distributions, the variance affects the scale and the width. If it is overestimated, the distribution essentially becomes a distance measures from the means, and impulses will be assigned to groups of similar interimpulse spacing (like in k-means clustering). Thus there is little consequence if the variance is overestimated, so long as we do not flatten the distribution completely. Therefore we pick σ_k to be $\approx 10\bar{\sigma}$, where $\bar{\sigma} = \frac{1}{K}\sum_k \sigma_k$ or some approximation of this; in simulations, this has shown to give satisfactory results (discussed in Section 3.1.1).

This method is purely heuristic, so there is no guaranty that even if this process is followed, a suitable $\boldsymbol{\Theta}$ will be generated. To improve our chances, instead of just considering the mean candidate with the top score (3.1.1), consider b of them to generate multiple starting parameter sets. That is, there will be many potential μ_1, generate a $\boldsymbol{\Theta}$ for each of $\mu_{1,j}$ corresponding to the top b MCS_j. Through experimentation, we found $b = 3K$ to be a sufficient. The entire process for generating multiple $\boldsymbol{\Theta}$ is outlined in Algorithm 3.1.

Algorithm 3.1 Sequential search parameter initialization algorithm

Given the observed impulse times $\{\tau_i \in [0, 3T_{\max}]\}$ for K sources, to generate b $\boldsymbol{\Theta}$, starting with $k = 1$ do the following

1. Assume that τ_1 is from source k and compute the interimpulse spacing to the other impulses (i.e. $\tau_i - \tau_1$, $\forall i > 1$). Consider all interimpulse spacings $< T_{\max}$ as mean candidates $\{\mu_{k,j}\}$.
2. For each mean candidate $\mu_{k,j}$
   (a) Compute $\{s_n = \tau_1 + n\mu_{k,j}\}$, and
   (b) Starting from the first τ_n select the τ_i such that $|\tau_i - s_n| < \varepsilon$, where ε is some user-defined tolerance. If no τ_i can be found, stop and collect the values found into $\{\tau_i^*\}$.
      i. Compute $L_j(\mu_{k,j})$ on $\{\tau_i^*\}$ and $ECR_j = \frac{|\{\tau_n\}|}{|\{\tau_i^*\}|}$, where here $|\cdot|$ denotes the cardinality of the set.
3. Compute (3.1.1) for each candidate.
   (a) If at the actual first impulse, save the top b candidates as μ_k for each of the respective b starting points. Proceed to the next step with just one of the b starting points.
   (b) Otherwise, just save the top candidate as μ_k.
4. Remove $\{\tau_i^*\}$ corresponding to μ_k and increment k. Renumber the remaining impulses so that the first one is τ_1.
5. Go back to step 1.
   (a) Keep repeating until k reaches K, this will complete one $\boldsymbol{\Theta}^{[0]}$.
   (b) After a starting parameter set is completed, go on to complete the next set until all b are generated.

3.1.1 An example sequential search

Figure 3.1.1 shows a mixture of two impulse trains with $\mathcal{T}_k \sim \mathcal{N}(\mu_k, \sigma_k^2) = (10, 0.1^2)$ and $(30, 0.1^2)$. We will use these impulse times to illustrate the sequential search. The next step is to identify the candidates; Figure 3.1.2 shows the first few candidate μ_1 (there may actually be more than shown depending on $T_{\max}$). Then, we test the

first candidate $\mu_{1,1} = 1.77$ to see if there are any clicks at the multiples of it. As shown in Figure 3.1.3, we do not find any beyond the 2 impulses that generated $\mu_{1,1}$ so,

$$ECR_1 = \frac{2}{\left\lfloor \frac{61.74}{1.77} \right\rfloor + 1} = \frac{2}{35} = 0.06.$$

To be complete, we should compute the entire MCS_1, but already we see that ECR_1 is very low, which indicates of a poor match/candidate.

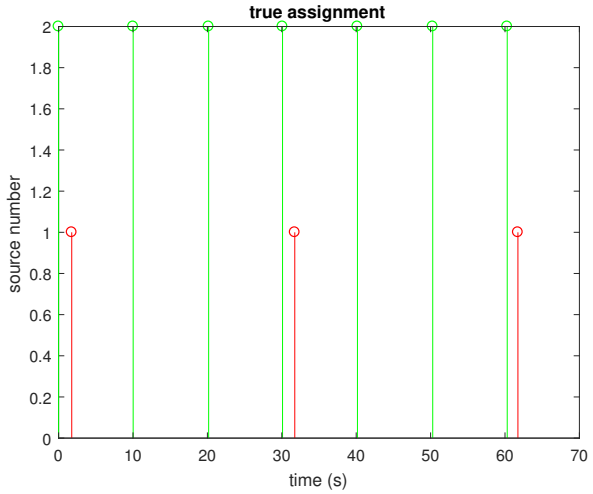


Figure 3.1.1: Impulse times from a sources with $(\mu_k, \sigma_k^2) = (10, 0.1^2)$ and $(30, 0.1^2)$.

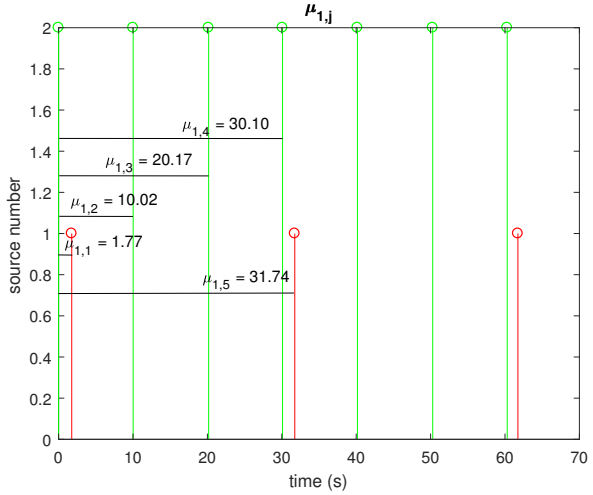


Figure 3.1.2: First 5 mean candidates identified out of the impulse times.

To contrast, we will do the same for the next candidate $\mu_{1,2} = 10.02$. The results of the check are shown in

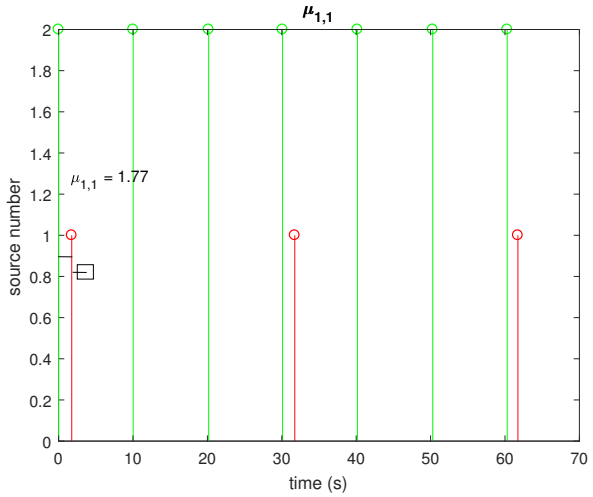


Figure 3.1.3: We check $k\mu_{1,1} \pm \varepsilon$ sequentially for $k = 2, 3, \ldots$ to see if there are any clicks corresponding to this mean. The values of $k\mu_{1,1} \pm \varepsilon$ are indicated by boxes. No match is found at $2\mu_{1,1}$, so we stop the search.

Figure 3.1.4. In this case we find 5 additional impulses that match $\mu_{1,2}$, so

$$ECR_2 = \frac{2+5}{\left\lfloor \frac{61.74}{10.02} \right\rfloor + 1} = \frac{7}{7} = 1.0.$$

This is the best ECR possible, so even without computing MCS_2, $\mu_{1,2}$ seems like much a better candidate than $\mu_{1,1}$. According to the algorithm, from here we would get the MCS_j for the other $\mu_{1,j}$. But when we come back to $\mu_{1,2}$, we would remove all the green pulses and just be left with the red ones to extract another value using the same process.

Based on these 10 impulses with $T_{\max} = 100, \varepsilon = 1$ and $K = 2$, the $3K$ sets of means output are

$$\mu_1, \mu_2 = \begin{cases} 10.02, & 29.98 \\ 20.17, & 29.98 \\ 30.10, & 29.98 \\ 50.29 & 29.98 \\ 60.30, & 29.98 \\ 61.74, & 58.53 \end{cases}$$

We see that the first choice is quite close to the actual means of $10, 30$; it is well within the range AM and EMV can safely operate in (Figure 2.3.2).

Through this example, to some degree, we have shown that this method has the potential to work. Since we do not need to find the means exactly for successful separation, measuring the difference from the actual parameters for each candidate set is only partially instructive. Instead, it is better to observe its performance in conjunction with the methods in Chapter 2. In particular, recall Table 2.4.2 where we used A2.3 and AM and EMV on data sets with $K = 1, 2, 3, 4, 5$, and we were able to to achieve low error rates.

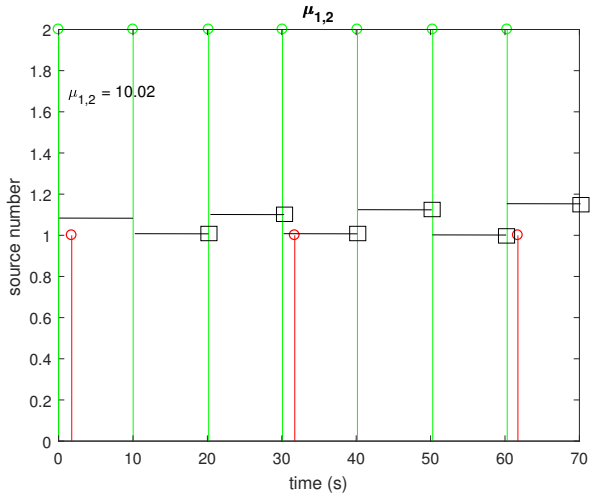


Figure 3.1.4: We check $k\mu_{1,2} \pm \varepsilon$ sequentially for $k = 2, 3, \ldots$ to see if there are any clicks corresponding to this mean. The values of $k\mu_{1,1} \pm \varepsilon$ are indicated by boxes. Several matches are found.

3.2 $\delta\tau$-histogram

Here we will describe, the $\delta\tau$-histogram [5] with some adjustments for our application/system model. For a set of impulse times of interleaved impulse trains, consider every possibly pairing and compute the interimpulse spacing for each. This is an $O(M^2)$ operation. In application, the maximum spacing is usually known, making it an $O(M)$ operation. Forming a histogram of interimpulse spacings, peaks emerge at the actual interimpulse spacings for each source (and their multiples). We can extract an estimate for $\boldsymbol{\Theta}$ from these histogram peaks.

Conveniently, it can be shown that this histogram is equivalent to the integral over the histogram bins of the autocorrelation of the impulse time function [5]. As a consequence of working on a digital system, we will form the autocorrelation on a sampled impulse time function and get the same result as if we computed the autocorrelation and then integrated. Specifically, for the set of impulse times $\{\tau_i\}_{i=1}^{M}$, the impulse time function is

$$f(t) = \sum_i \delta(t - \tau_i), \tag{3.2.1}$$

a signal with an impulse at each impulse time. Its autocorrelation is

$$\begin{aligned} r(s) &= \int f(t) f(t-s) dt \\ &= \int \sum_i \delta(t - \tau_i) \sum_j \delta(t - \tau_j - s) dt. \end{aligned}$$

The integrand is only non-zero when the difference between impulse times i, j is equal to the lag s

$$s = \tau_i - \tau_j.$$

So we can rewrite

$$r(s) = \sum_i \sum_j \delta((\tau_i - \tau_j) - s). \tag{3.2.2}$$

Each $r(s)$ is effectively a count of the number of pairs of impulse times with spacing s. If we define the start and end points of the n-th bin as (x_n, y_n), then we can write the histogram as

$$h[n] = \int_{x_n}^{y_n} r(s)ds.$$

The sampled version of $f(t)$ is defined as

$$f[n] = \int_{n\gamma}^{(n+1)\gamma} f(t),$$

where γ is the chosen sampling interval. That is, $f[n]$ is the number of impulses that lie in $[n\gamma, (n+1)\gamma]$. The corresponding autocorrelation is calculated using sums instead of integrals

$$r[s] = \sum_n f[n]f[n-s]$$

Note the n and the lag s here are sample indexes and not time. Multiplying the index by γ gives time. Since a binning operation was done when $f[n]$ was generated, a lag of s refers not only to a difference of $s\gamma$, but actually encompasses the range of differences $(s-1)\gamma$ to $(s+1)\gamma$. That is

$$r[n] = h[n] = \int_{(n-1)\gamma}^{(n+1)\gamma} r(s)ds.$$

In general, $r[s]$ should be computed for $s = 1, 2, \ldots, \lfloor T/\gamma \rfloor$. If a maximum possible interimpulse interval, $T_{\max}$, is known then $r[s]$ only needs to be computed up to the corresponding index. The same applies for a lower bound if a minimum possible interval is known. The choice of γ is an important topic discussed after the method is completely described.

A large histogram peak indicates that the interimpulse interval repeats many times. However, sub-harmonics (i.e. integer multiples of the actual pulse spacing) also show up as peaks, so picking the highest peaks for the μ_k is not always the best choice. Instead, we should pick peaks that are closest to the expected histogram height. In general, the height of a histogram is the total number of samples multiplied by the probability a sample falls within the range of a bin. Let $\mathcal{F}(\tau)$ describe the mixture of distributions. For example,

$$\mathcal{F}(\tau) = \alpha_1\mathcal{T}_1(\tau) + \alpha_2\mathcal{T}_2(\tau) + \cdots,$$

where α_k are the mixing proportions. Then the theoretical height of bin s would be

$$\Gamma(s) = M \int_{(s-1)]\gamma}^{(s+1)\gamma} \mathcal{F}(\tau)d\tau.$$

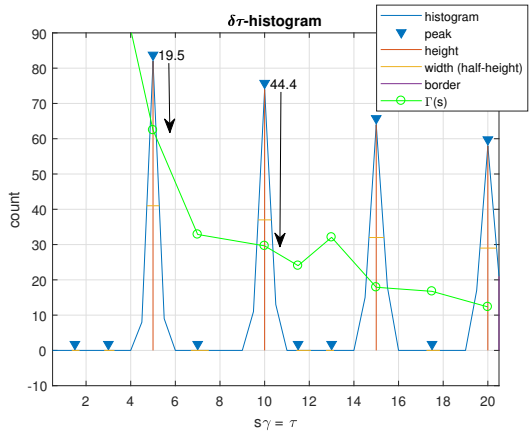
But we do not know the $\mathcal{T}_k(\tau)$ or the α_k,so instead we make the simplifying assumption that the $\mathcal{T}_k(\tau)$ are unimodal (e.g. Gaussian) with disjoint peaks (i.e. they do not overlap). This means that if there is a peak in bin s, we assume that a μ_k occurs at the center (i.e. $\mu_k = s\gamma$). Then the theoretical height of this peak is

$$\Gamma(s) = \frac{T}{s\gamma} \int_{(s-1)]\gamma}^{(s+1)\gamma} \mathcal{T}_k(\tau)d\tau. \tag{3.2.3}$$

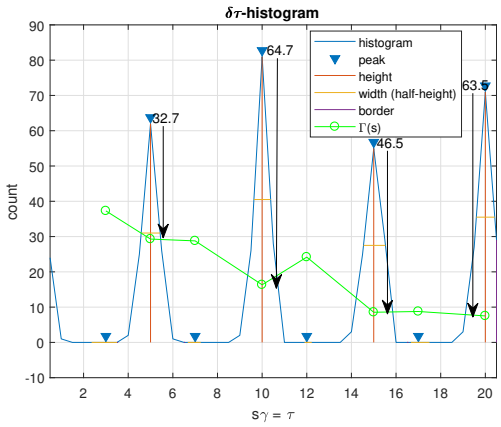
The factor before the integral is the approximate number of impulses from source with μ_k during a time period of T from 2.2. In the case that two or more $\mathcal{T}_k(\tau)$ overlap at s, then (3.2.3) would underestimate the height of the histogram. However, a conservative estimate is preferable. To compute (3.2.3), we need to know $\mathcal{T}_k$. We already

assume $\mu_k = s\gamma$ but we also need σ_k if $\mathcal{T}_k$ are Gaussian. Variance is a measure of the spread of the data, and therefore is related to the width of the peaks. We measure peak width as the distance between its boundaries at half its height[1] and use this as an estimate of σ_k (it is better to overestimate σ_k slightly than to underestimate it). Given a (μ_k, σ_k), the integral in (3.2.3) can be computed.

Since we pick a conservative $\Gamma(s)$, we expect that the actual μ_k should have peaks that are near but surpass it. Thus, we consider the peaks that are greater than $\Gamma(s)$, and then pick the K peaks that are closest to $\Gamma(s)$ as our estimates. This gives some protection against sub-harmonics. Sub-harmonics occur because the n-th order difference of an impulse train with interimpulse spacing μ_k will generate $T/\mu_k - n$ interimpulse spacings of $n\mu_k$. The threshold at the same point $s = \frac{n\mu_k}{\gamma}$ is $\Gamma(\frac{n\mu_k}{\gamma}) \leq \frac{T}{n\mu_k}$. Under most circumstances $\frac{T}{\mu_k} \geq 1$, so $T/\mu_k - n \gg \frac{T}{n\mu_k}$. That is, the height of a sub-harmonic peak is much greater than $\Gamma(s)$. Thus picking the peaks closest to $\Gamma(s)$ (using a simple difference measure) excludes most sub-harmonics. For example, Figure 3.2.1a shows a histogram and threshold for a source with $\mu = 5$. We see the most prominent peaks occurs at μ and its sub-harmonics, but peaks at sub-harmonics are much farther away from $\Gamma(s)$ than the peak at μ is.



(a) Histogram and threshold for a single source $\mathcal{T} \sim \mathcal{N}_0(5, 0.1^2)$.



(b) Histogram and threshold for 2 sources: $\mathcal{T}_1 \sim \mathcal{N}_0(5, 0.1^2)$ and $\mathcal{T}_2 \sim \mathcal{N}_0(10, 0.1^2)$.

Figure 3.2.1: Example $\delta\tau$-histograms with $\gamma = 0.5$ and $T_{\max} = 20$ for $M = 100$. The black arrows indicate the distance between the peaks and $\Gamma(s)$.

It is possible to have a μ_j that happens to be an integer multiple of μ_k. In this case, contribution from μ_j would add to the peak corresponding to a sub-harmonic of μ_k. This is illustrated in Figure 3.2.1b which has two sources: $\mu_1 = 5$ and $\mu_2 = 10$. As before, the sub-harmonics of μ_1 are far from $\Gamma(s)$, but this time the peak at $\tau = 10$ is especially high (higher than the original sub-harmonic value) due to the addition of the impulses from the second source. To avoid ignoring the potential conflation between μ_j and the sub-harmonic of a μ_k, we pick the K farthest from $\Gamma(s)$ in addition to the K closest peaks. Thus, we have $2K$ potential peaks and extract μ_k and σ_k from their position and width, respectively. We consider every combination without replacement as a potential $\boldsymbol{\Theta}$. If there are $< K$ peaks that surpass $\Gamma(s)$, multiple sources might have the same μ. In this case we consider all combinations with replacement for the potential $\boldsymbol{\Theta}$. The entire process for generating multiple $\boldsymbol{\Theta}$ from the observed impulse times is outlined in Algorithm 3.2, which we refer to as A3.2.

If γ is too large, then all data points will fall into the same bin and there will be either one or no peaks. On the other hand, if γ is too small, data from the same distribution will be fractured into many bins and there will be

[1] Please refer to width (half height)

Algorithm 3.2 $\delta\tau$-histogram parameter initialization algorithm

Given the observed impulse times $\{\tau_i\}$ for K sources and $T_{\max}$, to generate multiple $\boldsymbol{\Theta}$ with precision γ, do the following

1. Form a impulse time series $f[n]$ that has 1's at the impulse times and 0's everywhere else
   (a) The corresponding time indexes are $\lfloor \tau_1/\gamma \rfloor$ to $\lfloor \tau_M/\gamma \rfloor$ with each point being γ apart.
2. Compute the autocorrelation, $r[s]$, of $f[n]$ for $0 < s < \left\lfloor \frac{T_{\max}}{\gamma} \right\rfloor$.
3. Find the peaks of $r[s]$ and their widths.
4. For each peak compute $\Gamma(s)$ according to (3.2.3) using the corresponding lag of each peak as μ_k and the width as σ_k. This (μ_k, σ_k^2) for the peak are the parameters collected into $\boldsymbol{\Theta}$ if the peak is chosen.
5. Compare the height of the peaks with $\Gamma(s)$. Considering the subsets that are greater than $\Gamma(s)$:
   (a) If there are no peaks that pass, adjust γ and start again.
      i. Increase γ if there were many small peaks that did not pass
      ii. Decrease γ if there were very few peaks that did not pass (or no peaks at all)
   (b) If there $< K$ peaks, then take all combinations *with replacement* of these peaks as the output.
   (c) Otherwise, consider the difference $r[s] - \Gamma[s]$. Choose the K peaks that have the lowest difference and the K peaks that have the highest difference. Take all combinations *without replacement* of these $2K$ peaks as the output.

many small peaks (e.g. Figure 3.2.2). Literature suggests setting γ to the approximate jitter (i.e. σ_k) [5, 22]; this is consistent with our simulation observations. In practice, σ_k are unknown and γ is tuned to the observations. One possibility is to consider many values of γ, apply A3.2 and a timing separation algorithm for each of the generated $\boldsymbol{\Theta}$, and choose the result with largest (2.2.1) as the final output. However, this can be extremely computationally costly. An alternative is to judge a candidate histogram and adjust γ if not satisfactory. For example in Figure 3.2.2, even though γ is too small, we can clearly see that there are 4 significant peaks of which one probably corresponds to μ (if we increase γ, we will get something like Figure 3.2.1a). Therefore, we suggest starting with a small γ and increasing it until the number of peaks that pass $\Gamma(s)$ is approximately

$$\sum_{k=1}^{K} \left\lfloor \frac{T_{\max}}{\hat{\mu}_k} \right\rfloor,$$

where $\hat{\mu}_k$ are the locations of peaks greater than $\Gamma(s)$ indexed in ascending order (i.e. $\hat{\mu}_1 < \hat{\mu}_2$). If $\{\sigma_k\}$ do not vary much between observations, then there is no need to tune γ every time.

As we saw, the $\delta\tau$-histogram approach is complicated by the occurrence of sub-harmonics as peaks [23]. We minimize this complication by picking peaks based on their distance from $\Gamma(s)$. Our estimates serve as input to the pulse-separation algorithms, which will likely reject the sub-harmonics. Consequently, although computational cost is increased, performance is not compromised by incorrectly picking sub-harmonics.

3.2.1 Alternate peak picking (*)

Before the Algorithm 3.2 was developed, there was an earlier version which we used to generate some of the results presented in this paper. Ideally, we everything would use the more current $\delta\tau$-histogram method but did not redo

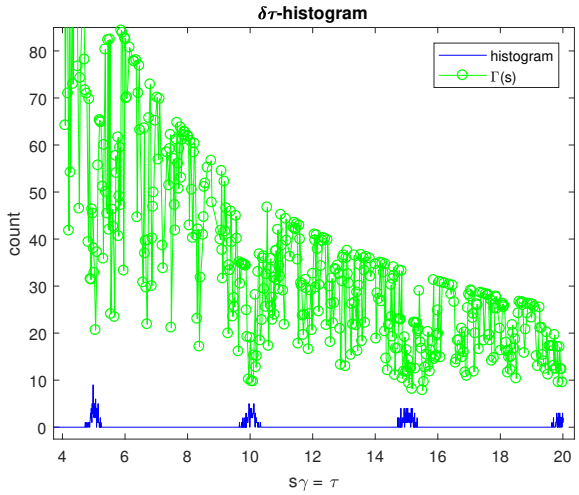


Figure 3.2.2: $\delta\tau$-histogram with $\gamma = 0.01$ and $T_{\max} = 20$ for a single source $\mathcal{T} \sim \mathcal{N}_0(5, 0.1^2)$ and $M = 100$.

simulations for everything (since we were focused on finding better alternatives). The method is very similar to the one just described, the main difference is in the selection of candidate peaks and the σ_k are set to $10\bar{\sigma}$ rather than being extracted from the peak widths.

The difference in peak selection is that instead of the heuristics used in step 5 of Algorithm 3.2 to avoid sub-harmonics, we just consider the top b, a user defined integer, unique sets of highest peaks. Specifically, if there are c peaks each at a height of $v_1, v_2, \ldots, v_c$. There are $\binom{c}{K}$ possible combinations. For each combination, the score is sum of the peaks values. For example, if the combination of peaks are $\{1, 2, 10\}$, then the score is $v_1 + v_2 + v_{10}$. We choose the b combinations with highest scores. Through experimentation, we found $b = 3K$ to be a sufficient just like with the sequential search.

When referring to this method we will use Algorithm 3.2* to distinguish it from the more current version described above.

As far as performance goes, there are a few tables in Section 2.3.4 that present results from Algorithm 3.2* with acceptable error rates, but for direct comparison with Algorithm 3.2, consider Table 3.2.1. The table compares P_e averaged over 100 trials using best guess for the two initialization methods for $K = 2, 3, 4, 5$ with $\mu_k \in [1, 100]$, $\sigma = 0.1$. As intended, Algorithm 3.2 is much better than Algorithm 3.2* in every instance.

Table 3.2.1: Average error rates over 100 trials for best guess with A2.4 and A3.2* and A2.4 with different numbers of sources with $\mu_k \in [1, 100]$, $\sigma = 0.1$.

K	2	3	4	5
Algorithm 3.2*	0.0241	0.1233	0.2225	0.3377
Algorithm 3.2	0.0173	0.0611	0.1211	0.2405

3.2.2 CDIF and SDIF

As mentioned previously, there are ways to deal with the sub-harmonics more directly. We would be remiss if we did not mention the cumulative difference (CDIF) histogram [47] and the sequential difference (SDIF) histogram [6]. There are a number of details to CDIF and SDIF, but essentially there are three steps:

1. Form a difference histogram or histograms and identify a candidate interimpulse interval
2. Remove all impulses associated with candidate using a sequential search (something like Algorithm 3.1)
3. If impulses remain, go back to step 1 and repeat with the remainder.

There are two main problems with this method. First, it requires a sequential search which, as discussed earlier, is designed for exactly periodic sources and does not fair well with large variance or jitter. Second, also due to the sequential searching, the end result of CDIF and SDIF is a complete assignment of impulses in addition to the source interimpulse intervals. Applying Chapter 2 after these methods would be redundant. For this reason we did not explore their use for parameter estimation.

3.2.3 Complex autocorrelation

Another option for reducing (sub)harmonics[2] is using the **magnitude** of complex autocorrelation [5, 23] in place of (3.2.2). The complex autocorrelation of function $f(t)$ is defined as

$$\rho(s) = \int f(t)f(t-s)\exp(2\pi j\frac{t}{s})dt, \tag{3.2.4}$$

where $j = \sqrt{-1}$. It is just $r(s)$ with a complex exponential factor, but this factor will help to reduce the harmonics. Again let $f(t)$ be the time signal (3.2.1), so we can write

$$\rho(s) = \int \sum_i \delta(t-\tau_i) \sum_j \delta(t-s-\tau_j)\exp(2\pi j\frac{t}{s})dt.$$

The product of the sums are only non-zero when

$$t = \tau_i \quad \cap \quad t = s + \tau_j,$$

so

$$\begin{aligned} t - \tau_i &= t - s - \tau_j \\ \tau_i &= s + \tau_j \Leftrightarrow s = \tau_i - \tau_j. \end{aligned}$$

Thus we can rewrite

$$\rho(s) = \sum_i \sum_j \delta(s-(\tau_i-\tau_j))\exp(2\pi j\frac{\tau_i}{s}),$$

where we have been a bit sloppy with notation by dropping the integral and are substituting

$$\delta(x) = \begin{cases} 1 & x = 0 \\ 0 & x \neq 0 \end{cases} \tag{3.2.5}$$

which is something like the Kronecker delta function.

To see why this reduces the peaks of harmonics, let us first reconsider (3.2.2) and the size of its peaks. The following argument is adapted from [22, 23]. Let

$$f(t) = \sum_{n=0}^{M-1} \delta(t-n\mu). \tag{3.2.6}$$

[2] Technically speaking, we mean *sub-harmonics*, but we may use *harmonics* in its place for succinctness.

That is, there is an impulse at every; we have a signal with single periodic source with no jitter. Then we have

$$r(s) = \sum_{n=0}^{M-1} \sum_{m=0}^{n-1} \delta(s - (n\mu - m\mu)).$$

In the more general case, the second sum would go from 0 to $M-1$ as well, but for this problem we do not care about $s \leq 0$, so we just omit those terms from the summation. Furthermore define $\ell = n - m$ so

$$\begin{aligned} r(s) &= \sum_{n=0}^{M-1} \sum_{m=0}^{n-1} \delta(s - (n-m)\mu) \\ &= \sum_{\ell=0}^{M-1} (M-\ell)\delta(s - \ell\mu). \end{aligned} \tag{3.2.7}$$

Note that this function is only non-zero at $s = \ell\mu$. To see how this expression is derived, note that $\ell = n - m = 1$ corresponds to the first order difference. ($\ell = 0 \Leftrightarrow s = 0$ is trivial and not interesting; it corresponds to a difference of 0 and just gives the total count of impulses $r(0) = M$) That is, the difference between adjacent impulses is exactly μ, and there are exactly $M-1$ first order differences (you need a pair of impulses per each difference). Following the same logic, $\ell = 2$ corresponds to the second order difference of 2μ, the first harmonic, of which there are $M-2$. In general, for the ℓ-th order difference/$\ell-1$-th harmonic, there are $N-\ell$ corresponding interimpulse intervals. Recall, we talked about this earlier (but not so precisely) when we reasoned to picking peaks closest and farthest from the threshold.

Now let us contrast this with the the complex autocorrelation for the same source (3.2.6).

$$\begin{aligned} \rho(s) &= \sum_{n=0}^{M-1} \sum_{m=0}^{n-1} \delta(s - (n\mu - m\mu)) \exp(2\pi j \frac{n\mu}{s}) \\ &= \sum_{n=0}^{M-1} \sum_{m=0}^{n-1} \delta(s - (n-m)\mu) \exp(2\pi j \frac{n}{n-m}). \end{aligned} \tag{3.2.8}$$

In the second line, due to $\delta(s-(n-m)\mu)$, we may comfortably substitute $s = (n-m)\mu$. Then again let $\ell = n-m$, where every value of ℓ indicates the level of the difference or harmonic level. We want to rearrange the sum in terms of ℓ like (3.2.7), but the added complex exponential is not only in terms of ℓ making direct simplification difficult. Recall (3.2.5), so the sum is essentially the sum of the complex exponential for the $m, n \mapsto \ell, n$. Therefore consider Table 3.2.2, where we compute the complex exponentials for $\ell = 1, 2, 3$. For each ℓ, there are exactly $M - \ell$ unique pairs of m, n. Notice for $\ell = 1$, the complex exponential is always 1, so $\rho(\mu) = M - 1$ just like (3.2.7). This is means that peak/count of the true interimpulse interval is unchanged. The situation is more complex for $\ell > 1$. For $\ell = 2$, observe that all terms are 1 or -1, so the sum is either 1 if M is even or 0 if M is odd. That is

$$\rho(2\mu) = \begin{cases} 1 & \mod(M, 2) = 0 \\ 0 & \mod(M, 2) = 1 \end{cases},$$

where $\mod(a, b)$ evaluates to the remainder of a/b. Something similar hold for $\ell = 3$, but instead of 2 terms, it requires 3 to sum to 0. That is,

$$\rho(3\mu) = \begin{cases} 1 & \mod(M, 3) = 0 \\ 1 + \exp(\frac{2\pi}{3} j) & \mod(M, 3) = 1 \\ 0 & \mod(M, 3) = 2 \end{cases}.$$

This is all due to the periodic nature of the complex exponential. In fact, we can generalize the result to the following

$$\rho(\ell\mu) = \begin{cases} 1 & \mod (M,\ell) = 0 \\ 1+\exp(2\pi j\frac{1}{\ell}) & \mod (M,\ell) = 1 \\ 1+\exp(2\pi j\frac{1}{\ell})+\exp(2\pi j\frac{2}{\ell}) & \mod (M,\ell) = 2 \\ \vdots & \vdots \\ 1+\exp(2\pi j\frac{1}{\ell})+\exp(2\pi j\frac{2}{\ell})+\cdots+\exp(2\pi j\frac{\ell-2}{\ell}) & \mod (M,\ell) = \ell-2 \\ 0 & \mod (M,\ell) = \ell-1 \end{cases}$$
$$= \sum_{n=0}^{\mod (M,\ell)} \exp(2\pi j\frac{n}{\ell}).$$

To better understand this quantity, consider Theorem 3.1 and its Corollary 3.1.

Theorem 3.1. [3] *Let $p_n = e^{2\pi j\frac{n}{N}}$ for $n = 0,1,2,\ldots,N-1$ be a set of N evenly spaced points around the unit circle in the complex plane (i.e. the primitive roots of unity on the complex plane). For any $N > 1$, the sum of N evenly spaced points in the complex plane around the unit circle is 0. That is*

$$\sum_{n=0}^{N-1} e^{2\pi j\frac{n}{N}} = 0.$$

Proof. Let $r = e^{\frac{2\pi j}{N}}$, then realize we have a geometric series with a well known result for the sum of the first N terms:

$$\sum_{n=0}^{N-1} r^n = \frac{1-r^N}{1-r}.$$

Then if we plug in our r,

$$\sum_{n=0}^{N-1} e^{2\pi j\frac{n}{N}} = \frac{1-(e^{\frac{2\pi j}{N}})^N}{1-e^{\frac{2\pi j}{N}}} = \frac{1-\overbrace{e^{2\pi j}}^{1}}{1-e^{\frac{2\pi j}{N}}} = \frac{0}{1-e^{\frac{2\pi j}{N}}} = 0.$$

Note, we can be sure that the denominator is non-zero since it is required $N > 1$. □

Corollary 3.1. *The maximum magnitude of the sum of the first $0 < M < N$ points occurs when $M = \frac{N}{2}$. That is*

$$\arg\max_M \left|\sum_{n=0}^{M-1} e^{2\pi j\frac{n}{N}}\right| = \frac{N}{2},$$

and correspondingly

$$\max_M \left|\sum_{n=0}^{M-1} e^{2\pi j\frac{n}{N}}\right| = \left|\sum_{n=0}^{\frac{N}{2}-1} e^{2\pi j\frac{n}{N}}\right| = \frac{2}{\sqrt{2-e^{-2\pi j\frac{1}{N}}-e^{+2\pi j\frac{1}{N}}}}.$$

Note, M is an integer so if N is odd, then the maximum will occur at $M = \lfloor\frac{N}{2}\rfloor$ and/or $M = \lceil\frac{N}{2}\rceil$.

[3] This is not a novel result, but I could not find a suitable reference, so I restated it here.

Proof. Use the same geometric sum result

$$\sum_{n=0}^{M-1} e^{2\pi j \frac{n}{N}} = \frac{1-(e^{\frac{2\pi j}{N}})^M}{1-e^{\frac{2\pi j}{N}}} = \frac{1-e^{2\pi j \frac{M}{N}}}{1-e^{2\pi j \frac{1}{N}}}.$$

The magnitude of a number A is defined

$$|A| = \sqrt{AA^*},$$

but since the square root is a monotonic function, maximizing its argument, AA^*, is the same as maximizing the square root. Therefore, we wish to maximize

$$\begin{aligned}\left(\frac{1-e^{2\pi j \frac{M}{N}}}{1-e^{2\pi j \frac{1}{N}}}\right)\left(\frac{1-e^{2\pi j \frac{M}{N}}}{1-e^{2\pi j \frac{1}{N}}}\right)^* &= \left(\frac{1-e^{2\pi j \frac{M}{N}}}{1-e^{2\pi j \frac{1}{N}}}\right)\left(\frac{1-e^{-2\pi j \frac{M}{N}}}{1-e^{-2\pi j \frac{1}{N}}}\right) \\ &= \frac{1-e^{-2\pi j \frac{M}{N}}-e^{+2\pi j \frac{M}{N}}+1}{1-e^{-2\pi j \frac{1}{N}}-e^{+2\pi j \frac{1}{N}}+1} \\ &= \frac{2-e^{-2\pi j \frac{M}{N}}-e^{+2\pi j \frac{M}{N}}}{2-e^{-2\pi j \frac{1}{N}}-e^{+2\pi j \frac{1}{N}}}\end{aligned}$$

as a function of M for a given $N > 1$ under the constraint that $M < N$. Then note, the denominator does not depend on M, so we can just consider the numerator for our objective function

$$J(M) = 2 - e^{-2\pi j \frac{M}{N}} - e^{+2\pi j \frac{M}{N}}.$$

Note M is also restricted to integers, but let us first try to solve it without constraints. With this relaxation, we will take the derivative and set to 0 to find the critical points.

$$\begin{aligned}J'(M) &= -(-2\pi j \frac{1}{N})e^{-2\pi j \frac{M}{N}} - (2\pi j \frac{1}{N})e^{+2\pi j \frac{M}{N}} \\ &= (2\pi j \frac{1}{N})(e^{-2\pi j \frac{M}{N}} - e^{+2\pi j \frac{M}{N}}).\end{aligned}$$

Then solving for zero

$$\begin{aligned}J'(M) = 0 = (2\pi j \frac{1}{N})(e^{-2\pi j \frac{M}{N}} - e^{+2\pi j \frac{M}{N}}) \\ 0 = e^{-2\pi j \frac{M}{N}} - e^{+2\pi j \frac{M}{N}} \\ e^{+2\pi j \frac{M}{N}} = e^{-2\pi j \frac{M}{N}}.\end{aligned} \tag{3.2.9}$$

Taking the natural log of both sides we get

$$2\pi j \frac{M}{N} = -2\pi j \frac{M}{N}$$

which can only be satisfied if $M = 0$. This is a trivial and uninteresting result, as this corresponds to summing no terms, resulting in a sum of 0. For magnitude 0 is the lowest value, so this is a minimum. However, this is not the only solution to (3.2.9). The complex exponential is periodic on the interval $\theta \in [0, 2\pi)$; that is,

$$e^{j(\theta+2k\pi)} = e^{j\theta}$$

for any integer k (positive, negative, or zero). This means we can multiply by either side of (3.2.9) by $e^{j2\pi k}$ without invalidating the equality. First multiply the LHS by $e^{j2\pi(1)}$, this gives

$$e^{+2\pi j \frac{M}{N}} = e^{-2\pi j \frac{M}{N}} e^{j2\pi} = e^{j(2\pi - 2\pi \frac{M}{N})}.$$

Equating the exponents (i.e. we take the natural log of both sides)

$$\begin{aligned} j2\pi\frac{M}{N} &= j(2\pi - 2\pi\frac{M}{N}) \\ \frac{M}{N} &= 1 - \frac{M}{N} \\ \frac{2M}{N} &= 1 \\ M &= \frac{N}{2}. \end{aligned}$$

Next we compute the second derivative and plug the value in to see if it is a maximum or a minimum.

$$\begin{aligned} J''(M) &= (2\pi j\frac{1}{N})((-2\pi j\frac{1}{N})e^{-2\pi j\frac{M}{N}} - (+2\pi j\frac{1}{N})e^{+2\pi j\frac{M}{N}}) \\ &= -(2\pi j\frac{1}{N})^2(e^{-2\pi j\frac{M}{N}} + e^{+2\pi j\frac{M}{N}}) \\ &= -j^2\frac{4\pi^2}{N^2}(e^{-2\pi j\frac{M}{N}} + e^{+2\pi j\frac{M}{N}}) \\ &= \frac{4\pi^2}{N^2}(e^{-2\pi j\frac{M}{N}} + e^{+2\pi j\frac{M}{N}}). \end{aligned}$$

Now plug in $M = \frac{N}{2}$

$$\begin{aligned} J''(\frac{N}{2}) &= \frac{4\pi^2}{N^2}(e^{-2\pi j\frac{\frac{N}{2}}{N}} + e^{+2\pi j\frac{\frac{N}{2}}{N}}) \\ &= \frac{4\pi^2}{N^2}(e^{-\pi j} + e^{+\pi j}) \\ &= \frac{4\pi^2}{N^2}(-2) = -\frac{8\pi^2}{N^2}. \end{aligned}$$

Recall, the second derivative test states that if we have function $f(x)$ and a critical point c such that $f'(c) = 0$, then the critical point corresponds to a local maximum if $f''(c) < 0$ or a local minimum if $f''(c) > 0$. This follows directly from the result that a function is concave down if $f''(x) < 0$ and concave up if $f''(x) > 0$. When $f''(x) = 0$, there is a change in concavity, an inflection point. Above we see that $J''(\frac{N}{2}) < 0$, so it corresponds to a local maximum. Furthermore, we can verify that $M = 0$ is also a minimum since

$$J''(0) = \frac{4\pi^2}{N^2}(e^0 + e^0) = \frac{8\pi^2}{N^2} > 0.$$

These two solutions cover the points within our range (i.e. $M < N$ and $N > 1$), but to be sure consider the general case where we multiply the LHS of (3.2.9) with $e^{j2\pi k}$. After taking the natural logarithm of both sides we have

$$\begin{aligned} 2\pi j\frac{M}{N} &= 2\pi kj - 2\pi j\frac{M}{N} \\ \frac{M}{N} &= k - \frac{M}{N} \\ \frac{2M}{N} &= k \\ M &= \frac{kN}{2}. \end{aligned}$$

Then for $k = 2$, $M = N$ which violates the condition (and also takes us back to Theorem 3.1). Since the solution is obviously monotonically increasing with respect to k, $k = 3, 4, \ldots$ will also violate the condition. To be complete, the negative values of k can also be ignored since they will give $M < 0$ which is not of any interest. □

These two results give us the bounds for the sum

$$\left| \sum_{n=0}^{\text{mod }(M,\ell)} \exp(2\pi j \frac{n}{\ell}) \right| \in \left[0, \frac{2}{\sqrt{2 - e^{-2\pi j \frac{1}{\ell}} - e^{+2\pi j \frac{1}{\ell}}}} \right].$$

It is not easy to understand how the maximum behaves as a function of ℓ, so we plot it for a some values of ℓ in Figure 3.2.3. It is approximately linear with a slope of 0.3147. That is,

$$\frac{2}{\sqrt{2 - e^{-2\pi j \frac{1}{\ell}} - e^{+2\pi j \frac{1}{\ell}}}} \approx 0.3147\ell.$$

Note, this does not come from the linear regression line on these 100 points. Instead we notice the intercept appears to be 0, so we simply compute the slope as the mean of difference between adjacent points (points are all exactly 1 unit apart). When actually computed, regression line is

$$0.3175\ell + 0.0661$$

with $R^2 = 0.99998$. The crude approximation has $R^2 = 0.9994$. The difference in accuracy is very marginal, so we use the simpler approximation, 0.3147ℓ, in further discussions.

