TECHNICAL RESEARCH REPORT

Modifications of the Euclidean Algorithm for Isolating Periodicities from a Sparse Set of Noisy Measurements

by S.D. Casey, B.M. Sadler

T.R. 95-105

Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry

Modifications of the Euclidean Algorithm For Isolating Periodicities From a Sparse Set of Noisy Measurements

Stephen D. Casey[1][2]
Department of Mathematics
and Statistics
The American University
4400 Massachusetts Ave., N.W.
Washington, DC 20016–8050
e-mail: scasey@american.edu

Brian M. Sadler
Army Research Laboratory
2800 Powder Mill Road
Adelphi, MD 20783
e-mail: sadler@arl.mil

EDICS: SP 3.6

[1]Research partially supported by Air Force Office of Scientific Research Grant F49620–94–1–0196
[2]Current address: Institute for Systems Research, University of Maryland, College Park, MD 20742–3311

Abstract

Modifications of the Euclidean algorithm are presented for determining the period from a sparse set of noisy measurements. The elements of the set are the noisy occurrence times of a periodic event with (perhaps very many) missing measurements. This problem arises in radar pulse repetition interval (PRI) analysis, in bit synchronization in communications, and other scenarios. The proposed algorithms are computationally straightforward and converge quickly. A robust version is developed that is stable despite the presence of arbitrary outliers. The Euclidean algorithm approach is justified by a theorem which shows that, for a set of randomly chosen positive integers, the probability that they do not all share a common prime factor approaches one quickly as the cardinality of the set increases. In the noise–free case this implies convergence with only ten data samples, independent of the percentage of missing measurements. In the case of noisy data simulation results show, for example, good estimation of the period from one hundred data samples with fifty percent of the measurements missing and twenty five percent of the data samples being arbitrary outliers.

1 Introduction: Statement of the Problem

Our problem begins with a set of measurements of a periodic process. We wish to determine the period of the process from this set, which we shall assume is (perhaps very) sparse and noisy. We assume our data is a finite set of real numbers

$$S = \{s_j\}_{j=1}^{n}, \text{ with } s_j = k_j\tau + \phi + \eta_j, \tag{1}$$

where τ (the period) is a fixed positive real number, the k_j's are non–repeating positive integers, ϕ (the phase) is a real random variable uniformly distributed over the interval $[0, \tau)$, and the η_j's are zero–mean independent identically distributed (iid) error terms. We assume that the η_j's have a symmetric probability density function (pdf), and that $|\eta_j| < \frac{\tau}{2}$ for all j. By assuming that the s_j's are increasing, we can interpret the s_j's as occurrence times with time gaps or jumps determined by the k_j's. (Later we will have occasion to reorder the data for analysis; this does not alter the time–of–occurrence interpretation). For example, letting $k_1 = 1$, $k_2 = 3$, $k_3 = 5$, ..., $k_n = 2n - 1$, corresponds to missing every other occurrence of a periodic event with period τ. The period is the unique maximum value of τ such that the k_j are all integers. Note that for any solution τ that fits the form of S, the numbers $\frac{\tau}{2}, \frac{\tau}{3}, \ldots$ are also possible solutions. Estimates of τ in turn lead to estimates of ϕ and the k_j's.

The problem of finding τ given the set (1) arises in different ways that are often associated with pulse train analysis. These include radar pulse repetition interval (PRI) analysis [31], bit synchronization in communication systems [10], and neuron firing rate analysis. More generally, the problem arises when one observes events that occur in integer multiples of some fixed, but unknown, real number. The problem is to determine that number. For example, in communications the elements of S may be the times of a pseudorandomly occurring change in the carrier frequency, where the change rate is governed by a shift register output. The set S may also arise from noisy zero–crossing times of a sinusoid.

We solve for τ by using a modified Euclidean algorithm to find what is essentially the greatest common divisor of the set S. Subsequent algorithms are modifications of this first procedure. The theoretical basis for the algorithm is given in Sections 2, 3, 4, and the Appendix. The algorithm

is computationally straightforward, requires little input data, and given reasonable data, quickly converges to the value of τ with very high probability. A fundamental result is given by Theorem 3.1, which states that for a set of uniformly distributed positive integers distributed over an arbitrarily large lattice, the probability that they all do not share a common prime factor approaches one quickly as the cardinality of the set increases.