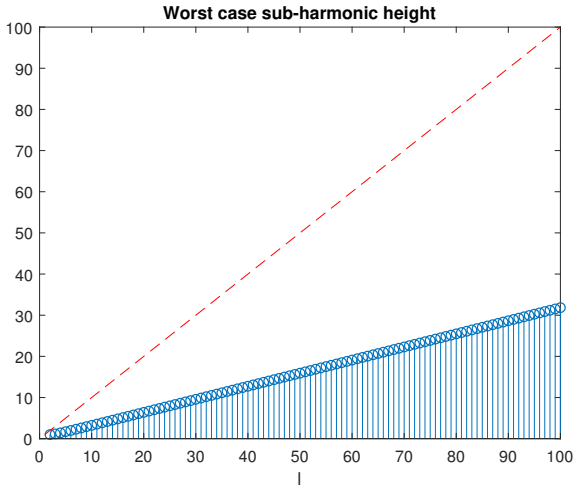


Figure 3.2.3: Plot of the worst case harmonic height for different values of ℓ. The red dotted line has a height of ℓ.

To clarify further, consider the complete expression for the complex autocorrelation is then

$$\rho(s) = (M-1)\delta(s-\mu) + \sum_{\ell=2}^{M-1} \sum_{n=0}^{\text{mod }(M,\ell)} \delta(s - \mu\ell) \exp(2\pi j \frac{n}{\ell}). \tag{3.2.10}$$

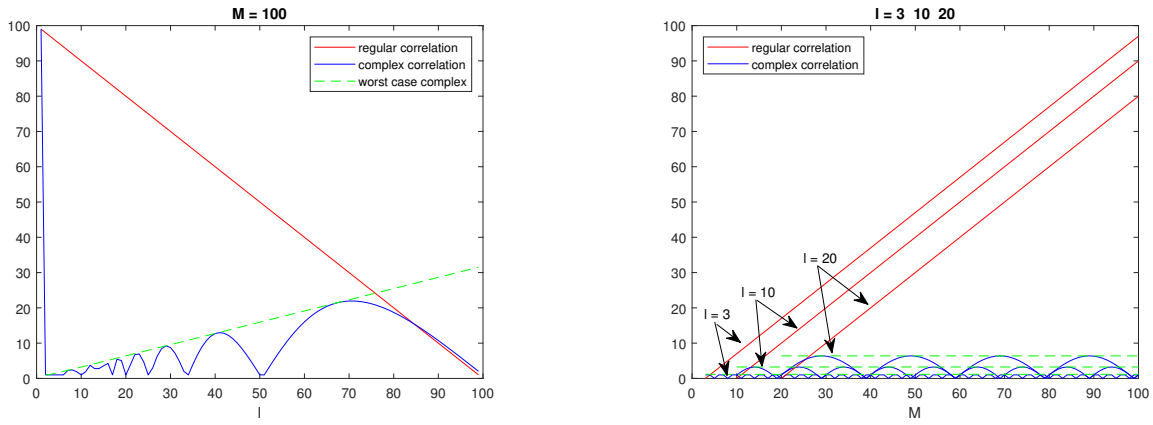
(Again $\ell = 0 \Leftrightarrow s = 0$ is trivial and $\rho(0) = M$, but we omit it in the sum as it follows a different form) We compare this to (3.2.7) with some slight rearrangement

$$r(s) = (M-1)\delta(s-\mu) + \sum_{\ell=2}^{M-1} (M-\ell)\delta(s-\ell\mu).$$

As noted earlier, in both cases, the first term is the same. However in $r(s)$ the height of the ℓ-th harmonic is $M-\ell$, whereas with $\rho(s)$, the height is $|\sum_{n=0}^{\mod (M,\ell)} \exp(2\pi j \frac{n}{\ell})|$ which is approximately bounded with 0.3147ℓ. The heights of harmonics for $r(s)$ are decreasing with ℓ, and the heights of harmonics for $\rho(s)$ are decreasing with ℓ. They intersect at approximately

$$\begin{aligned} 0.3147\ell &= M - \ell \\ \ell &= \frac{M}{1.3147} \approx 0.7606M. \end{aligned}$$

That is, the harmonics in $\rho(s)$ will be lower than harmonics in $r(s)$ for $\ell < 0.7606M$. Above this threshold, there is no guaranty. But due to the nature of $\mod (M,\ell)$, the values above the threshold are close to $M-\ell$. To see this, consider Figure 3.2.4 which displays the heights of peaks for regular correlation in red and complex correlation in blue (the worst case for the complex correlation is also shown as a green dotted line). Only when $\ell > 0.7606M$, then $|\rho(s)| \approx r(s)$, otherwise $|\rho(s)| \ll r(s)$. Furthermore when $\ell > 0.7606M$, $r(\ell\mu) < 0.2394M$. For most interesting datasets, this will be considerably less than $r(\mu) = M-1$; meaning that if present, these harmonic peaks will be insignificant in comparison to the true mean impulse spacing. Additionally, it is unusual to care about very high order harmonics. For example, if the true interimpulse interval is 10 and $M = 1000$, the $0.7606M$ harmonic corresponds to an interimpulse interval of 760. In most experiments, interesting values would be around the same scale (e.g. perhaps interimpulse spacings of 1-100 for this example).



(a) $M = 100$ (b) $\ell = 3, 10, 20$

Figure 3.2.4: This is the expected heights of the ℓ-th harmonics for $r(s)$ (red) and $\rho(s)$ (blue) at the listed M for a purely periodic source.Worst case harmonic height shown as green dotted lines for the different ℓ.

Note for (3.2.6), we made the assumption that the source started emitting at $t = 0$, for a single source, there is no problem with this assumption since we can always shift the time origin to where the first impulse is. However, the same cannot be said if there are multiple sources; under the assumption of no overlap, they will not all start at

the same time, and thus the first click from all sources cannot be adjusted back to time 0. Therefore, we need to assert that the claims made above will hold for the more general case of (3.2.6)

$$f_\phi(t) = \sum_n d(t - (n\mu + \phi)),$$

where here $\phi > 0$ is some arbitrary delay (often called phase). Using this, we recompute (3.2.7)

$$\begin{aligned} r(s) &= \sum_{n=0}^{M-1} \sum_{m=0}^{n-1} \delta(s - ((n\mu + \phi) - (m\mu + \phi))) \\ &= \sum_{n=0}^{M-1} \sum_{m=0}^{n-1} \delta(s - (n\mu - m\mu)). \end{aligned}$$

The ϕ cancel out, and then substituting $\ell = n - m$, we get the same result as before. Similarly for (3.2.8) we plug in $f_\phi(t)$

$$\begin{aligned} \rho(s) &= \sum_{n=0}^{M-1} \sum_{m=0}^{n-1} \delta(s - (n\mu + \phi) - (m\mu + \phi))) \exp(2\pi j \frac{n\mu - \phi}{s}) \\ &= \sum_{n=0}^{M-1} \sum_{m=0}^{n-1} \delta(s - (n-m)\mu) \exp(2\pi j \frac{n\mu - \phi}{(n-m)\mu}) \\ &= \sum_{n=0}^{M-1} \sum_{m=0}^{n-1} \delta(s - (n-m)\mu) \exp(2\pi j \frac{n}{n-m}) \exp(2\pi j \frac{-\phi}{(n-m)\mu}). \end{aligned}$$

This is the same as before, but with the added delay factor

$$\exp(2\pi j \frac{-\phi}{(n-m)\mu}).$$

Realize that for a given $\ell = n - m$, this quantity is constant with respect to n. Therefore, we have

$$\begin{aligned} \rho(\ell\mu) &= \sum_{n=0}^{\text{mod }(M,\ell)} \exp(2\pi j \frac{n}{\ell}) \exp(-2\pi j \frac{\phi}{\ell\mu}) \\ &= \exp(-2\pi j \frac{\phi}{\ell\mu}) \sum_{n=0}^{\text{mod }(M,\ell)} \exp(2\pi j \frac{n}{\ell}). \end{aligned}$$

Since $|\exp(-2\pi j \frac{\phi}{\ell\mu}| = 1$, the delay ϕ will only adjust the direction in the complex plane, but not the magnitude of the harmonics. Therefore, the claims above will hold regardless of the first impulse time.

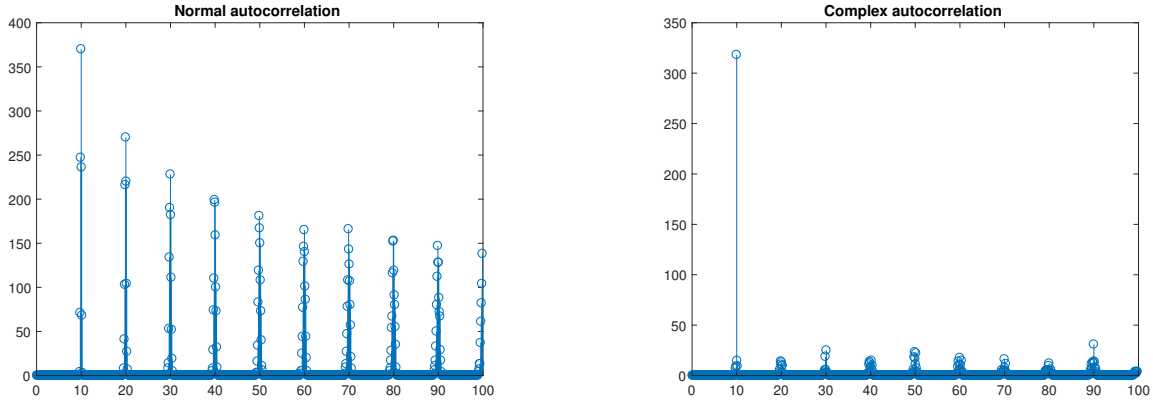
Figure 3.2.4 shows the values for a purely periodic source, but we are interested in sources that have some jitter (i.e. $\tau_i - \tau_{i-1} \sim \mathcal{T}$). As discussed in Section 3.2, jitter has the direct effect of lowering the bin height. More specifically, if the interimpulse interval is denoted by the random variable $X \sim \mathcal{T}$, then the second harmonic is actually the random variable

$$Y = X + X,$$

where $Y \sim \mathcal{T} * \mathcal{T}$ where $*$ denotes convolution [29]. If $X \sim \mathcal{N}(\mu, \sigma^2)$, then $Y \sim \mathcal{N}(2\mu, 2\sigma^2)$. Going further, let the ℓ-th harmonic be $Y = \ell X$, then, again assuming $X \sim \mathcal{N}(\mu, \sigma^2)$, $Y \sim \mathcal{N}(\ell\mu, \ell\sigma^2)$. For a jittered source, if we

look at $r(s)$, the harmonic peaks will be shifted to $\ell\mu$, but the peaks will also be flattened (i.e. wider and shorter) because of the increased variance. With $\rho(s)$, harmonic peaks will be attenuated due to the cancellation from the summing of the complex exponential. However, cancellation only occurs when terms collect at a particular lag s. Many terms will collect at the s near the $\ell\mu$, but it will not be perfect at each s as it was with the purely periodic case. When more terms collect at an s, the propensity for cancellation is greater. Thus the amount of cancellation will be less at larger ℓ because there will be less terms collecting at each s.

To demonstrate this fact, consider Figure 3.2.5 which shows $r(s)$ and $|\rho(s)|$ for $M = 1000$ samples from a single source with $\mathcal{T} \sim \mathcal{N}(10, 0.1^2)$. Note that the harmonics do not exactly follow any of the lines in Figure 3.2.4; this is effect of jitter. The peak corresponding to the mean interimpulse interval is highest in both, but the peaks of the harmonics in $r(s)$ are clearly much larger than those in $\rho(s)$.



(a) $r(s)$ (b) $\rho(s)$

Figure 3.2.5: This is $\delta\tau$-histogram computed using both the regular autocorrelation, $r(s)$, and the complex autocorrelation, $\rho(s)$, for $M = 1000$ samples from a single source with $\mathcal{T} \sim (10, 0.1^2)$.

Table 3.2.2: The complex exponentials in (3.2.8)

(a) $\ell = 1$

ℓ	n	m	$\exp(2\pi j \frac{n}{\ell})$
1	1	0	$\exp(2\pi j) = 1$
⋮	2	1	$\exp(2\pi j 2) = \exp(4\pi j) = 1$
⋮	⋮	⋮	⋮
1	$M-1$	$M-2$	$\exp(2(M-1)\pi j) = 1$

(b) $\ell = 2$

ℓ	n	m	$\exp(2\pi j \frac{n}{\ell})$	mod (n, ℓ)
2	2	0	$\exp(2\pi j \frac{2}{2}) = 1$	0
⋮	3	1	$\exp(2\pi j \frac{3}{2}) = \exp(3\pi j) = -1$	1
⋮	4	2	$\exp(2\pi j \frac{4}{2}) = \exp(4\pi j) = 1$	0
⋮	5	3	$\exp(2\pi j \frac{5}{2}) = \exp(5\pi j) = -1$	1
⋮	⋮	⋮	⋮	
2	$M-1$	$M-3$	$\exp((M-1)\pi j)$	mod $(M-1, 2)$

(c) $\ell = 3$

ℓ	n	m	$\exp(2\pi j \frac{n}{\ell})$	mod (n, ℓ)
3	3	0	$\exp(2\pi j \frac{3}{3}) = \exp(2\pi j) = 1$	0
⋮	4	1	$\exp(2\pi j \frac{4}{3}) = \exp(\frac{8\pi}{3} j) = \exp(\frac{2\pi}{3} j)$	1
⋮	5	2	$\exp(2\pi j \frac{5}{3}) = \exp(\frac{10\pi}{3} j) = \exp(\frac{1\pi}{3} j)$	2
⋮	6	3	$\exp(2\pi j \frac{6}{3}) = \exp(4\pi j) = \exp(2\pi j)$	0
⋮	7	4	$\exp(2\pi j \frac{7}{3}) = \exp(\frac{14\pi}{3} j) = \exp(\frac{2\pi}{3} j)$	1
⋮	8	5	$\exp(2\pi j \frac{8}{3}) = \exp(\frac{16\pi}{3} j) = \exp(\frac{1\pi}{3} j)$	2
⋮	⋮	⋮	⋮	
3	$M-1$	$M-4$	$\exp(2\pi j \frac{M-1}{3})$	mod $(M-1, 3)$

3.3 PRI/ICI map

The PRI or ICI[4] map described in [48, 22] is a method that applies the complex autocorrelation (3.2.4) at different sections of time to produce a map of when different interimpulse intervals are present. It is like Short Time Fourier Transform (STFT) where the frequency content of a signal is displayed as a function of time. However in terms of implementation, it is more similar to the wavelet transform since the size of interval bins and time period where $\rho(s)$ is computed varies.

More specifically,

$$D(t,\tau) = \int_{s\in W(t,\tau)} f(s)f(s-\tau)\exp(2\pi j\frac{s}{\tau})ds \tag{3.3.1}$$

denotes the PRI map value at a time t and PRI τ, where $W(t,\tau)$ denotes the time window over which we compute the complex autocorrelation. Again in this case, the function f is the impulse time function (3.2.1). The time window $W(t,\tau)$ is a window centered around t with width $v\tau$, where $v \in \mathbb{N}$ is the desired number of PRI for each bin (it is a user defined parameter). That is

$$W(t,\tau) = [t - \frac{v}{2}\tau, t + \frac{v}{2}\tau].$$

A duration of $v\tau$ has exactly enough time for v successive interimpulse spacings of length τ. In other words, if there is a source with a PRI of τ, we should expect to see approximately v of them in the window $W(t,\tau)$; this normalizes the intensity by giving each τ-bin an equivalent amount of data.

$D(t,\tau)$ is a continuous function, but when we implement we need to discretize. That is, we must select some set values for t and τ to evaluate. For t, evenly spaced values will suffice, however, the spacing should be that such that there is some overlap, or at least no gaps between the successive $W(t,\tau)$. To ensure this, one should consider the minimum value of τ and compute the window width in time. In the τ direction, how the windows are spaced depend on the system model. Let τ_n denote the n-th occurring ICI. In [48, 22, 23], the assumed model for the ICI is

$$\tau_n = \mu + \varepsilon_n\mu, \tag{3.3.2}$$

where ε_n denotes the relative deviation from the mean ICI, μ. That is, it is a realization of the uniformly distributed random variable on the range $[-\frac{\varepsilon_{\max}}{2}, +\frac{\varepsilon_{\max}}{2}]$. The total deviation from μ is $\varepsilon_n\mu$, so the variation is proportional to μ. For this reason, they recommend τ-windows of width $\varepsilon_{\max}\tau$. That is,

$$[(1 - \frac{\varepsilon_{\max}}{2})\tau, (1 + \frac{\varepsilon_{\max}}{2})\tau].$$

With windows proportional to τ, the jitter will more accurately be captured and the μ more correctly identified. For a similar reason, [22] recommends spacing the range of τ by a geometric progression (e.g. the center of the i-th bin is given by $\tau_i = r^{i-1}\tau_1$, where $r > 1$ is some constant) rather than uniform spacing. However, our model is quite different; instead we claim that the ICI are iid realizations of $\mathcal{T} \sim \mathcal{N}(\mu,\sigma^2)$. To put this into the same form as (3.3.2), let $X \sim \mathcal{N}(0,1)$ and let x_n be the n-th iid realization of X. Then our model is for ICI is

$$\tau_n = \mu + \sigma x_n.$$

Unlike (3.3.2), note that the total deviation from μ is not proportional to μ. Therefore, there is no reason to make windows proportional to τ; a uniform width of b should suffice. That is, we consider windows

$$[\tau - \frac{b}{2}, \tau + \frac{b}{2}].$$

[4] In this section we will use PRI, ICI, and interimpulse spacing interchangeably. They all mean the same thing.

Similar to time, we will also use uniform spacing for τ bin centers values. Also again, it is important to ensure that the bins overlap or at least do not leave any gaps in the range. Like discussed in Section 3.2, choice of b is important in terms of being able to discern the ICI present. According to our jitter model, $b \approx 3\sigma$ would be appropriate to center the impulses from the same timing source in the same bin.

Then the discretized PRI map value at spacing τ_j and at time t_i is

$$D(t_i, \tau_j) = \int_{\tau \in [\tau_j - \frac{b}{2}, \tau_j + \frac{b}{2}]} \int_{s \in W(t_i, \tau_j)} f(s) f(s-\tau) \exp(2\pi j \frac{s}{\tau_j}) ds d\tau. \tag{3.3.3}$$

Pay close attention to the quantities labeled τ and τ_j. To be more clear, Algorithm 3.3 lists the steps to implement the computation of (3.3). This should be computed for the all the predetermined $\{t_i\}$ and $\{\tau_j\}$. Note, as with $\rho(s)$ in Section 3.2.3, for analysis, we look at $|D(t_i, \tau_j)|$ rather than its complex value.

Algorithm 3.3 PRI map value computation

Given an impulse time signal $f(t)$, a time t_i, interimpulse spacing τ_j, and parameters v, b, the discretized PRI map value $D(t_i, \tau_j)$ is evaluated as follows:

1. Extract the window $[t_i - \frac{v}{2}\tau_j, t_i + \frac{v}{2}\tau_j]$ from the signal $f(t)$. Call this $f_e(t)$.
2. Compute the adjusted complex correlation

   $$\rho_e(\tau) = \int f_e(s) f_e(s-\tau) \exp(2\pi j \frac{s}{\tau_j}) ds.$$

   Note the complex exponential is computed with τ_j and not τ. This is a continuous unbounded function in τ, however because of $f_e(s)f_e(s-\tau)$, it only has non-zero values at a few values of τ.
3. Compute

   $$D(t_i, \tau_j) = \int_{\tau \in [\tau_j - \frac{b}{2}, \tau_j + \frac{b}{2}]} \rho_e(\tau) d\tau$$

   To say this another way, from $\rho_e(\tau)$ extract only the values corresponding to the PRI window $[\tau_j - \frac{b}{2}, \tau_j + \frac{b}{2}]$ and integrate/sum. (Alternatively, in the previous step, we can omit computation of lags outside this range)

For analysis, take the magnitude of $D(t_i, \tau_j)$.

3.3.1 Noise floor

In this section, we will to re-derive the noise floor found in [48, 22] but adjusted for our jitter model.

In the PRI map, we look at all possible interimpulse spacings, many of which probably do not correspond to spacings within a single source. These spurious spacings between sources are what we are referring to as the *noise* in the transform. Based on a few assumptions, we can estimate the expected height of the PRI transform bin due to this noise.

First let L denote the number pulse pairs that is counted for an arbitrary bin in the PRI transform. For the noise, we will be assuming a uniform distribution for the pulse spacings, so let ρ denote the pulse density of the signal. That is,

$$\rho = \frac{\text{total number of pulses}}{\text{total length of the signal}}.$$

Without loss of generality, let τ_k be the spacing the arbitrary bin is centered on; though no indexing is given, ρ should be computed for this bin. Then the signal extracted for the bin is given by the window $W(t, \tau_k)$ and has

length $v\tau_k$. This means that there on average $\rho v\tau_k$ pulses the signal extracted. In this bin, we only count the pulses pairs that have spacing in $[\tau_k - \frac{b}{2}, \tau_k + \frac{b}{2}]$, a window that is b-wide centered on τ. Again following the uniform assumption, ρb of the pulse pairs will have this spacing. We can combine these two quantities to give the expected value for L as product of the expected number of pulses for the time window and the expected portion of pulse pairs that should fall in the τ window. That is,

$$E[L] = \rho v\tau_k \times \rho b = \rho^2 b v\tau_k = \lambda.$$

For convenience, we denote this value with λ. Recall that the PRI map value is given by (3.3.1), so we can estimate the noise level as

$$D_N = \sum_{i=1}^{L} e^{j\theta_i},$$

where θ_i are iid random values uniformly distributed in $[0, 2\pi]$ (i.e. spurious spacings will have random phase). However we actually look at the magnitude of the PRI map, so we should consider

$$I_N = \sqrt{E[|D_N|^2]},$$

where

$$|D_N|^2 = D_N D_N^*,$$

and where D_N^* is the complex conjugate of D_N. Recall the the complex conjugate of a sum is the sum of the complex conjugates, so

$$D_N^* = \sum_{i=1}^{L} e^{-j\theta_i}.$$

Thus we have

$$\begin{aligned} I_N^2 = |D_N|^2 &= \sum_{i=1}^{L} e^{j\theta_i} \sum_{\ell=1}^{L} e^{-j\theta_\ell} \\ &= \sum_{i=1}^{L}\sum_{\ell=1}^{L} e^{j(\theta_i - \theta_\ell)}. \end{aligned}$$

But note, we can rewrite this sum as follows

$$\begin{aligned} |D_N|^2 &= \sum_{i=\ell} e^{j(\theta_i - \theta_\ell)} + \sum_{i\neq\ell} e^{j(\theta_i - \theta_\ell)} \\ &= \sum_{i=\ell} 1 + \sum_{i\neq\ell} e^{j(\theta_i - \theta_\ell)} \\ &= L + \sum_{i\neq\ell} e^{j(\theta_i - \theta_\ell)}. \end{aligned}$$

Since there are exactly L times that $i = \ell$. Now we want to compute the expectation of this. Recall

$$E[f(X)] = \int f(x)p(x)dx.$$

But in this case, we have 2 random variables L and $\theta_i - \theta_\ell$, so we need to consider $p(L, \theta_i - \theta_\ell)$; this is actually an interesting distribution containing both continuous and discrete values. We claim that L and $\theta_i - \theta_\ell$ are independent. That is, the number of pulse pairs will do not have influence on the random phases of each pair. Under

uniform distribution assumptions, we get that L is Poisson with $\lambda = \rho^2 bv\tau$. We also can compute $p(\theta_i - \theta_\ell)$ as the convolution of

$$p_\Theta(\theta) = \begin{cases} \frac{1}{2\pi} & 0 \leq \theta \leq 2\pi \\ 0 & \text{otherwise} \end{cases}$$

and if we let $\omega = -\theta$, then

$$p_\Omega(\omega) = \begin{cases} \frac{1}{2\pi} & -2\pi \leq \omega \leq 0 \\ 0 & \text{otherwise} \end{cases}.$$

Let $\phi = \theta_i - \theta_\ell = \theta + \omega$. Then

$$\begin{aligned} p_\Phi(\phi) &= \int p_\Theta(\phi - \omega) p_\Omega(\omega) d\omega \\ &= \int p_\Omega(\phi - \theta) p_\theta(\theta) d\theta \\ &= \frac{1}{2\pi} \int_0^{2\pi} p_\Omega(\phi - \theta) d\theta. \end{aligned}$$

In order to compute this, it useful to clearly define $p_\Omega(\phi - \theta)$. It is $p_\Omega(\omega)$ flipped (this actually gives $p_\Theta(\theta)$) and then shifted to the right by ϕ. That is,

$$p_\Omega(\phi - \theta) = \begin{cases} \frac{1}{2\pi} & \phi \leq \theta \leq \phi + 2\pi \\ 0 & \text{otherwise} \end{cases}.$$

This means we have 2 cases where the integral is non-zero. First, when $\phi < 0 \cap \phi + 2\pi > 0 \Rightarrow -2\pi < \phi < 0$,

$$\begin{aligned} p_\Phi(\phi) &= \frac{1}{2\pi} \int_0^{\phi + 2\pi} \frac{1}{2\pi} d\theta \\ &= \frac{1}{4\pi^2} \theta|_0^{\phi + 2\pi} \\ &= \frac{1}{4\pi^2} \phi + \frac{1}{2\pi}. \end{aligned}$$

Second when $\phi > 0 \cap \phi < 2\pi \Rightarrow 0 < \phi < 2\pi$,

$$\begin{aligned} p_\Phi(\phi) &= \frac{1}{2\pi} \int_\phi^{2\pi} \frac{1}{2\pi} d\theta \\ &= \frac{1}{4\pi^2} \theta|_\phi^{2\pi} \\ &= -\frac{1}{4\pi^2} \phi + \frac{1}{2\pi}. \end{aligned}$$

It is 0 elsewhere; this forms a triangle of height $\frac{1}{4\pi^2}$ centered at 0. The complete expression is

$$p_\Phi(\phi) = \begin{cases} \frac{1}{4\pi^2}\phi + \frac{1}{2\pi} & -2\pi < \phi \leq 0 \\ -\frac{1}{4\pi^2}\phi + \frac{1}{2\pi} & 0 < \phi < 2\pi \\ 0 & \text{otherwise} \end{cases}.$$

Now we can continue to compute the expectation. The expectation of a sum is the sum of the expected values of its terms (expectation is a linear operation), so let us focus on the terms in sum first

$$\begin{aligned}
E[e^{j(\theta_i-\theta_\ell)}] &= E[e^{j\phi}] \\
&= \int e^{j\phi} p_\Phi(\phi) d\phi \\
&= \int_{-2\pi}^{0} e^{j\phi}(\frac{1}{4\pi^2}\phi + \frac{1}{2\pi})d\phi + \int_{0}^{2\pi} e^{j\phi}(-\frac{1}{4\pi^2}\phi + \frac{1}{2\pi})d\phi.
\end{aligned}$$

Recall,

$$\int e^{cx} dx = \frac{1}{c} e^{cx} + C,$$

where c is some constant, and C is the constant of integration. Also

$$\int x e^{cx} dx = e^{cx}(\frac{cx-1}{c^2}).$$

Thus

$$\begin{aligned}
E[e^{j\phi}] &= \frac{1}{4\pi^2}\int_{-2\pi}^{0} \phi e^{j\phi} + \frac{1}{2\pi}\int_{-2\pi}^{0} e^{j\phi} d\phi \\
&+ \frac{-1}{4\pi^2}\int_{0}^{2\pi} \phi e^{j\phi} + \frac{1}{2\pi}\int_{0}^{2\pi} e^{j\phi} d\phi \\
&= \frac{1}{4\pi^2} e^{j\phi}(\frac{j\phi-1}{j^2})|_{-2\pi}^{0} + \frac{1}{2\pi}\frac{1}{j} e^{j\phi}|_{-2\pi}^{0} \\
&+ \frac{-1}{4\pi^2} e^{j\phi}(\frac{j\phi-1}{j^2})|_{0}^{2\pi} + \frac{1}{2\pi}\frac{1}{j} e^{j\phi}|_{0}^{2\pi} \\
&= \frac{1}{4\pi^2} e^{j\phi}(1-j\phi)|_{-2\pi}^{0} + \frac{1}{2\pi}\frac{1}{j}(e^{0} - e^{-2\pi j}) \\
&+ \frac{-1}{4\pi^2} e^{j\phi}(1-j\phi)|_{0}^{2\pi} + \frac{1}{2\pi}\frac{1}{j}(e^{2\pi j} - e^{0}) \\
&= \frac{1}{4\pi^2}\left(e^{0}(1-j(0)) - e^{2\pi j}(1-j(-2\pi))\right) \\
&+ \frac{-1}{4\pi^2}\left(e^{2\pi j}(1-j(2\pi)) - e^{0}(1-j(0))\right) \\
&= \frac{1}{4\pi^2}(1-1-2\pi j) - \frac{1}{4\pi^2}(1-2\pi j-1) \\
&= \frac{-2\pi j}{4\pi^2} + \frac{2\pi j}{4\pi^2} = 0.
\end{aligned}$$

Then we have

$$\begin{aligned}
E[|D_N|^2] &= E[L + \sum_{i\neq\ell} e^{j(\theta_i-\theta_\ell)}] \\
&= E[L] + \sum_{i\neq\ell} E[e^{j(\theta_i-\theta_\ell)}] \\
&= \rho^2 bv\tau + \sum 0 \\
&= \lambda.
\end{aligned}$$

Therefore

$$I_N = \sqrt{\lambda}.$$

Any threshold related to the noise floor should be proportional to I_N.

Note when working with actual data, we do not typically know ρ a-priori, thus we must estimate it. Let $M_{W(t,\tau_k)}$ denote the number of pulses found in the window $W(t,\tau_k)$. Then we estimate

$$\rho \approx \frac{M_{W(t,\tau_k)}}{v\tau_k}.$$

This makes

$$\lambda \approx \left(\frac{M_{W(t,\tau_k)}}{v\tau_k}\right)^2 bv\tau_k = \frac{b}{v\tau_k} M^2_{W(t,\tau_k)}.$$

[48, 22] relates this term to the probability of false alarm using the threshold Γ

$$P_{fa} = \Pr\{|D_N| > \Gamma\} = 1 - \Gamma \int_0^\infty J_1(\Gamma s)\exp(\lambda(J_0(s)-1))ds, \tag{3.3.4}$$

where J_0 and J_1 are Bessel functions of order 0 and 1 respectively. The complete derivation can be found in Appendix A of [48]; as a brief summary, it uses the characteristic functions to get an expression for the pdf of $|D_N|$ again assuming the number of arrivals is Poisson as above. Though our values for λ are computed differently, (3.3.4) requires no modification for our use. Importantly, we want to be able to determine Γ, but (3.3.4) cannot be solved for Γ directly. Instead we build a table of P_{fa} for values λ, Γ, which we can use later to look up Γ for λ at the desired P_{fa}. Values for $\Gamma \in [0,10]$ and $\lambda \in [0.20]$ are shown in Figure 3.3.1. The general trend is as Γ increases, P_{fa} decreases, and Γ must be higher for the same P_{fa} as λ increases. That is, let P^*_{fa} be the probability of false alarm for λ^*, Γ^*, then for $\lambda > \lambda^*$, we require $\Gamma > \Gamma^*$ to achieve P^*_{fa} for λ, Γ.

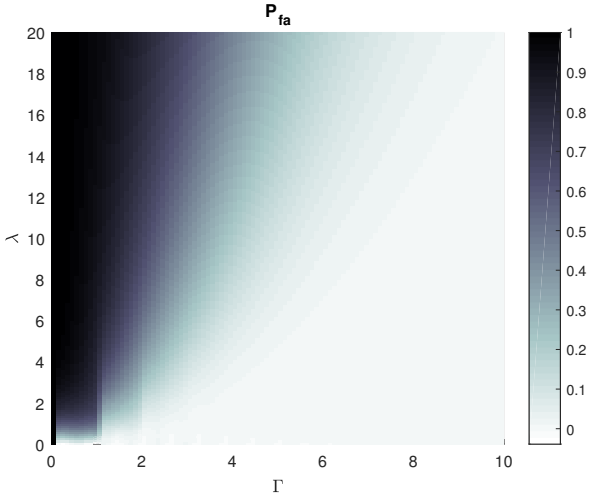


Figure 3.3.1: This is a plot of the P_{fa} for λ and Γ values as shown.

Note for Figure 3.3.1, we needed to compute the integral in (3.3.4). We approximated the integral using the rectangular approximation

$$\int f(x)dx \approx \sum_i f(x_i)\Delta x,$$

where $\Delta x = x_i - x_{i-1}$. Furthermore, the bounds of integration are $(0, \infty)$, but this is okay since it works out that there exists some b such that

$$\int_b^\infty J_1(\Gamma s)\exp(\lambda(J_0(s)-1))ds = 0$$

because Bessel functions are decreasing functions. That is there exists some b such that $\int_b^\infty J_v(z) = 0$. In practice, we found $b = 2000$ to be abundantly sufficient for the values we considered (also we used $\Delta x = 0.01$). A typical example of the integrand can be seen in Figure 3.3.2. For $\lambda = 20$ and $\Gamma = 0.5$, there is no change in the function value is essentially 0 when $s > 1$, and so there would be benefit in those terms.

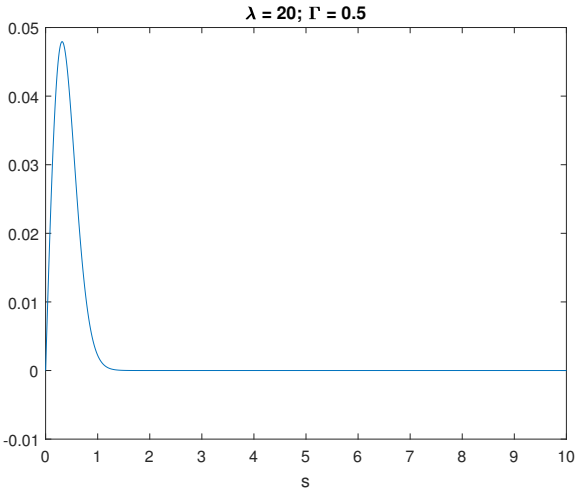
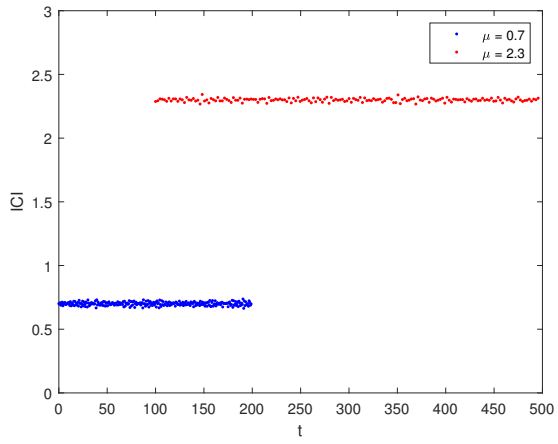
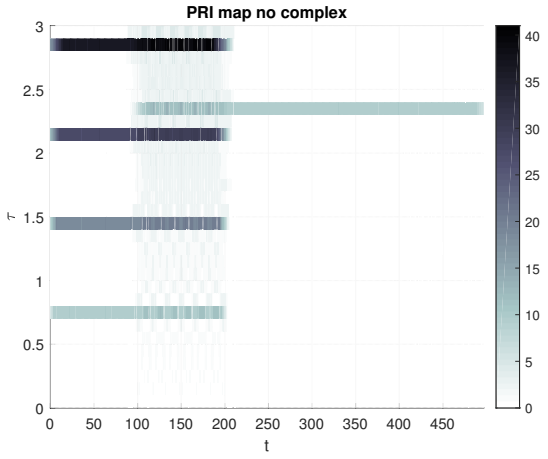


Figure 3.3.2: This is a plot of $J_1(\Gamma s)\exp(\lambda(J_0(s)-1))$.

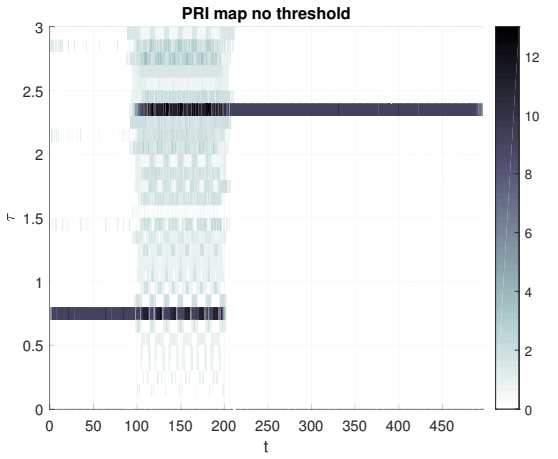
Finally, we show an example of the threshold Γ for $P_{fa} = 0.05$ applied to a PRI map on simulated data in Figure 3.3.3. It cleans up the map to the point we can clearly see where there sources of different ICI are present. This is where the methods of [22, 48] stop, the presence of sources is known, but the total number of sources (e.g. there may be multiple at a detected ICI), amount of jitter (i.e. timing distribution parameters other than μ), and which impulses belong to each source is unknown. This is our novel contribution; we extract the multiplicity and parameters and then we can classify the segments using our timing separation algorithms.



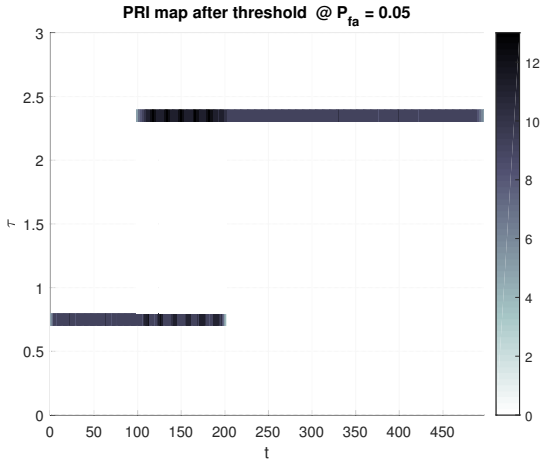
(a) This is a plot of the actual ICI for the example.



(b) This is the PRI map computed using regular autocorrelation.



(c) This is the PRI map.



(d) This is the PRI map after thresholding with Γ for $P_{fa} = 0.05$.

Figure 3.3.3: This is an example of the ICI map for two sources with $\boldsymbol{\Theta} = \{(0.7, 0.01^2), (2.3, 0.01^2)\}$ at times as shown in Figure 3.3.3a. We take the mixture of impulse times and generate the PRI map according to Algorithm 3.3 using $b = 0.1$ and $\tau_j - \tau_{j-1} = 0.1$ and $t_i - t_{i-1} = 1$. The result is shown in Figure 3.3.3b. Also for comparison, we present Figure 3.3.3b which is the same as Figure 3.3.3b but we use $r(s)$ instead of $\rho(s)$. Using $\rho(s)$ clearly reduces the harmonics, but there is still some noise. In Figure 3.3.3d, we show final result after thresholding the map with Γ for $P_{fa} = 0.05$.

3.3.2 Extracting $\boldsymbol{\Theta}$

Recall the goal is to find a set(s) of parameters $\boldsymbol{\Theta}^{[0]}$ to use for Algorithm 2.8. In this section we will discuss how to do this for $|D(t_i, \tau_j)|$ for a single t_i, and in the Section 3.3.3 we will discuss how to process the different time steps together.

For a single t_i, extracting parameters from $|D(t_i, \tau_j)|$ is analogous to the problem Algorithm 3.2 is trying to solve. We approach it in a similar manner: find a threshold based to identify candidates from peaks across τ_j. While it is numerically different, the PRI map is designed to be an estimate for the histogram of only the true interimpulse intervals. We model this by

$$|D(t_i, \tau_j)| \approx M_{W(t_i,\tau_j)} \Pr\{x \in [\tau_j - \frac{b}{2}, \tau_j + \frac{b}{2}]\} \triangleq \hat{D}(t_i, \tau_j), \tag{3.3.5}$$

where $M_{W(t_i,\tau_j)}$ is the true number of ICI[5] in window $W(t_i, \tau_j)$ and x is an ICI (e.g. using notation from Section 2.1, $\tau_{k,i} - \tau_{k,i-1}$,). Note we do not include $|\cdot|$ on $\hat{D}(t_i, \tau_j)$ to simplify notation, but it is the estimate for the magnitude of $D(t_i, \tau_j)$. In the following, we will expand (3.3.5) and use it to help with finding $\boldsymbol{\Theta}$.

If we assume that all sources are Gaussian, then in actuality, the pdf for seeing an interimpulse spacing x is given by the general Gaussian mixture model

$$f(x) = \sum_k \phi_k \mathcal{T}_k(x),$$

where ϕ_k is the mixing proportion and as before $\mathcal{T}_k \sim \mathcal{N}(\mu_k, \sigma_k^2)$. Note, In order to conserve the rules of probability $0 \leq \phi_k \leq 1$ and $\sum_k \phi_k = 1$. Recall then, that probability over the interval $[\tau_j - \frac{b}{2}, \tau_j + \frac{b}{2}]$ is given by.

$$\begin{aligned}
\Pr\{x \in [\tau_j - \frac{b}{2}, \tau_j + \frac{b}{2}]\} &= \int_{\tau_j - \frac{b}{2}}^{\tau_j + \frac{b}{2}} f(x) ds \\
&= \int_{\tau_j - \frac{b}{2}}^{\tau_j + \frac{b}{2}} \left(\sum_k \phi_k \mathcal{T}_k(s) \right) ds \\
&= \sum_k \phi_k \left(\int_{\tau_j - \frac{b}{2}}^{\tau_j + \frac{b}{2}} \mathcal{T}_k(s) ds \right).
\end{aligned} \tag{3.3.6}$$

For the timing problem ϕ_k are greatly influenced by $\mathcal{T}_k$. Loosely, as mentioned in Section 2.2, for a time period T, the number of impulse from a source with mean μ_k is approximately $\frac{T}{\mu_k}$. This means

$$M \approx \sum_k \frac{T}{\mu_k}$$

and correspondingly

$$\phi_k \approx \frac{T/\mu_k}{M}.$$

However, as in Algorithm 3.2, it makes sense to evaluate the peaks in $|D(t_i, \tau_j)|$ individually to determine their validity as candidates for $\boldsymbol{\Theta}$. But the above expressions require knowing all distributions at once. To overcome this difficulty, we will make a few simplifying assumptions and then prove that we get a more conservative but still valid threshold.