In the noise–free case our algorithm is equivalent to the Euclidean algorithm and converges with very high probability given only $n = 10$ data samples, independent of the number of missing measurements. Simulation examples demonstrate successful estimation of τ for $n = 10$ with 99.99% of the possible measurements missing. In fact, with only 10 data samples, it is possible to have the percentage of missing measurements arbitrarily close to 100%. There is, of course, a cost, in that the number of iterations the algorithm needs to converge increases with the percentage of missing measurements. The algorithm will degrade as n is reduced from 10 (see Table 2), which is consistent with the theory (Theorem 3.1, Proposition 3.1, and Table 1).

In the presence of noise (non–zero η_j's in (1)) and false data (or outliers), there is a tradeoff between the number of data samples, the amount of noise, and the percentage of outliers. The algorithm will perform well given low noise for $n = 10$, but will degrade as noise is increased. However, given more data, it is possible to reduce noise effects and speed up convergence by binning the data, and averaging across bins. Binning can be effectively implemented by using an adaptive threshhold with a gradient operator, allowing convergence in a single iteration in many cases.

The problem of finding τ from S has been considered by Fogel and Gavish [9], who used spectrum analysis of a delta–train function with impulses at the occurrence times. This approach is equivalent to a periodogram–based method for spectral analysis of point processes as developed by Bartlett [1]. General techniques for PRI analysis include dividing the total number of pulses observed by the number of pulse intervals, as well as use of pulse interval histograms [31]. More recently Gray et al. have derived the maximum likelihood estimator of τ for the iid Gaussian noise case [12]. In their work, missing measurements and outliers were not considered.

The more general problem of time series spectrum analysis using zero–crossings has been con-

sidered by a number of authors. Kedem uses zero–crossing counts on an interval for spectrum analysis of random signals [18]. This method does not seem to be conducive to cases of missing measurements. Jones [16] and Parzen [22] developed a multiplicative missing observation model (AM model) for spectrum analysis of stationary processes with periodically missing measurements. The resulting nonparametric estimates are based on suitably modified periodograms. This work was extended by Bloomfield [2] and Scheinok [29] to include missing measurements where the probability of a miss is modeled by a Bernoulli process. Identification of parametric (ARMA) models from time series with missing measurements has also been considered by a number of authors, e.g., see Rosen and Porat [27] and references therein. Recently Dandawate [6] and Giannakis and Zhou [11] have proposed using cyclic statistics of time series with periodically missing measurements. Generally speaking, the above referenced models for missing measurements assume access to uniformly spaced amplitude samples, and are therefore not directly applicable in our setup.

Kay and Sudhaker proposed the use of the occurrence time of zero–crossings to obtain DFT coefficients [17]. This approach is related to a class of pulse time modulation techniques in which sample amplitudes are suitably transformed to zero–crossing interval widths [8]. This method requires addition of an auxiliary signal before detection of the zero–crossings and is therefore not applicable here.

The paper is organized as follows. Section 2 gives the development of a modified Euclidean algorithm for finding τ that is robust to noise. Given this estimate, we then discuss estimating ϕ and the k_j's. Section 3 is a discussion of the result that for a set of randomly chosen positive integers, the probability that they do not all share a common prime factor approaches one quickly as the cardinality of the set increases. We then propose, in Section 4, further modifications of the algorithm which are effective given increased noise and outliers. Section 5 contains simulation results and discussion of performance for the various proposed algorithms. Lengthy proofs are presented in the Appendix.

2 A Modified Euclidean Algorithm for Finding τ

Given the set S as in (1), we modify the Euclidean algorithm to develop a procedure for finding τ. This involves some basic number theory. References for this material include Hardy and Wright [13], Ireland and Rosen [14], Knuth [19], Leveque [21], Rosen [26], and Schroeder [30].

The Euclidean algorithm is a division process for the set of integers $\mathbf{Z}$. The algorithm is based on the property that, given two positive integers a and b, $a > b$, there exists two positive integers q and r such that

$$a = q \cdot b + r \,,\, 0 \leq r < b \,.$$

If $r = 0$, we say that b divides a, and denote this by $b|a$. This property of the set of integers, combined with the fact that if $a, b \in \mathbf{Z} \setminus \{0\}$ then $a \cdot b \neq 0$ ($\mathbf{Z}$ has no zero divisors), make $\mathbf{Z}$ a unique factorization domain. Thus in $\mathbf{Z}$ every non–zero element may be written as the product of powers of irreducible integers, or primes. The Euclidean algorithm also holds in more general algebraic structures called Euclidean domains.