[5] Only the ICI arising from individual whales' click trains, not between clicks in trains from different whales.

First we develop an expression for (3.3.5) assuming a single source at with $\mu_k = \tau_j$; a peak in the PRI map indicates a PRI. The probability (3.3.6) becomes

$$\Pr\{x \in [\tau_j - \frac{b}{2}, \tau_j + \frac{b}{2}]\} = \int_{\tau_j - \frac{b}{2}}^{\tau_j + \frac{b}{2}} \mathcal{T}_k(s)ds = \int_{\tau_j - \frac{b}{2}}^{\tau_j + \frac{b}{2}} \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(s-\tau_j)^2}{2\sigma_k^2}} ds. \tag{3.3.7}$$

Given a σ_k, this could be evaluated exactly, but we do not know it. Previously in Section 3.2, to determine σ_k we considered using the average value or estimating it from the width of the peak. An average σ_k may not be known or appropriate in all cases, and for the PRI map relation peak width to σ_k is unclear due to the variable sized bins. So instead we develop an entire expression for $\hat{D}(t_i, \tau_j)$ which we can then optimize for σ_k based on $|D(t_i, \tau_j)|$.

We approximate $M_{W(t_i,\tau_j)}$ with $E[M_{W(t_i,\tau_j)}]$ under the assumption that the source is emitting for the entire time $[t_i - \frac{v}{2}\tau_j, t_i + \frac{v}{2}\tau_j]$ denoted by the duration

$$T = (t_i + \frac{v}{2}\tau_j) - (t_i - \frac{v}{2}\tau_j) = v\tau_j.$$

That is, we need to find how many interimpulse spacings we can fit in T on average. Let $x_{k,i} \sim \mathcal{T}_k$ be the iid interimpulse spacings from source k. Since we let $\mathcal{T}_k = \mathcal{N}(\mu_k, \sigma_k^2)$, then a duration of L_k interimpulse spacings is the sum

$$\sum_{i=1}^{L_k} x_{k,i} \sim \mathcal{N}(L_k\mu_k, L\sigma_k^2). \tag{3.3.8}$$

(it is a well known result that the sum of Gaussian random variables is also a Gaussian random variable where the first and second moments are summed) Recognize, $M_{W(t_i,\tau_j)} = L_k$, so we find

$$E[M_{W(t_i,\tau_j)}] = E[L_k] = \sum_{L_k=1}^{\infty} L_k \Pr\{T - \varepsilon < \sum_{i=1}^{L_k} x_{k,i} \leq T\},$$

where ε is some value. We consider durations in the range $(T - \varepsilon, T)$ to allow for some variation– the last emission from a source may not be exactly at the end of the bin (similarly, the first impulse may not be right at the beginning of the bin). That is, the duration of the click train is approximately $L_k\mu_k$, but it is not guaranteed that there exists a L_k such that $T = L_k\mu_k$ exactly. Since T is a real finite number there will exist some integer L_k for which

$$L_k\mu_k \leq T \leq (L_k + 1)\mu_k.$$

Subtracting μ_k

$$\begin{aligned} L\mu_k - \mu_k \leq \quad & T - \mu_k \quad \leq (L_k + 1)\mu_k - \mu_k \\ (L-1)\mu_k \leq \quad & T - \mu_k \quad \leq L_k\mu_k. \end{aligned}$$

It follows then that

$$T - \mu_k \leq L_k\mu_k \leq T. \tag{3.3.9}$$

In other words, there exists a L_k for which the duration of the click train falls in this interval. Thus, we set $\varepsilon = \mu = \tau_j$.

Continuing, using (3.3.8), we can write

$$E[L_k] = \sum_{L_k=1}^{\infty} L_k \int_{T-\varepsilon}^{T} \frac{1}{\sqrt{2\pi L\sigma_k^2}} e^{-\frac{(s-L_k\mu_k)^2}{2L_k\sigma_k^2}} ds. \tag{3.3.10}$$

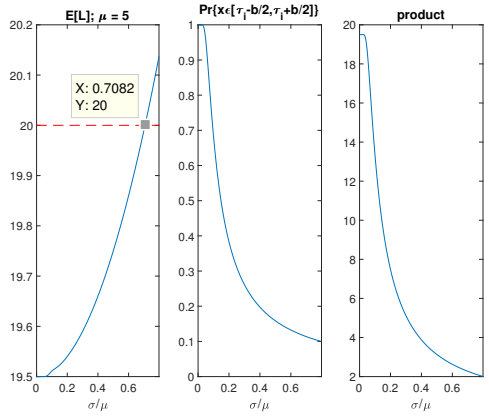
This does not have an analytical expression, but can easily solved numerically given a σ_k. Though it is an infinite sum, there exists some L_k^* such that for $L_k \geq L_k^*$, $\Pr\{T - \varepsilon < \sum_{i=1}^{L_k} x_{k,i} \leq T\} \approx 0$. This is not a rigorous

proof: As L_k grows, the center of the distribution moves more to the right, and eventually the entire mass of the distribution will be outside of the range $(T-\varepsilon, T)$. By monitoring the probability (i.e. the integral in (3.3.10)) during calculation, we can determine L_k^*, and stop computation.

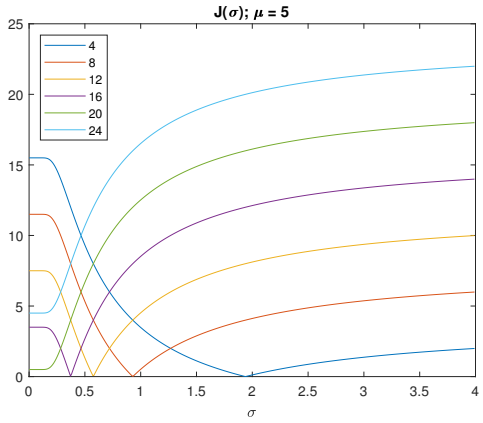
Combining (3.3.7) and (3.3.10), we get a function of σ_k. We can find σ_k by solving

$$J(\sigma_k) = ||D(t_i,\tau_k)| - E[L_k] \int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \mathcal{T}_k(s)ds| = 0. \tag{3.3.11}$$

We do not have an analytical expression for this, and typical numerical optimization methods will not work since the objective function is not convex. Rather than going through a rigorous proof to show that it is not convex, we provide a counter example shown in Figure 3.3.4; clearly, Figure 3.3.4b is not strictly convex. If a single example is not strictly convex, then that is enough to conclude that (3.3.11) is not strictly convex. Another important result shown in Figure 3.3.4b, is that (3.3.11) may never be 0.



(a) We evaluate the functions (left) (3.3.10), (center) (3.3.7), and (right) (3.3.10) $\times$ (3.3.7). For (3.3.10), the red dotted line marks $\frac{T}{\mu}$, and the marker shows that it intersects (3.3.10) at approximately $\frac{\sigma}{\mu} = \frac{1}{\sqrt{2}}$.



(b) We compute (3.3.11) using the result in 3.3.4a for values of $|D(t_i,\tau_j)|$ as shown in the legend.

Figure 3.3.4: Here we show an example of (3.3.11) for $\mu = 5$, $\sigma \in [0.01, 4]$, and $T = 100$ to show that it is not convex. In 3.3.4a, we show the intermediate parts. Since the parts are not strictly convex, the final result is not strictly convex either.

Lack of convexity means the usual gradient descent methods will not work well. However if the range for σ_k are known, a grid search can work. For example, in Figure 3.3.4b, we just evaluated $J(\sigma_k)$ for many values σ_k; it is easy to pick a minimum out of these results. A more robust method is the Golden Section Search (GSS); it imposes no restrictions on the function other than that an extremum exists [49]. The basic idea of GSS is to find an interval that contains the extremum and then successively make it smaller by testing values within the interval.

Next we extend the result to multiple identical sources. First suppose that we have 2 iid sources with $\mu_k = \tau_j$ at time t_i. Then we have

$$E[M_{W(t_i,\tau_j)}] = E[L_1 + L_2] = 2E[L_1]$$

as the expected value of the sum of independent random variables is the sum of their expectations and L_1 and L_2 are identical. This generalizes; if there are K identical sources,

$$E[M_{W(t_i,\tau_j)}] = E[KL_1] = KE[L_1].$$

In the case of identical sources, (3.3.6) actual remains the same as (3.3.7). This is because

$$\phi_k \approx \frac{E[L_k]}{E[M_{W(t_i,\tau_j)}]} = \frac{E[L_1]}{KE[L_1]} = \frac{1}{K}.$$

That is, the mixing proportions are uniform, so

$$\Pr\{x \in [\tau_j - \frac{b}{2}, \tau_j + \frac{b}{2}]\} = \sum_{k=1}^{K} \phi_k \left(\int_{\tau_j - \frac{b}{2}}^{\tau_j + \frac{b}{2}} \mathcal{T}_k(s) ds \right) = \sum_{k=1}^{K} \frac{1}{K} \left(\int_{\tau_j - \frac{b}{2}}^{\tau_j + \frac{b}{2}} \mathcal{T}_k(s) ds \right) = \int_{\tau_j - \frac{b}{2}}^{\tau_j + \frac{b}{2}} \mathcal{T}_k(s) ds.$$

Under this framework, we have

$$J(\sigma_k, K) = ||D(t_i, \tau_k)| - KE[L_1] \int_{\tau_j - \frac{b}{2}}^{\tau_j + \frac{b}{2}} \mathcal{T}_k(s) ds|. \tag{3.3.12}$$

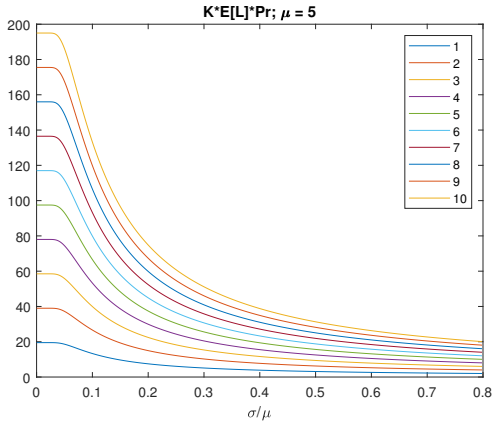
Given a K, we can optimize this for a σ_k as done for (3.3.11). Similarly, given a σ_k, we can optimize this for a K. Figure 3.3.5 shows how the function behaves as both a function of σ_k and K. Let

$$\hat{D}_K(t_i, \tau_k) = KE[L_1] \int_{\tau_j - \frac{b}{2}}^{\tau_j + \frac{b}{2}} \mathcal{T}_k(s) ds. \tag{3.3.13}$$

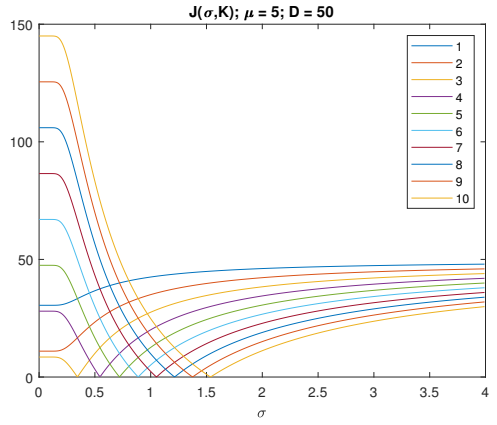
An important feature shown is that even though $\hat{D}_K(t_i, \tau_k)$ increases as K increases,

$$\lim_{\sigma_1 \to \infty} \hat{D}_K(t_i, \tau_k) = 0 \tag{3.3.14}$$

no matter the value of K and μ_1 (i.e. τ_j). The larger, the σ_1, the flatter the distribution becomes, until it eventually becomes 0 (i.e. $K \times 0 = 0$).



(a) We evaluate the function $K \times$ (3.3.10) $\times$ (3.3.7). For (3.3.10). The values of K are shown in the legend.



(b) We compute (3.3.12) using the result in 3.3.5a for the values of K shown in the legend.

Figure 3.3.5: Here we show an example of (3.3.12) for $\mu = 5$, $\sigma \in [0.01, 4]$, and $T = 100$. We arbitrarily set $D(t_i, \tau_j) = 50$. In 3.3.5a, we show the intermediate part to show it monotonically increases with K.

On the other end, the maximum predicted height for the bin $W(t_i, \tau_j)$ for K identical sources with $\mu = \tau_j$ is

$$\tilde{D}_K(t_i, \tau_j) = \lim_{\sigma_1 \to 0} KE[L_1] \int_{\tau_j - \frac{b}{2}}^{\tau_j + \frac{b}{2}} \mathcal{T}_k(s) ds.$$

Since,

$$\delta(t) = \lim_{\sigma \to 0} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{t^2}{2\sigma^2}} \tag{3.3.15}$$

where $\delta(t)$ is the Dirac-delta function [50]. That is

$$\delta(t) = \begin{cases} \infty & t = 0 \\ 0 & t \neq 0 \end{cases}$$

and

$$\int_{t_1}^{t_2} \delta(t) = \begin{cases} 1 & 0 \in [t_1, t_2] \\ 0 & \text{otherwise} \end{cases}.$$

Using the product rule for limits we can express $E[L_1]$ and $\int_{\tau_j - \frac{b}{2}}^{\tau_j + \frac{b}{2}} \mathcal{T}_k(s) ds$ as $\sigma_1 \to 0$ then get exact expression for $\tilde{D}(t_i, \tau_j)$. First

$$\lim_{\sigma_1 \to 0} E[L_1] = \lim_{\sigma_1 \to 0} \sum_{L_1=1}^{\infty} L_1 \int_{T-\varepsilon}^{T} \frac{1}{\sqrt{2\pi L_1 \sigma_1^2}} e^{-\frac{(s - L_1\mu_1)^2}{2L_1\sigma_1^2}} ds.$$

Even though the standard deviation of our Gaussian is $\sigma_1\sqrt{L_1}$, (3.3.15) can still be used as

$$\sigma_1 \to 0 \Rightarrow \sigma_1\sqrt{L_1} \to 0$$

for any given L_1. Therefore we have

$$\lim_{\sigma_1 \to 0} E[L_1] = \sum_{L_1=1}^{\infty} L_1 \int_{T-\varepsilon}^{T} \delta(s - L_1\mu_1) ds.$$

Going from the definition of the Dirac-delta function

$$\int_{T-\varepsilon}^{T} \delta(s - L_1\mu_1) ds = \begin{cases} 1 & L_1\mu_1 \in [T-\varepsilon, T] \\ 0 & \text{otherwise} \end{cases}.$$

Letting $\varepsilon = \mu_1$

$$\lim_{\sigma_1 \to 0} E[L_1] = \left\lfloor \frac{T}{\mu_1} \right\rfloor.$$

This result is tied closely to the discussion for (3.3.9). Further

$$\begin{aligned} \lim_{\sigma_1 \to 0} \Pr\{x \in [\tau_j - \frac{b}{2}, \tau_j + \frac{b}{2}]\} &= \lim_{\sigma_1 \to 0} \int_{\tau_j - \frac{b}{2}}^{\tau_j + \frac{b}{2}} \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(s-\tau_j)^2}{2\sigma_1^2}} ds \\ &= \int_{\tau_j - \frac{b}{2}}^{\tau_j + \frac{b}{2}} \delta(s - \tau_j) ds \\ &= 1. \end{aligned}$$

Therefore

$$\tilde{D}_K(t_i,\tau_j) = K\left\lfloor \frac{T}{\tau_j} \right\rfloor$$

which follows our intuition. This leads to the following result:

Fact 3.1. *If*

$$|D(t_i,\tau_j)| < \tilde{D}_K(t_i,\tau_j) = K\left\lfloor \frac{T}{\tau_j} \right\rfloor$$

there exists a σ_1 *such that*

$$\hat{D}_K(t_i,\tau_j) = |D(t_i,\tau_j)|.$$

Proof. As stated, $\tilde{D}_K(t_i,\tau_j)$ is the maximum value for a given K for $\hat{D}_K(t_i,\tau_j)$. Coupling this with (3.3.14) and the understanding that $\hat{D}_K(t_i,\tau_j)$ is continuous in σ, there must exist some σ for which $|D(t_i,\tau_j)|$ is reached. □

This can be applied to understand why some values of K reach 0 while others do not in Figure 3.3.5b.

So far we can compute $\hat{D}_K(t_i,\tau_j)$, but in order to use it as a threshold like in Algorithm 3.2, we need to argue that relevant $|D(t_i,\tau_j)|$ values will surpass it. Consider the following:

Claim 3.1. *Let*

$$\hat{D}_1(t_i,\tau_j) = E[L_1]\int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \mathcal{T}_k(s)ds = \left(\sum_{L_1=1}^{\infty} L_1 \int_{T-\varepsilon}^{T} \frac{1}{\sqrt{2\pi L\sigma_1^2}} e^{-\frac{(s-L_1\tau_j)^2}{2L_1\sigma_1^2}} ds\right)\left(\int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(s-\tau_j)^2}{2\sigma_1^2}} ds\right),$$

denote the predicted bin height $\hat{D}(t_i,\tau_j)$ *assuming a single source with* $\mu=\tau_j$*. Then, assuming that there are a total of* K *sources*

$$\hat{D}_1(t_i,\tau_j) \le \hat{D}(t_i,\tau_j),$$

where $\hat{D}(t_i,\tau_j)$ *is the estimate for the bin height assuming a source with* $\mu=\tau_j$ *and using all of the* $K-1$ *additional sources.*

Proof. In the case of only iid sources this is straightforward from ((3.3.13))

$$\hat{D}_1(t_i,\tau_j) \le \hat{D}_K(t_i,\tau_j)$$
$$E[L_1]\int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \mathcal{T}_k(s)ds \le KE[L_1]\int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \mathcal{T}_k(s)ds$$
$$1 \le K.$$

However if there non-identical sources (i.e. sources with $\mu \neq \tau_j$), then we need to compute

$$\hat{D}(t_i,\tau_j) = E[M_{W(t_i,\tau_j)}]\Pr\{x \in [\tau_j \pm \frac{b}{2}]\}.$$

Under the assumption of independence,

$$E[M_{W(t_i,\tau_j)}] = E[\sum_{k=1}^{K} L_k] = \sum_k E[L_k].$$

Also from ((3.3.6))

$$\Pr\{x \in [\tau_j \pm \frac{b}{2}]\} = \sum_k \phi_k \int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \mathcal{T}_k(s)ds.$$

The mixing proportions are unknown, but we can estimate them

$$\hat{\phi}_k = \frac{E[L_k]}{\sum_j E[L_j]}.$$

Substitute this into the above equations and we get

$$\begin{aligned}
\hat{D}(t_i,\tau_j) &= \left(\sum_n E[L_n]\right)\left(\sum_k \frac{E[L_k]}{\sum_n E[L_n]} \int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \mathcal{T}_k(s)ds\right) \\
&= \sum_k E[L_k] \int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \mathcal{T}_k(s)ds \\
&= \underbrace{E[L_1] \int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \mathcal{T}_1(s)ds}_{\hat{D}_1(t_i,\tau_j)} + \sum_{k=2}^{K} E[L_k] \int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \mathcal{T}_k(s)ds. \qquad (3.3.16)
\end{aligned}$$

Note the second term $\sum_{k=2}^{K} E[L_k] \int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \mathcal{T}_k(s)ds \geq 0$ since all of its parts are non-negative. Therefore it is trivial to see that indeed

$$\begin{aligned}
\hat{D}_1(t_i,\tau_j) &\leq \hat{D}_1(t_i,\tau_j) + \sum_{k=2}^{K} E[L_k] \int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \mathcal{T}_k(s)ds = \hat{D}(t_i,\tau_j). \\
0 &\leq \sum_{k=2}^{K} E[L_k] \int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \mathcal{T}_k(s)ds
\end{aligned}$$

□

Corollary 3.2. *Let $\hat{D}_N(t_i,\tau_j)$ denote ((3.3.13)) for N iid sources. Then $K \geq N$,*

$$\hat{D}_N(t_i,\tau_j) \leq \hat{D}(t_i,\tau_j).$$

Proof. Assume that the sources are numbered such that $k = 1,2,\ldots,N$ all have $\mu_k = \tau_j$, and the sources have $\mu_k \neq \tau_j$ for $k = N+1,\ldots,K$. Consider again (3.3.16)

$$\begin{aligned}
\hat{D}(t_i,\tau_j) &= \sum_k E[L_k] \int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \mathcal{T}_k(s)ds \\
&= \sum_{n=1}^{N} E[L_1] \int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \mathcal{T}_1(s)ds + \sum_{k=N+1}^{K} E[L_k] \int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \mathcal{T}_k(s)ds \\
&= \underbrace{NE[L_1] \int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \mathcal{T}_1(s)ds}_{\hat{D}_N(t_i,\tau_j)} + \sum_{k=N+1}^{K} E[L_k] \int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \mathcal{T}_k(s)ds.
\end{aligned}$$

Again, the second term $\sum_{k=N+1}^{K} E[L_k] \int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \mathcal{T}_k(s)ds$ is non-negative, so it is trivial to see that indeed

$$\hat{D}_N(t_i,\tau_j) \leq \hat{D}(t_i,\tau_j).$$

□

That is, ignoring contributions from other possible sources with $\mu \neq \tau_j$ we will underestimate $\hat{D}(t_i, \tau_j)$. Going back to (3.3.5), we assume $\hat{D}(t_i, \tau_j) \approx |D(t_i, \tau_j)|$. Then it follows $\hat{D}_N(t_i, \tau_j) \leq |D(t_i, \tau_j)|$, so it can be used as a threshold. To avoid confusion from here on, we will use N to indicate the multiplicity of the source with $\mu = \tau_j$, and K will denote the total number of sources as usual. This is why the subscript on (3.3.13) is N instead of K in the previous inequality.

Note, underestimation also means in practice when we optimize (3.3.12) we might use a smaller σ_k or larger K to match $|D(t_i, \tau_j)|$. To overcome this ambiguity caused by the double jeopardy, we will output up to two sets of $(\mu, \sigma), N$ for each τ_j. There are three possibilities for the relationship of $\hat{D}_N(t_i, \tau_j)$ and $|D(t_i, \tau_j)|$:

1. $\hat{D}_N(t_i, \tau_j) < |D(t_i, \tau_j)|$; the PRI map value passes the threshold
2. $\hat{D}_N(t_i, \tau_j) = |D(t_i, \tau_j)|$; the PRI map value and the threshold are the same
3. $\hat{D}_N(t_i, \tau_j) > |D(t_i, \tau_j)|$; the PRI map value is below the threshold

We save at most one parameter set from cases 1 and one from case 2 since they are the cases that "satisfy" the threshold $\hat{D}_N(t_i, \tau_j)$. Importantly, there may be many values of N that fall in case 1, and similarly for case 2. For example, see Figure 3.3.5. When $N = 1, 2$, $\hat{D}_N(t_i, \tau_j) < |D(t_i, \tau_j)|$ for all σ. Intuition is that we should save parameter set that gets closest to $|D(t_i, \tau_j)|$ which usually (if not always) corresponds to the larger N. This covers case 1. We are in the second case when $N > 2$; there exists a σ such that $\hat{D}_N(t_i, \tau_j) = |D(t_i, \tau_j)|$, a realization of Fact 3.1. In order to pick which N to save parameters from, we apply the intuition that simpler is better and pick the smallest N. This is analogous to the use of MDL. In total this means we can test each $|D(t_i, \tau_j)|$ with $\hat{D}_N(t_i, \tau_j)$ with increasing N and we should encounter case 1 and/or case 2 to get the set(s) of parameters. Case 3, where the PRI map value is below $\hat{D}_N(t_i, \tau_j)$, indicates there is no source with $\mu = \tau_j$ in the signal, and increasing N will not change this. So testing higher N is frivolous (i.e. we should stop). The processed is outlined in Algorithm 3.4.

Algorithm 3.4 Method for extracting parameter sets and multiplicities from $|D(t_i, \tau_j)|$

To extract parameters $(\theta_1, N_1), (\theta_2, N_2)$ from a source with $\mu_k = \tau_j$ exists considering $|D(t_i, \tau_j)|$, set $N = 1$

1. Find σ_k by optimizing (3.3.12) at N over the predefined range $[\sigma_{LB}, \sigma_{UB}]$ (e.g. use golden section search)
2. Use σ_k to compute $\hat{D}_N(t_i, \tau_j)$
3. Check $D_t = |D(t_i, \tau_j)| - \hat{D}_N(t_i, \tau_j)$
   (a) If $D_t > 0$, it means we pass the threshold. Save $N_1 = N$ and $\theta_1 = (\tau_j, \sigma_k)$.
   (b) If $D_t \leq 0$, the predicted value is at or below the threshold
      i. If $|D_t| < \varepsilon$, where ε is some arbitrary small number, we say that the threshold is met. Save $N_2 = N$ and $\theta_2 = (\tau_j, \sigma_k)$
      ii. Stop looking for terms
4. $N++$ and go back to 1. (we only reach this point if case (a) in previous step).

At the end (θ_1, N_1) and/or (θ_2, N_2) may have been unassigned.

Again Algorithm 3.4 will be applied to peaks values of $|D(t_i, \tau_j)|$. For a given t_i, if p is the total number of peaks, there will be at most 2^p sets of parameters to test. At each p, we need to choose one set or the other– a binary decision. For an n-length binary word, there are 2^n possible words (leaves of the binary tree). These binary words corresponds exactly to our parameter sets. Let $\tilde{\Theta}$ denote the *super parameter set* which is the collection of at

most 2 estimates parameter sets for each peaks found in the PRI map (this is more clearly defined in Section 3.3.3). We output the $\mathbf{\Theta}^{[0]}$ from the p-length binary words (removing empty sets).

Finally, we can improve performance (i.e. reduce the number of peaks we need to evaluate) by first considering the threshold Γ based on the noise floor from [48, 22] described in Section 3.3.1 before applying Algorithm 3.4.

3.3.3 Partitioning the PRI map

In the previous section we discussed how to generate the super parameter set $\tilde{\Theta}_i$ for time t_i, but it would not make sense to apply our timing separation algorithms to the impulses at just time t_i (or even $t_i - t_{i-1}$)– there might only be one impulse to classify! It would make more sense to consider at least the impulses contained in the window $W(t_i, \tau_j)$, but this is also ambiguous since we have multiple τ_j to consider (though perhaps we could just use the largest window). Even if we could use $W(t_i, \tau_j)$, there will only be on the order of v impulses (e.g. 10). As seen in Section 2.2.3, the performance of our algorithms is unpredictable with very short segments. Furthermore, trying to make sense of many short segments of assignments is almost as troublesome as assigning the impulses in the first place. Instead, the goal is to process as many t_i together as possible; in this section we will discuss how to determine which t_i should be associated based on their $\tilde{\Theta}_i$.

First let us introduce the notation. Let $\tilde{\mathbf{\Theta}}$ denote the set of super parameters. That is

$$\tilde{\mathbf{\Theta}} = \begin{bmatrix} \tilde{\Theta}_1 & \tilde{\Theta}_2 & \cdots & \tilde{\Theta}_{N_T} \end{bmatrix},$$

where N_T is the number of total number of time steps for the duration of the map T (i.e. $T = t_{N_T} - t_i$) and each

$$\tilde{\Theta}_i = \begin{bmatrix} \theta_{i,1}^{(pk)} & \theta_{i,2}^{(pk)} & \cdots & \theta_{i,M_i}^{(pk)} \end{bmatrix},$$

where M_i is the number of peaks that pass Algorithm 3.4 at t_i, and then each

$$\theta_{i,j}^{(pk)} = \left[\hat{\theta}_{i,j,1} | \hat{\theta}_{i,j,2} \right]$$

corresponds to the parameter sets and multiplicities at $\mu_k = \tau_j$. That is,

$$\hat{\theta}_{ijk} = \left\{ (\mu_{ijk}, \sigma_{ijk}^2), N_{ijk} \right\}$$

where we have dropped the commas from the subscript for brevity and N_{ijk} denotes the multiplicity of the source with parameters $(\mu_{ijk}, \sigma_{ijk}^2)$. Note it is possible for

$$\hat{\theta}_{ijk} = \emptyset,$$

however if $\hat{\theta}_{ij1} = \emptyset$, $\hat{\theta}_{ij2} \neq \emptyset$ and vice versa (when both are empty, that peak's entry is pruned).

First we will check $\tilde{\Theta}_i$ sequentially in increasing i and check to see if $\tilde{\Theta}_i \equiv \tilde{\Theta}_{i-1}$. That is, if $\tilde{\Theta}_{i-1}$ contain all of $\hat{\theta}_{ijk}$ (but perhaps at different j, k) and no other parameters, then we say that $\tilde{\Theta}_i$ *completely matches* the previous time step $\tilde{\Theta}_{i-1}$ and are a part of the same *segment* or *partition*. In the end, we will process each segment as one set of of mixed impulse trains. Note in practice, we adopted the notion of only comparing μ and N, since they take set values. Whereas σ can be any value (within a range) making finding equality difficult. Also as noted in other parts of this chapter, supplying the wrong σ is not very detrimental especially if it is overestimated.

However, completely matching segments alone may not give enough segments that are long enough to get meaningful results from our classification algorithms. So we introduce the concept of a *partial match*. We say $\tilde{\Theta}_i$ is a partial match of $\tilde{\Theta}_{i-1}$ if in $\tilde{\Theta}_{i-1}$ there exists at least one $\hat{\theta}_{ijk}$. This concept is extended to the more useful *partial matching segment*. We say that two completely matching segments

$$S_1 = \{\tilde{\Theta}_i, \tilde{\Theta}_{i+1}, \ldots, \tilde{\Theta}_{i+L_1}\}$$

and

$$S_2 = \{\tilde{\Theta}_j, \tilde{\Theta}_{j+1}, \ldots, \tilde{\Theta}_{j+L_2}\},$$

of length L_1 and L_2 respectively are partial matching segments if:

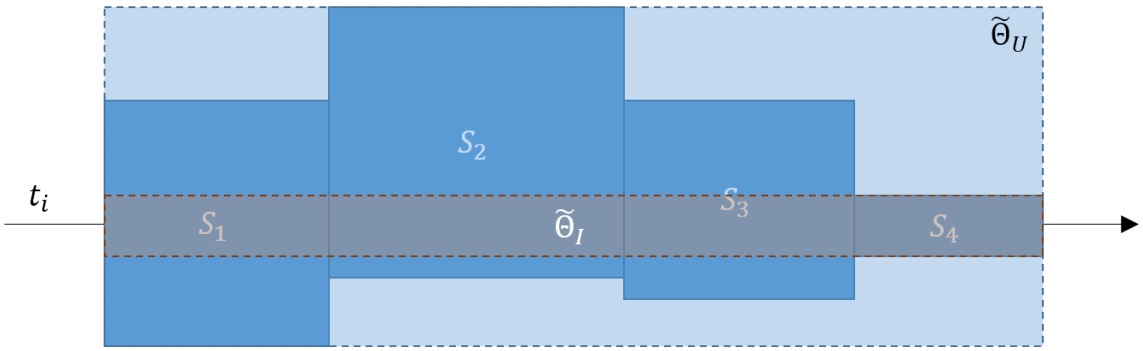
- They are non-overlapping. Without loss of generality assume $i < j$, then segments are non-overlapping if $t_{i+L_1} < t_j$.
- $\tilde{\Theta}_j$ is a partial match of $\tilde{\Theta}_{i+L_1}$.

For this partial matching segment, we define *union super parameter set*, $\tilde{\Theta}_U$, to be the union of parameter sets and multiplicities from the completely matching segments. This is what we will draw $\mathbf{\Theta}^{[0]}$ from later. Also we define $\tilde{\Theta}_I$, the *minimum matching parameters set*, as the intersection of parameter sets and multiplicities from the completely matching segments. The minimum matching parameter set is used to determine if the partial matching segment can be extended with the next segment. Specifically, when all sources are different from the initial partial match even if the last segments contain a partial match, we should just start a new segment instead. Figure 3.3.6 illustrates the what qualifies as a partial match. This is the best we can do if we make sequential decisions about segments, but perhaps there are better segmentation schemes (e.g. try to optimize the average length of partial matching segments). We outline the entire process of segmenting the PRI map into these partial matching segments, ζ, in Algorithm 3.5.

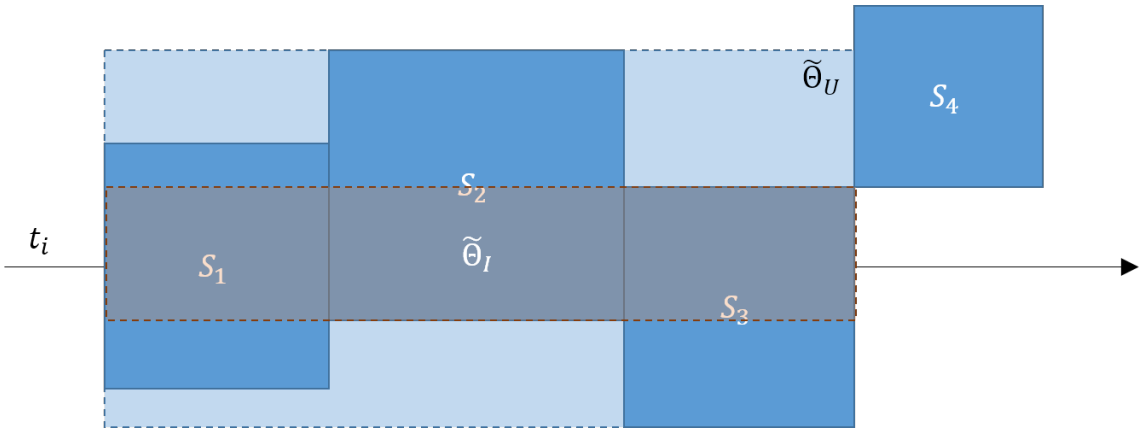
At the end of Algorithm 3.5, we will have a set of partial matching segments $\{\zeta_n\}$ for which we have representative parameters $\tilde{\Theta}_{U_n}$. As mentioned previously we utilize $\tilde{\Theta}_{U_n}$ to generate the $\mathbf{\Theta}^{[0]}$ for the segment to be used in Algorithm 2.8. It is difficult to quantify the performance of this algorithm without looking at actual data and the resulting classification error rates. For this we direct you to Section 6.1.

On an important note, this segmentation algorithm is based on the notion that the timing separation algorithm is able to handle missing impulses especially in long durations. Consider Figure 3.3.6 again. There are parameters in S_1 that are missing for the entire time of S_2 that show up again in S_3. This corresponds to a gap or drop out of the sources corresponding to these parameters. Furthermore, while it is implied in the figure, there is no requirement that S_i, S_{i+1} are adjacent. Any amount of non-zero time may pass between the ending of S_i and the beginning of S_{i+1}.

While the methods of Chapter 2 can tolerate some missing impulses (e.g. Section 6.2), they were not designed to handle them directly. Though, perhaps A2.4 has some potential to work on gaps. The idea for the PRI map was to use it for practical applications in Chapter 6. However, it is interesting to and illustrative to consider how this method works on the ideal timing data. For more on this see Section 3.3.4.



(a) This is an example of $\zeta = \{S_1, S_2, S_3, S_4\}$.



(b) This is an example of $\zeta = \{S_1.S_2, S_3\}$; S_4 is not included since $\tilde{\Theta}_I \cap S_4 = \emptyset$.

Figure 3.3.6: Here we illustrate what qualifies as a partial matching segment and show how $\tilde{\Theta}_U$ and $\tilde{\Theta}_I$. Let the horizontal direction indicate time, increasing left-to-right, and let the vertical direction indicate unique parameter sets. That is, if two segments occupy the same vertical height, then they have the corresponding parameter set in common. The solid lined boxes indicate completely matching segments and the dotted lined boxes indicate $\tilde{\Theta}_U$ and $\tilde{\Theta}_I$ as labeled.

Algorithm 3.5 PRI map segmentation for partial segments

Given $\tilde{\mathbf{\Theta}}$ for a PRI map, we partition the times into matching segments S_j, and then partial matching segments ζ_j. First to find matching segments, set segment index $j = 1$ and initialize segment 1 as $S_1 = \{\tilde{\Theta}_1\}$ with representative super parameter $\tilde{\Theta}_{S_1} = \tilde{\Theta}_1$. Also set time index $i = 2$.

1. Check $\tilde{\Theta}_i \equiv \tilde{\Theta}_{S_j}$
   (a) if true, then add $\tilde{\Theta}_i$ into $\tilde{\Theta}_{S_j}$
   (b) if false, then we start a new segment
      i. $j++$
      ii. Initialize $S_j = \{\tilde{\Theta}_i\}$ with representative $\tilde{\Theta}_{S_j} = \tilde{\Theta}_i$
2. $i++$ and repeat until no more $\tilde{\Theta}_i$ to process

If the durations of S_j are satisfactory, then you do not need to proceed. But if the segments are too short, we merge the completely matching segments into partial matching segments. By:
Initialize the first partial matching segment as $\zeta_1 = \{S_1\}$ with representatives union match $\tilde{\Theta}_{U_1} = \tilde{\Theta}_{S_1}$ and minimum match $\tilde{\Theta}_{I_1} = \tilde{\Theta}_{S_1}$. We reuse the index variables j as the partial segment match index and i as the complete match segment index. Set $j = 1$ and $i = 2$.

1. Compare $\tilde{\Theta}_{S_i}$ with $\tilde{\Theta}_{I_j}$
   (a) If they are a partial match
      i. Add S_i to ζ_j
      ii. Update $\tilde{\Theta}_{U_j} = \tilde{\Theta}_{U_j} \cup \tilde{\Theta}_{S_i}$
      iii. Update $\tilde{\Theta}_{I_j} = \tilde{\Theta}_{I_j} \cap \tilde{\Theta}_{S_i}$
   (b) Otherwise start a new partial segment
      i. $j++$
      ii. $\tilde{\Theta}_{U_j} = \tilde{\Theta}_{S_i}$
      iii. $\tilde{\Theta}_{I_j} = \tilde{\Theta}_{S_i}$
2. $i++$ and repeat until no more $\tilde{\Theta}_{S_i}$ to process

3.3.4 Direct application to $\rho(s)$

The complex autocorrelation $\rho(s)$ was discussed at length in Section 3.2.3, however a method to extract $\mathbf{\Theta}^{[0]}$ was never established. Though since it is analogous to the $\delta\tau$-histogram, Algorithm 3.2 or any of its variants could be used. In this section we discuss the adaptation of the method in Section 3.3.2 to give an alternative for parameter estimation using $\rho(s)$.

Algorithm 3.4 is based on (3.3.5) which is the idealization of $|D(t_i,\tau_j)|$ as a histogram of the only the true interimpulse spacings. We can say the same for $\rho(s)$ and apply Algorithm 3.4 directly. We define bins around τ_j in the same manner (i.e. $[\tau_j - \frac{b}{2}, \tau_j + \frac{b}{2}]$) to compute

$$\rho[\tau_j] = \int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} \rho(s)ds$$

which is used as the substitute of $|D(t_i,\tau_j)|$. Here we use $[\cdot]$ to differentiate the value in the bin centered at the distinct τ_j from the continuous evaluation of $\rho(s)$.

Additionally, we can also apply the threshold from the noise floor (3.3.4) to reduce the number of peaks in $\rho[\tau_j]$ we need to process. Previously,

$$E[L] \triangleq \lambda = \rho v\tau_j \times \rho b = \rho^2 bv\tau_j$$

where ρ, the impulse density, is estimated as

$$\rho \approx \frac{M_{W(t_i,\tau_j)}}{v\tau_j}.$$

While we still have b, we no longer have v or $W(t_i,\tau_j)$. Instead in this case we are using the entire duration of the signal T (rather than $v\tau_j$), so we adjust

$$\rho \approx \frac{M}{T},$$

and

$$\lambda = \rho T \times \rho b = \rho^2 Tb.$$

Due to our uniform definition of bins for τ_j, λ will be the same value for every τ_j. That is, we get the same threshold value for every bin. Note, sometimes ρ will be a small number (most impulsive signals are sparse) which makes λ also small and Γ small in turn. When Γ is too low, it does not reject any values.

To improve results (especially in the case of low Γ), we add in a third criterion from [23] based on the normal autocorrelation (3.2.4) which we discretize to

$$r[\tau_j] = \int_{\tau_j-\frac{b}{2}}^{\tau_j+\frac{b}{2}} r(s)ds.$$

Due to the complex exponential in (3.2.4), it is clear that

$$|\rho[\tau_j]| \leq r[\tau_j].$$

However, in Section 3.2.3 we argued that almost always

$$|\rho(\ell\mu)| \ll r(\ell\mu)$$

for the $\ell > 1$ harmonics, but

$$|\rho(\mu)| \approx r(\mu).$$

This means we can use $r[\tau_j]$ as a threshold for $|\rho[\tau_j]|$ to further help isolate fundamental PRI by eliminating peaks from harmonics. Specifically, we use the threshold

$$\beta r[\tau_j], \tag{3.3.17}$$

where β is a user defined constant. Like in [23], we found $\beta = 0.15$ to be generally satisfactory though experimentation. Though in some situations (e.g. a single source), we needed to reduce β in order to find any peaks. We incorporate this modification into the total process listed in Algorithm 3.6. The resulting $\tilde{\Theta}$ can be used to generate the $\mathbf{\Theta}^{[0]}$ from the allowable binary words. An example of the application of Algorithm 3.6 is shown in Figure 3.3.7.