The Euclidean algorithm yields the greatest common divisor of two (or more) elements of $\mathbf{Z}$. The *greatest common divisor* of two integers a and b, denoted by $\gcd(a, b)$, is the product of the powers of all prime factors p that divide both a and b. We may represent the algorithm applied to a, b, $a > b$, as follows:

$$\begin{aligned}
a &= b \cdot q_1 + r_1 &&: \quad 0 < r_1 < b \\
b &= r_1 \cdot q_2 + r_2 &&: \quad 0 < r_2 < r_1 \\
&\vdots &&\quad \vdots \\
r_{k-2} &= r_{k-1} \cdot q_k + r_k &&: \quad 0 < r_k < r_{k-1} \\
r_{k-1} &= r_k \cdot q_k \,.
\end{aligned}$$

The procedure terminates when $r_{k+1} = 0$. This gives $\gcd(a, b) = r_k$.

This procedure can be extended to work on S. The symbol $\gcd(k_1, \ldots, k_n)$ is the greatest common divisor of the set $\{k_j\}$, i.e., the product of the powers of all prime factors p that divide each k_j. Note that this is not the pairwise gcd of the set $\{k_j\}$. If $\gcd(k_1, \ldots, k_n) = 1$, the set

$\{k_j\}$ is called *mutually relatively prime.* If, however, $\gcd(k_i, k_j) = 1$ for all $i \neq j$, the set $\{k_j\}$ is called *pairwise relatively prime.* If a set is pairwise relatively prime, it is mutually relatively prime. However, the converse is not true (for example, consider the set $\{35, 21, 15\}$). The computation of the gcd of a set of more than two integers uses the following proposition. There is also a natural extension of the gcd to multiples of a fixed $\tau > 0$.

Proposition 2.1

$$(i.) \quad \gcd(k_1\tau, \ldots, k_n\tau) = \tau \gcd(k_1, \ldots, k_n)\,,$$

$$(ii.) \quad \gcd(k_1, \ldots, k_n) = \gcd(k_1, \ldots, k_{n-2}, (\gcd(k_{n-1}, k_n)))\,.$$

Proof : See Leveque [21], page 16. □

The standard Euclidean algorithm, as shown above, involves repeated division. In our problem, we are dealing with numbers that are essentially "noisy integers." Remainder terms could be noise, and thus could be non–zero numbers arbitrarily close to zero. Subsequent iterations in the procedure may involve dividing by such small values, which would result in arbitrarily large numbers. Thus, the standard algorithm is unstable under perturbation by noise. However, the algorithm may be changed so that the process of subtraction replaces division by making use of the following proposition. So that $(k_j - k_{j+1}) \in \mathbf{N}$, we assume that the k_j's are sorted in descending order.

Proposition 2.2

$$\gcd(k_1, \ldots, k_n) = \gcd((k_1 - k_2), (k_2 - k_3), \ldots, (k_{n-1} - k_n), k_n)\,.$$

Proof : See the Appendix. □

We will also need the following related result.

Proposition 2.3

$$\gcd((k_1 - k_2), (k_2 - k_3), \ldots, (k_{n-1} - k_n)) = \gcd((k_1 - k_n), (k_2 - k_n), \ldots, (k_{n-1} - k_n))\,.$$

Proof : See the Appendix. □

We are now ready to develop our modified algorithm. Here, and throughout the rest of the paper, we will assume the s_j's are sorted in *descending* order, i.e., $s_1 \geq s_2 \geq \cdots \geq s_n$. This allows a more straightforward visualization of our algorithm. We form a new set by subtracting adjacent pairs of these numbers, given by $s_j - s_{j+1}$. After this first operation, the phase information has been subtracted out, and the resulting set has the simpler form

$$S' = \{s'_j\}_{j=1}^{n-1}, \text{ with } s'_j = K_j\tau + \eta'_j,$$

where $K_j = k_j - k_{j+1}$ and $\eta'_j = \eta_j - \eta_{j+1}$. In subsequent iterations of the algorithm, the data will maintain this same general form. Because of the η_j perturbations we establish a threshold η_0 and, after the subtraction of adjacent pairs, we declare all numbers in the interval $[0, \eta_0]$ to be zero and eliminate them from the set. Choice of η_0 is dictated by the distribution of the η_j's, with $0 < \eta_0 < \frac{\tau}{2}$ (see Section 4). We then append zero, sort, subtract adjacent pairs, and threshold. By appending zero, we adjoin the previous non–zero minimum to the set. The algorithm is continued by iterating this process of appending zero, sorting, subtracting, and eliminating the elements in $[0, \eta_0]$. It terminates when all but one of the elements are in $[0, \eta_0]$, i.e.,"equal to zero." By the propositions above, this element is equal to $\gcd(K_1, \ldots, K_{n-1}) \cdot \tau \pm$ error term.