Algorithm 3.6 Parameter estimation for $\rho[\tau_j]$

Given the autocorrelation $r[\tau_j]$ and complex autocorrelation $\rho[\tau_j]$, for a time signal of length T containing M impulses with user defined P_{fa} and β

1. Find the noise threshold Γ
   (a) Compute $\rho = \frac{M}{T}$
   (b) Compute $\lambda = \rho^2 T b$
   (c) For the choice of P_{fa} look up Γ for the associated λ (e.g. Figure 3.3.1)
2. Apply Γ. That is, define
$$\rho_N[\tau_j] = \begin{cases} \rho[\tau_j] & \rho[\tau_j] > \Gamma \\ 0 & \rho[\tau_j] \leq \Gamma \end{cases}.$$
   (a) If $\rho_N[\tau_j] = 0 \forall \tau_j$, we cannot differentiate the signals from random arrivals given P_{fa}, so stop here.
3. Apply autocorrelation threshold
   (a) Define
$$\rho_\beta[\tau_j] = \begin{cases} \rho_N[\tau_j] & \rho[\tau_j] > \beta r[\tau_j] \\ 0 & \rho[\tau_j] \leq \beta r[\tau_j] \end{cases}$$
   (b) If $\rho_\beta[\tau_j] = 0 \forall \tau_j$ and $\beta > 0$, β is too large. According to step 2, some source should exist. Reduce β (we use $\beta = \beta - 0.1$) and repeat previous step.
   (c) If $\rho_\beta[\tau_j] = 0 \forall \tau_j$ and $\beta = 0$, there is some sort of error, stop here.
4. Find peaks in $\rho_\beta[\tau_j]$
5. For each peak in $\rho_\beta[\tau_j]$ apply Algorithm 3.4 and collect the results into $\tilde{\Theta}$

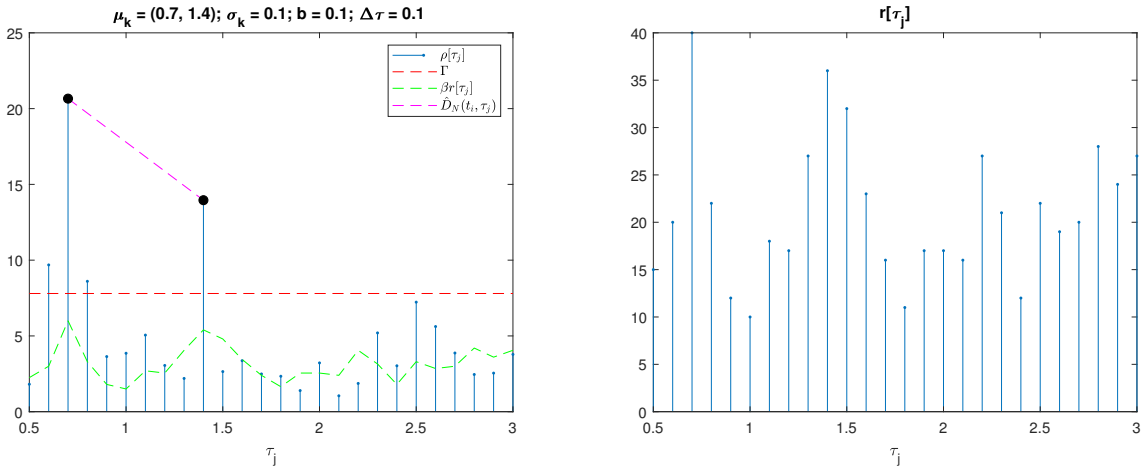


Figure 3.3.7: (Left) This is an example of the application of Algorithm 3.6 applied to a set of 100 mixed impulses with $\boldsymbol{\Theta} = \{(0.7, 0.1^2), (1.4, 0.1^2)\}$. The threshold Γ for $P_{fa} = 0.05$ is shown as a dotted red line. We use $\beta = 0.15$ and the black dots which represent the peaks in $\rho_\beta[\tau_j]$. To compute $\hat{D}_N(t_i, \tau_j)$ we optimize σ in the range $[0.01, 0.5]$. In this case from $\tilde{\Theta}$ we only get one possible $\boldsymbol{\Theta}^{[0]} = \{(0.7, 0.1226^2), (1.4, 0.0878^2)\}$. This $\boldsymbol{\Theta}^{[0]}$ used with AM and A2.3 results in a 0.04 classification error rate. (Right) For comparison we present the corresponding $\delta\tau$-histogram for the same parameters.

3.4 Hough transform

As discussed in [51], the Hough transform can be used to identify PRI from waterfall or raster plots of the observed impulse times. First we will discuss the transform and then its application to extracting PRI.

The Hough transform is an image processing technique used to find straight lines in a binary image. The details of the transform can be found in most image processing texts, such as [52], but this is a brief overview: Assuming that the image is made up of straight lines, every point of brightness is comes from a line. A line can be described completely with two parameters (e.g. point and slope), so for each point, we can collect the parameters for the set of lines it could have come from. Once this is done for every point, count how many times each parameter pair is found (similar to a histogram). The ones with the highest counts correspond to the lines found in the image (essentially the peaks in a histogram).

For an illustrative example, consider an image made up of points from a single line described by

$$y = x + 1, \tag{3.4.1}$$

where y corresponds to the vertical position (e.g. pixel number) and x corresponds to the horizontal position in the image. Specifically, consider the points $(x, y) = (0, 1), (-1, 0), (1, 2)$; these are shown as the black squares in Figure 3.4.1. Visually, it is clear that these points form the line, but we can use the Hough transform to confirm. Here we will consider the parameters m, slope, and b, intercept. Recall that a line can be described with the following equation

$$y = mx + b.$$

Similarly, any point (x, y) can be described by a "line" in the (m, b) parameter space

$$b = y - mx.$$

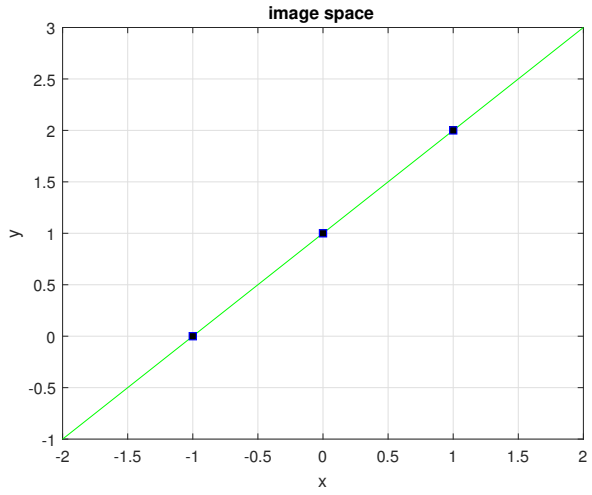


Figure 3.4.1: A simple image made up of three points, shown as black squares, from a line, shown in green but not a part of the image.

If we plug in the points in the image, we get the parameter lines in parameter space (i.e. $b = 1, b = m, b = 2 - m$) as shown in Figure 3.4.2. Note that all of the lines converge at $(m, b) = (1, 1)$ which exactly corresponds to (3.4.1). In terms of parameter accumulation, $(m, b) = (1, 1)$ would have a count of 3, whereas everywhere else would have a count of 1 or 0. Thus we would conclude that the image is composed of the single line (3.4.1). Where there are even more points, the count would even be higher.

In practice, there are some modifications from the methods as described above due to the nature of pixels and to allow for some variation from a perfectly straight line of points. Also the parameters (m, b) are not typically used since it is not possible to describe a vertical line with it; we used them in the example since the point-slope equation is very familiar to most people.

The connection to finding lines in an image and finding the PRI from impulse times lies is the waterfall plot or raster display. Raster displays are how radar pulses are sometimes viewed. In such a display the brightness indicates presence of an impulse and horizontal axis represents time as usual, but in this case the vertical axis also represents time in steps of the width of the display. That is, if all the impulse detection times are on a line, chop the line into segments of length τ(not to be confused with the τ_i denoting impulse time) and stack them sequentially to get the raster display. A scaled waterfall plot with $\tau = 5.5$ for a mixture of sources with PRIs of 5.0 and 5.7 is shown Figure 3.4.3. To be more specific, these sources follow the model as described in Section 2.1, with $\mathcal{T}_1 \sim \mathcal{N}_0(5.0, 0.01^2)$ and $\mathcal{T}_2 \sim \mathcal{N}_0(5.7, 0.01^2)$. Upon inspection, a human would recognize that there are a number of lines, but there are only two different slopes. [51] discusses the exact relationship between PRI and the slopes of the lines.

However, similar to sequential search, when the σ is large, the lines in the waterfall plot breakdown. Consider Figure3.4.4a which is the same as Figure 3.4.3 but with with $\mathcal{T}_1 \sim \mathcal{N}_0(5.0, 0.2^2)$ and $\mathcal{T}_2 \sim \mathcal{N}_0(5.7, 0.2^2)$; the variance is much higher. Some sort of lines are recognizable, but there is considerably more variation. Figure 3.4.4b shows the same plot but with the ideal straight lines overlaid to better see the discrepancies. It is possible that this method could be adapted to work better, but our efforts thus far have not been fruitful.

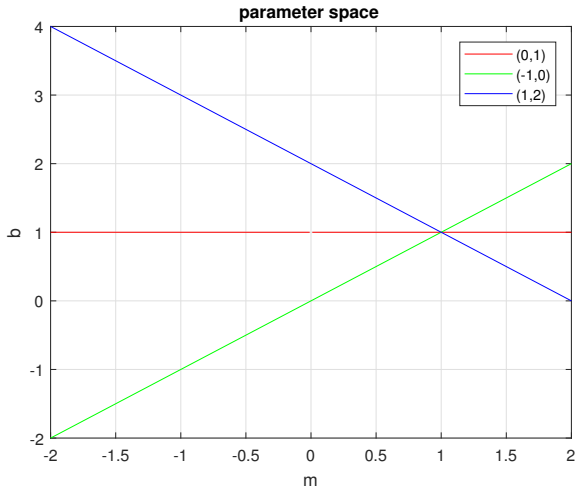


Figure 3.4.2: The (m, b) plot for the points in Figure 3.4.1.

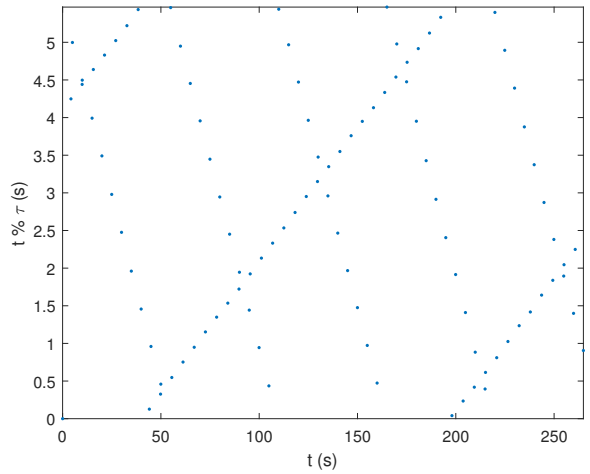
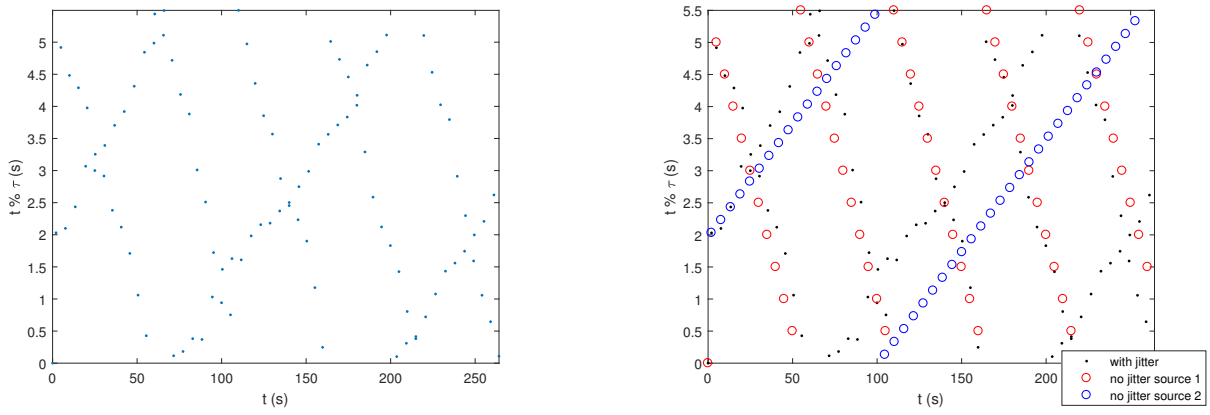


Figure 3.4.3: This is a waterfall plot for two sources with PRIs 5.0 and 5.7 and Gaussian jitter with $\sigma^2 = 0.01^2$.



(a)

(b) Same as 3.4.4a, but with

Figure 3.4.4: This is a waterfall plot (not evenly scaled) for two sources with PRIs 5.0 and 5.7 and Gaussian jitter with $\sigma^2 = 0.2^2$.

4
Detection

Recall that the main motivation of this work is to separate mixtures of sperm whale clicks, but in the previous chapters, we discussed separation of impulses based solely on their timing. This is because the duration of a sperm whale click is very short so a click can be approximated as an impulse, and the time of a click is very easy to pick out just by looking at the amplitude. Identifying clicks and their times is the focus of this chapter.

Since clicks are recognizable just based off of their amplitude, the one idea is to apply a threshold to pick out the spikes. There are many variations and improvements on this idea such as taking the envelope, root mean square (RMS), or a Taeger-Kaiser measure (TK) of the signal or doing a Page test (see Appendix D)[44]. While these methods work rather well (and are what we have used in other sections of this paper), they are generally just picking out high energy points. We would like to see if we can get improved results by combining our classification algorithm with the detection step.

We cast the joint detection-classification problem as a Bayesian detection problem. Consider this system model for the received signal

$$S(t) = \sum_{k=1}^{K} \sum_{i=0}^{N_k-1} h_k(t - \tau_{k,i}) + W(t),$$

where K is the total number of sources , N_k is the number of clicks sent out per source k, h_k is the repeated signal from source k (e.g. the click or the pulse), $\tau_{k,i}$ is the time the i-th signal from source k is emitted, and $W(t)$ is noise. Again we assume the interimpulse spacing $\tau_{k,i} - \tau_{k,i-1} > 0$ is distributed according to some prior known distribution $\mathcal{T}_k(t)$. Further we assume that the support of the signals are much shorter than the pulse spacing, so there is no overlap of pulses. For simplicity, we will assume that the noise is Gaussian; $W(t) \sim \mathcal{N}(0, n_0)$. And as in our timing algorithm we assume that the interimpulse distributions are Gaussian also $\mathcal{T}_k \sim \mathcal{N}(\mu_k, \sigma_k^2)$– ignore the truncation factor for now.

From this signal, we want to estimate the transmitted signals $h_k(t)$, and the times at which they are transmitted $\boldsymbol{\tau}_k = \{\tau_{k,i}\}$.

4.1 Single source

If there are multiple sources, we have to use the timing algorithm to classify the detected clicks. Let us first try to develop a detector based on this model for a single source. Consider the received signal

$$S(t) = \sum_{i=0}^{N-1} h(t-\tau_i) + W(t),$$

with $\tau_i - \tau_{i-1} \sim \mathcal{N}(\mu, \sigma^2)$. From this signal, we want to estimate the transmitted signal $h(t)$, and the times at which they are transmitted $\boldsymbol{\tau} = \{\tau_i\}$.

Random processes are difficult to work with, so instead let us consider the sampled version of this signal. That is

$$S[k] = S(k\Delta) = \sum_{i=0} h[k-\ell_i] + W[k]$$

for $k = 0, \ldots, K_0$ (we use K_0 to distinguish from K which we have been using for total number of sources) where we have substituted $t = k\Delta$ and $\tau_i = \ell_i\Delta$ for integer k, ℓ_i. The assumption is that Δ is small enough such that $|\tau_i - \ell_i\Delta|$ is negligible. Note $\frac{1}{\Delta}$ is the sample rate. When we sample the random process $W(t)$, we get the random vector $W[k]$ where each sample is iid $\sim \mathcal{N}(0, n_0) \Rightarrow \mathbf{W} \sim \mathcal{N}(0, n_0 I)$ where $\mathbf{W}$ is the vector of $W[k]$ for all k. Sampling makes the interimpulse distribution

$$\mathcal{T}(\tau_i - \tau_{i-1}) = \mathcal{T}(\Delta\ell_i - \Delta\ell_{i-1}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(\Delta\ell_i - \Delta\ell_{i-1} - \mu)^2}{2\sigma^2}).$$

We just have to be careful to multiply ℓ_i by Δ before using $\mathcal{T}$.

Let $\mathbf{h}$ denote $\{h[k]\}$ and $\boldsymbol{\ell}$ denote the collection of impulse indexes. Then it makes sense to choose $\mathbf{h}$ and $\boldsymbol{\ell}$ to maximize

$$\Pr\{\mathbf{h}, \boldsymbol{\ell} | \mathbf{S}\} = \frac{\Pr\{\mathbf{S}, \mathbf{h}, \boldsymbol{\ell}\}}{\Pr\{\mathbf{S}\}}.$$

But note when we maximize over $\mathbf{h}$ and $\boldsymbol{\ell}$, the actual value of the denominator becomes irrelevant. Thus we are mainly concerned with maximizing

$$\begin{aligned} \Pr\{\mathbf{S}, \mathbf{h}, \boldsymbol{\ell}\} &= \Pr\{\mathbf{h}, \boldsymbol{\ell}\} \Pr\{\mathbf{S} | \mathbf{h}, \boldsymbol{\ell}\} \\ &= \Pr\{\mathbf{h}\} \Pr\{\boldsymbol{\ell}\} \Pr\{\mathbf{S} | \mathbf{h}, \boldsymbol{\ell}\} \end{aligned}$$

Here we assume that $\mathbf{h}$ and $\boldsymbol{\ell}$ are independent. In terms of sperm whales, the shape $\mathbf{h}$ is a function of individual's anatomy, direction with respect to the hydrophone, and the water properties. The timing of clicks is mostly dependent on the activity the animal is engaging in (e.g. foraging or navigation). Animals may change click shape depending on activity as well (e.g. make it louder), but for now we will ignore this relationship– we would need a considerable amount of knowledge to form a joint pdf. Something similar could be said for radars. In any case, let us analyze these three terms individually.

For the first term, we let $\Pr\{\mathbf{h}\}$ be uniform for all possible impulses. Here *all possible impulses* means $\mathbf{h}$ can be any waveform that adheres to the known constraints. This can be quite involved (e.g. frequency characteristics, interpulse interval, max amplitude), but for now we will only enforce that the length of an impulse is $T_{max} = \Delta H \ll \tau_i - \tau_{i-1}$.

The second term is given by the assumption that interimpulse intervals are iid according to $\mathcal{T}$. Hence

$$\Pr\{\ell\} = \prod_{i=1} \mathcal{T}(\Delta\ell_i - \Delta\ell_{i-1}).$$

This is essentially the objective function we were maximizing in Chapter 2.

For the last term, $\Pr\{\mathbf{S}|\mathbf{h},\boldsymbol{\ell}\}$, since

$$W[k] = S[k] - \sum_i h[k-\ell_i],$$

we have

$$\begin{aligned}\Pr\{\mathbf{S}|\mathbf{h},\boldsymbol{\ell}\} &= (2\pi)^{-\frac{K_0}{2}}|n_0 I|^{-\frac{1}{2}}\exp(-\frac{1}{2n_0}\sum_{k=1}^{K_0}(S[k]-\sum_i h[k-\ell_i])^2)\\ &= (2\pi n_0)^{-\frac{K_0}{2}}\exp(-\frac{1}{2n_0}\sum_{k=1}^{K_0}(S[k]-\sum_i h[k-\ell_i])^2)\end{aligned}$$

As the $\log$ is monotonic increasing, it is common practice to maximize the log-likelihood function. That is we choose $\mathbf{h},\boldsymbol{\ell}$ to maximize

$$\log\Pr\{\mathbf{S},\mathbf{h},\boldsymbol{\ell}\} = \log\Pr\{\mathbf{h}\} + \log\Pr\{\boldsymbol{\ell}\} + \log\Pr\{\mathbf{S}|\mathbf{h},\boldsymbol{\ell}\}$$

In terms of maximization, the first term falls away, except for the fact that it restricts $\mathbf{h}$ to only the impulses we expect. The second term becomes

$$\begin{aligned}\log\Pr\{\ell\} &= \sum_{i=1}\log\mathcal{T}(\Delta\ell_i - \Delta\ell_{i-1})\\ &= \sum_{i=1}\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{(\Delta\ell_i-\Delta\ell_{i-1}-\mu)^2}{2\sigma^2})\right)\\ &= \sum_{i=1}-\frac{1}{2}\log 2\pi\sigma^2 - \frac{(\Delta\ell_i-\Delta\ell_{i-1}-\mu)^2}{2\sigma^2}\\ &= \left(-\frac{1}{2}\log 2\pi\sigma^2 \underbrace{\sum_{i=1}1}_{N}\right) - \frac{1}{2\sigma}\sum_i(\Delta\ell_i-\Delta\ell_{i-1}-\mu)^2\end{aligned}$$

N some constant, but dependent on $\boldsymbol{\ell}$

And the last term is

$$\begin{aligned}\log\Pr\{\mathbf{S}|\mathbf{h},\boldsymbol{\ell}\} &= \log\left((2\pi n_0)^{-\frac{K_0}{2}}\exp(-\frac{1}{2n_0}\sum_{k=1}^{K_0}(S[k]-\sum_i h[k-\ell_i])^2)\right)\\ &= -\frac{K_0}{2}\log(2\pi n_0) - \frac{1}{2n_0}\sum_{k=1}^{K_0}(S[k]-\sum_i h[k-\ell_i])^2\end{aligned}$$

Note that the first term of the last term is just a constant not dependent on $\mathbf{h}$ or $\boldsymbol{\ell}$, so we can ignore it in terms of the maximization. Thus the objective function to maximize is

$$p \triangleq -\frac{1}{2}\log 2\pi\sigma^2(N-1) - \frac{1}{2\sigma}\sum_{i=1}^{N-1}(\Delta\ell_i-\Delta\ell_{i-1}-\mu)^2 - \frac{1}{2n_0}\sum_{k=1}^{K_0}(S[k]-\sum_{i=0}^{N-1}h[k-\ell_i])^2 \tag{4.1.1}$$

Note $N = f(\boldsymbol{\ell})$ is the number of impulses. A common technique to maximize a function is to take its derivative/gradient and set it to 0 and solve– needs to be a concave function to get a global maximum. Rewrite the last

sum

$$\begin{aligned}
\sum_{k=1}^{K_0}(S[k]-\sum_{i=0}^{N-1}h[k-\ell_i])^2 &= \sum_{k=1}^{K_0}(S_k-\sum_{i=0}^{N-1}h_{k-\ell_i})^2 \\
&= \sum_{i=0}^{N-1}\sum_{k=0}^{H-1}(S_{k+\ell_i}-h_k)^2+\sum_{k=0}^{\ell_0-1}S_k^2+\sum_{k=\ell_0+H}^{\ell_1-1}S_k^2+\sum_{k=\ell_1+H}^{\ell_2-1}S_k^2+\cdots \\
&\quad +\sum_{k=\ell_{N-2}+H}^{\ell_{N-1}-1}S_k^2+\sum_{k=\ell_{N-1}+H}^{K_0}S_k^2 \\
&= \sum_{i=0}^{N-1}\sum_{k=0}^{H-1}(S_{k+\ell_i}-h_k)^2+\sum_{k=0}^{\ell_0-1}S_k^2+\sum_{i=1}^{N-1}\sum_{k=\ell_{i-1}+H}^{\ell_i-1}S_k^2+\sum_{k=\ell_{N-1}+H}^{K_0}S_k^2
\end{aligned}$$

Now taking the the partial derivative of the objective with respect to h_q where $q \in \{0,1,2,\ldots,H-1\}$

$$\begin{aligned}
\frac{\partial p}{\partial h_q} &= \frac{\partial}{\partial h_q}\left(-\frac{1}{2}\log 2\pi\sigma^2(N-1)-\frac{1}{2\sigma}\sum_{i=1}^{N-1}(\Delta\ell_i-\Delta\ell_{i-1}-\mu)^2-\frac{1}{2n_0}\sum_{k=1}^{K_0}(S[k]-\sum_{i=0}^{N-1}h[k-\ell_i])^2\right) \\
&= \frac{\partial}{\partial h_q}\left(-\frac{1}{2n_0}\sum_{k=1}^{K_0}(S[k]-\sum_{i=0}^{N-1}h[k-\ell_i])^2\right) \\
&= -\frac{1}{2n_0}\frac{\partial}{\partial h_q}\left(\sum_{i=0}^{N-1}\sum_{k=0}^{H-1}(S_{k+\ell_i}-h_k)^2+\sum_{k=0}^{\ell_0-1}S_k^2+\sum_{i=1}^{N-1}\sum_{k=\ell_{i-1}+H}^{\ell_i-1}S_k^2+\sum_{k=\ell_{N-1}+H}^{K_0}S_k^2\right) \\
&= -\frac{1}{2n_0}\frac{\partial}{\partial h_q}\left(\sum_{i=0}^{N-1}(S_{q+\ell_i}-h_q)^2\right) \\
&= -\frac{1}{2n_0}\sum_{i=0}^{N-1}\frac{\partial}{\partial h_q}\left(S_{q+\ell_i}^2-2S_{q+\ell_i}h_q+h_q^2\right) \\
&= -\frac{1}{2n_0}\sum_{i=0}^{N-1}(-2S_{q+\ell_i}+2h_q) \\
&= +\frac{1}{n_0}\sum_{i=0}^{N-1}(S_{q+\ell_i}-h_q)
\end{aligned}$$

Setting this to 0 and solving

$$\begin{aligned}
0 &= \frac{1}{n_0}\sum_{i=0}^{N-1}\left(S_{q+\ell_i}-\hat{h}_q\right) \\
0 &= \sum_{i=0}^{N-1}S_{q+\ell_i}-N\hat{h}_q \\
\hat{h}_q &= \frac{1}{N}\sum_{i=0}^{N-1}S_{q+\ell_i} \qquad (4.1.2)
\end{aligned}$$

As expected, the estimate for the impulse is the mean of all occurrences of the impulse in the received signal. Since the noise is zero mean, it gets canceled out through averaging. Plugging (4.1.2) back into (4.1.1)

$$\begin{aligned} p &= -\frac{1}{2}\log 2\pi\sigma^2(N-1) - \frac{1}{2\sigma}\sum_{i=1}^{N-1}(\Delta\ell_i - \Delta\ell_{i-1} - \mu)^2 - \frac{1}{2n_0}\sum_{k=1}^{K_0}(S[k] - \sum_{i=0}^{N-1}\hat{h}[k-\ell_i])^2 \\ &= -\frac{1}{2}\log 2\pi\sigma^2(N-1) - \frac{1}{2\sigma}\sum_{i=1}^{N-1}(\Delta\ell_i - \Delta\ell_{i-1} - \mu)^2 \\ &\quad -\frac{1}{2n_0}\left(\sum_{i=0}^{N-1}\sum_{k=0}^{H-1}(S_{k+\ell_i} - \hat{h}_k)^2 + \sum_{k=0}^{\ell_0-1}S_k^2 + \sum_{i=1}^{N-1}\sum_{k=\ell_{i-1}+H}^{\ell_i-1}S_k^2 + \sum_{k=\ell_{N-1}+H}^{K_0}S_k^2\right) \end{aligned}$$

Now we need to optimize over $\{\ell_i\}$ and N. Here it is not easy to take the derivatives with respect to N and ℓ_i. Let us go term by term again to see if we can develop any intuition. Recall we are trying to maximize p.

The first term is linear and decreasing with respect to only N, so we want N as small as possible. In fact, the best value would be $\hat{N} = 0$, but this may cause the last terms to blow up as we will see later.

For the second term assume that $\hat{N} \geq 2$, otherwise the sum is 0. Then we can take the derivative and set to 0. Consider optimization for ℓ_r for $r \in \{1, 2, \ldots, N-1\}$

$$\begin{aligned} \frac{\partial}{\partial\ell_r}\left(-\frac{1}{2\sigma}\sum_{i=1}^{N-1}(\underbrace{\Delta\ell_i - \Delta\ell_{i-1}}_{L} - m)^2\right) &= -\frac{1}{2\sigma}\frac{\partial}{\partial\ell_r}\left(\sum_{i=1}^{N-1}L^2 - 2L\mu - \mu^2\right) \\ &= -\frac{1}{2\sigma}\frac{\partial}{\partial\ell_r}\left(\sum_{i=1}^{N-1}(\Delta\ell_i - \Delta\ell_{i-1})^2 - 2(\Delta\ell_i - \Delta\ell_{i-1})\mu\right) \\ &= -\frac{1}{2\sigma}\frac{\partial}{\partial\ell_r}\left(\sum_{i=1}^{N-1}\Delta^2(\ell_i^2 - 2\ell_i\ell_{i-1} + \ell_{i-1}^2) - 2\Delta\mu(\ell_i - \ell_{i-1})\right) \\ &= -\frac{1}{2\sigma}\left(\Delta^2(2\ell_r - 2\ell_{r-1}) - 2\Delta\mu + \Delta^2(-2\ell_{r+1} + 2\ell_r) - 2\Delta(-1)\mu\right) \end{aligned} \tag{4.1.3}$$

$$\begin{aligned} &= -\frac{1}{2\sigma}2\Delta^2\left(\ell_r - \ell_{r-1} + \ell_r - \ell_{r+1}\right) \\ &= -\frac{\Delta^2}{\sigma}\left(2\ell_r - \ell_{r-1} - \ell_{r+1}\right). \end{aligned}$$

Note last two terms in the fourth line arise from the fact that ℓ_r shows up when $i = r \Rightarrow \ell_i = \ell_r$ and $i = r+1 \Rightarrow \ell_{i-1} = \ell_r$. Now set the derivative to 0

$$\begin{aligned} 0 &= -\frac{\Delta^2}{\sigma}\left(2\hat{\ell}_r - \ell_{r-1} - \ell_{r+1}\right) \\ 0 &= 2\hat{\ell}_r - \ell_{r-1} - \ell_{r+1} \\ \hat{\ell}_r &= \frac{\ell_{r-1} + \ell_{r+1}}{2}. \end{aligned} \tag{4.1.4}$$

This means the ℓ_r should be evenly spaced. The derivative for ℓ_0 is different since it only shows up if $i = 1$ in

$i-1=0$. Substituting this into (4.1.3) from the previous derivative

$$\begin{aligned}\frac{\partial}{\partial \ell_0}\left(-\frac{1}{2\sigma}\sum_{i=1}^{N-1}(\Delta\ell_i-\Delta\ell_{i-1}-m)^2\right) &= -\frac{1}{2\sigma}\left(\Delta^2(-2\ell_1+2\ell_0)-2\Delta(-1)\mu\right)\\ &= -\frac{1}{2\sigma}2\Delta\left(\Delta(-\ell_1+\ell_0)+\mu\right)\\ &= -\frac{\Delta}{2\sigma}\left(\Delta(\ell_0-\ell_1)+\mu\right).\end{aligned}$$

Setting this to 0

$$\begin{aligned}0 &= -\frac{\Delta}{2\sigma}\left(\Delta(\hat{\ell}_0-\ell_1)+\mu\right)\\ -\mu &= \Delta(\hat{\ell}_0-\ell_1)\\ -\frac{\mu}{\Delta} &= \hat{\ell}_0-\ell_1\\ \hat{\ell}_0 &= \ell_1-\frac{\mu}{\Delta}.\end{aligned}$$

Note, we have been sloppy with notation in that ℓ_i are integer valued so when we take ratios we must round. This result means that $\hat{\ell}_0$ should be one mean spacing prior to the next transmission time. Coupling this with (4.1.4), we get that the optimum choice for impulse times, ignoring any shape information, is to have them each spaced exactly μ apart. This is not too surprising.

Now for the last term (i.e. the shape information)

$$\begin{aligned}p_3 &= -\frac{1}{2n_0}\left(\sum_{i=0}^{N-1}\sum_{k=0}^{H-1}(S_{k+\ell_i}-\hat{h}_k)^2+\sum_{k=0}^{\ell_0-1}S_k^2+\sum_{i=1}^{N-1}\sum_{k=\ell_{i-1}+H}^{\ell_i-1}S_k^2+\sum_{k=\ell_{N-1}+H}^{K_0}S_k^2\right)\\ &= -\frac{1}{2n_0}\left(\sum_{i=0}^{N-1}\sum_{k=0}^{H-1}(S_{k+\ell_i}-\frac{1}{N}\sum_{j=0}^{N-1}S_{k+\ell_j})^2+\sum_{k=0}^{\ell_0-1}S_k^2+\sum_{i=1}^{N-1}\sum_{k=\ell_{i-1}+H}^{\ell_i-1}S_k^2+\sum_{k=\ell_{N-1}+H}^{K_0}S_k^2\right).\end{aligned}$$

It is a function of both N and $\boldsymbol{\ell}$. First, consider $N=0\Rightarrow\boldsymbol{\ell}=\emptyset$

$$p_3(N=0)=-\frac{1}{2n_0}\sum_{k=0}^{K_0}S_k^2, \tag{4.1.5}$$

essentially the autocorrelation of the received signal multiplied by a factor involving n_0. Typically, n_0 is unknown, but one option is to use the root mean square of the entire signal

$$\hat{n}_0=\sqrt{\frac{1}{K_0}\sum_{k=0}^{K_0}S_k^2}. \tag{4.1.6}$$

Since the signals will be included, n_0 will be overestimated. However, assuming that the the number and length of impulses are small in comparison to the total length of the signal, the amount over should be inconsequential. In fact with $N=0$, the over all objective function value would just be (4.1.5). The first two terms disappear since they originate from

$$\log\prod_{i=1}^{N-1}\mathcal{T}(\tau_i-\tau_{i-1}),$$

the log-likelihood of the interimpulse intervals. If there are no interimpulse intervals (or impulses), this term should not contribute to the total in any way.

Further consider $N = 1 \Rightarrow \boldsymbol{\ell} = \{\ell_0\}$

$$\begin{aligned} p_3(N=1,\ell_0) &= -\frac{1}{2n_0}\left(\sum_{k=0}^{H-1}(S_{k+\ell_0}-S_{k+\ell_0})^2+\sum_{k=0}^{\ell_0-1}S_k^2+\sum_{k=\ell_0+H}^{K_0}S_k^2\right) \\ &= -\frac{1}{2n_0}\left(\sum_{k=0}^{\ell_0-1}S_k^2+\sum_{k=\ell_0+H}^{K_0}S_k^2\right). \end{aligned}$$

This makes the complete objective function

$$p = -\frac{1}{2n_0}\left(\sum_{k=0}^{\ell_0-1}S_k^2+\sum_{k=\ell_0+H}^{K_0}S_k^2\right).$$

Again the first two terms are lost since there is no interimpulse interval with just one impulse. Recall we are trying to maximize, to see how, rearrange the sum

$$p = -\frac{1}{2n_0}\sum_{k=0}^{K_0}S_k^2+\frac{1}{2n_0}\sum_{k=\ell_0}^{H-1+\ell_0}S_k^2 \tag{4.1.7}$$

The first term is constant for any given received signal and any choice of ℓ_0, and the n_0 factor in the second term can be ignored with respect to optimization. Thus choice ℓ_0 in this case is just dependent on the last sum. This can be computed for every ℓ_0; there are $K_0 - H$ values to check.

Furthermore consider $N = 2 \Rightarrow \boldsymbol{\ell} = \{\ell_0, \ell_1\}$

$$\begin{aligned} p_3(N=2,\{\ell_0,\ell_1\}) &= -\frac{1}{2n_0}\left(\sum_{i=0}^{2-1}\sum_{k=0}^{H-1}(S_{k+\ell_i}-\frac{1}{2}\sum_{j=0}^{1}S_{k+\ell_j})^2+\sum_{k=0}^{\ell_0-1}S_k^2+\sum_{i=1}^{1}\sum_{k=\ell_{i-1}+H}^{\ell_i-1}S_k^2+\sum_{k=\ell_1+H}^{K_0}S_k^2\right) \\ &= -\frac{1}{2n_0}\Bigg(\sum_{k=0}^{H-1}(S_{k+\ell_0}-\frac{1}{2}(S_{k+\ell_0}+S_{k+\ell_1}))^2+\sum_{k=0}^{H-1}(S_{k+\ell_1}-\frac{1}{2}(S_{k+\ell_0}+S_{k+\ell_1}))^2 \\ &\quad +\sum_{k=0}^{\ell_0-1}S_k^2+\sum_{k=\ell_0+H}^{\ell_1-1}S_k^2+\sum_{k=\ell_1+H}^{K_0}S_k^2\Bigg) \\ &= -\frac{1}{2n_0}\Bigg(\sum_{k=0}^{H-1}(\frac{1}{2}S_{k+\ell_0}-\frac{1}{2}S_{k+\ell_1})^2+\sum_{k=0}^{H-1}(\frac{1}{2}S_{k+\ell_1}-\frac{1}{2}S_{k+\ell_0})^2+\sum_{k=0}^{\ell_0-1}S_k^2+\sum_{k=\ell_0+H}^{\ell_1-1}S_k^2 \\ &\quad +\sum_{k=\ell_1+H}^{K_0}S_k^2\Bigg) \\ &= -\frac{1}{2n_0}\Bigg(\frac{1}{4}\sum_{k=0}^{H-1}\left((S_{k+\ell_0}-S_{k+\ell_1})^2+(S_{k+\ell_1}-S_{k+\ell_0})^2\right)+\sum_{k=0}^{\ell_0-1}S_k^2+\sum_{k=\ell_0+H}^{\ell_1-1}S_k^2 \\ &\quad +\sum_{k=\ell_1+H}^{K_0}S_k^2\Bigg). \end{aligned}$$

Now note for any $a, b \in \mathbb{R}$

$$(a-b)^2 = a^2 - 2ab + b^2 = (b-a)^2.$$

So we rewrite the above equation

$$\begin{aligned} &= -\frac{1}{2n_0}\left(\frac{1}{4}\sum_{k=0}^{H-1}\left(2(S_{k+\ell_0}-S_{k+\ell_1})^2\right) + \sum_{k=0}^{\ell_0-1} S_k^2 + \sum_{k=\ell_0+H}^{\ell_1-1} S_k^2 + \sum_{k=\ell_1+H}^{K_0} S_k^2\right) \\ &= -\frac{1}{2n_0}\left(\frac{1}{2}\sum_{k=0}^{H-1}(S_{k+\ell_0}-S_{k+\ell_1})^2 + \underbrace{\sum_{k=0}^{\ell_0-1} S_k^2 + \sum_{k=\ell_0+H}^{\ell_1-1} S_k^2 + \sum_{k=\ell_1+H}^{K_0} S_k^2}_{L}\right), \end{aligned}$$

where we use L to denote the "leftovers" from whatever we choose to be $\boldsymbol{\ell}$. Again we want to maximize, but there is a negative factor in the very front, so actually we want to minimize all the sums. To minimize the leftovers, we should choose $\boldsymbol{\ell}$ to pick out the highest energy points in the signal. To minimize the first term, we want to choose ℓ_0 and ℓ_1 such that the points that they pick out are very similar. That is we want matching parts of the signal so that the sum of their squared difference is small.

We observe something curious though if we expand out the sum

$$\begin{aligned} \sum_{k=0}^{H-1}(S_{k+\ell_0}-S_{k+\ell_1})^2 &= \sum_{k=0}^{H-1} S_{k+\ell_0}^2 - 2S_{k+\ell_o}S_{k+\ell_1} + S_{k+\ell_1}^2 \\ &= -2\sum_{k=0}^{H-1} S_{k+\ell_o}S_{k+\ell_1} + \sum_{k=0}^{H-1} S_{k+\ell_0}^2 + \sum_{k=0}^{H-1} S_{k+\ell_1}^2 \\ &= -2\sum_{k=0}^{H-1} S_{k+\ell_o}S_{k+\ell_0+\underbrace{(\ell_1-\ell_0)}_{x}} + \sum_{k=0}^{H-1} S_{k+\ell_0}^2 + \sum_{k=0}^{H-1} S_{k+\ell_1}^2 \end{aligned}$$

The first term is something like the correlation function with time lag $x = \ell_1 - \ell_0$, and we want to maximize (due to the -2) the sum/correlation function over all lags. What is strange is that we also want the energy (i.e. last two terms) to be low. However, if we include those two terms in with the leftovers we almost get a full autocorrelation of the received signal, there is just a $\frac{1}{2}$ on the parts of the signal where we say there is an impulse. Consider Proposition 4.1; it tells us we should apply the reducing factor to the largest term if we want to minimize. This is in agreement with our earlier intuition: the location of impulses should correspond to the highest energy parts of the signal. However, we still need to account for p_1 and p_2 which can no longer be ignored.

Proposition 4.1. *For $a, b \in \mathbb{R}$ where $a < b$ and some $\alpha > 1 \Rightarrow \frac{1}{\alpha} < 1$,*

$$\frac{1}{\alpha}a + b > a + \frac{1}{\alpha}b.$$

That is, the largest term in the sum should have the factor if the goal is minimization. This can be generalized to more terms and factors.

Proof. First multiply $a < b$ by $1 - \frac{1}{\alpha}$

$$\begin{aligned} (a &< b) \times (1 - \frac{1}{\alpha}) \\ a - \frac{1}{\alpha}a &< b - \frac{1}{\alpha}b. \end{aligned} \tag{4.1.8}$$

Then note

$$\begin{aligned} a - \frac{1}{\alpha}a &= a + b - (\frac{1}{\alpha}a + b) \\ b - \frac{1}{\alpha}b &= a + b - (a + \frac{1}{\alpha}b). \end{aligned}$$

Plugging this into (4.1.8)

$$\begin{aligned} a + b - (\frac{1}{\alpha}a + b) &< a + b - (a + \frac{1}{\alpha}b) \\ \Big(- (\frac{1}{\alpha}a + b) &< -(a + \frac{1}{\alpha}b) \Big) \times -1 \\ \frac{1}{\alpha}a + b &> a + \frac{1}{\alpha}b. \end{aligned}$$

□

In summary, the objective function is

$$\begin{aligned} p &= -\frac{1}{2}\log 2\pi\sigma^2(2-1) - \frac{1}{2\sigma}\sum_{i=1}^{2-1}(\Delta\ell_i - \Delta\ell_{i-1} - m)^2 \\ &- \frac{1}{2n_0}\left(\frac{1}{2}(-2\sum_{k=0}^{H-1} S_{k+\ell_o}S_{k+\ell_0+(\underbrace{\ell_1-\ell_0}_{x})} + \sum_{k=0}^{H-1} S_{k+\ell_0}^2 + \sum_{k=0}^{H-1} S_{k+\ell_1}^2) + \sum_{k=0}^{\ell_0-1} S_k^2 + \sum_{k=\ell_0+H}^{\ell_1-1} S_k^2 + \sum_{k=\ell_1+H}^{K_0} S_k^2\right) \\ &= -\frac{1}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma}(\Delta\ell_1 - \Delta\ell_0 - m)^2 \\ &- \frac{1}{2n_0}\left(-\sum_{k=0}^{H-1} S_{k+\ell_o}S_{k+\ell_0+(\underbrace{\ell_1-\ell_0}_{x})} + \frac{1}{2}\sum_{k=0}^{H-1} S_{k+\ell_0}^2 + \frac{1}{2}\sum_{k=0}^{H-1} S_{k+\ell_1}^2 + \sum_{k=0}^{\ell_0-1} S_k^2 + \sum_{k=\ell_0+H}^{\ell_1-1} S_k^2 + \sum_{k=\ell_1+H}^{K_0} S_k^2\right) \\ &= -\frac{1}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma}(\Delta\ell_1 - \Delta\ell_0 - m)^2 \\ &+ \frac{1}{2n_0}\left(\sum_{k=0}^{H-1} S_{k+\ell_o}S_{k+\ell_0+(\underbrace{\ell_1-\ell_0}_{x})} - \left(\frac{1}{2}\sum_{k=0}^{H-1} S_{k+\ell_0}^2 + \frac{1}{2}\sum_{k=0}^{H-1} S_{k+\ell_1}^2 + \sum_{k=0}^{\ell_0-1} S_k^2 + \sum_{k=\ell_0+H}^{\ell_1-1} S_k^2 + \sum_{k=\ell_1+H}^{K_0} S_k^2\right)\right) \end{aligned} \tag{4.1.9}$$

Given an ℓ_0 and ℓ_1, we can easily compute this value– the problem is we need to check for each ℓ_0, ℓ_1. Proposition 4.2 explains that this will require $O(K_0^2)$ comparisons which is often feasible.

Proposition 4.2. *For K_0 elements $\{0, 1, 2, \ldots, K_0 - 1\}$, there are $\frac{1}{2}(K_0^2 - K_0)$ unique pair combinations without replacement.*

Proof. Without loss of generality, we can cover all unique pair combinations without replacement (i, j), where i, j are the index of the elements, if we consider all pairs for which $i < j$. Consider a $K_0 \times K_0$ matrix where each element ij corresponds to a comparison between time indexes i, j. The points where $i < j$ corresponds to the upper triangle right above the diagonal. The diagonal has $K_0 - 0$ elements; the next layer up has $K_0 - 1$ elements and so forth until we reach the top which has $K_0 - (K_0 - 1) = 1$ elements. There are $K_0 - 1$ layers excluding the

diagonal. Summing these terms

$$K_0 - 1 + \cdots + 2 + 1 = \sum_{n=1}^{K_0-1} n = \frac{1}{2}(K_0 - 1 + 1)(K_0 - 1)$$
$$= \frac{1}{2}(K_0^2 - K_0).$$

This is a typical arithmetic series. □

For $N = 3 \Rightarrow \boldsymbol{\ell} = \{\ell_0, \ell_1, \ell_2\}$, we can come up with a similar expression, though it will not simplify as nicely. Also, in the same manner as $N = 2$, we would have to consider all $\ell_0 < \ell_1 < \ell_2$ which is like the upper pyramid (3D) of a K_0 cube. Like with Proposition 4.2, the precise number of comparisons can be found, but it will be $O(K_0^3)$. It follows then for arbitrary N, we will need to compare all $\ell_0 < \ell_1 < \cdots < \ell_{N-1}$ which is $O(K_0^N)$.

Finally, we must compare all amounts N to make the detection decision. Doing this will require

$$O(1) + O(K_0) + O(K_0^2) + O(K_0^3) + \cdots + O(K_0^{N_{max}}) = O(K_0^{N_{max}}).$$

Typically K_0 is quite large (e.g. high sample rate) and N_{max} (which theoretically should be K_0/H but more is more realistically around $\Delta K_0/\mu$) is often large too. So doing it exactly is computationally infeasible. With this in mind, we consider the greedy approximation in the following.

4.1.1 Sequential estimation

The first greedy approach we will call *sequential estimation*. Rather than trying to process the entire signal at once, we will detect each impulse sequentially (i.e. an online vs a batch method). Though it will not guarantee optimality, it makes the problem feasible. As discussed before, we will assume we know $H, n_0, \mathcal{T} \sim \mathcal{N}(\mu, \sigma^2)$. However for n_0, we can use (4.1.6) as discussed previously.

First we must find a suitable ℓ_0. According to (4.1.7), we just need to pick the highest energy section that is H long. In order to prevent us from just picking noise, we compare our maximum with the noise energy over the length of an impulse

$$\sum_{k=0}^{H-1} W^2[k] \approx \sum_{k=0}^{H-1} n_0 = Hn_0. \tag{4.1.10}$$

If the energy does not surpass this, we should not declare an impulse. In such cases, we should move the window over and keep going until we either find one or reach the end.

Assuming a first impulse is found, we have an ℓ_0. Then with that first impulse identified, we look at the rest of the signal for another one. That is we just evaluate (4.1.9) with the given ℓ_0 for each ℓ_1 and pick the maximum. Note, $\ell_1 \in [\ell_0 + H + 1, K_0]$ since ℓ_1 cannot come before ℓ_0; this means we only need to check $O(K_0)$ values which is very feasible. However, like when finding ℓ_0, we should not just accept any maximum; the maximum should exceed the threshold of just a single detection. That is (4.1.7) evaluated for the entire signal. We will refer to this value as the single detection threshold (SDT). Note to make computations easier, once we have identified ℓ_0, we can ignore everything prior to make things simple. This simplification makes the SDT just the energy of the signal after the impulse

$$\sum_{k=\ell_0+H}^{K_0} S_k^2.$$

Once we find ℓ_1, we can treat this new impulse as the new "first" impulse and repeat the process of finding a "second" impulse ℓ_3. We keep going repeating this until we reach the end of the signal. Algorithm 4.1 gives an outline for the method.

Algorithm 4.1 Sequential estimation algorithm

Given a signal $S(t)$ containing noise with power n_0, sampled to $\{S_k\}$ at a rate of $1/\Delta$, which contains impulses from a single source according to $H, \mathcal{T} \sim \mathcal{N}(\mu, \sigma^2)$, the shape of the source impulse $\mathbf{h}$, and the times of impulses $\boldsymbol{\ell}$ can be approximated with the following procedure:

1. Find ℓ_0
   (a) Set the detection window width $W = \mu\Delta$.
   (b) Compute $D(\ell) = \sum_{k=0}^{H-1} S_{k+\ell}^2$ for $\ell \in [0, W-1]$
   (c) Find $\max D(\ell)$
      i. If $\max D(\ell) < Hn_0$, slide the window over one W and start again
      ii. Otherwise, $\hat{\ell}_0 = \arg\max D(\ell)$
2. Use *last identified impulse* at ℓ_i (initially ℓ_0) to find *next impulse* ℓ_{i+1} (initially ℓ_1)
   (a) Ignore all signal prior to ℓ_i
   (b) Compute (4.1.9) for all remaining indexes using ℓ_i as "ℓ_0" and find the maximum
   (c) Compare the maximum with the SDT, (4.1.7), evaluated for this new truncated signal.
      i. If the maximum is not greater than the SDT, say that there are no more impulses and terminate
      ii. Otherwise, set the index of the maximum to ℓ_{i+1}
3. Repeat previous step using the newly found impulse ℓ_{i+1} as the new *last identified impulse*

To show that this works, consider the recording from a single sperm whale from [45] (the same data set used in Section 2.5.2). Specifically, we consider a 10 second segment that contains 6 clicks. One of the prerequisites for our detection algorithm is we need to know $\mathcal{T}$ and H. Under the assumption that $H = 2400$ (with $1/\Delta = 48000$, this means each impulse is 0.05 seconds long), we use [44]'s simple SNR threshold detector to extract times, and we compute $\mathcal{T} \sim \mathcal{N}(1.5738, 9.5384 \times 10^{-4})$. The signal, and detection times from both the simple detector and Algorithm 4.1 are shown in Figure 4.1.1. Our sequential estimator finds the same number of impulses as the simple threshold detector. The times are very similar, but simple detector's times are just 153-155 samples (≈ 0.0032 seconds) later than Algorithm 4.1. This is well within an acceptable range.

Note, the smaller impulses are actually paired with the larger impulses. The smaller ones are either the result of multi-path from the larger, or are actually the initial muted click[1]. In any case, we are adopting the convention that the time of the larger ones are what we are interested in, and appending the smaller impulses (if present) after the larger ones.

Though we are getting promising results on actual impulses, a major weakness of Algorithm 4.1 is that it requires knowledge of $\mathcal{T}$ which is typically not the case. Previously we were able to overcome this problem of parameter estimation in Section 2.3, but this required the knowledge of the detection times. Let us consider to multiple source problem to see if we can find any clues to help with this.

[1] It is understood that sperm whales produce their clicks at the front of their head (via the "monkey lips") projecting sound backwards, then reflecting the sound back forward through a focusing lens. Though the "click" is the emission that gets focused, sometimes the sound generated at the front of the head also is picked up in recordings [53].

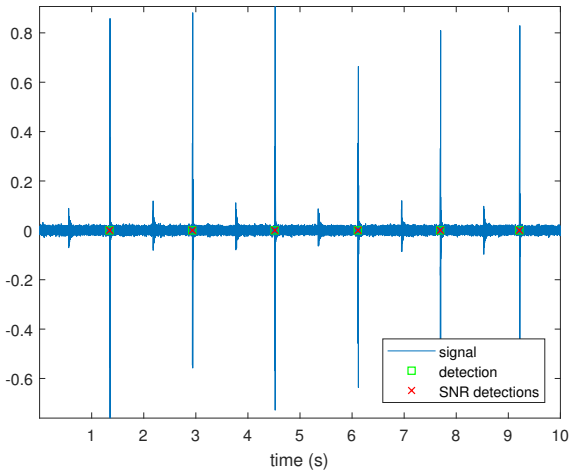


Figure 4.1.1: Except of a recording from a single sperm whale. Green squares indicate the detection times from Algorithm 4.1, and the red x mark the detection times from [44]'s simple SNR threshold detector.

4.2 Multiple source

The method in Section 4.1.1 can be extended to multiple sources. The general idea is to again step through the signal sequentially assigning impulses, but at each for assignment, consider one of the K sources and maintain multiple solution paths as in Chapter 2 outputting the one with the highest objective function in the end.

As with the signal source case, we first will discretize the signal

$$S(t) = \sum_{r=1}^{R} \sum_{i=0}^{N_r-1} h_r(t - \tau_{r,i}) + W(t) \Rightarrow S(\Delta k) = S[k] = \sum_{r=1}^{R} \sum_{i=0}^{N_r-1} h_r[k - \ell_{r,i}] + W[k]$$

for $k = 0, \ldots, K_0$ where we have substituted $t = k\Delta$ and $\tau_{r,i} = \ell_{r,i}\Delta$ for integer $k, \ell_{r,i}$. Note $r \in \{1, 2, \ldots, R\}$ denotes the source. Again, the assumption is that Δ is small enough such that $|\tau_{r,i} - \ell_{r,i}\Delta|$ is negligible. Note $\frac{1}{\Delta}$ is the sample rate. When we sample the random process $W(t)$, we just get the random vector $W[k]$ where each sample is iid $\sim \mathcal{N}(0, n_0) \Rightarrow \mathbf{W} \sim \mathcal{N}(0, n_0\mathbf{I})$ where $\mathbf{W}$ is the vector of $W[k]$ for all k. This makes the interimpulse distribution

$$\mathcal{T}_r(\tau_{r,i} - \tau_{r,i-1}) = \mathcal{T}(\Delta\ell_{r,i} - \Delta\ell_{r.i-1}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(\Delta\ell_{r,i} - \Delta\ell_{r,i-1} - m)^2}{2\sigma^2}).$$

We just have to be careful to multiply $\ell_{r,i}$ by Δ before using $\mathcal{T}_r$. Up till now, this is almost exactly like the single source case, except now we have multiple sources. Note, it is assumed that the impulses from either source will not overlap.

What we want to find is $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_R\}$ and $\mathbf{L} = \{\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \ldots, \boldsymbol{\ell}_R\}$, that is the impulses shape for each source and their associated timings. As in the previous case, we wish to choose these values as to maximize

$$\Pr\{\mathbf{H}, \mathbf{L}|\mathbf{S}\} = \frac{\Pr\{\mathbf{S}, \mathbf{H}, \mathbf{L}\}}{\Pr\{\mathbf{S}\}}.$$

But when we maximizing over $\mathbf{H}$ and $\mathbf{L}$, the value of the denominator becomes irrelevant. Thus we are mainly

concerned with maximizing

$$\begin{aligned}\Pr\{\mathbf{S},\mathbf{H},\mathbf{L}\} &= \Pr\{\mathbf{S}|\mathbf{H},\mathbf{L}\}\Pr\{\mathbf{H},\mathbf{L}\}\\ &= \Pr\{\mathbf{S}|\mathbf{H},\mathbf{L}\}\Pr\{\mathbf{H}\}\Pr\{\mathbf{L}\}. \end{aligned} \tag{4.2.1}$$

As before, we assume that the impulse shapes and timings are independent. Further, we assume that the sources are independent from each other. So we can write

$$\Pr\{\mathbf{H}\} = \Pr\{\mathbf{h}_1\}\Pr\{\mathbf{h}_2\}\cdots\Pr\{\mathbf{h}_R\}$$

$$\Pr\{\mathbf{L}\} = \prod_{r=1}^{R}\Pr\{\boldsymbol{\ell}_r\} = \prod_{r=1}^{R}\prod_{i=1}\mathcal{T}(\Delta\ell_{r,i} - \Delta\ell_{r,i-1}).$$

We make the assumption that $\Pr\{\boldsymbol{h}_r\}$ are all identically distributed as uniform over all impulse shapes of length $T_{max} = \Delta H$. This is the same as before, this just restricts our impulse shapes to a given length, and then we can ignore this term. And as noted, $\Pr\{\mathbf{L}\}$ is constructed based on our assumptions of source independence and iid interimpulse spacings within sources (this was the heart of the timing algorithm). Once again we will have that $\Pr\{\mathbf{S}|\mathbf{H},\mathbf{L}\} = \Pr\{\mathbf{W}\}$, since

$$W[k] = S[k] - \sum_{r=1}^{R}\sum_{i=0}^{N_r-1} h_r[k-\ell_{r,i}],$$

and $\mathbf{W} \sim \mathcal{N}(0, n_0\mathbf{I})$, so

$$\Pr\{\mathbf{S}|\mathbf{H},\boldsymbol{L}\} = (2\pi n_0)^{-\frac{K_0}{2}}\exp(-\frac{1}{2n_0}\sum_{k=1}^{K_0}(S[k] - \sum_{r=1}^{R}\sum_{i=0}^{N_r-1} h_r[k-\ell_{r,i}])^2).$$

This is very similar to single source case, so many results will be the same. Again, to make things easier we will consider maximizing the log of (4.2.1), so

$$\begin{aligned}\log\Pr\{\mathbf{L}\} &= \sum_{r=1}^{R}\sum_{i=1}^{N_r-1}\log\mathcal{T}_r(\Delta\ell_{r,i} - \Delta\ell_{r,i-1})\\ &= \sum_{r=1}^{R}\sum_{i=1}^{N_r-1}\log\left(\frac{1}{\sqrt{2\pi\sigma_r^2}}\exp(-\frac{(\Delta\ell_{r,i} - \Delta\ell_{r,i-1} - m_r)^2}{2\sigma_r^2})\right)\\ &= \sum_{r=1}^{R}\left(\left(-\frac{1}{2}(N_r-1)\log(2\pi\sigma_r^2)\right) - \frac{1}{2\sigma_r^2}\sum_{i=1}^{N_r-1}(\Delta\ell_{r,i} - \Delta\ell_{r,i-1} - m_r)^2\right).\end{aligned}$$

Note, N_r, the number of impulses from source r, is a function $\boldsymbol{\ell}_r$. Also

$$\begin{aligned}\log\Pr\{\mathbf{S}|\mathbf{H},\boldsymbol{L}\} &= \log\left((2\pi n_0)^{-\frac{K_0}{2}}\exp(-\frac{1}{2n_0}\sum_{k=1}^{K_0}(S[k] - \sum_{r=1}^{R}\sum_{i=0}^{N_r-1} h_r[k-\ell_{r,i}])^2)\right)\\ &= -\frac{K_0}{2}(2\pi n_0) - \frac{1}{2n_0}\sum_{k=1}^{K_0}(S[k] - \sum_{r=1}^{R}\sum_{i=0}^{N_r-1} h_r[k-\ell_{r,i}])^2.\end{aligned}$$

The first term above is just a constant not dependent on $\mathbf{H}$ or $\mathbf{L}$, so we can ignore it in terms of the maximization. Thus the objective function to maximize is

$$p \triangleq \sum_{r=1}^{R} \left(\left(-\frac{1}{2}(N_r - 1)\log(2\pi\sigma_r^2) \right) - \frac{1}{2\sigma_r^2} \sum_{i=1}^{N_r-1} (\Delta\ell_{r,i} - \Delta\ell_{r,i-1} - m_r)^2 \right) - \frac{1}{2n_0} \sum_{k=1}^{K_0} \left(S[k] - \sum_{r=1}^{R} \sum_{i=0}^{N_r-1} h_r[k - \ell_{r,i}] \right)^2 . \tag{4.2.2}$$

Again if we want to find $\hat{h}_{r,q}$, an estimate for the q-th value in the impulse of source r, we take the partial derivative again with respect to it and set to 0. The result will be essentially the same as (4.1.2) since all terms unrelated to source $r$are ignored

$$\hat{h}_{r,q} = \frac{1}{N_r} \sum_{i=0}^{N_r-1} S_{q+\ell_{r,i}}. \tag{4.2.3}$$

It is a simple average.

As before, there exists no easy way maximize for $\mathbf{L}$ or $\boldsymbol{\ell}_r$. However, we can evaluate p for a given $\mathbf{L}$. Though it is computationally infeasible to try and compute this the p for every $\mathbf{L}$, if we could, we could just pick $\mathbf{L}$ corresponding to the the largest p. Instead, we consider a greedy approximation like Section 4.1.1 in the following section.

Note, (4.2.2) can be rewritten as

$$p = \left(\sum_r \sum_i \log \mathcal{T}_r(\Delta\ell_{r,i} - \Delta\ell_{r,i-1}) \right) + \left(\sum_{k=1}^{K_0} \log f \left(S[k] - \sum_{r=1}^{R} \sum_{i=0}^{N_r-1} h_r[k - \ell_{r,i}] \right) \right), \tag{4.2.4}$$

where $f \sim \mathcal{N}(0, n_0)$

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}},$$

and $h_r[k] = 0$ for $k \leq 0$ and $k > H$ (i.e. it is bounded). So the term $S[k] - \sum_{r=1}^{R} \sum_{i=0}^{N_r-1} h_r[k - \ell_{r,i}]$ is really just the signal minus the impulses.

4.2.1 Greedy approach

In the timing algorithms of Chapter 2, the assumption was that the assignment of the current point only depends on the timing of the previous point. In the same manner let us make the simplifying approximation that the detection of the next point only depends on the previous. That is, we will do the classification sequentially.

In Section 4.1.1, we start with a single impulse and then try to detect the next one based on the first detection. At each time point after $\ell_i + H - 1$, the end time of the most previous impulse, we compare the objective function values for

- No other impulses assigned (i.e. we are at the end of the signal)
- We say there is impulse at the time point

If the second value is greater then an impulse should be detected. This idea will work for the multiple sources as well.

First, assume we are given the first impulse (time) ℓ_0 but do not know which source it belongs too. For example, we use some simple SNR/energy detector (we did this for the single source case too). Similar to how we do the

timing separation algorithms in Chapter 2, we can assume it belongs to each possible source and extend from there choosing the best output of the choices in the end. But for simplicity of we describe the algorithm assuming that the first impulse belongs to the first source.

So we start checking for new detections at $v = \ell_0 + H$, where v is the current index of where the algorithm has worked up to. For simplicity, define $\ell_0 = 0$. As we assign each time point we also declare an assignment

$$z_k = \begin{cases} r & S[k] \text{ is from } r \\ 0 & o.w. \end{cases}.$$

A $z_k = 0$ indicates that $S[k]$ is just the noise. Let $\mathbf{z}(v) = [z_v, z_{v+1}, \ldots, z_{v+H-1}]$ be the assignment of the next H time points. We determine if $\mathbf{z}(v)$ belongs to source r or is noise by comparing the associated values of $p(v)$ from (4.2.4). Depending on $\mathbf{z}^{-1}(v) = [z_0, z_1, z_2, \ldots, z_{v-1}]$ (i.e the assignment of points leading up to v), $p(v)$ is computed differently.

- Assuming $\mathbf{z}(v)$ is all noise

$$p(v) = \sum_{s=1}^{R} \sum_{i=0}^{N_s-1} \log \mathcal{T}_r(\Delta\ell_{s,i} - \Delta\ell_{r,N_s-1}) + \sum_{k=0}^{v+H-1} \log f\left(S[k] - \sum_{s=1}^{R} \sum_{i=0}^{N_s-1} \hat{h}_s[k - \ell_{s,i}]\right)$$

- Assuming $\mathbf{z}(v)$ belongs to source r and $r \in \mathbf{z}^{-1}(v)$

$$\begin{aligned} p(v) &= \sum_{s=1}^{R} \sum_{i=0}^{N_s-1} \log \mathcal{T}_r(\Delta\ell_{s,i} - \Delta\ell_{r,N_s-1}) + \sum_{k=0}^{v+H-1} \log f\left(S[k] - \sum_{s=1}^{R} \sum_{i=0}^{N_s-1} \hat{h}_s[k - \ell_{s,i}]\right) \\ &= \log \mathcal{T}_r(\Delta v - \Delta\ell_{r,N_r-1}) + \sum_{s=1}^{R} \sum_{i=1}^{N_s-1} \log \mathcal{T}_r(\Delta\ell_{s,i} - \Delta\ell_{s,i-1}) \\ &\quad + \sum_{k=0}^{v+H-1} \log f\left(S[k] - \hat{h}_r[k - v] - \sum_{s=1}^{R} \sum_{i=0}^{N_s-1} \hat{h}_s[k - \ell_{s,i}]\right) \end{aligned}$$

  Note before computing the above $\hat{\mathbf{h}}_r$ is updated with the new points according to (4.2.4).

- Assuming $\mathbf{z}(v)$ belongs to source r but $r \notin \mathbf{z}^{-1}(v)$

$$\begin{aligned} p(v) &= \sum_{s=1}^{R} \sum_{i=0}^{N_s-1} \log \mathcal{T}_r(\Delta\ell_{s,i} - \Delta\ell_{r,N_s-1}) + \sum_{k=0}^{v+H-1} \log f\left(S[k] - \sum_{s=1}^{R} \sum_{i=0}^{N_s-1} \hat{h}_s[k - \ell_{s,i}]\right) \\ &= \sum_{s\neq r} \sum_{i=1}^{N_s-1} \log \mathcal{T}_r(\Delta\ell_{s,i} - \Delta\ell_{s,i-1}) + \sum_{k=0}^{v+H-1} \log f\left(S[k] - \hat{h}_r[k - v] - \sum_{s\neq r} \sum_{i=0}^{N_s-1} \hat{h}_s[k - \ell_{s,i}]\right) \end{aligned}$$

  Again, here the second line is formatted the same as the previous case. There is no addition to the timing term and the estimate for $\hat{\mathbf{h}}_r$ is just the added points

We assign $\mathbf{z}(v)$ to the r that maximizes $p(v)$ and then increment $v = v + H$. Though in the case $p(v)$ for noise is higher, we only increment $v = v + 1$. We stop looking for detections when $v + H > K_0$.

To test this methodology we generate two sources with $\boldsymbol{\Theta} = \{(1.5, 0.13^2), (1.3, 0.13^2)\}$ and arbitrary shapes; we set the SNR to 10. We seed the algorithm above with the first impulse time, and the result is shown in Figure 4.2.1. Unfortunately, what happens is that while we do detect every impulse, we assign every impulse to source 1.

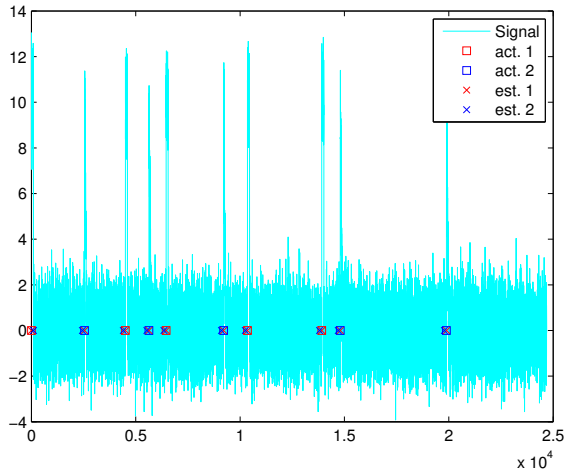


Figure 4.2.1: Detection algorithm directly applied to a 2 source problem

We do have some detections for source 2, but it corresponds to noise immediately preceding all impulses except the first. Our first detection for source 2 is the block of noise after the seeded impulse. This happens since assignment to a source reduces $p(v)$, and then once a source is assigned to noise, it will continue to detect noise blocks.

One problem is that we are making detection decisions based on $[0, v + H - 1]$, we should also consider the rest of the signal. Like with single source, we can compute the $p(v)$ values above for rest of the signal (or at least an extended time) and choose the maximum amongst the results. for the next detection. However this only improves the accuracy on detection times, we still are unable to determine which impulses belong to the second source. However, if we provide the first impulse from the second source as well, we get results like Figure 4.2.2. There is one flip-flop error, but otherwise, each impulse is detected and assigned to the right source.

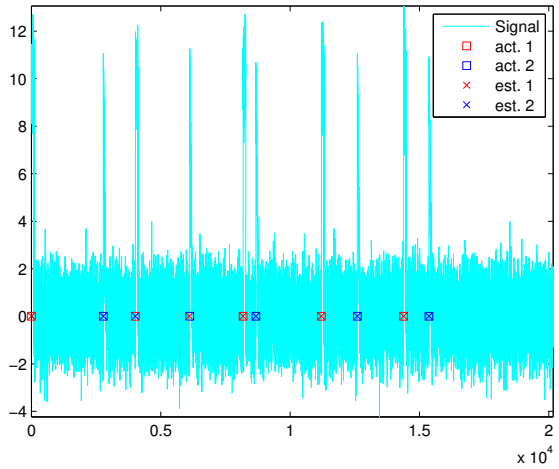


Figure 4.2.2: Detection algorithm directly applied to a 2 source problem with peak searching

What this means is that we have a problem with initialization. For this method to work, we need to know:

- The timing parameters $\boldsymbol{\Theta}$
- Locations of the first impulse for each of the R sources

In a practical problem these are unlikely to be given. Perhaps, we could take a portion of the signal run a regular detector (Appendix D) and use the methods in Chapter 3 to get $\boldsymbol{\Theta}$ and then Chapter 2 to assign the impulses, but then we could also just do that for the whole signal. It seems doubtful that using the detection method described in this section would provide any improvement over doing the detection and classification separately (especially if we need to initialize by first doing them separately). To this end, rather than pursuing this detection algorithm further, we focus our efforts in improving the methods in Chapter 3 Chapter 2. Though we were unable to find a suitable detection method, the work in this Chapter greatly influenced the methods in Section 6.3.

5 Working with sperm whale clicks

Separating sperm whale click trains is the intended application of this research. In this chapter, we take a closer look at techniques to extract the information used in other parts of the paper (e.g. impulse times and assignments) and analyze some of that data.

5.1 Detection

All the algorithms of Chapters 2 and 3 operate on click times. Detection is the practice of extracting click times (and sometimes click shape) from the recorded signal. Sperm whale clicks can be approximated as impulses, and this are easy to recognize based on their amplitude. Therefore, as pointed out in Chapter 4, threshold detectors are often used. Algorithm 5.1 outlines a simple detector, for more details consult [44].

Algorithm 5.1 A single channel threshold detector adapted from [44].

For a signal y

1. Find the envelope using the Hilbert transform: $y \to y_H$ (see Appendix D)
2. Take magnitude of result: $|y_H|$
3. Run a peak detector on result (e.g. in MATLAB use `findpeaks`)
   (a) Threshold: $5 \times \overline{|y_H|}$, where $\overline{|y_H|}$ is the mean of $|y_H|$
   (b) Minimum peak distance: 24 ms (this is the approximate length of a usual (sperm whale) click)

The locations of the peaks are the detection times

5.1.1 Preprocessing

While the detection methods can be applied directly to the recorded signals, it never hurts to do some preprocessing to make the detection easier. For our application, it means remove things that are not sperm whale clicks.

It is well known and documented that sperm whale clicks have energy between 2-20 kHz [54, 55]. Therefore, as a first step we apply a band-pass filter to all recordings. Figure5.1.1 shows an example filter that is applied to the recordings.

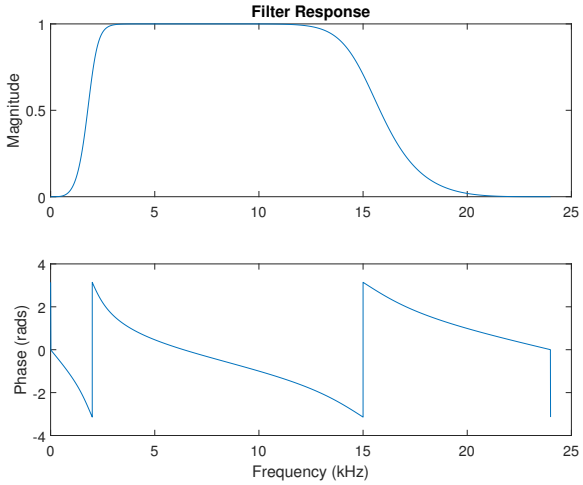


Figure 5.1.1: This is a 4-th order Butterworth filter with a 2-15 kHz passband.

Related to filtering the spectrum, another helpful thing to do is to remove tonals. Tonals are constant tones that arise as an artifact due to things such as ship noise or the undulation of the cable in the case of a towed hydrophone array. They show up as large vertical lines in the spectrum. One mitigation procedure is to flatten the spectrum by normalizing each frequency by its magnitude. That is, let $Y(\omega)$ be the Fourier transform of $y(t)$. Then we get the normalized signal $\tilde{y}$ by

$$\tilde{y}(t) = \mathcal{F}^{-1}\left\{\frac{Y(\omega)}{|Y(\omega)|}\right\},$$

where $\mathcal{F}^{-1}$ represents the inverse Fourier transform.

5.2 Ground truthing

In order to verify timing separation algorithms (e.g. compute the classification error rate), it is important to know which clicks actually belong to which whale. That is, we need to know the ground truth. For many of the previous results, we simulated click times, so we knew the actual assignment. However, this is much more difficult obtain on actual sperm whale clicks. In the case we have recordings from a single animal, we can create pseudo-mixtures by overlaying different parts of the signal (e.g. Section 2.5.2). However, this information is rare, and we can only do so many experiments with the data we have. Thus it is necessary to be able to identify a ground truths in recordings of multiple animals. We describe two methods in this section: TDOA and multipath.

5.2.1 TDOA method

In the case that there are multiple receivers, each receiver is physically separated by some distance. Let a be the distance in meters from the source to the first receiver and b be the distance from the source to the second receiver. Assume that the speed of sound is $c = 1500$ m/s.[1] Then a click will take a/c s to arrive at the first receiver, and b/c s to arrive at the second receiver. If τ is the time of the click, then we will see the click recorded in the first channel at $\tau + a/c$ and in the second channel at $\tau + \frac{b}{c}$. If we take the first receiver as reference, the time difference of arrival (TDOA) is the difference in time that the click arrives at the second receiver as opposed to the first. That is, the TDOA is

$$(\tau + \frac{b}{c}) - (\tau + \frac{a}{c}) = \frac{b-a}{c}.$$

Here a negative value would imply that it arrives at the second receiver before the first. For a physical illustration of this, see Figures 5.2.1c and 5.2.1a.

Above, we have described a near field model, but sources in different locations will also exhibit a similar difference in TDOA even if we assume a far field model as shown in Figure 5.2.1b. In this situation, there is a simple trigonometric transform to go from TDOA to Angle of Arrival (AOA or θ)

$$\theta = \cos^{-1}(\frac{c}{d}\text{TDOA}).$$

Note, it is possible that two sources have the same TDOA if they are at the same θ but at different distances. However, in that case, perhaps amplitude can help to distinguish sources. Regardless, the conclusion here is that signals with different TDOA are in different locations, and therefore, assuming stationary sources and receivers, are different sources.

Even though sperm whales and towed arrays are not stationary, over short periods of time the assumption of stationary sources and receivers is generally acceptable. Thus, we will develop a classification method based on this theory.

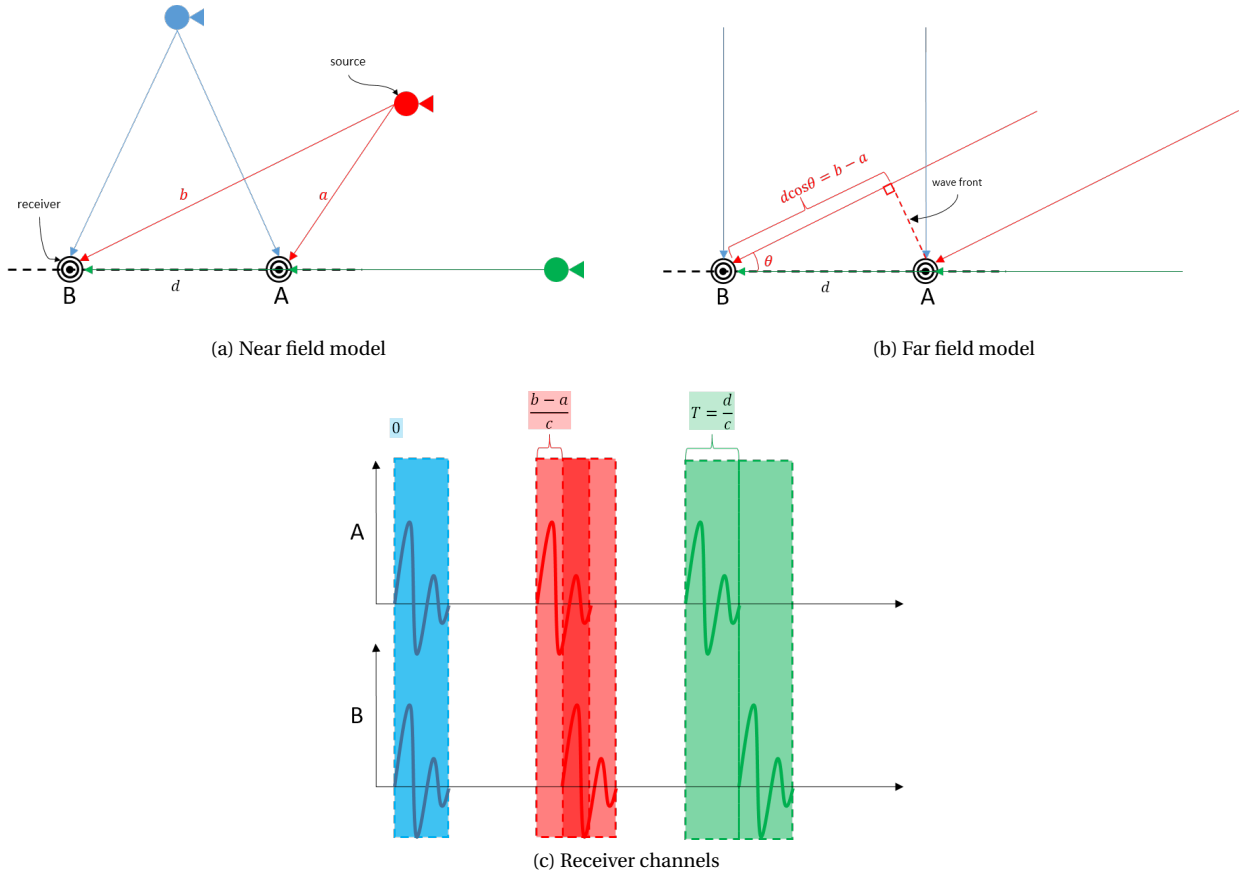
There are a number of methods that are used to estimate TDOA from an array of receivers; for example, MUSIC is an eigenvector method that exploits the orthogonality of the signal and noise subspaces [56]. However, generally, this requires that a noise subspace exists; that is, we have more receivers than sources. With PAM monitoring for sperm whales, this is a rare occurrence. However, since we are dealing with clicks, we can determine TOA of each vocalization in each channel with precision and then compute TDOA between channels of associated clicks. We more formally outline this process in Algorithm 5.2.

At the end of Algorithm 5.2, we can aggregate all TDOA plot against time. For example, see Figure 5.2.2 which comes from recordings off a towed array.[2] There are a number of curves that are indicative of different sources slowly moving over time. In this case they are likely whales, but the detection method used also picked up some non-whale impulses. A clear artifact are the detections corresponding to TDOA $= 0$; these are almost surely sounds from the boat or the towed line itself. Therefore we need to further verify which detections are actually whale clicks (e.g. manually look at the detection).

We do not need to check every detection, just the ones we think could be whales. As we just mentioned, these are the detections seen as curves of slowly varying TDOA. For humans, it is relatively easy for us to recognize these patterns, but would also be helpful and interesting to automate or at least semi-automate this the process of finding curves and selected data.

[1] This is a very rough approximation, but it is sufficient for our motivating example. The speed of sound varies quite a bit in the actual ocean depending on the temperature, pressure, and salinity of the water. See: https://dosits.org/tutorials/science/tutorial-speed/

[2] Thank you to Yvonne Barkley for sharing data from the HICEAS 2010 survey of the Northwest Hawaiian islands.



(a) Near field model

(b) Far field model

(c) Receiver channels

Figure 5.2.1: This is an illustration of TDOA for near and far field situations.

Algorithm 5.2 TDOA extraction from a pair of channels

Given a recordings from pair of channels $y_1(t), y_2(t)$ we compute TDOA

1. Find the clicks in each channel (e.g. Section 5.1). Let the collection of click times for source k be $\boldsymbol{\tau}_k = \{\tau_{k,i}\}$, where $\tau_{k,i}$ is the i-th click in the k-th channel
2. Compute $\text{TDOA}_{\max} = \frac{D}{c}$, where D is the physical distance between the hydrophones and c is the speed of sound.
3. Remove associated clicks
   (a) Collect $\mathcal{T} = \{\boldsymbol{\tau}_1, \boldsymbol{\tau}_2\}$ and sort based on amplitude
   (b) Let τ_A be the time of the largest click with the largest amplitude. Find all $\tau_{k,i} \in \mathcal{T}$ such that
   $$\tau_A - \text{TDOA}_{\max} < \tau_{k,i} < \tau_A + \text{TDOA}_{\max}$$
   and remove them from $\mathcal{T}$, note τ_A should not be removed.
   (c) Find the next largest amplitude, get τ_A and repeat b. Stop when no more τ_A can be found.
4. Sort the pruned $\mathcal{T}$ now by click time. Let τ_i be the i-th time in $\mathcal{T}$.
5. Starting with $i = 1$, Compute TDOA for τ_i
   (a) Extract signals
   $$\tilde{y}_1(t) = \begin{cases} y_1(t) & t - 1.2\text{TDOA}_{\max} < t < t + 1.2\text{TDOA}_{\max} \\ 0 & \text{otherwise} \end{cases}$$
   $$\tilde{y}_2(t) = \begin{cases} y_2(t) & t - 1.2\text{TDOA}_{\max} < t < t + 1.2\text{TDOA}_{\max} \\ 0 & \text{otherwise} \end{cases}$$
   we consider slightly longer than $\text{TDOA}_{\max}$ just in case any detections are on the edge
   (b) Compute $r(t) = \tilde{y}_1(t) * \tilde{y}_2(t)$, where $*$ denotes correlation.
   (c) Find the estimate for TDOA, $\hat{s} = \arg\max_s r(s)$
6. Use TDOA to infer associated click time for other channel. Let $k(\tau_i)$ denote the channel which τ_i came from. Define $\hat{\boldsymbol{\tau}}_1, \hat{\boldsymbol{\tau}}_2$ as the impulse times inferred by TDOA. Initially they are empty.
   (a) If $k(\tau_i) = 1$
      i. add $\tau_{1,i} = \tau_i$ to $\hat{\boldsymbol{\tau}}_1$
      ii. add $\tau_{2,i} = \tau_i - \hat{s}$ to $\hat{\boldsymbol{\tau}}_2$
   (b) If $k(\tau_i) = 2$
      i. add $\tau_{1,i} = \tau_i + \hat{s}$ to $\hat{\boldsymbol{\tau}}_1$
      ii. add $\tau_{2,i} = \tau_i$ to $\hat{\boldsymbol{\tau}}_2$
7. Increment i, and then repeat steps 5 and 6 until a TDOA for all impulses in $\mathcal{T}$ have been found.

Step 6 can be omitted if finding TDOA is the only interest. For our application, knowing the times of impulses corresponding to each TDOA is important.

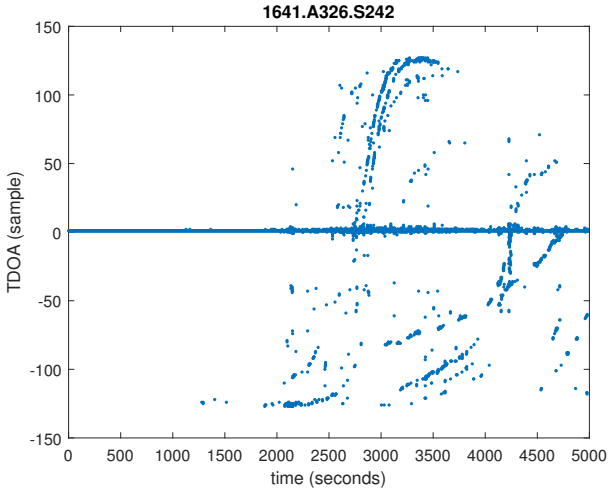


Figure 5.2.2: This is a plot of a TDOA over time for the 1641.A326.S242 survey from HICEAS 2010. Note, TDOA is shown in sample number, $f_s = 192000$ samples per second.

5.2.2 Multipath as a second source

Multipath is a well known phenomena that occurs as a result of multiple propagation paths (hence the name multi-path). A simple example is an echo; first the original sound is heard and then a slightly distorted copy is heard after a slight delay (i.e. the echo). Any echo is a multipath arrival, and the original sound is the direct arrival. Multiple propagation paths occur since we do not live in an infinite space without obstructions. In the case of wireless telecommunications, the signal will bounce off of the ground, walls, buildings, and the like. With underwater communication (e.g. radar or sperm whale clicks), the signals bounce off the ocean bottom and the surface of the water as shown in Figure 5.2.3.

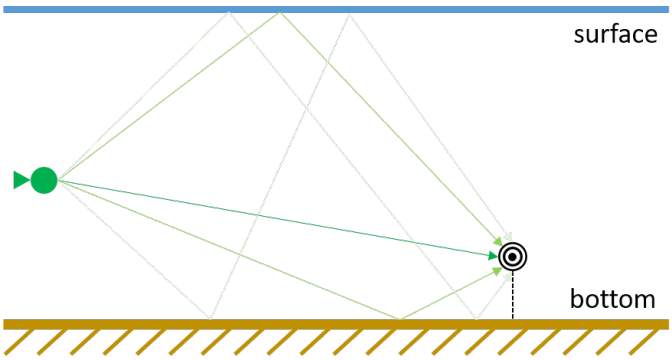
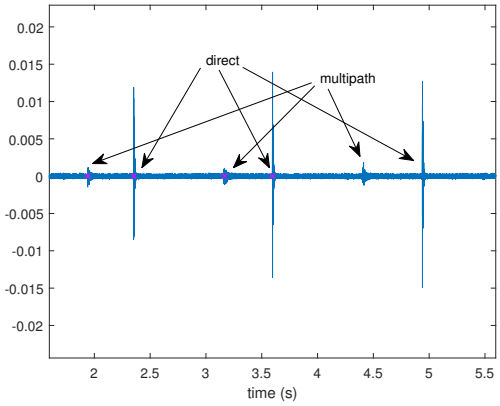


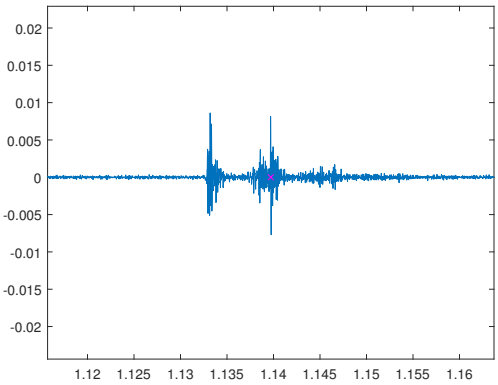
Figure 5.2.3: The arrows indicate the multiple propagation paths the can occur in underwater settings. The direct arrival is in dark green. In lighter shades, we have the multipath arrivals that typically include the bottom reflection or surface reflection, and also sometimes the bottom-surface or the surface-bottom reflections.

Typically, multipath is regarded as a form of unwanted interference and efforts are taken to remove it from the signal for further processing. However, multipath arrivals can also be used to help with classification and/or localization of sources. [46] For our purpose, if we have a recording of a single animal with direct and multipath arrivals, we can just regard the direct and multipath arrivals as two separate sources. Sometimes it is easy to distinguish between the direct and multipath; a good example of this can be found in the [45] data set. Figure 5.2.4 illustrates this.

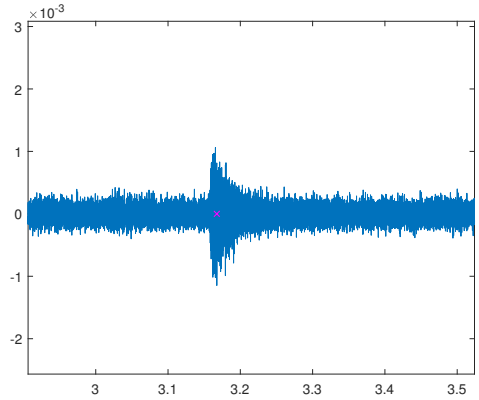
More formally, we can use Algorithm 5.1 with a low threshold to pick out all arrival times. Then we can manually go through the signal and annotated what we observe (i.e. direct arrival, multipath, false detection). These labeled times can serve as a sample dataset.



(a) Examples of multipath and direct arrivals in the recording



(b) Example of direct arrival



(c) Example of multipath arrival

Figure 5.2.4: Direct and multipath arrivals are clearly distinguishable in [45].

5.3 The nature of click trains

In order to develop some intuition about how click trains should behave, we take a look at a sample recording from [45] where a solely a single whale vocalizing has been identified (just as we have been doing in previously); henceforth we will refer to this as *the recording*. The first 5 minutes of the recording is shown in Figure 5.3.1. Previously, the clicks and times were identified an extracted in [46] which we will use for our analysis. In total there are 1102 clicks spanning just under 25 minutes.

5.3.1 ICI

Since we want to understand the timing of impulses, let us consider the ICI. Since this recording is just from a single whale, we can compute the ICI by taking the difference between adjacent clicks (i.e. $\tau_i - \tau_{i-1}$). The ICI for the entire recording is shown in Figure 5.3.2. We see that a majority of the values are under 2 seconds, but there are a few that are much larger. As was noted in [14] and corroborated by [19], ICI for sperm whales ranges from

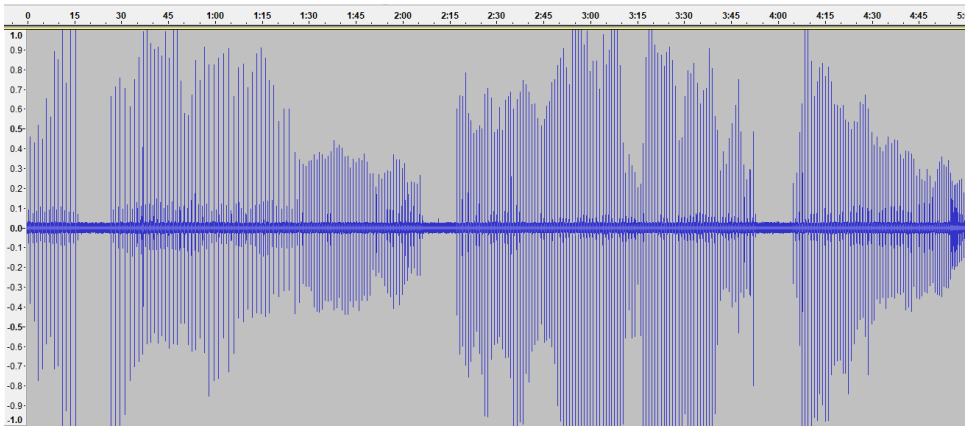
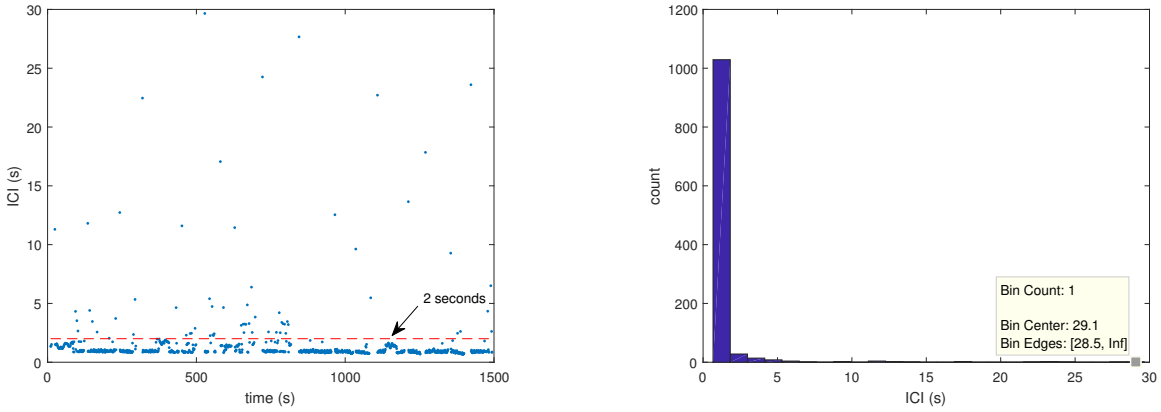


Figure 5.3.1: First 5 minutes of a recording from [45].

0.5-2 seconds, so what we are seeing makes sense. Here the larger ICI correspond to gaps in the recording. These gaps can be clearly seen in Figure 5.3.1, and like mentioned before, the gaps may be due to the whale turning its head or stopping to do something. Using 2 seconds as a cutoff, we will lose 65 "ICI" which means ≈ 0.04 clicks are missed every second. And the average length of the gaps (> 2 seconds) is 6.98 with a standard deviation of 6.95. The spread is rather large; the range is $2.04 - 29.66$. This is just a sample size of one as well, so we need to take any generalizations with a grain of salt.



(a) ICI. The red dotted line shows the cutoff of typical sperm whale ICI.

(b) Histogram of ICI

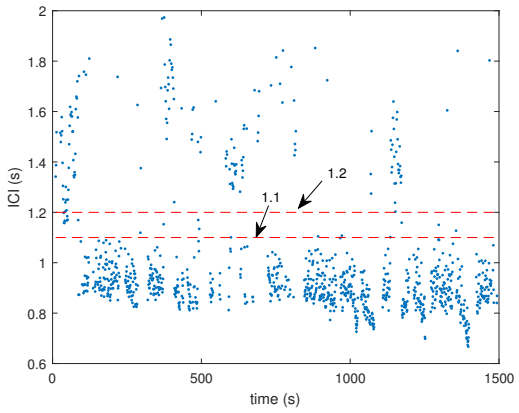
Figure 5.3.2: ICI for the recording

The pruned ICI are shown in Figure 5.3.3. There appears to be at least 2 distinct modes or $\mathcal{T}$ that this whale is switching between; there are distinct differences in time and values. The main one is shorter with $\mu = 0.87$ and more compact than the larger one which is more spread out with a median around 1.5.

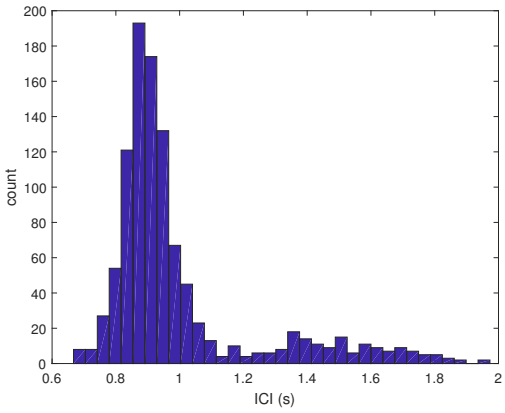
To further support this hypothesis consider Figure 5.3.4 which shows the 1-step difference in ICI. That is

$$(\tau_i - \tau_{i-1}) - (\tau_{i-1} - \tau_{i-2}).$$

This measure gives us some idea about the consistency of the ICI. We see that most values are close to 0 (within ± 0.2), but then there are a few larger ones around ± 0.8 which is roughly the difference in the means we see in Figure 5.3.3. This is promising, if we can just identify the different sections where different $\mathcal{T}$ are present, then we can apply Chapter 2 methods to the appropriate segments. Recall, that the PRI/ICI map (Section 3.3) gives us exactly the times when different ICI are present.

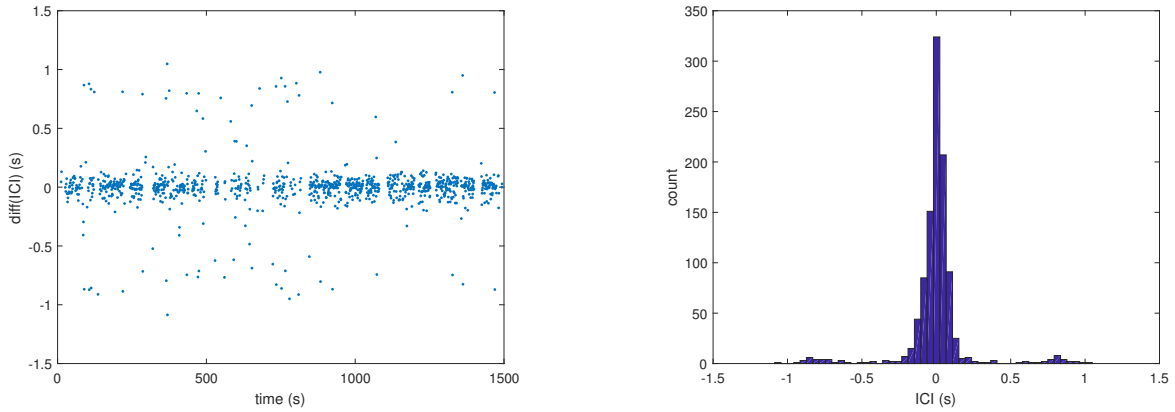


(a) ICI. The dotted red lines are here to emphasize that there is a split in the ICI values, but it is not an exclusive separation (i.e. there is some overlap).



(b) Histogram of ICI

Figure 5.3.3: ICI < 2 for the recording



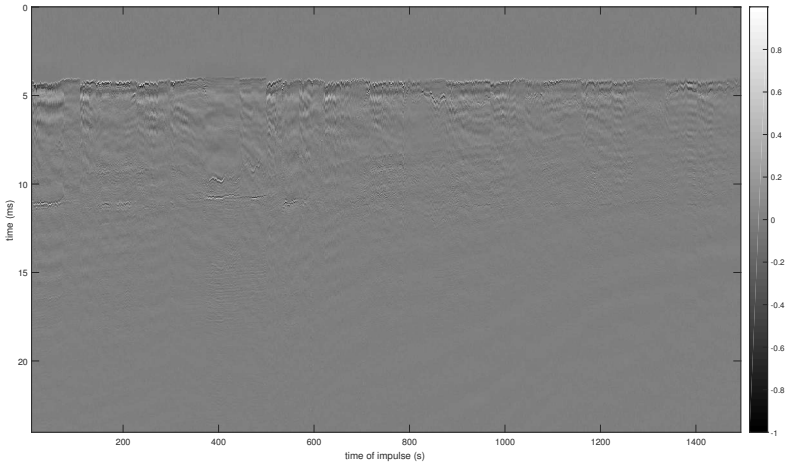
(a) 1-step difference of ICI

(b) Histogram of 1-step difference of ICI

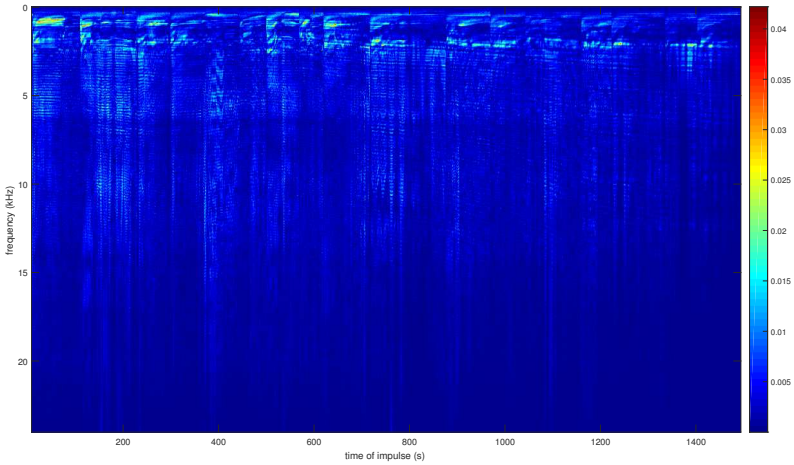
Figure 5.3.4: 1-step difference of ICI < 2 for the recording

5.3.2 Click shape

In addition to impulse spacings, we also want to analyze the shape characteristics of clicks and how they evolve over time. Figure 5.3.5 shows waterfall plots for the extracted clicks in the recording. These waterfall plots are not the same as the waterfall plots discussed in Section 3.4. Here a waterfall plot is where we take the information of a single click (e.g. the time series directly or its Fourier transform) and plot its values via a color in a vertical line. These lines are ordered horizontally with respect to their time of occurrence. With these plots, we are able to see how the shape of clicks vary over time. While there is some general similarity for all click shapes, like we saw with the ICI, there appear to be some distinct sections for different shapes. It is likely the sections correspond to the different ICI sections, but we need to more carefully cross reference. Furthermore within sections, adjacent clicks are generally very similar, but we do note slow variations over time which are visualized as smooth curves in the waterfall plots. This slow variation of shape is what is exploited in most other click classification methods. In Section 6.3 we show one way to incorporate this information with our methods.



(a) Waterfall of time series



(b) Waterfall of frequency spectra

Figure 5.3.5: Waterfall plots for the clicks in the recording

6

Practical timing separation for sperm whales

The majority of this paper assumes an ideal timing impulse separation problem. The outline of this problem is described in Section 2.1, but we will list the main points again here

- All sources emit impulses according to interimpulse distributions $\mathcal{T}_k$, and each interimpulse spacing is independent with respect to the others.
- All sources emit continuously for the entire duration of the recording.

For most situations these are not practical considerations, but adjustments can be made to handle more realistic scenarios (like we did in Section 2.3). The initial motivation for the research was to improve PAM for marine mammals; specifically we focus on sperm whale click trains and with hopes that the results can extend to other echolocating species.

As seen in Section 5.3, for sperm whales, we need to make concessions for the following:

- $\mathcal{T}_k$ cannot be considered stationary over long periods of time
- Some clicks get missed in the recording

Furthermore, historically, the shape of clicks have played a large role in the classification of click trains. That is, clicks that look similar are purported to come from the same animal, and there are a number of methods that take advantage of this [13, 14, 15, 16, 17]. Previously, we were arguing that timing information can separate clicks, but we would be remiss to completely ignore shape information. In this chapter we will also discuss how to incorporate click shape information.

6.1 Using the PRI map

As seen in Section 5.3, sperm whale click trains appear to be stable about some mean ICI over short periods of time. This leads to the idea that if we can identify these short periods and their parameters, we can apply the methods of Chapter 2 to classify the clicks at those times. As discussed at length in Section 3.3, this can be done using the PRI map of [48, 22]. In this section we provide a few examples to illustrate its use.

6.1.1 A good simulation

First we would like to show the method work on an ideal case. Consider three sources with

$$\boldsymbol{\Theta} = \{(0.7, 0.01^2), (2.3, 0.01^2), (1.5, 0.01^2)\}$$

at times as shown in Figure 6.1.1. We take the mixture of impulse times and generate the PRI map according to Algorithm 3.3 using $v = 10$, $b = 0.1$ and $\tau_j - \tau_{j-1} = 0.1$ and $t_i - t_{i-1} = 1$ and apply the threshold Γ for $P_{fa} = 0.05$.. The result is shown in Figure 3.3.3b with the true ICI overlaid. By all accounts, the map is a good representation of the underlying ICI. This makes parameter extraction and segmentation easy. In this case, we get three ζ_i shown in Figure 6.1.3 with where the means in $\tilde{\Theta}_i$ can be easily inferred by looking at the map. That is they are the correct means. Using AM and A2.2, the classification error rate for ζ_1 is 0.0182, 0.0204 for ζ_2, and 0 for ζ_3. This is an excellent result, as it was designed to be.

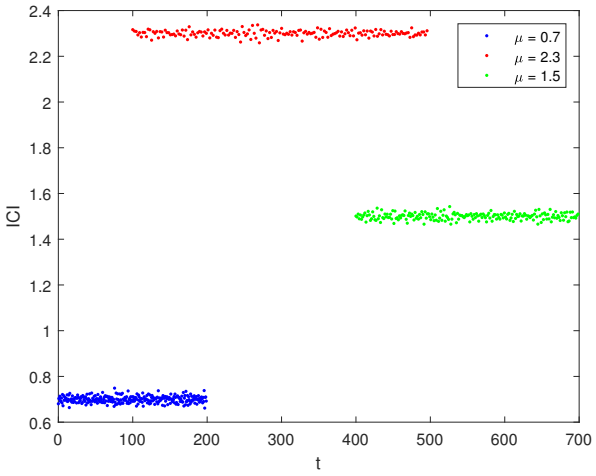


Figure 6.1.1: These are the simulated ICI for the example in Section 6.1.1.

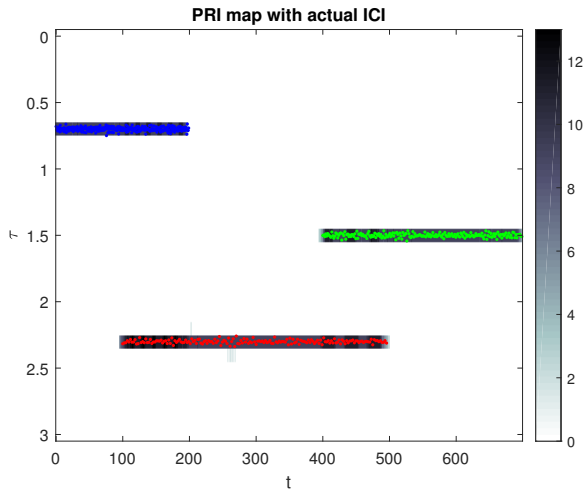


Figure 6.1.2: This is the PRI map for the impulse times in Figure 6.1.1 Note that the y-axis is has been flipped. The true ICI values are plotted above the map in the same colors as before.

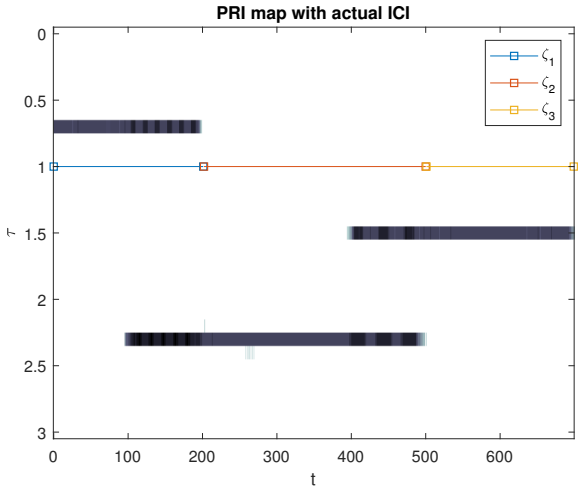


Figure 6.1.3: This is the PRI map for the impulse times in Figure 6.1.1 where the times of the 3 ζ_i are also mapped out.

6.1.2 A not so good simulation

The example in Section 6.1.1 was very good because the ICI were very close to their mean values. That is, σ_k was small. Such a small σ_k are not as realistic with sperm whales, $\sigma_k \approx 0.1$ would give a better model. So we repeat the same test, but this time with $\boldsymbol{\Theta} = \{(0.7, 0.1^2), (2.3, 0.1^2), (1.5, 0.1^2)\}$. The ICI are shown in Figure 6.1.4. There is a definitely a lot more variation.

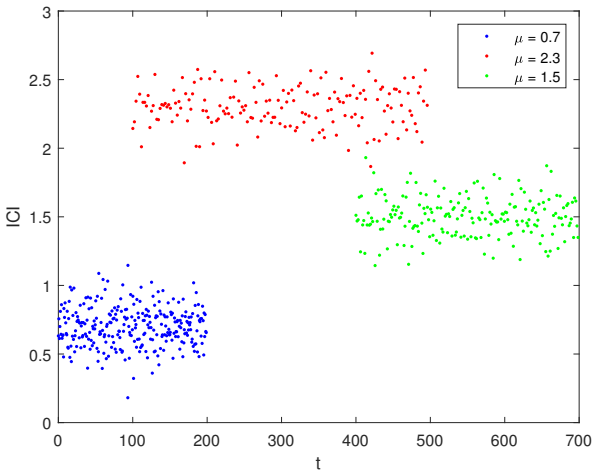


Figure 6.1.4: These are the simulated ICI for the example in Section 6.1.2.

Figure 6.1.5 shows the computed map (using the same parameters as before). There is agreement with the actual ICI of sources, but this means the larger variation is also translated in the map. Correspondingly, we estimate many different source parameters. So unlike the previous example, we get many ζ_i of which many are deemed too short to classify. Only 19 of the 72 ζ_i contain a significant number of clicks (for clarification we say the ζ_i should contain v sources). These are the segments shown in Figure 6.1.6. While these ζ_i cover most of the time, there are some significant gaps.

Additionally, P_e for each of these ζ_i is plotted as a function of its segment start time in Figure 6.1.7. The error rate is less than 0.1 except for 3 segments. So even though the segments are fractured, the classification of those segments is mostly satisfactory.

Perhaps we can improve the results for this example if we increase b and $\tau_j - \tau_{j-1}$ to better capture the larger variation due to an increased σ_k. However, this would reduce the resolution of the parameter estimates.

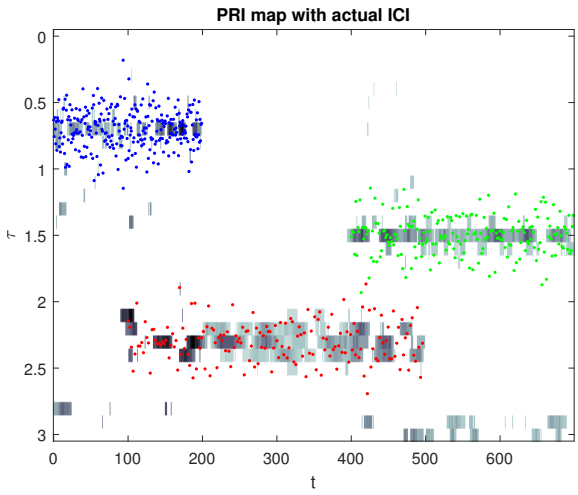


Figure 6.1.5: This is the PRI map for the impulse times in Figure 6.1.4 Note that the y-axis is has been flipped. The true ICI values are plotted above the map in the same colors as before.

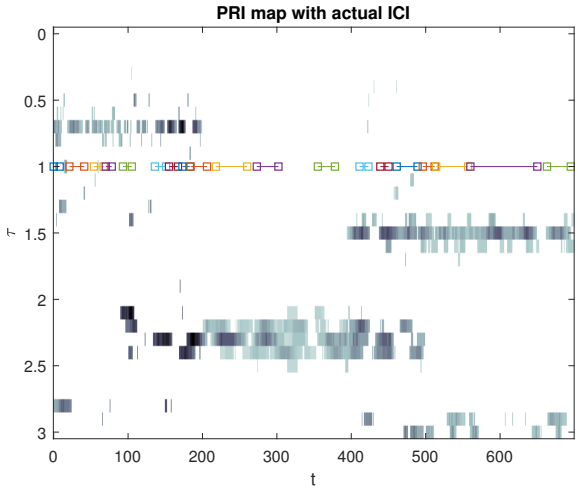


Figure 6.1.6: This is the PRI map for the impulse times in Figure 6.1.4 where the times of the 19 ζ_i that are long enough to classify are also mapped denoted by different colored lines.

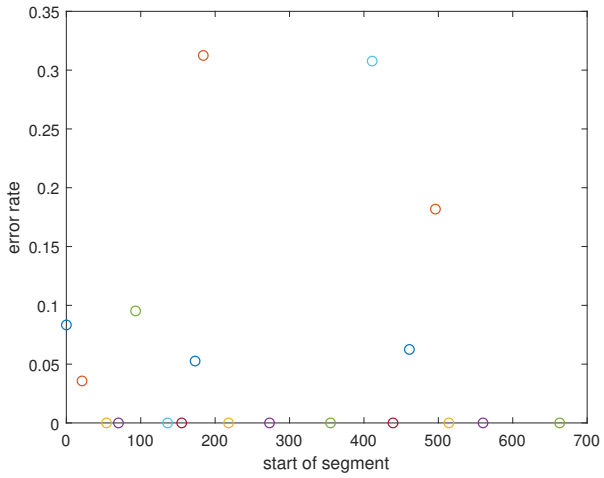


Figure 6.1.7: This is a plot of the error rate for each segment shown in Figure 6.1.6. Each dot's horizontal position indicates the start time of its associated segment. The color of the dot also corresponds to the color of the segment.

6.1.3 On multipath

Recall in Section 5.2.2, we discussed how to use multipath to create a test dataset from single animal recordings. We use that method to the dataset found in [45], and then apply the PRI map and classification methods of the previous examples. Refer to Section 5.3.1 to see how the ICI for the direct arrivals behaves. The ICI behavior of the multipath arrivals is nearly the same since they occur about the same amount of time after the direct arrival, though sometimes they are unable to be detected.

The resulting PRI map generated using the same parameters as the previous sections is shown in Figure 6.1.8. To reinforce that the map illustrates the true ICI, we do overlay the actual ICI on the map in Figure 6.1.9 though it is a little difficult to see the map beneath them.

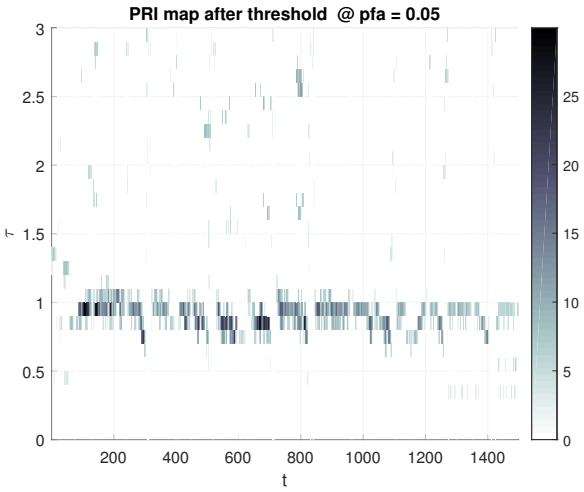


Figure 6.1.8: These are the PRI map for direct and multipath arrivals in [45].

When we estimate parameters and then segment this map, we generate many ζ_i, and like Section 6.1.2, many contain less than v clicks, so we do not attempt to classify. For the ones that contain v or more clicks, we classify and compute the error rate shown in Figure 6.1.10. Though there are a number of high error rate segments, most of the error rates are very good. This is even more impressive when we recognize that a combination of a multipath and direct arrival is corresponds to $\mu_1 = \mu_2$ in Section 2.2.2. It is the hardest mixture to separate, so perhaps the higher error segments are inevitable.

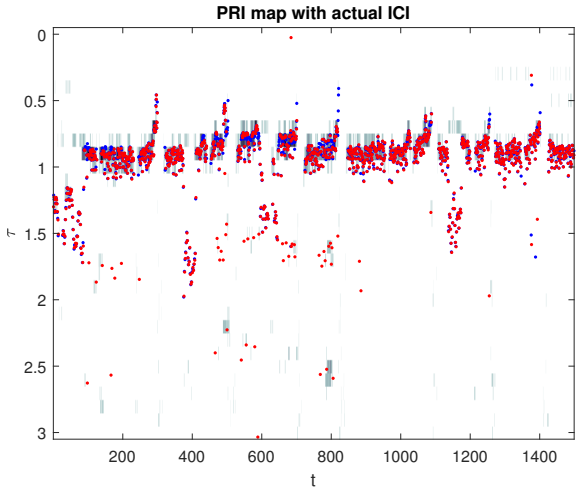


Figure 6.1.9: This is the PRI map in Figure 6.1.8 true ICI values are plotted above the map (Note that the y-axis is has been flipped). The blue dots are the direct arrivals, and the red dots are the multipath arrivals. It is hard to see the blue dots since there is a red dot that almost always immediately follows it. In this scale, this means the red dots usually overlaps the blue.

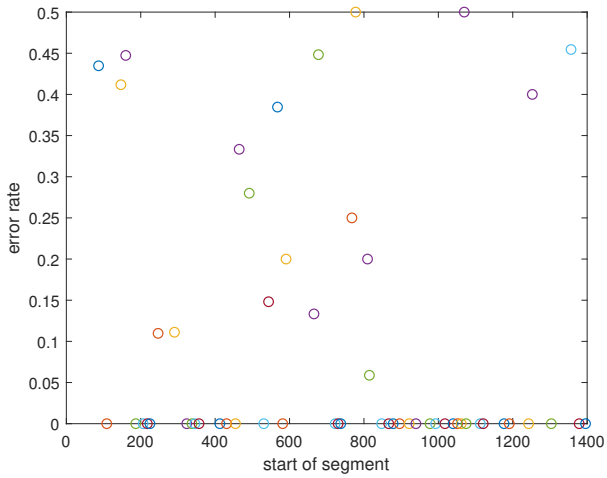


Figure 6.1.10: This is a plot of the error rate for each segment found extracted from the map in Figure 6.1.8. Each dot's horizontal position indicates the start time of its associated segment.

6.2 Missing impulses

For timing-only deinterleaving, [57] addressed problems with PRI jitter and missing pulses by using clustering, whereas [58] and [59] used the square sine wave transform and Fourier transform respectively. [60] used fuzzy adaptive resonance theory to help with this problem as well. However, without going into detail, these algorithms can be summarized as follows:

1. Get a candidate PRI (using clustering, sine wave transform, etc.).
2. Use candidate PRI in a sequential search (SS). If a sequence is found, assign it to the PRI and then remove it.

Repeat these steps until all impulses have been removed.

Similarly we can summarize our method as follows:

1. Estimate timing parameters (Chapter 3)
2. Using timing parameters for timing separation algorithm (Chapter 2)

A major difference is that we do not sequentially estimate timing parameters, but find them all at once. However, we can remove the influence of PRI/timing parameter estimation if supplant the estimation with the actual values in step 1. We can then focus on the step 2s and compare the difference.

For the SS, we use the method outlined in [7] which [57] refers to in their paper. In our implementation of SS, a pulse is said to be at target time x if τ, the time of arrival of the pulse, is within $x \pm \varepsilon$. This tolerance, ε, should be a function of the jitter to prevent loss, so we let $\varepsilon = 3\sigma$. Also, because we are giving the actual PRI, a sequence with the given PRI exists, so we simply accept any sequence greater than 3 impulses rather than comparing the total (weighted) count with a threshold.

We repeat the simulation from Section 2.2.2, with A2.2 and SS. We report P_e extend the range of σ_k to $[10^{-3}, 10^{1}]$ and average over 100 trials. The results are shown in Figure 6.2.1. A2.2 outperforms SS throughout, but both algorithms break down at higher values of jitter.

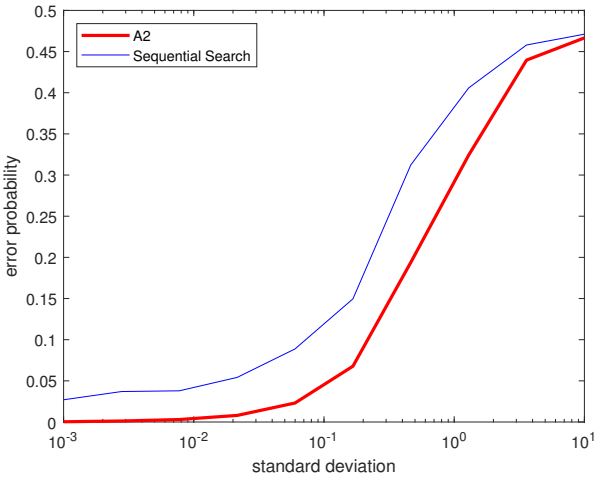


Figure 6.2.1: Error rates averaged over 100 trials as a function of σ_k for the SS algorithm and A2.2 with $\mu_1 = 1, \mu_2 = \pi$.

Missed impulses do happen in practice and complicate classification. The algorithms of Chapter 2 were not designed to handle this directly yet, but could be adjusted to include likelihood factors for missed impulses. Without

getting into this, we assess the performance of the current algorithms in the presence of missed detections. We repeat the experiment from Figure 6.2.1 using $\sigma_k = 10^{-1}$ and a rate of missing impulses, ζ, increasing from 0 to 0.5. In each trial, $M = 100$ impulse times are generated, and ζM impulses are removed at random. Results are shown in Figure 6.2.2. Even without modification to handle missing impulses, our algorithm matches or outperforms SS (which has built-in concessions for missing impulses).

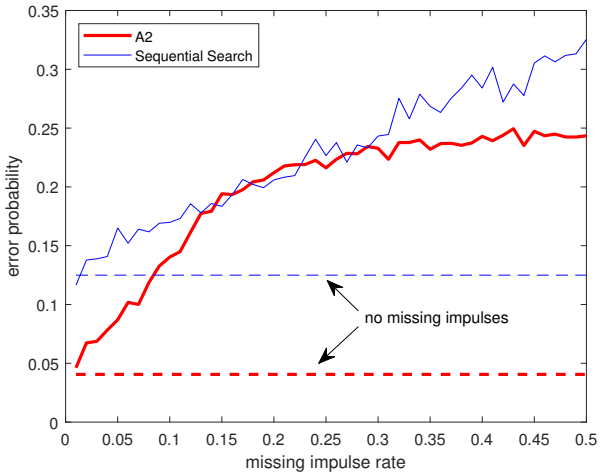


Figure 6.2.2: Error rates over 100 trials between SS algorithm and A2.3 with $\mu_1 = 1, \mu_2 = \pi$ and $\sigma_k = 0.1$ for different missing impulse rates.

6.3 Inclusion of click shape

In this section, we will discuss how to incorporate the impulse/click shape with the timing information. Consider the same model as in Chapter 4 (some of this discussion will be repeated from the chapter)

$$s(t) = \sum_{k=1}^{K} \sum_{i=0}^{N_k-1} h_k(t - \tau_{k,i}) + W(t), \tag{6.3.1}$$

where again we say that $W(t)$ noise. In Chapter 2, we only looked at the timing, $\tau_{k,i}$, but now we also want to consider the pulse shape, $h_k(t)$. We assume $h_k(t)$ are different for each source, and the differences in structure can be used for classification. For example, in our main application, sperm whale clicks, the impulses or clicks are known to have a multipulse structure that varies due to differences in physiology between individuals and on the positioning of the receiver in relation to the whale [61].

The signal $s(t)$ is almost always processed as the sampled version $s[n]$, so we will develop our algorithms for the sampled signal. Thus $h_k(t)$ is replaced by a discrete time version $h_k[n]$. Similarly, the noise is replaced by $W[n]$. The problem is now to find the best classification $\hat{\mathbf{D}}$ and impulse shapes $\{\hat{\mathbf{h}}_k\}$, where $\mathbf{h}_k$ is$\{h_k[n]\}_{n=1}^{H}$, the impulse shape for the k-th source. What is "best" depends on the context, but in one sense, we can maximize

$$\Pr\{\mathbf{D}, \{\mathbf{h}_k\} | \mathbf{s}, \{\tau_i\}\}, \tag{6.3.2}$$

where $\mathbf{s} = \{s[n]\}$ and $\{\tau_i\}$ are the detected click times.[1] to find the best assignment and pulse shapes. To be clear we assume

- The click times $\{\tau_i\}$ have been detected and the associated clicks at each τ_i have been extracted
- We know the number of sources K
- We know the timing distributions $\mathcal{T}_k$

As in Chapter 2, we hope to relax these assumptions later. With this in mind, we can take the log of (6.3.2) and use the definition of conditional probability[2] to write

$$\log \Pr\{\mathbf{D}, \{\mathbf{h}_k\}|\mathbf{s}, \{\tau_i\}\} = \log\left(\frac{\Pr\{\mathbf{D}, \{\mathbf{h}_k\}, \mathbf{s}, \{\tau_i\}\}}{\Pr\{s, \{\tau_i\}\}}\right).$$

With respect to maximization, the numerator is just a scaling factor so we can ignore it. Thus we have

$$\begin{aligned}
\log\left(\Pr\{\mathbf{D}, \{\mathbf{h}_k\}, \mathbf{s}, \{\tau_i\}\}\right) &= \log\left(\Pr\{\mathbf{s}|\mathbf{D}, \{\mathbf{h}_k\}, \{\tau_i\}\} \Pr\{\mathbf{D}, \{\mathbf{h}_k\}, \{\tau_i\}\}\right) \\
&= \log\left(\Pr\{\mathbf{s}|\mathbf{D}, \{\mathbf{h}_k\}, \{\tau_i\}\} \Pr\{\mathbf{D}, \{\tau_i\}|\cancel{\{\mathbf{h}_k\}}\cancel{\Pr\{\{\mathbf{h}_k\}\}}\right) \\
&= \log\left(\Pr\{\mathbf{s}|\mathbf{D}, \{\mathbf{h}_k\}, \{\tau_i\}\} \Pr\{\{\tau_i\}|\mathbf{D}\}\cancel{\Pr\{\mathbf{D}\}}\right) \\
&= \log \Pr\{\mathbf{s}|\mathbf{D}, \{\mathbf{h}_k\}, \{\tau_i\}\} + \log \Pr\{\{\tau_i\}|\mathbf{D}\} \qquad (6.3.3)
\end{aligned}$$

To get here we just used the definition of conditional probability. In the second line, we drop the conditioning on $\{\mathbf{h}_k\}$ in the second factor since the joint probability of the impulse times and assignments is independent of the impulse shapes.[3] Furthermore, as in Chapter 4, $\Pr\{\{\mathbf{h}_k\}\}$ can be ignored with respect to optimization if we do not place any constraints on the impulse shapes. That is, $\Pr\{\{\mathbf{h}_k\}\}$ is assumed to be uniform over its universe. Similarly, in the following line, $\Pr\{\mathbf{D}\}$ can be ignored. In the absence of any other information, all $\mathbf{D}$ are equally likely (i.e. $\Pr\{\mathbf{D}\}$ is assumed to be uniform over its universe). Our final objective function (6.3.3) contains two terms. Realize the second term $\log \Pr\{\{\tau_i\}|\mathbf{D}\}$ is actually the log-likelihood of the timing, (2.2.1), and it appears then that the first term the log-likelihood of the impulse shape (we will confirm this shortly). This indicates that the problem is somehow separable between shape and timing.

Since we have discussed the second term and its optimization at length already in Chapter 2, let us inspect the first one more closely. First we will assume that the noise $W[n]$ is white Gaussian, $W[n] \sim \mathcal{N}(0, n_0)$ where n_0 is known[4]. Furthermore, we assume that all $h_k[n]$ has finite support $[T_1, T_2]$, i.e., $h_k[n] = 0$ for $n \notin [T_1, T_2]$, where in general $T_1 \leq 0$ (the pulse detection will usually find the peak of the pulse). Then

$$\log \Pr\{\mathbf{s}|\mathbf{D}, \{\mathbf{h}_k\}, \{\tau_i\}\} \equiv \log \ell(\mathbf{D}, \{\mathbf{h}_k\}, \{\tau_i\}; \mathbf{s}) \triangleq L(\mathbf{D}, \{\mathbf{h}_k\}, \{\tau_i\}; \mathbf{s}),$$

[1] We are being loose with the $\Pr\{\}$, probability of an event, notation. Some of the arguments are discrete, for which this makes sense, but some are continuous. If X is a continuous random variable and x is some realization of it, $\Pr\{X = x\} = 0$. We do not actually mean this, instead when we say $\Pr\{x\}$ we are actually referring to $\Pr\{X \in [x - \delta/2, x + \delta/2]\}$ where δ is an infinitesimally small number; such a quantity is not necessarily 0.

[2] For events A, B,

$$P(A|B) \triangleq \frac{P(A,B)}{P(B)} \Leftrightarrow P(A,B) = P(A|B)P(B) = P(B|A)P(A).$$

[3] Assuming A, B are independent events

$$P(A,B) = P(A)P(B).$$

Then

$$P(A|B) = \frac{P(A,B)}{P(B)} = \frac{P(A)P(B)}{P(B)} = P(A),$$

and similarly

$$P(B|A) = P(B).$$

[4] The noise power n_0 can be easily estimated from the impulse free part of the signal.

where

$$L(\mathbf{D}, \{\mathbf{h}_k\}, \{\tau_i\}; \mathbf{s}) = -\sum_{k=1}^{K} \sum_{i} \frac{1}{2n_0} \sum_{n=T_1}^{T_2} (s[n - \Delta\tau_{S_k(i)}] - h_k[n])^2 + C, \tag{6.3.4}$$

where C is a constant that can be ignored with respect to optimization (it depends on n_0 and the length of the signal, but does not change for a given recording), Δ is the inverse of the sampling frequency, and we have used the assumption that pulses do not overlap.

Putting everything together, the solution is finding $\mathbf{D}, \{\mathbf{h}_k\}$ such that they maximize the log-likelihood (6.3.3) = (2.2.1) + (6.3.4). That is,

$$\hat{\mathbf{D}}, \{\hat{\mathbf{h}}_k\} = \arg \max_{\mathbf{D}, \{\mathbf{h}_k\}} -\sum_{k=1}^{K} \sum_{i} \frac{1}{2n_0} \sum_{n=T_1}^{T_2} (s[n - \Delta\tau_{S_k(i)}] - h_k[n])^2 + \sum_{k=1}^{K} \sum_{i} \log \mathcal{T}_k(\tau_{S_k(i)} - \tau_{S_k(i-1)}). \tag{6.3.5}$$

As in Section 2.3 this is an optimization over the discrete $\mathbf{D}$ and the continuous $\{\mathbf{h}_k\}$. Given an assignment $\mathbf{D}$, only the first term of (6.3.5) depends on $\{\mathbf{h}_k\}$ as (6.3.4); this is a standard ML problem (or least square problem) with the likelihood maximizing solution

$$\hat{h}_k[n] = \frac{1}{N_k} \sum_{i=1}^{N_k} s[n - \Delta\tau_{S_k(i)}]. \tag{6.3.6}$$

This confirms what we said before, the first term measures shape information.

On the other hand if an $\{\mathbf{h}_k\}$ is given, there is no simple analytical solution for $\mathbf{D}$. However, we know how to deal with (2.2.1) via Section 2.2. Recall, we discussed how the optimum assignment for the impulse τ_i is only dependent on the optimum path for each unique K-tuple. Now we will show the same is still true for (6.3.5) for given $\{\mathbf{h}_k\}$. The difference now is we have the extra term (6.3.4). Again assume we have an optimum path leading to impulse τ_i, and we need to decide which source to assign it to. Ignoring the (2.2.1) part, the best decision would be to choose

$$\hat{k} = \arg \min_{k} \sum_{n=T_1}^{T_2} (s[n - \Delta\tau_i] - h_k[n])^2.$$

That is, choose the source whose impulse shape matches the signal around τ_i closest. Here, the key point is that this decision has no dependence on any previous or future impulse times. This is less restrictive than timing part, so it adds no additional dependance to the total objective function. That is, (6.3.5) will depend optimum path for each unique K-tuple. To illustrate this, we revisit the same example in Section 2.2 which we used to originally motivate our algorithms (refer back to there for notation). First, let

$$f_k(\tau_i) = -\frac{1}{2n_0} \sum_{n=T_1}^{T_2} (s[n - \Delta\tau_{S_k(i)}] - h_k[n])^2, \tag{6.3.7}$$

this is the amount added to the objective function from (6.3.4) if the impulse at τ_i is assigned to source k. Consider $\mathcal{P}_{100} = [\ldots, 3, 1, 1, 2, 1, 2, 2]$, then $(d_1(\mathcal{P}_{100}), d_2(\mathcal{P}_{100}), d_3(\mathcal{P}_{100})) = (2, 0, 6)$. When impulse 101 is assigned to each of the three sources, one of the following terms is added to the log-likelihood

$$\begin{aligned} f_1(\tau_{101}) + \log \mathcal{T}_1(\tau_{101} - \tau_{98}) \quad &\text{if } D_{101} = 1 \\ f_2(\tau_{101}) + \log \mathcal{T}_2(\tau_{101} - \tau_{100}) \quad &\text{if } D_{101} = 2 \\ f_3(\tau_{101}) + \log \mathcal{T}_3(\tau_{101} - \tau_{94}) \quad &\text{if } D_{101} = 3 \end{aligned}$$

Whereas if $\mathcal{P}_{100} = [\ldots, 3, 2, 2, 1, 1, 2, 2]$, the K-tuple is still $(2, 0, 6)$, and terms potentially added are still the same

$$\begin{aligned} f_1(\tau_{101}) + \log \mathcal{T}_1(\tau_{101} - \tau_{98}) \quad & \text{if } D_{101} = 1 \\ f_2(\tau_{101}) + \log \mathcal{T}_2(\tau_{101} - \tau_{100}) \quad & \text{if } D_{101} = 2 \\ f_3(\tau_{101}) + \log \mathcal{T}_3(\tau_{101} - \tau_{94}) \quad & \text{if } D_{101} = 3 \end{aligned}$$

With regards to the assignment of the 101st impulse these two paths are identical. On the other hand if $\mathcal{P}_{100} = [\ldots, 3, 2, 1, 1, 2, 1, 2, 2]$, the K-tuple is $(2, 0, 7)$], and we get the following modified terms

$$\begin{aligned} f_1(\tau_{101}) + \log \mathcal{T}_1(\tau_{101} - \tau_{98}) \quad & \text{if } D_{101} = 1 \\ f_2(\tau_{101}) + \log \mathcal{T}_2(\tau_{101} - \tau_{100}) \quad & \text{if } D_{101} = 2 \\ f_3(\tau_{101}) + \log \mathcal{T}_3(\tau_{101} - \tau_{93}) \quad & \text{if } D_{101} = 3 \end{aligned}$$

Even with the added term, it is clear that these terms do not depend on what happened in the part of the path before the 3, indicated by $\ldots$. Thus, the likelihood for the decision at time $i+1$ only depends on the tail of the path, and this is exactly characterized by K-tuple. Therefore, given an $\{\mathbf{h}_k\}$, we optimize using the same algorithms as before, A2.1, A2.2, A2.3, A2.4, but with the (6.3.5) as the objective function instead of (2.2.1).

Since we have optimization methods that work given a $\mathbf{D}$ or given a $\{\mathbf{h}_k\}$, we consider an alternating maximization approach again. That is, from an initial $\mathbf{D}^{[0]}$, find $\{\mathbf{h}_k\}^{[1]}$. Then use $\{\mathbf{h}_k\}^{[1]}$ to find $\mathbf{D}^{[1]}$. Then use $\mathbf{D}^{[1]}$ to find $\{\mathbf{h}_k\}^{[2]}$, and so on until the $\mathbf{D}, \{\mathbf{h}_k\}$ converge or some maximum number of iterations is reached. Note $.^{[i]}$ indicates the i-th iteration. Making incremental improvements, to (6.3.5) does not guarantee we will find the global maximum, but we should at least be able to find a local max. Whether or not the local max is the global one will largely depend on where the algorithm is started. We decided to generate $\mathbf{D}^{[0]}$ using a Section 2.2 algorithm on the data ignoring impulse shape information, and it has shown good results. We will refer to this method as the *batch method*, and it is summarized in Algorithm 6.1. It should be noted that if we use a sub-optimal timing separation algorithm, we cannot say that (6.3.5) will always improve through the iterations. However, we can keep track of the objective function values and then output the $\mathbf{D}, \{\mathbf{h}_k\}$ corresponding to the highest.

Algorithm 6.1 The batch method using timing information and impulse shape

Given the signal $s[n]$, sampling width Δ, noise power n_0, K sources, distributions $\mathcal{T}_k$, impulse times $\{\tau_i\}_{i=1}^M$,

1. Get $\mathbf{D}^{[0]}$
   (a) Use a timing separation algorithm (e.g. A2.1, A2.2, A2.3, A2.4) on $\{\tau_i\}$ with K and $\mathcal{T}_k$
   (b) $i = 0$
2. Using $\mathbf{D}^{[i]}$, find $\{\mathbf{h}_k\}^{[i+1]}$ with (6.3.6)
3. Using $\{\mathbf{h}_k\}^{[i+1]}$, find $\mathbf{D}^{[i+1]}$
   (a) Use a timing separation algorithm but replace (2.2.1) with (6.3.5) as the objective function
4. Check for convergence
   (a) If $\mathbf{D}^{[i+1]} = \mathbf{D}^{[i]}$, then stop
   (b) Otherwise $i = i + 1$ and go to step 2.

Alternatively, use the shape-only algorithm for step one to generate $\{\mathbf{h}_k\}^{[0]}$ and use to find $\mathbf{D}^{[1]}$, alternating appropriately through iterations.

Algorithm 6.2 is another method to approximately maximize (6.3.5). The idea of the approach is use the same structure as Section 2.2 algorithms with (6.3.5), but to update the estimates for $\mathbf{h}_k$ as we try to assign each impulse. That is, when computing $f_k(\tau_i)$, the $h_k[n]$ is first computed by

$$\hat{h}_k[n] = \frac{1}{\tilde{N}_k} \sum_{\tau_{k,i} \in [0,\tau_i]} s[n - \Delta\tau_{k,i}], \tag{6.3.8}$$

where $\tau_{k,i}$ indicates the i-th are the impulse assigned to source k (here not necessarily the true assignment), $\tilde{N}_k$ is the number of impulses assigned to source k up to τ_i

$$\tilde{N}_k = \sum_{\tau_{k,i} \in [0,\tau_i]} 1.$$

These estimates are only optimal if the signal ended at τ_i, as opposed to using (6.3.6) which gives the ML estimates for shapes using the entire signal. Thus using (6.3.8) will not guarantee a maximized output. However, it has the benefit that it does not need go through the data over multiple iterations until it converges, and in simulations, its results are promising. We will refer to Algorithm 6.2 as the *online method*. Note, the online method can be modified in the same ways we modified A2.1 (i.e. removing STO paths, or only maintaining a survivor paths); we just need to make sure in step 2b we use (6.3.5) as the objective function.

Algorithm 6.2 Online method using timing information and impulse shape

Given the signal $s[n]$, sampling interval Δ, noise power n_0, K sources, distributions $\mathcal{T}_k$, and impulse times $\{\tau_i\}_{i=1}^M$

1. Initialize the assignment of τ_1 such that in the first K paths, τ_1 is labeled $1, \ldots, K$
   (a) Additionally, for each path k, set $\mathbf{h}_k$ to the first impulse shape (this is actually applying (6.3.8))
   (b) Set $i = 1$
2. For each path consider the extensions where τ_{i+1} is assigned to source $k = 1, \ldots, K$
   (a) Update $(d_1(\mathcal{P}_i), \ldots, d_K(\mathcal{P}_i))$ for each extended path as follows: if the extension is $\hat{D}_{i+1} = k$ then
   $$d_k(\mathcal{P}_i) = 0$$
   $$d_j(\mathcal{P}_i) = d_j(\mathcal{P}_i) + 1$$
   (b) Taking into account the assignment of τ_i to source k, recompute $\mathbf{h}_k$ according to (6.3.8)
   (c) Compute (6.3.5) using the above information for each extended path
3. Put the paths with same $(d_1(\mathcal{P}_i), \ldots, d_K(\mathcal{P}_i))$ into a group
4. In each group, eliminate all paths except the one with largest objective function
5. Check if finished
   (a) If $i = M$, finished, choose the most likely path as the assignment
   (b) Otherwise, increment i and go to step 2

One more important modification is the *shape-only algorithm*. As the name implies shape-only algorithm classifies the impulses based on shape information only. To do this, just use Algorithm 6.2, but instead of using

(6.3.5) as the objective, use

$$-\sum_{k=1}^{K}\sum_{i}\frac{1}{2n_0}\sum_{n=T_1}^{T_2}(s[n-\Delta\tau_{S_k(i)}]-h_k[n])^2.$$

That is, (6.3.5) without timing information. Again, this is a greedy approach, and optimality is not guaranteed, but it also has shown good results. Also as an alternative, the shape-only algorithm can be used to initialize the batch method with a shapes instead of an impulse assignments. A key benefit of the shape-only algorithm is that it does not require any knowledge of $\mathcal{T}_k$.

6.3.1 Parameter estimation

In practice, the noise $W[n]$ and the timing distribution $\mathcal{T}_k$ are unknown. Here we have been assuming that they are known to be Gaussian, but the parameters are unknown. That is

$$W[n] \sim \mathcal{N}(0, n_0)$$
$$\mathcal{T}_k \sim \mathcal{N}_0(\mu_k, \sigma_k^2).$$

For a given recording, assuming we know know the detection times τ_i and the length of an impulse H, we can easily extract the impulse free parts of the signal– call this $w[n]$. There are gaps in $w[n]$ (where the impulses are extracted from), but the areas outside of the gaps are just samples of $W[n]$. Therefore we can use them estimate n_0 using standard ML estimation. That is

$$n_0 = \frac{1}{N}\sum_{i=1}^{N}\left(w[i]-\frac{1}{N}\sum_{j=1}^{N}w[j]\right)^2,$$

where N is the total number of samples in $w[n]$; we have been loose with indicating the indices and gaps.

It is more difficult to find the parameters of $\mathcal{T}_k$, and is not realistic to assume that they are known. This is the topic of Section 2.3 and Chapter 3, but these these methods are only based on the impulse times. Now we have access to shape information. A simple idea to make use of shape is to use the shape-only algorithm to generate an initial assignment which we then can use to get an ML estimate for $\boldsymbol{\Theta}$. However, such an estimate is only based on impulse shapes, and is a greedy estimate at that. As with AM, we can improve the estimate over iterations by recomputing $\boldsymbol{\Theta}$ every time we get a new assignment. This application is straightforward for the online-method, the algorithm is just re-run many times with recomputed $\boldsymbol{\Theta}$ till convergence. For the batch-method, we iterate already, so $\boldsymbol{\Theta}$ should either be recomputed either immediately preceding step 3 or right after.

6.3.2 Results for impulse shape algorithms

There are situations where classification by impulse shape is sufficient (e.g. very clear and different impulse shapes), and there are situations where classification only by timing is a better choice (e.g. the impulse shapes are unclear or vary over time). There are also cases in between. In this section, we evaluate the error probability of the batch method, online method, and shape-only algorithm in various signal-to-noise ratios (SNRs) and timings.

The original motivation of this project was to separate sperm whale click trains, so we will use sperm whale clicks as our example impulses. Actual sperm whale clicks are extracted from the dataset mentioned in Section 5.3. We select clicks from two different sections, and then we align and average them to give a two different representative clicks. Specifically, the final clicks have $H = 0.024$ s (1154 samples at rate $f_s = 48$ kHz) and are normalized to unit power. They are shown in Fig. 6.3.1.

Similar to Figure 2.3.2, we will fix one timing distribution and vary the other. That is, impulse times are generated with parameters $\boldsymbol{\Theta} = \{(0.5, 0.13^2), (\mu_2, 0.13^2)\}$ using the data generation algorithm (Algorithm 2.5). However,

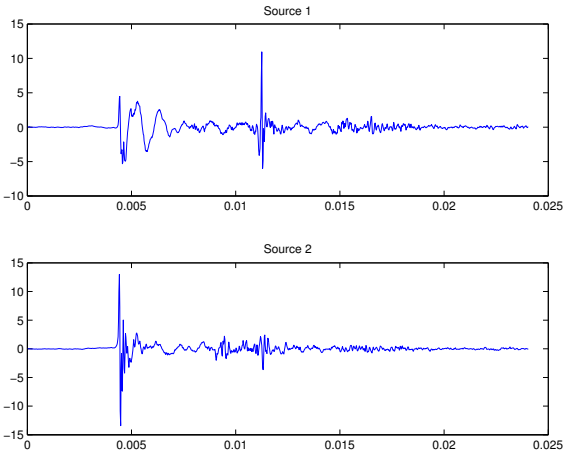


Figure 6.3.1: These are the normalized clicks for (top) source 1 and (bottom) source 2.

care is taken to ensure that times do not result in overlapping impulses (i.e. each impulse time is at least H apart, datasets that violate this are discarded). Enough impulses are generated such that they span approximately 10 seconds; a typical set will contain 25-40 impulse times total.

With the impulse shapes and times, we construct the entire signal in two steps. First, we make the noise-free signal

$$s_p[n] = \sum_{k=1}^{K} \sum_{i=0}^{N_k-1} h_k[n - \Delta\tau_{k,i}].$$

Then a noise sequence $\tilde{w}[n] \sim \mathcal{N}(0,1)$ of the same length is also generated. With $s_p[n]$ and $\tilde{w}[n]$, a signal of arbitrary SNR α can be made according to

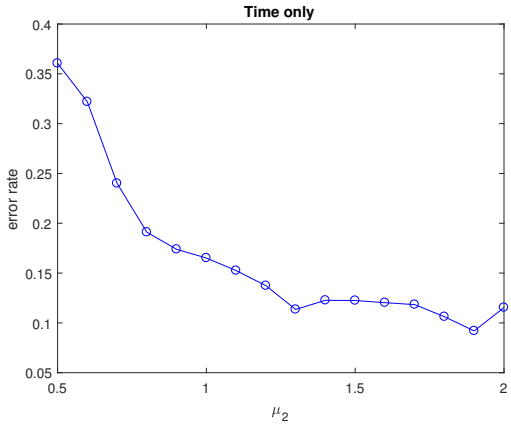
$$\sqrt{\alpha} s_p[n] + \tilde{w}[n]. \tag{6.3.9}$$

The datasets considered have μ_2 from 0.5 to 2.0 (parameters are chosen to mimic actual sperm whale timings) and α from 0.5 to 3. For each μ_2, α pair 100 different timings and signals are generated. The error rate is computed for each signal, and the rate averaged across all 100 trials is reported and shown in Figure 6.3.2. For these results, we use the techniques in Section 6.3.1 to give n_0 and $\mathcal{T}_k$ (including iterating up to 50 times to improve $\boldsymbol{\Theta}$). The STO variant of the algorithms are used to save on computation time.

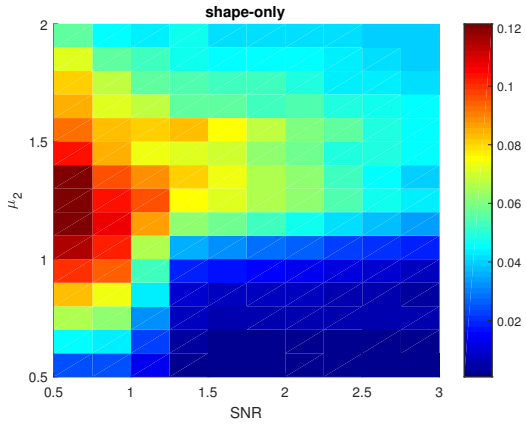
As expected, and seen in previous results where only timing information is used, performance is poor when μ_2 is close the μ_1 and improves as their distance grows. Also as expected, the shape-only algorithm does better at higher SNR. However, its performance also exhibits some dependence with the timing mean which is most apparent when the SNR< 1. Recall that the estimates for the pulse shapes are made sequentially, so the order of the pulses, which is dependent on the timing parameters, affects the final result. Specifically, we see a degradation in performance when $\mu_2 \approx 2.5\mu_1$. So when there are ≈ 2.5 pulses from one source before the other, the greedy scheme makes the wrong decisions. If timing information is added to the shape-only algorithm (i.e. the online method), we see that there is only a very marginal improvement ($\bar{P}_e = 0.0429$ versus $\bar{P}_e = 0.0444$ for shape-only). Whereas, the batch method shows significant improvement– almost everything is classified correctly. This is not completely surprising, the online method is a greedy approximation, so it should perform worse than the more optimal batch method.

The results indicate that the batch method is the best choice, and that the online method is not really useful.

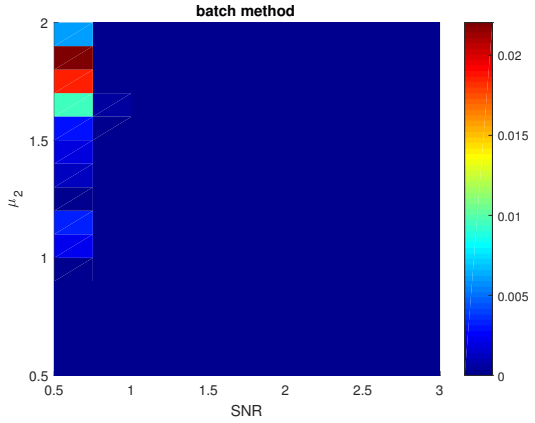
If the interest is just in speed, the shape-only method appears to suffice. However it should be noted that we constructed a simulation that matched our assumptions (i.e. white noise and a pulse shapes that do not vary) exactly. In the ocean, the noise is not white and pulse shapes from sperm whales vary slightly between clicks (as shown in Figure 5.3.5). So when applied to real data, the results will be different. We would need to improve our simulation model to get more accurate results and/or update the system model (6.3.1) to develop a better method.



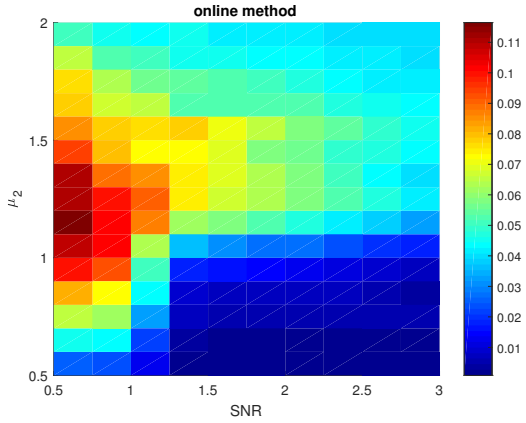
(a) This is the average error probability of A2.3 initialized by the shape-only results for parameters for $\mu_1 = 0.5$ and μ_2 as shown above.



(b) This is the average error probability of the shape-only algorithm for $\mu_1 = 0.5$ with μ_2 and SNR as shown



(c) This is the average error probability of the batch method for $\mu_1 = 0.5$ with μ_2 and SNR as shown



(d) This is the average error probability of the online method for $\mu_1 = 0.5$ with μ_2 and SNR as shown

Figure 6.3.2: Average error rates of a time-only method (A2.3), shape-only algorithm, online method, and batch method for sperm whale like timing ($\boldsymbol{\Theta} = \{(0.5, 0.13^2), (\mu_2, 0.13^2)\}$) and clicks at varied SNR are shown here. Parameter estimation is done as in Section 6.3.1.

7 Conclusion

In this thesis we explored the use of timing to separate sources with the intention of separating sperm whale click trains. To begin, we started with the idealized timing separation problem outlined in Section 2.1 where the number and Gaussian timing distributions of sources are known. Brute force separation is computationally infeasible, but with inspiration from the Viterbi algorithm, we developed a feasible method to separate mixtures optimally. To further improve computational performance, we made made simplifying approximations and showed that the classification error rate of the simplified algorithms are close to optimal one. Under the assumption of truncated Gaussian timing distributions, a lower bound was found for mixtures of two sources. This lower bound was tested verified through simulations, and it was found that the $c_v = \sigma/\mu$ of sources is related to error rate. The actual relationship is complex, but generally, error rates are higher when c_v is higher. The important result here is that the amount of relative jitter, c_v, determines the viability of separation by timing. That is if c_v is too high, a different method should be considered. Though it is also important to note that even in the presence of jitter, our timing separation algorithms still perform better than other methods, like sequential search, that do not handle jitter optimally.

Additionally, we relaxed the assumption of knowing the timing distribution parameters and presented methods to jointly estimate parameters and separate impulses. For this mixed optimization problem, we considered AM and EMV, and they were shown to be reliable around a certain range about the true parameters. That is, they are somewhat sensitive to the initialization point. So next we focused efforts in developing heuristics to get good initial estimates for the parameters of timing distributions. In particular, methods based on the $\delta\tau$-histogram seemed to be the most useful. Equipped with parameter estimation, we relaxed the assumption of also knowing the number of sources. Then we showed using the minimum description length principle was able to discern the number of sources.

We explored the use of timing to improve detection methods. Using a Bayesian framework and assuming that the impulse shape does not change between emissions we got an objective function to maximize that consists of the timing log-likelihood function plus a likelihood function related to the shapes of impulses detected. This optimization problem is harder than timing-only separation; knowledge of the impulse shapes from each source is necessary but impossible to estimate without knowing all the detections. We took a greedy approach to come up with an approximate method, but we still needed to assume we knew the timing parameters and at least the first detection times for each source. Due to these assumptions, these methods are impractical. However, the analysis

for the detection problem inspired a classification method using impulse shape and timing. Also using the same idea, we proposed a method for classification using impulse shape only that could be used as an initialization point for AM or EMV.

This inclusion of impulse shape was related to the goal of adapting our timing based separation methods for sperm whale click train separation. In addition to utilizing shape, we needed to account for the non-ideal timing problem where we assume that all whales constantly emit for the entire duration of recordings. The PRI map was used to help segment recordings into times where we can assume the idealized model. We also developed methods based on the PRI model to extract timing parameters and and estimate the number of sources. It is still a work in progress, but the results so far are promising. Existing detection methods are not always perfect and whales might skip clicks for one reason or another, so missing impulses are a concern as well. However even without any change in the timing separation algorithm, we outperformed a sequential search method which has concessions built in to handle a missed impulses.

7.1 Future directions

There is still more work to be done with regards to a practical implementation for sperm whale click trains. The PRI map segmentation and parameter extraction works, but it needs improvement in producing longer segments. This might be accomplished by adjusting map parameters, adding in some post filtering operation to average the map values, or averaging of parameters during the segmentation operation. One likely reason for short PRI map segments is missed clicks in a train showing up as gaps in the map. We can achieve longer segments by closing these gaps, but then we will likely need to account for these missed clicks more directly. For this, A2.4 is one approach that we can expand upon. Additionally, the number of sources output for a PRI map segment may overestimate the actual number of sources. The use of MDL has been useful in preventing overestimation, but it is worth exploring if there are any modifications to the separation algorithm itself that can be made to discourage the use of extra sources.

With both missed clicks and extra sources, the real issue is management of outlier ICIs. In the ideal timing separation problem, we assumed that these outliers do not exist; this is why we discounted solution paths with STO. However, in real mixtures, true outliers show up, and these outliers can lead to very low likelihood values which would normally eliminate the solution even though it is the true assignment. One solution would be to adjust the penalty an outlier incurs on the overall likelihood, but exactly how to adjust the value so that to avoid false outliers needs to be ascertained.

We did present a method where we married timing separation with impulse shape classification, but there is more room to grow here too. The method is based on the assumption that click shape for each source is constant which is not necessarily true (e.g. as the whale moves, the channel changes). Rather we should allow for slow variations click-to-click. Perhaps the online method presented is a step in the right direction, but it needs to be expanded upon. Another approach would be to impose distributions on a feature (or sets of features) as we did with the timing. However, this presents the challenge of deciding which features to use and determining their distribution.

In this paper we were focused on developing an optimal method for the separation of renewal processes. While there are other methods that separate based on timing only, many assume a different model, and then those that do, use heuristics to modify a method that assumes no jitter to handle the jitter. It would be easy to simulate data for which our algorithms do better and vice versa. To this end, we felt that it was unfair and unnecessary to compare with these methods, at least in order to prove that our proposed method works in theory. However, when tackling the problem of separating actual sperm whale clicks, comparisons are needed to provide evidence that we can do better or at least provide some benefit over existing algorithms. While we could try to assert that our assumed model is the correct one via analysis of verified data, in the end it comes down to which method produces the best result regardless of the assumptions made. Regrettably, as discussed above, we did not reach a

final method for practical separation and so were not able to compare yet.

Finally, we have always been assuming a single sensor, it would be interesting to see how timing be used in a multiple sensor problem. With more information, we should be able to achieve even better separation. An initial approach might be to do the separation on each sensor and then cross validate the results. Also, though the timing is considered is the time between the same click in different channels and not ICI, we used pairs of hydrophones to do manual separation by TDOA. Finding a way to automate this process would be very valuable. It is possible ICI regularity could be exploited to help in the automation as well.

A

Generating samples from distributions

In order to validate performance of algorithms, we need to be able to generate controlled sampled data. An important step in this process is drawing samples from some specified distribution. Typically there will always be some utilities in software to pull a (pseudo)random number from a uniform distribution on the interval $[0, 1]$ (e.g. `rand()` in C) [1] and often also from the standard normal distribution $\mathcal{N}(0, 1)$ (e.g. `randn()` in MATLAB). In this Appendix, we will describe how to pull samples from any specified distribution under the assumption that these utilities exist. Appendix A.1 will discuss the method for an arbitrary Gaussian, whereas Appendix A.2 will present methods for any pdf specified by an $f(x)$.

A.1 Arbitrary Gaussian $\mathcal{N}(\mu, \sigma^2)$ from $\mathcal{N}(0, 1)$

Assume we have a method to get draw samples x form $X \sim \mathcal{N}(0, 1)$. We can generate the samples y from $Y \sim \mathcal{N}(\mu, \sigma^2)$ using the following transformation

$$y = \sigma x + \mu. \tag{A.1.1}$$

Proof. The above transformation corresponds to $Y = \sigma X + \mu$. If this Y has the proper moments, then we verify the operation. First note

$$\begin{aligned} E[X] &= 0 \\ E[X^2] &= 1. \end{aligned}$$

Then consider

$$\begin{aligned} E[Y] &= E[\sigma X + \mu] \\ &= \sigma E[X] + \mu \\ &= 0 + \mu = \mu \end{aligned}$$

[1] Getting a good (pseudo)random number is not a trivial process and is of particular interest itself. However, this is outside the scope of this paper.

This is correct, we want the expected value of Y to be μ. Now let us check the second moment

$$\begin{aligned} E[(Y - E[Y])^2] &= E[Y^2 - 2YE[Y] + E[Y]^2] \\ &= E[Y^2] - 2E[Y]^2 + E[Y]^2 \\ &= E[(\sigma X + \mu)^2] - 2\mu^2 + \mu^2 \\ &= E[\sigma^2 X^2 + 2\mu\sigma X + \mu^2] - \mu^2 \\ &= \sigma^2 \underbrace{E[X^2]}_{1} + 2\mu\sigma \underbrace{E[X]}_{0} + \underbrace{\mu^2 - \mu^2}_{0} \\ &= \sigma^2. \end{aligned}$$

□

A.1.1 Arbitrary truncated Gaussian $\mathcal{N}_a(\mu, \sigma^2)$ from $\mathcal{N}(\mu, \sigma^2)$

In our notation, $\mathcal{N}_a(\mu, \sigma^2)$ indicates a Gaussian distribution with mean μ and variance σ^2 that has been cut off on the left at a (distributions can also be truncated from above too). More specifically, if $X \sim \mathcal{N}_a(\mu, \sigma^2)$, then

$$\Pr\{X < a\} = 0.$$

And importantly the following is not necessarily true

$$\begin{aligned} E[X] &= \mu \\ E[(X - E[X])^2] &= \sigma^2. \end{aligned} \tag{A.1.2}$$

Importantly, μ and σ^2 are the mean and variance of the distribution that is truncated, not the moments of the truncated distribution. However note, in some cases $\mu \gg a$ and σ^2 is not large, so the effect of truncation is negligible and (A.1.2) ends up being essentially true. See Appendix B.1 for more info.

Getting samples from a truncated distribution is simple if a method to draw samples from the underlying (i.e. non-truncated) distribution exists. Let $\tilde{x}$ be a sample drawn from the underlying distribution, keep it as a sample from X if $\tilde{x} > a$. Otherwise, reject $\tilde{x}$ and draw another one. This can be repeated as many times as desired. Note, the same process can be used to truncate from above as well – just throw away any samples $> b$ if b is the upper bound.

A.1.2 Arbitrary Gaussian to standard Gaussian

Similar to the first transformation, we can go the other way to transform an arbitrary Gaussian $Y \sim \mathcal{N}(\mu, \sigma^2)$ to the standard one $X \sim \mathcal{N}(0, 1)$ using the following transformation

$$x = \frac{1}{\sigma}(y - \mu).$$

Proof. This is the inverse of (A.1.1), so the proof is just the opposite. We simply evaluate the moments of the transform to verify that indeed has mean 0 and variance of 1.

$$\begin{aligned} E[X] &= E[\frac{1}{\sigma}(Y - \mu)] \\ &= \frac{1}{\sigma}(E[Y] - \mu) \\ &= \frac{1}{\sigma}(\mu - \mu) = 0. \end{aligned}$$

And similarly

$$\begin{aligned} E[(X - E[X])^2] &= E[(\frac{1}{\sigma}(Y-\mu) - 0)^2] \\ &= \frac{1}{\sigma^2}\underbrace{E[(Y-\mu)^2]}_{\triangleq\sigma^2} \\ &= \frac{1}{\sigma^2}\sigma^2 = 1. \end{aligned}$$

□

A.2 Arbitrary distribution from a random number generator

In this section we will briefly describe Gibbs sampling [62] and the Metropolis-Hastings algorithm [63] which are methods draw samples from an arbitrary pdf specified by $p(x)$.

Metropolis (Appendix A.2.1) and Gibbs (Appendix A.2.2) are two ways to get the "same" result. Which one is better will depend on your application. For example, Gibbs sampling a 1D function can be immediately paralleled for efficiency, but the same cannot be said for Metropolis. However, for multiple dimension problems, it is not possible to parallel the procedure for either Gibbs or Metropolis since the next sample always depends on the previous one. With Metropolis, there is also a lot more fine tuning and user-set variables than Gibbs.

A.2.1 Metropolis-Hastings

Algorithm A.1 describes a produce to samples from an arbitrary pdf $p(x)$.[2] It assumes we are sampling a continuous value, but the procedure directly generalizes to discrete values (just replace any integrals with sums and skip any "function sampling" steps). It is written to get samples x, 1D values, but it is exactly the same for a multidimensional $\mathbf{x}$.

While the algorithm will work as is, there are a few more things to consider. First, we might choose a bad starting value way out of the range of the function. Then it would take some time for the jumps to get into the functions range. In other words, it would take some time before the samples started to actually be relevant, so the use of a "burn-in" period is advised. That is, throw away the first set of samples (e.g. the first 1000).

Another source of error is that Metropolis tends to produce correlated samples. This can pose a problem if we want to produce independent samples – if samples are correlated, they are not independent. The correlation occurs because, to some degree, the current sample is generated from the previous. For example, Figure A.2.1 shows the absolute value of the first 1000 autocorrelation coefficients (a measure of the level of correlation, see Appendix A.2.3) from a 10^6 sample generation run. Ideally, it would be low everywhere, but until the lag reaches 200, the correlation coefficient is not small. A simple solution is just take every 200th sample; this is called "thinning," and the improved result is also shown in Figure A.2.1. However, the act of thinning presents another problem: when we thin, we lose samples. In the example shown we go from 10^6 samples down to 5000, 0.5% of the original amount. To overcome this, we can just generate more samples. Say we only keep every L-th sample, and we want N total samples, then we must use the algorithm to generate LN samples which may not be feasible if N and/or L is large.

[2] If we are given an arbitrary positive function (i.e. $f(x) \geq 0 \forall x$) instead, we can convert it to a valid pdf by scaling it. That is use $p(x) = \frac{f(x)}{\int f(x)dx}$.

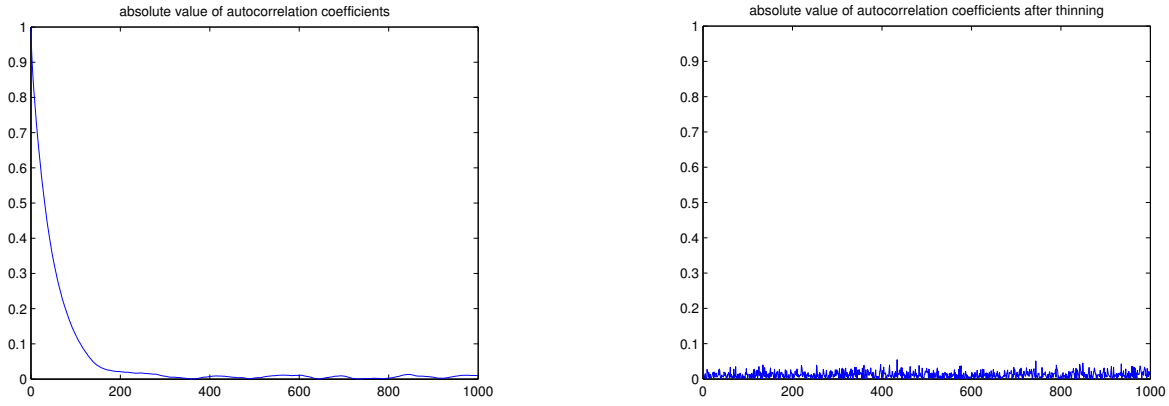


Figure A.2.1: (Left) Using J_1, these are the absolute value of the first 1000 correlation coefficients after removing the burn in period. (Right) After removing every 200 samples (out of a 10^6 sample generation run), we get a different set of coefficients.

Algorithm A.1 Metropolis-Hastings algorithm

Given a distribution $p(x)$ to draw samples from:

1. Pick a starting value $x^{(0)}$. Result should be independent of the choice. Set $t = 1$.
2. Pull a sample from the jumping/proposal distribution $J(x^*|x^{(t-1)})$.
   (a) This distribution should be symmetric. That is $J(x_1|x_2) = J(x_2|x_1)$
   (b) Typically, we define an auxiliary random variable
   $$Y = X_1 - x_2$$
   where Y's distribution is even (i.e. $p_Y(y) = p_Y(-y)$). For example, $Y \sim \mathcal{N}(0,1)$, or Y is uniform on the symmetric interval $[-0.5, 0.5]$. This distribution should be "easy" to sample from in the sense that the system/code has some built-in capability to do so (e.g normal or uniform distributions). Using this auxiliary variable we can also write
   $$X_1 = Y + x_2$$
   Now it is clear that $p(X_1|x_2)$ (i.e. J) is simply $p_Y(y)$. Thus we conclude that J is symmetric since $p(X_1|x_2) \equiv p_Y(x_1 - x_2) = p_Y(-(x_1 - x_2)) = p_Y(x_2 - x_1) \equiv p(X_2|x_1)$.
      i. Many systems can pull a value from $Z \sim \mathcal{N}(0,1)$. To get $Z' \sim \mathcal{N}(\mu, \sigma^2)$, use
      $$Z' = \mu + \sigma Z$$
      Indeed you can check
      $$E[Z'] = E[\mu + \sigma Z] = \mu + \sigma \underbrace{E[Z]}_{0} = \mu$$
      $$\begin{aligned} E[(Z' - \mu)^2] &= E[(\mu + \sigma Z - \mu)^2] \\ &= \sigma^2 \underbrace{E[Z^2]}_{1} \end{aligned}$$
      $E[Z^2] = 1$ because $E[(Z-0)^2] = 1 \Rightarrow E[Z^2] = 1$. We are mainly interested in changing the width of the distribution (σ) for this application– moving the mean would make the distribution unsymmetrical.
      ii. Many systems can pull a value $U \sim$ Uniform$([0,1])$. To change this to a sample on the interval $[-b, b]$, we just need to scale and shift. First note, the interval is $b - (-b) = 2b$ long, so we stretch our interval by multiplying the samples with $2b$. This gives us $2bU \sim$ Uniform$([0, 2b])$. To center, shift to the left by subtracting b giving us $(2bU - b) \sim$ Uniform$([-b, b])$. It is possible that you could run into numerical issues with the scaling if b is particularly large or the system does not pull samples with fine enough granularity, but for most cases you should not have to worry.
   (c) To generate the sample x^* from the distribution, pull a sample of Y from the system and add it to $x^{(t-1)}$
   (d) The choice of J does not matter so much as long as it is symmetric. However some J will converge faster than others.

Continued in Algorithm A.1.

Algorithm A.1 Metropolis-Hastings algorithm (continued)

3. Evaluate the probability of the candidate sample $p(x^*)$
   (a) If $p(x^*) > p(x^{(t-1)})$, then set $x^{(t)} = x^*$.
   (b) If $p(x^*) \leq p(x^{(t-1)})$, pull a sample u from a uniform distribution on $[0, 1]$ (system should have this built in).
      i. If $u < r = \frac{p(x^*)}{p(x^{(t-1)})}$, set $x^{(t)} = x^*$
      ii. If $u \geq r$, set $x^{(t)} = x^{(t-1)}$. That is, we reject this new sample
4. Increment t and go to step 2.

We stop when some specified maximum number of samples is reached or we meet some convergence criteria. For example, if the statistics of the first half of the samples match the second half. That is, take all of the samples, split them in half and compute the mean for each. If the means match, then we have enough samples. Another approach would be to start multiple sampling chains at once, and compare the statistics between them. The idea is the same as comparing the first half of samples with the second half; stop when the statistics match.

A.2.2 Gibbs sampling

Suppose we want to draw samples from an arbitrary unscaled pdf (i.e. improper; does not sum to 1) which is specified by some sampled values

$$\{p(x_i) : x_i\}_{i=1}^{N_X}$$

where x_i is the value of the event sorted such that $x_{i-1} < x_i$ and $p(x_i)$ is the probability of the event. Typically these values arise from sampling a function $f(x)$ at x_i (usually regularly spaced) or perhaps if there was a histogram of events that needed to be replicated/simulated. The values are bounded in $[x_1, x_{N_X}]$, and assume that outside of these specified values it the probability is 0.

The idea of Gibbs sampling is to pull a random number from $[0, 1]$ and find its corresponding value on the CDF to take as the sample. The exact procedure is described in Algorithm A.2. Note, if you are given a proper pdf, just skip the first step. Furthermore if instead a continuous function is given, one option is to simply sample it to get to get to the starting point. Alternatively, just use it directly and ignore any approximations given.

Gibbs does not have the same issue with correlated samples like Metropolis since each sample is generated without any dependence on the previous ones. Additionally, this means this process can be directly paralleled; that is, we can run the procedure then combine the results to maximize efficiency.

Multiple dimension Gibbs

Above is a method to sample from a pdf via Gibbs sampling. It is only 1D, but it can be generalized in the following manner. Consider we have some (unscaled) multivariate pdf given by $f(\mathbf{x})$, where $\mathbf{x}$ is an $M \times 1$ vector and M is the number of variables. Note, $f : \mathbb{R}^M \to \mathbb{R}$. For example, $\mathbf{x} = [x, y]^T$ and

$$\begin{aligned} f(\mathbf{x}) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\ &= x + y \end{aligned}$$

Equivalently, we could have specified the function

$$f(x, y) = x + y$$

but we opt for the vector notation to keep it more general. We want to sample $f(\mathbf{x})$, and to do so we will simply apply our 1D sampling method multiple times. The procedure is outlined in Algorithm A.3.

Algorithm A.2 Gibbs sampling algorithm

Given a set of values non-negative values $\{p(x_i) : x_i\}_{i=1}^{N_X}$ approximating some distribution, draw a samples from it using the following procedure

1. Make the pdf proper

$$\tilde{p}(x_i) = \frac{p(x_i)}{\int_{x_1}^{x_{N_X}} p(x)dx},$$

where $\tilde{p}(x_i)$ so

$$\int_{x_1}^{x_{N_X}} \tilde{p}(x)dx = \int_{x_1}^{x_{N_X}} \frac{p(x)}{\int_{x_1}^{x_{N_X}} p(y)dy}dx$$

Note, we only have a few discrete values for x_i and $p(x_i)$ so we cannot actually compute the integral. So instead, we approximate the integral with the area of trapezoids. That is

$$\int_{x_1}^{x_{N_X}} p(x)dx \approx \frac{1}{2}\sum_{i=2}^{N_X}(x_i - x_{i-1})(p(x_i) + p(x_{i-1})$$

2. Create the CDF

$$\begin{aligned} F(x_i) &= \int_{x_1}^{x_i} \tilde{p}(x)dx \\ &\approx \frac{1}{2}\sum_{j=2}^{i}(x_j - x_{j-1})(\tilde{p}(x_j) + \tilde{p}(x_{j-1}) \end{aligned}$$

This function takes values $[0, 1]$ from $[x_1, x_{N_X}]$.

3. Pull a sample u from $U \sim$ uniform$([0, 1])$.

4. Get sample x from X (the arbitrary random variable) by finding the value x such that $F(x) = u$. In the case that u does not refer to an exact $F(x_i)$, we linearly interpolate from its nearest neighbors. That is, if $F(x_j) < u < F(x_k)$, then form a line $\hat{F}(x)$ between the two points to estimate the CDF. Without loss of generality, assume $x_j < x_k$, so we can use the point-slope equation for a line

$$\begin{aligned} \hat{F}(x) - F(x_j) &= \frac{F(x_k) - F(x_j)}{x_k - x_j}(x - x_j) \\ \hat{F}(x) &= \underbrace{\frac{F(x_k) - F(x_j)}{x_k - x_j}}_{m}(x - x_j) + F(x_j) \end{aligned}$$

Then we choose the sample as x such that $\hat{F}(x) = u$. That is

$$\begin{aligned} u &= m(x - x_j) + F(x_j) \\ u - F(x_j) &= m(x - x_j) \\ u - F(x_j) + mx_j &= mx \\ x &= \frac{1}{m}\left(u - F(x_j) + mx_j\right) \end{aligned}$$

An alternative here would just be to round.

Algorithm A.3 Multidimensional Gibbs sampling

Given some $f(\mathbf{x})$ representing a distribution, draw samples from it with the following:

1. Choose some initial starting value $\mathbf{x}^{(0)}$ which likes within the range of possible values for $\mathbf{x}$. Set $t = 1$.
2. Visit each component of $\mathbf{x}$ and sample
   (a) Get sample $x_1^{(t)}$ from $p(x_1|x_2^{(t-1)}, x_3^{(t-1)}, \ldots, x_M^{(t-1)}) \equiv f(x_1, x_2^{(t-1)}, x_3^{(t-1)}, \ldots, x_M^{(t-1)})$.
      i. Plug in $x_2^{(t-1)}, x_3^{(t-1)}, \ldots, x_M^{(t-1)}$ into $f(\mathbf{x})$ to get a 1D function where x_1 is the variable. Call this function $\tilde{f}(x_1)$.
      ii. Evaluate $\tilde{f}(x_1)$ at sufficiently many points along x_1's effective range (i.e. the values where the function is mostly located) to get $\{x_1(i), \tilde{f}(x_1(i))\}$. For example if $\tilde{f}(x_1) \sim \mathcal{N}(\mu, \sigma^2)$, then we would choose many points on $[\mu - 3\sigma^2, \mu + 3\sigma^2]$ since 99.7% of a normal distribution's density lies on this range.
      iii. Pull a sample from $\{x_1(i), \tilde{f}(x_1(i))\}$ using Algorithm A.2
   (b) Get sample $x_2^{(t)}$ from $p(x_2|x_1^{(t)}, x_3^{(t-1)}, x_4^{(t-1)}, \ldots, x_M^{(t-1)}) \equiv f(x_1^{(t)}, x_2, x_3^{(t-1)}, x_4^{(t-1)}, \ldots, x_M^{(t-1)})$. Here we use the new sample for $x_1^{(t)}$ rather than the previous one.
   (c) Get sample $x_3^{(t)}$ from $p(x_3|x_1^{(t)}, x_2^{(t)}, x_4^{(t-1)}, x_5^{(t-1)}, \ldots, x_M^{(t-1)}) \equiv f(x_1^{(t)}, x_2^{(t)}, x_4^{(t-1)}, x_5^{(t-1)}, \ldots, x_M^{(t-1)})$.
   (d) ...
   (e) Get sample $x_M^{(t)}$ from $p(x_M|x_1^{(t)}, x_2^{(t)}, \ldots, x_{M-1}^{(t)}) \equiv f x_1^{(t)}, x_2^{(t)}, \ldots, x_{M-1}^{(t)}, x_M)$.
3. Increment t, then repeat the sweep (i.e. Step 2).

We stop generating new samples when reach some maximum number of samples or we can set up some convergence criteria. One typical criterion is if the statistics of the first half of the samples match the second half. For example, take all of the samples, split them in half and compute the mean for each. If the means match, then we have enough samples. Another approach would be to start multiple sampling chains at once, and compare the statistics between them.

A.2.3 Correlation vs correlation coefficient

In this appendix we talk about the *correlation coefficient* $\rho(X,Y) = \frac{E[(X-E[X])(Y-E[Y])]}{\sqrt{E[(X-E[X])^2]E[(Y-E[Y])^2]}}$ which is somewhat related *correlation* $r(X,Y) = E[XY]$.[3] In both quantities, the higher the value the more the correlated (i.e. similar) the variables. are. We can say that the process is wide sense stationary (WSS)[4] since we are sampling from the same distribution each time. Thus we can generalize the autocorrelation coefficient to

$$\rho(\tau) = \frac{E[(X_t - \mu_X)(X_{t+\tau} - \mu_X)]}{\sigma_X^2}$$

where μ_X and σ_X^2 denote the mean and variance of the process X respectively. (Similarly we could write the correlation $r(\tau) = E[X_t X_{t+\tau}]$) To compute this for our samples, we use the formula to estimate

$$\hat{\rho}(k) = \frac{1}{s_X^2}\frac{1}{N-|k|}\sum_{i=1}^{N-k}(x_i - m_X)(x_{i+k} - m_X) \tag{A.2.1}$$

where x_i denotes the i-th sample generated, N denotes the total number of samples, and m_X and s_X^2 are the sample mean and variance of all the samples. That is

$$m_X = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$s_X^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - m_X)^2$$

In MATLAB, we can easily compute (A.2.1) using the built in `xcorr` function with the `'unbiased'` flag. `xcorr` actually computes an estimate of $r(k) = E[X_i X_{i+k}]$, so it outputs $\sum_{i=1}^{N-k} x_i x_{i+k}$. With the `'unbiased'` flag, it multiplies the previous result with $\frac{1}{N-|k|}$. This is almost what we want, we just need to divide by s_X^2 and subtract m_X from all the samples first. Thus the script is:

```
[r,lags]=xcorr(X-mean(X),'unbiased');
r=r/cov(X);
```

`lags` contains indexes k, the sample lags, and `r` contains the correlation coefficients.

[3] In statistics, it appears as if correlation is defined by the correlation coefficient. However, in signal processing correlation is defined as the second non-central moment.

[4] Basically, this means that the process does not change over time.

B
Effect of truncation

B.1 Gaussian

Let $Y \sim \mathcal{N}(\mu, \sigma^2)$, so that its pdf is

$$f_Y(y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu)^2}{2\sigma^2}}.$$

Let X be a random variable that is Y but truncated from above by B and below by A. That is, $\Pr\{X < A \cup X > B\} = 0$. As discussed in [35], to find the pdf for X, we just need to normalize by the remaining values. That is, if

$$f_X(x) = \begin{cases} f_Y(x) & A < x < B \\ 0 & \text{otherwise} \end{cases},$$

then

$$\int f_X(x)dx \neq 1,$$

which makes $f_X(x)$ an invalid pdf. This is easy to prove; since

$$\int f_Y(y)dy = 1,$$

then

$$\int_A^B f_Y(y)dy \leq 1.$$

If A, B are finite, then the inequality is strict.

In general, the the pdf of a Y truncated to (A, B) is given by

$$f_X(x) = \begin{cases} \frac{f_Y(x)}{\int_A^B f_Y(x)dx} & A < x < B \\ 0 & \text{otherwise} \end{cases}.$$

That is, the truncated distribution is simply re-normalized to the truncated range. So for our random variable:

$$\begin{aligned} f_X(x) &= \frac{f_Y(x)}{\int_A^B f_Y(s)ds} \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \left(\frac{1}{\sqrt{2\pi\sigma^2}} \int_B^A e^{-\frac{(s-\mu)^2}{2\sigma^2}} ds \right)^{-1} \end{aligned}$$

for $A < x < B$. This does not always result in a nice expression, however in the case of Gaussians, it can worked out [35, 64]. Furthermore, analytical expression for the moments have also been derived [35].

In this paper, we are particularly interested in $\mathcal{N}_0(\mu, \sigma^2)$, that is

$$\begin{aligned} A &= 0 \\ B &= \infty. \end{aligned}$$

The corresponding moments are

$$\begin{aligned} E[X] &= \mu + \sigma\lambda(\alpha) \\ E[(X - E[X])^2] &= \sigma^2(1 - \delta(\alpha)), \end{aligned}$$

where $\alpha = -\frac{\mu}{\sigma}$ and

$$\begin{aligned} \lambda(\alpha) &= \frac{\phi(\alpha)}{Q(\alpha)} \\ \delta(\alpha) &= \lambda(\lambda(\alpha) - \alpha), \end{aligned}$$

where ϕ and Q are the pdf and complementary CDF of the standard normal distribution respectively. In Figure B.1.1, we compare the actual moments of X with μ and σ^2. Specifically, we look at $\mu = 0, 0.1, 0.2, 0.3, \ldots, 10$, and $\sigma^2 = 10^{-6}, \ldots, 100$ (on a log-scale). The difference between the actual moments and μ, σ increases as μ decreases and σ^2 increases. For $\sigma^2 < 1$, the estimation error for μ is very small. Except for very low $\mu \approx 0$, the difference for σ^2 is also very small if σ^2 is roughly less than 10^{-1}. For larger variances, the value of μ_k comes in to play.

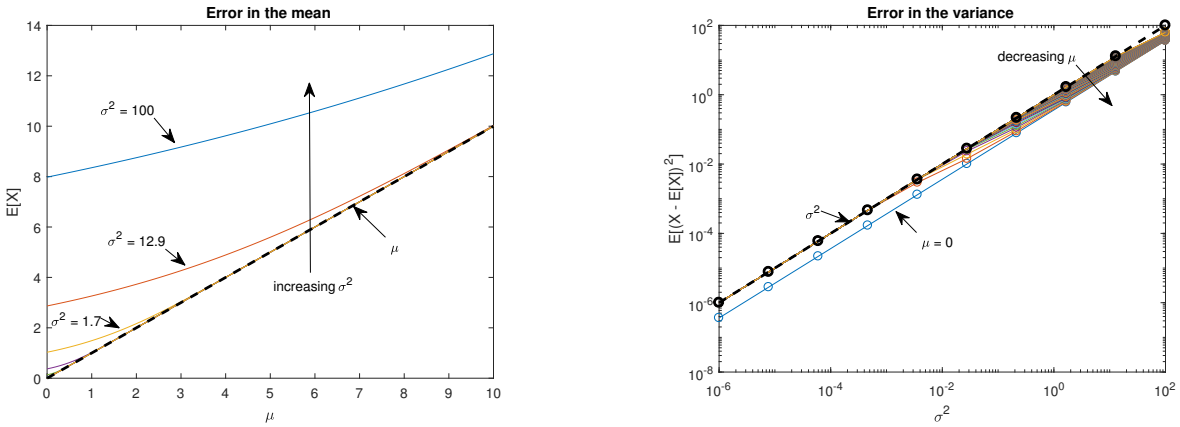


Figure B.1.1: For $X \sim \mathcal{N}_0(\mu, \sigma^2)$, we compute its mean and variance and compare with the input parameters μ and σ^2 (shown as thick black dashed lines).

C
Complex numbers

Let z denote a complex number which can be represented in two forms. First, we can express it in real and imaginary parts

$$z = a + jb,$$

where a is the real part, b is the imaginary part, and $j = \sqrt{-1}$ is the imaginary number. Sometimes we will write $\Re\{z\} = a$ and $\Im\{z\} = b$. Alternatively, we can use polar form

$$z = re^{j\theta},$$

where r is the magnitude (which is also denoted as $|z|$)and θ is the phase. This is best illustrated with an image of the complex plane as shown in Figure C.0.1. We can convert between these two forms easily

$$r = \sqrt{a^2 + b^2}$$

$$\theta = \tan^{-1}\frac{b}{a}$$

$$a = r\cos\theta$$

$$b = r\sin\theta.$$

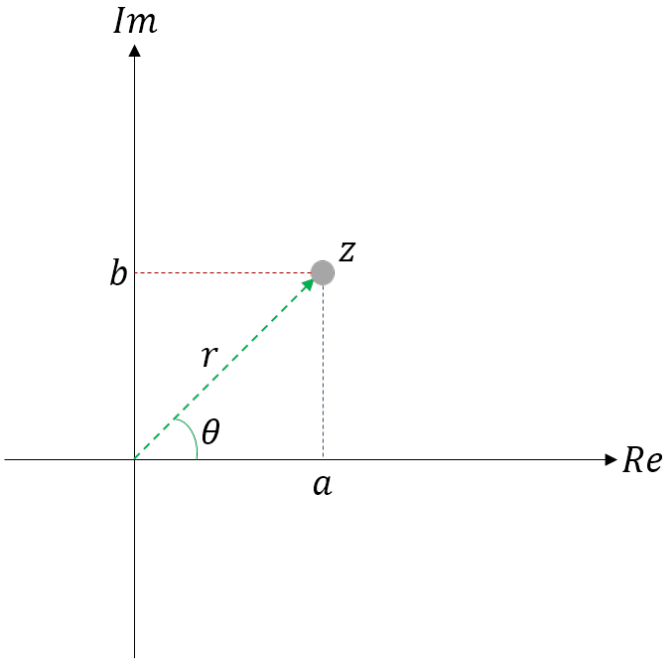


Figure C.0.1: The complex number z in the plane

C.1 Complex conjugate

The complex conjugate of z is defined as

$$\begin{aligned} z^* &\triangleq a - jb \\ &\triangleq re^{-j\theta}. \end{aligned}$$

Fact C.1. *Let z_1 and z_2 be complex numbers and $z_3 = z_1 + z_2$. Then $z_3^* = z_1^* + z_2^*$.*

Proof. Simply compute z_3^*.

$$\begin{aligned} z_3^* &= (z_1 + z_2)^* \\ &= \Re\{z_1 + z_2\} - j\Im\{z_1 + z_2\}. \end{aligned}$$

Let

$$\begin{aligned} z_1 &= a_1 + jb_1 \\ z_2 &= a_2 + jb_2, \end{aligned}$$

where a_i, b_i denote the real and imaginary component of z_i. Then

$$\begin{aligned} z_1 + z_2 &= a_1 + jb_1 + a_2 + jb_2 \\ &= a_1 + a_2 + j(b_1 + b_2). \end{aligned}$$

This means

$$\Re\{z_1 + z_2\} = a_1 + a_2,$$

and

$$\Im\{z_1 + z_2\} = b_1 + b_2.$$

Thus

$$\begin{aligned} z_3^* &= a_1 + a_2 - j(b_1 + b_2) \\ &= \underbrace{a_1 - jb_1}_{z_1^*} + \underbrace{a_2 - jb_2}_{z_2^*}. \end{aligned}$$

□

Fact C.2. *Let z_1, z_2 be complex numbers and $z_3 = z_1 z_2$. Then $z_3^* = z_1^* z_2^*$.*

Proof. In this case it is useful to consider the exponential notation. That is,

$$z_k = r_k e^{j\theta_k}.$$

So we have

$$z_3 = r_1 e^{j\theta_1} r_2 e^{j\theta_2} = r_1 r_2 e^{j(\theta_1+\theta_2)}.$$

And then

$$\begin{aligned} z_3^* &= r_1 r_2 e^{-j(\theta_1+\theta_2)} \\ &= r_1 e^{-j\theta_1} r_2 e^{-j\theta_2} \\ &= z_1^* z_2^*. \end{aligned}$$

□

Fact C.3. *Let z_1, z_2 be complex numbers and $z_3 = \frac{z_1}{z_2}$. Then $z_3^* = \frac{z_1^*}{z_2^*}$.*

Proof. This result follows almost exactly the same argument as the previous. Again, using exponential notation

$$z_3 = \frac{r_1 e^{j\theta_1}}{r_2 e^{j\theta_2}} = \frac{r_1}{r_2} e^{j(\theta_1-\theta_2)}.$$

So

$$\begin{aligned} z_3^* &= \frac{r_1}{r_2} e^{-j(\theta_1-\theta_2)} \\ &= \frac{r_1}{r_2} e^{-j\theta_1+j\theta_2} \\ &= \frac{r_1}{r_2} e^{+(-j\theta_1)} e^{-(-j\theta_2)} \\ &= \frac{r_1 e^{-j\theta_1}}{r_2 e^{-j\theta_2}} = \frac{z_1^*}{z_2^*}. \end{aligned}$$

□

D

Energy functions for thresholds

As a brief summary, the detection algorithms consists of two steps: First, the signal/waveform needs to be converted into some non-negative form to which we can apply the threshold. After the conversion, we set a threshold to and say that any values above it constitutes a detection.

This appendix is dedicated to discussing the functions used to transform the signal in the first step. This information can be found in [44] and many other books, but we felt it was important enough to at least include a summary of the different methods here. There are many different non-negative transformations that we can use, but they all sort of stem from the notion that the transmitted signal is modulated to some carrier frequency. That is

$$s(t) = x(t)\cos(\omega_c t + \phi)$$

where $s(t)$ is the transmitted signal, $x(t)$ is the *message* signal and $\cos$ is the frequency modulation to ω_c. This oscillation does not provide any information with regards to message, so we actually want to ignore it. Here is a short list:

- absolute value: $\hat{x}(t) = \begin{cases} s(t) & s(t) \geq 0 \\ -s(t) & s(t) < 0 \end{cases}$
  - The signal is still messy and the oscillations are still there
  - Not a great choice, but it is a starting point
- Root mean square (rms): $\hat{x}(t) = \sqrt{\frac{1}{T}\int_{t-\frac{T}{2}}^{t+\frac{T}{2}} s^2(t)}$
  - The idea here is that if we average just the noise, we should get 0
- Teager Kaiser (TK): $\hat{x}(t) = \sqrt{\max(0, s^2(t) - s(t-dt)s(t+dt))}$
  - Assume dt is a small quantity such that $x(t+dt) \approx x(t-dt) \approx x(t) \approx x_0$. Then we have the trigonometric identity
    $$\cos(\alpha-\beta)\cos(\alpha+\beta) = \cos^2\alpha - \sin^2\beta,$$

so

$$\begin{aligned} s(t-dt)s(t+dt) &= \underbrace{(x_0\cos(\omega_c t))}_{s(t)}{}^2 - x_0^2\sin^2(\omega_c dt) \\ x_0^2\sin^2(\omega_c dt) &= s^2(t) - s(t-dt)s(t+dt). \end{aligned}$$

And for small x, $\sin(x) \approx x$ since the Taylor series expansion is $\sin(x) = x - \frac{x^3}{6} + \frac{x^5}{120} + \cdots$. Thus

$$x_0^2(\omega_c dt)^2 = s^2(t) - s(t-dt)s(t+dt).$$

Note $\omega_c dt$ is some constant, so we can say that the RHS is proportional to x_0^2, and then just take the square root to get $\hat{x}_0$.

- Typically dt is chosen to be the sample rate

- Envelope: $\hat{x}(t) = \mathcal{F}^{-1}\{W_C\mathcal{F}\{s(t)\}\}$ where $W_C = \begin{cases} 1 & \omega \geq 0 \\ 0 & \omega < 0 \end{cases}$
  - We can go into a lot of math here, and there are other methods to capturing the envelope (e.g. full wave rectifier and low pass filter, Hilbert transform), but throwing away the negative frequency components gives the analytic signal.

We compare these methods on $x(t)$ as a linear increasing function (i.e. $s(t)$ is a chirp); the result is shown in Figure D.0.1. Except for absolute value, they all perform relatively the same. In practice, we typically use TK since the computational burden is small.

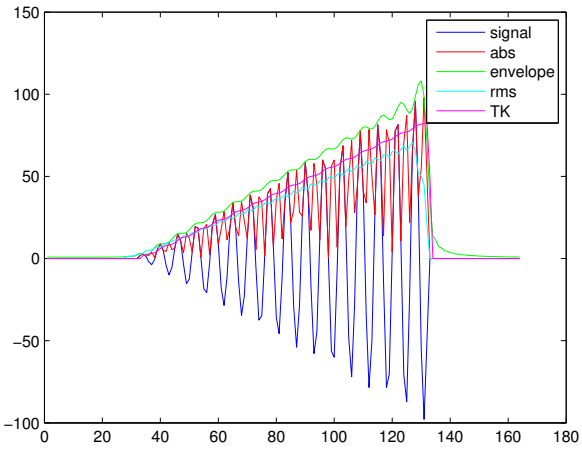


Figure D.0.1: Comparison between the different non-negative transformations on a chirp. For RMS $T = 5$, and in TK dt is the sampling interval.

E

Miscellaneous theorems and proofs

Theorem E.1. *For any distribution $f(x)$ and arbitrary number ε, there exists a value X such that*

$$\int_X^\infty f(x)dx < \epsilon.$$

Proof. For every probability density function $f(x)$ there is an associated cumulative distribution function $F(x) = \int_{-\infty}^{\infty} f(x)dx$ which satisfies the following properties:

1. monotonic increasing
2. right continuous
3. $\lim_{x\to-\infty} F(x) = 0$
4. $\lim_{x\to+\infty} F(x) = 1$

If we express the last property using the formal definition of a limit we for every $\varepsilon > 0$ there exists a value c such that

$$|F(x) - 1| < \varepsilon$$

for every $x > c$. Note, the first, third and forth property implies $F(x) \in [0, 1]$ so $F(x) - 1 \leq 0$ always. Thus

$$|F(x) - 1| = -(F(x) - 1)$$

so

$$\begin{aligned} |F(x) - 1| &< \varepsilon \\ 1 - F(x) &< \varepsilon \\ \int_{-\infty}^{\infty} f(s)ds - \int_{-\infty}^{x} f(s)ds &< \varepsilon \\ \int_{x}^{\infty} f(s)ds &< \varepsilon \end{aligned}$$

Now since $f(x) \geq 0$ (a property of density functions), clearly $f(x) < \varepsilon$ for all $x > c$, what we originally postulated. Realize that this means that for every $\varepsilon > 0$ there exists a value c such that $|f(x) - 0| < \varepsilon$ for all $x > c$, or $\lim_{x\to\infty} f(x) = 0$. □

Theorem E.2. *Let $T > 0$ be an exponentially distributed random variable with parameter λ. Then*

$$\Pr\{T > t + s | T > s\} = \Pr\{T > t\} \tag{E.0.1}$$

for all $t, s \geq 0$. That is, the exponential distribution is memoryless. This is a well known result (e.g. [65]), but we repeat it here for completeness.

Proof. This is easily shown since we know

$$F(t) = \Pr\{T \leq t\} = \begin{cases} 1 - e^{-\lambda t} & t \geq 0 \\ 0 & t < 0 \end{cases}$$

is the cdf for T, and we can denote the survival function (a.k.a. complementary cumulative distribution function) as

$$G(t) = \Pr\{T > t\} = 1 - F(t) = \begin{cases} e^{-\lambda t} & t \geq 0 \\ 0 & t < 0 \end{cases}.$$

Then using the definition of conditional probability

$$\begin{aligned} \Pr\{T > t + s | T > s\} &= \frac{\Pr\{T > t + s \cap T > s\}}{\Pr\{T > s\}} \\ &= \frac{\Pr\{T > t + s\}}{\Pr\{T > s\}} = \frac{e^{-\lambda(t+s)}}{e^{-\lambda s}} = e^{-\lambda t} = \Pr\{T > t\}. \end{aligned}$$

Note, it is easy to see that $T > t + s \cap T > s \equiv T > t + s$ by drawing a simple picture like Figure E.0.1. □

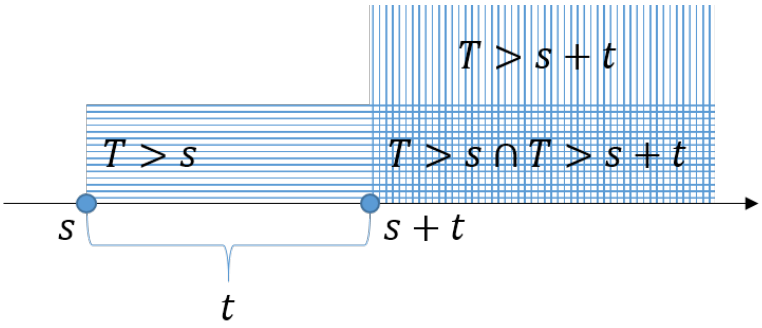


Figure E.0.1: This is an illustration of the events in (E.0.1).

Theorem E.3. *Let T and $t_i \geq 0$ be a iid random variables with a memoryless distribution $\mathcal{T}$. For any $\{t_i\}$ such that*

$$T = \sum_i t_i,$$

it can be said that

$$\mathcal{T}(T) = \mathcal{T}(\bigcap_i t_i).$$

In other words, no matter how the interval T is partitioned into $\{t_i\}$, the joint likelihood of $\{t_i\}$ is just equivalent to the likelihood of T.

Proof. Applying the definition of conditional probability to the LHS of (E.0.1), we have for any $t, s \geq 0$

$$\Pr\{T > t+s | T > s\} = \frac{\Pr\{T > t+s \cap T > s\}}{\Pr\{T > s\}} = \frac{\Pr\{T > t+s\}}{\Pr\{T > s\}}.$$

Again refer to Figure E.0.1 to understand $T > t+s \cap T > s \equiv T > t+s$. If we substitute this back into (E.0.1), then we have

$$\frac{\Pr\{T > t+s\}}{\Pr\{T > s\}} = \Pr\{T > t\} \Rightarrow \Pr\{T > t+s\} = \Pr\{T > t\} \Pr\{T > s\}. \tag{E.0.2}$$

Note this result stems directly from the definition of memoryless, so then that implies that if a random variable exhibits (E.0.2), then the random variable is memoryless. And also that if the random variable is memoryless, then it exhibits this property. (i.e. (E.0.2) is necessary and sufficient) Furthermore (E.0.2), can be easily extended to sums of more than 2 terms (e.g. let $t = u + v$).

So (E.0.2) is essentially the result that we want to show, except that it is for the probability of $T > t+s, T > t, t > s$, we need to show the same result for the likelihoods of $T = t+s, T = t, T = s$. For continuous random variables, the likelihood is given by the pdf. We also know that the exponential distribution is the only continuous distribution that is memoryless [65]. Therefore, let T be exponentially distributed with parameter λ, and we denote the distribution as $\mathcal{T}$. The likelihood of T is then

$$\mathcal{T}(T) = \lambda e^{\lambda T}.$$

Then consider times $t_i \geq 0$ such that

$$T = \sum_i t_i.$$

Assuming that the times t_i are iid with T, then

$$\begin{aligned}
\mathcal{T}(\bigcap_i t_i) &= \prod_i \mathcal{T}(t_i) \\
&= \prod_i \lambda e^{\lambda t_i} \\
&= \lambda e^{\lambda \sum t_i} = \lambda e^{\lambda T} = \mathcal{T}(T).
\end{aligned}$$

□

Bibliography

[1] W. Munk, P. Worcester, and C. Wunsch, *Ocean Acoustic Tomography*, ser. Cambridge Monographs on Mechanics. Cambridge University Press, 1995. 1

[2] D. K. Mellinger, K. M. Stafford, S. E. Moore, R. P. Dziak, and H. Matsumoto, "An overview of fixed passive acoustic observation methods for cetaceans," *Oceanograpy*, vol. 20, no. 4, pp. 36–45, 2007. 1

[3] W. W. Au, *Sonar of Dolphins*. Springer, New York, 1993. 1

[4] W. M. X. Zimmer, M. P. Johnson, P. T. Madsen, and P. L. Tyack, "Echolocation clicks of free-ranging cuvier's beaked whales (ziphius cavirostris)," *The Journal of the Acoustical Society of America*, vol. 117, no. 6, pp. 3919–3927, 2005. [Online]. Available: http://dx.doi.org/10.1121/1.1910225 1

[5] R. G. Wiley, *Electronic Intelligence: The Analysis of Radar Signals Second Edition (Artech House Radar Library (Hardcover))*. Artech House Publishers, 1993. 2, 52, 57, 60, 62

[6] D. J. Milojevic and B. M. Popovic, "Improved algorithm for the deinterleaving of radar pulses," *IEE Proceedings F - Radar and Signal Processing*, vol. 139, no. 1, pp. 98–104, Feb 1992. 2, 3, 52, 61

[7] H. K. Mardia, "New techniques for the deinterleaving of repetitive sequences," *IEE Proceedings F - Radar and Signal Processing*, vol. 136, no. 4, pp. 149–154, Aug 1989. 2, 3, 135

[8] J. B. Moore and V. Krishnamurthy, "Deinterleaving pulse trains using discrete-time stochastic dynamic-linear models," *IEEE Transactions on Signal Processing*, vol. 42, no. 11, pp. 3092–3103, Nov 1994. 2, 3

[9] J. Liu, H. Meng, Y. Liu, and X. Wang, "Deinterleaving pulse trains in unconventional circumstances using multiple hypothesis tracking algorithm," *Signal Processing*, vol. 90, no. 8, pp. 2581 – 2593, 2010, special Section on Processing and Analysis of High-Dimensional Masses of Image and Signal Data. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165168410000897 2, 3, 50

[10] L. de Lathauwer, B. de Moor, and J. Vandewalle, "Fetal electrocardiogram extraction by blind source subspace separation," *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 5, pp. 567–572, May 2000. 2

[11] Z. Chen, "A primer on neural signal processing," *IEEE Circuits and Systems Magazine*, vol. 17, no. 1, pp. 33–50, Firstquarter 2017. 2, 49

[12] C. Shannon, "Communication in the presence of noise," *Proceedings of the IRE,*, vol. 37, pp. 10–21, Jan. 1949. 2

[13] M. Andre and C. Kamminga, "Rhythmic dimension in the echolocation click trains of sperm whales: a possible function of identification and communication," *Journal of the Marine Biological Association of the United Kingdom*, vol. 80, no. 1, pp. 163–169, Feb 2000. 2, 127

[14] R. Bahl, T. Ura, and T. Fukuchi, "Techniques for segregation and classification of several vocalizing sperm whales for auv-based localization applications," in *Oceans 2003. Celebrating the Past ... Teaming Toward the Future (IEEE Cat. No.03CH37492)*, vol. 1, Sept 2003, pp. 457–463 Vol.1. 2, 122, 127

[15] X. C. Halkias and D. P. W. Ellis, "Estimating the number of marine mammals using recordings of clicks from one microphone," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 5, May 2006, pp. V–V. 2, 127

[16] J. Starkhammar, J. Nilsson, M. Amundin, S. A. Kuczaj, M. Almqvist, and H. W. Persson, "Separating overlapping click trains originating from multiple individuals in echolocation recordings," *The Journal of the Acoustical Society of America*, vol. 129, no. 1, pp. 458–466, 2011. [Online]. Available: http://dx.doi.org/10.1121/1.3519404 2, 127

[17] S. Zaugg, M. van der Schaar, L. Houégnigan, and M. André, "Extraction of pulse repetition intervals from sperm whale click trains for ocean acoustic data mining," *The Journal of the Acoustical Society of America*, vol. 133, no. 2, pp. 902–911, 2013. [Online]. Available: http://dx.doi.org/10.1121/1.4773278 2, 127

[18] P. Lepper, N. Dumortier, K. Dudzinski, and S. Datta, "Separation of complex excholocation signal trains from multiple bio-sonar sources," in *Proceedings of the international conference on underwater acoustic measurements: technologies and results*, pp. 913–918. 2

[19] E. T. Küsel, D. K. Mellinger, L. Thomas, T. A. Marques, D. Moretti, and J. Ward, "Cetacean population density estimation from single fixed sensors using passive acoustics," *The Journal of the Acoustical Society of America*, vol. 129, no. 6, pp. 3610–3622, 2011. [Online]. Available: http://dx.doi.org/10.1121/1.3583504 2, 122

[20] P. M. Baggenstoss, "Separation of sperm whale click-trains for multipath rejection," *The Journal of the Acoustical Society of America*, vol. 129, no. 6, pp. 3598–3609, 2011. [Online]. Available: http://link.aip.org/link/?JAS/129/3598/1 2

[21] ——, "The jonker-volgenant algorithm applied to click-train separation," *The Journal of the Acoustical Society of America*, vol. 135, no. 5, pp. 2485–2488, 2014. [Online]. Available: https://doi.org/10.1121/1.4869677 2

[22] O. L. Bot, J. Mars, C. Gervaise, and Y. Simard, "Rhythmic analysis for click train detection and source separation with examples on beluga whales," *Applied Acoustics*, vol. 95, pp. 37 – 49, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0003682X15000481 3, 60, 62, 72, 73, 77, 78, 88, 127

[23] K. Nishiguchi and M. Kobayashi, "Improved algorithm for estimating pulse repetition intervals," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 2, pp. 407–421, Apr 2000. 3, 60, 62, 72, 92, 93

[24] J. Young, A. Høst-Madsen, and E. M. Nosal, "Impulsive source separation with application to sperm whale clicks," in *2013 IEEE Digital Signal Processing and Signal Processing Education Meeting (DSP/SPE)*, Aug 2013, pp. 147–152. 3

[25] E.-M. Nosal, A. Høst-Madsen, and J. Young, "Using inter-click intervals to separate multiple odontocete click trains," *The Journal of the Acoustical Society of America*, vol. 134, no. 5, pp. 3987–3987, 2013. [Online]. Available: http://dx.doi.org/10.1121/1.4830532 3

[26] J. Young, A. Høst-Madsen, and E.-M. Nosal, "Improvements to using inter-click intervals to separate odontocete click trains from multiple animals," *The Journal of the Acoustical Society of America*, vol. 140, no. 4, pp. 3301–3301, 2016. [Online]. Available: http://dx.doi.org/10.1121/1.4970501 3

[27] J. Young, A. Høst-Madsen, and E. Nosal, "Deinterleaving of mixtures of renewal processes," *IEEE Transactions on Signal Processing*, vol. 67, no. 4, pp. 885–898, Feb 2019. 3

[28] P. Comon and C. Jutten, *Handbook of blind source separation: independent component analysis and blind deconvolution*. Elsevier, 2010. 4

[29] G. R. Grimmett and D. R. Stirzaker, *Probability and Random Processes, Third Edition*. Oxford University Press, 2001. 5, 13, 69

[30] R. Barbieri, E. C. Matten, A. A. Alabi, and E. N. Brown, "A point-process model of human heartbeat intervals: new definitions of heart rate and heart rate variability," *American Journal of Physiology - Heart and Circulatory Physiology*, vol. 288, no. 1, pp. H424–H435, 2004. [Online]. Available: http://ajpheart.physiology.org/content/288/1/H424 5

[31] T. Cover and J. Thomas, *Information Theory*. John Wiley, 1991. 5, 45

[32] L. L. Scharf, *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis*. Addison-Wesley, 1990. 6

[33] S. Verdú, *Multiuser Detection*. Cambridge, UK: Cambridge University Press, 1998. 6

[34] J. Proakis, *Digital Communications*. McGraw-Hill, 1995. 6

[35] W. H. Greene, *Econometric Analysis*, 5th ed. Upper Saddle River, NJ: Prentice Hall, 2003. 35, 158, 159

[36] H. Nguyen and B. Levy, "The expectation-maximization viterbi algorithm for blind adaptive channel equalization," *Communications, IEEE Transactions on*, vol. 53, no. 10, pp. 1671 – 1678, oct. 2005. 36, 37

[37] ——, "Blind and semi-blind equalization of cpm signals with the emv algorithm," *Signal Processing, IEEE Transactions on*, vol. 51, no. 10, pp. 2650 – 2664, oct. 2003. 36, 37

[38] G. J. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*. John Wiley & Sons, Inc., 1997. 36, 37, 38

[39] T. Marques, L. Thomas, J. Ward, N. DiMarzio, and P. L. Tyack, "Estimating cetacean population density using fixed passive acoustic sensors: An example with blainville's beaked whales," *Journal of the Acoustical Society of America*, vol. 125, no. 9, pp. 1982–1994, April 2009. 45

[40] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465 – 471, 1978. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0005109878900055 45

[41] P. Grünwald, "A tutorial introduction to the minimum description length principle," *ArXiv*, vol. math.ST/0406077, 2004. 45

[42] M. Wax and T. Kailath, "Determining the number of signals by information theoretic criteria," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '84.*, vol. 9, Mar 1984, pp. 232–235. 45

[43] W. v. Drongelen, *Signal processing for neuroscientists: introduction to the analysisof physiological signals*. Academic Press, 2017. 49

[44] W. M. X. Zimmer, *Passive Acoustic Monitoring of Cetaceans*. Cambridge University Press, 2011. 49, 98, 108, 109, 115, 163

[45] O. Adam, J.-F. Motsch, F. Desharnais, N. DiMarzio, D. Gillespie, and R. Gisiner, "Overview of the 205 workshop on detection and localization of marine mammals using passive acoustics," *Appl. Acoust.*, vol. 67, pp. 1060–1070, 2006. 50, 108, 121, 122, 123, 133

[46] E.-M. Nosal and L. N. Frazer, "Sperm whale three-dimensional track, swim orientation, beam pattern, and click levels observed on bottom-mounted hydrophones," *The Journal of the Acoustical Society of America*, vol. 122, no. 4, pp. 1969–1978, 2007. [Online]. Available: http://link.aip.org/link/?JAS/122/1969/1 50, 121, 122

[47] H. Mardia, "New techniques for the deinterleaving of repetitive sequences," *IEE Proceedings F (Radar and Signal Processing)*, vol. 136, pp. 149–154(5), August 1989. [Online]. Available: http://digital-library.theiet.org/content/journals/10.1049/ip-f-2.1989.0025 52, 61

[48] K. Nishiguchi, "Time-period analysis for pulse train deinterleaving," *Transactions of the Society of Instrument and Control Engineers*, vol. 40, no. 11, pp. 1114–1123, 2004. 72, 73, 77, 78, 88, 127

[49] J. Kiefer, "Sequential minimax search for a maximum," *Proceedings of the American mathematical society*, vol. 4, no. 3, pp. 502–506, 1953. 82

[50] B. Van Der Pol and H. Bremmer, *Operational calculus: based on the two-sided Laplace integral*, 3rd ed. American Mathematical Society, 1987. 84

[51] J. Perkins and I. Coat, "Pulse train deinterleaving via the hough transform," in *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*, vol. iii, Apr 1994, pp. III/197–III/200 vol.3. 94, 95

[52] R. C. Gonzalez and R. E. Woods, *Digital image processing*, 3rd ed. Dorling Kindersley, 2014. 94

[53] W. M. X. Zimmer, P. L. Tyack, M. P. Johnson, and P. T. Madsen, "Three-dimensional beam pattern of regular sperm whale clicks confirms bent-horn hypothesis," *Journal of the Acoustical Society of America*, vol. 117, no. 3, pp. 1473–1485, 2005. 108

[54] P. Madsen, M. Wahlberg, and B. Møhl, "Male sperm whale (physeter macrocephalus) acoustics in a high-latitude habitat: implications for echolocation and communication," *Behavioral Ecology and Sociobiology*, vol. 53, no. 1, pp. 31–41. [Online]. Available: http://dx.doi.org/10.1007/s00265-002-0548-1 116

[55] B. Møhl, M. Wahlberg, P. T. Madsen, A. Heerfordt, and A. Lund, "The monopulsed nature of sperm whale clicks," *The Journal of the Acoustical Society of America*, vol. 114, no. 2, pp. 1143–1154, 2003. [Online]. Available: http://dx.doi.org/10.1121/1.1586258 116

[56] J. Benesty, J. Chen, and Y. Huang, *Microphone Array Signal Processing*, ser. Springer Topics in Signal Processing. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 1. 117

[57] Y. Liu and Q. Zhang, "Improved method for deinterleaving radar signals and estimating pri values," *Sonar Navigation IET Radar*, vol. 12, no. 5, pp. 506–514, 2018. 135

[58] Z. Guoyi, Z. Xuzhou, and C. Shuo, "A pri jitter signal sorting method based on cumulative square sine wave interpolating," in *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*, Dec 2013, pp. 921–924. 135

[59] C. Ren, J. Cao, Y. Fu, and K. E. Barner, "Improved method for pulse sorting based on PRI transform," in *Signal Processing, Sensor/Information Fusion, and Target Recognition XXIII*, I. Kadar, Ed., vol. 9091, International Society for Optics and Photonics. SPIE, 2014, pp. 552 – 558. [Online]. Available: https://doi.org/10.1117/12.2050144 135

[60] A. W. Ata'a and S. N. Abdullah, "Deinterleaving of radar signals and prf identification algorithms," *IET Radar, Sonar Navigation*, vol. 1, no. 5, pp. 340–347, October 2007. 135

[61] W. M. X. Zimmer, P. T. Madsen, V. Teloni, M. P. Johonson, and P. L. Tayack, "Off-axis effects on the multipulse structure of sperm whale usual clicks with implications for sound production." *J. Acoust. Soc. Am.*, vol. 118, no. 5, pp. 3337–45, November 2005. 136

[62] A. Gelman, *Bayesian data analysis*, 3rd ed. Chapman & Hall, 2013. 150

[63] W. K. Hastings, "Monte carlo sampling methods using markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, April 1970. 150

[64] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous univariate distributions*. New York, NY: Wiley, 1994, vol. 1. 159

[65] M. H. DeGroot, *Probability and statistics*, 2nd ed. Reading, Mass.: Addison-Wesley Pub. Co., 1986. 166, 167