We will see in the next section that $\gcd(K_1, \ldots, K_{n-1}) \longrightarrow 1$ with probability 1 as $n \longrightarrow \infty$. Moreover, we will see that this convergence is very fast, which shows that the proposed modified Euclidean algorithm yields τ with a high probability for small ($n \approx 10$) to moderate ($n \approx 100$) values of n, depending on the distribution of the η_j's and the percentage of outliers.

The algorithm for finding τ given the set S is summarized as follows. Again, we assume that the original data set is in descending order, i.e., $s_j \geq s_{j+1}$.

Modified Euclidean Algorithm

1.) After the first iteration, append zero.

2.) Form the new set with elements $s_j - s_{j+1}$.

3.) Sort in descending order.

4.) Eliminate elements in $[0, \eta_0]$ from end of the set.

5.) Algorithm is done if left with a single element. Declare $\widehat{\tau} = s_1$. If not done, go to **(1.)**.

Remarks :

1. It is easily possible to generate low noise or even noise–free data sets such that the modified Euclidean algorithm does not converge to τ, even for an arbitrarily large number of data points. For example, consider $k_1 = 3, k_2 = 5, k_3 = 7, \ldots$ with true underlying period $\tau = 1$. To eliminate the phase, we do not save the first minimum. For this data, the algorithm will give $\tau = 2$. However, for even 10 data points, the theory developed in the following section shows that such data sets are extremely unlikely to occur. Another way of saying this is that if a large set of low noise or noise–free data leads the algorithm to an incorrect solution, then the corresponding k_j's exhibit a consistent pattern.

2. Given an estimate $\widehat{\tau}$ of τ, it is then possible to estimate ϕ and the k_j's. We give a brief discussion of these estimates. Fogel and Gavish [9, 10] have addressed the problem of estimating ϕ, and have produced an estimator $\widehat{\phi}$ for ϕ, given by

$$\widehat{\phi} = \frac{\widehat{\tau}}{2\pi} \arg \left\{ \sum_{j=1}^{n} \exp(2\pi i \frac{s_j}{\widehat{\tau}}) \right\} . \tag{2}$$

For a good estimate $\widehat{\tau}$ of τ, $\exp(2\pi i \frac{k_j \tau}{\widehat{\tau}}) \approx 1$. For $\eta_j \ll \tau \approx \widehat{\tau}$, $\eta_j / \widehat{\tau} \ll 1$, and so $\exp(2\pi i \eta_j / \widehat{\tau}) \approx \exp(0) = 1$. Therefore,

$$\begin{aligned}
& \frac{\widehat{\tau}}{2\pi} \arg \left\{ \sum_{j=1}^{n} \exp(2\pi i \frac{s_j}{\widehat{\tau}}) \right\} \\
= \; & \frac{\widehat{\tau}}{2\pi} \arg \left\{ \sum_{j=1}^{n} \exp(2\pi i \frac{k_j \tau}{\widehat{\tau}}) \exp(2\pi i \frac{\eta_j}{\widehat{\tau}}) \exp(2\pi i \frac{\phi}{\widehat{\tau}}) \right\} \\
\approx \; & \frac{\widehat{\tau}}{2\pi} \arg \left\{ \sum_{j=1}^{n} \exp(2\pi i \frac{\phi}{\widehat{\tau}}) \right\} \\
= \; & \frac{\widehat{\tau}}{2\pi} \arg \left\{ n \cdot \exp(2\pi i \frac{\phi}{\widehat{\tau}}) \right\}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{\widehat{\tau}}{2\pi}\left(\arg\{n\} + \arg\left\{\exp(2\pi i\frac{\phi}{\widehat{\tau}})\right\}\right) \\
&= \frac{\widehat{\tau}}{2\pi}\arg\left\{\exp(2\pi i\frac{\phi}{\widehat{\tau}})\right\} \\
&= \frac{\widehat{\tau}}{2\pi}\frac{2\pi\phi}{\widehat{\tau}} = \phi
\end{aligned}
$$

3. We present two methods of getting an estimate on the set of k_j's. Given a good estimate $\widehat{\phi}$, the first is to form the set

$$\sigma = \{k_j\tau + \phi + \eta_j - \widehat{\phi}\}_{j=1}^{n}.$$

Given the estimate $\widehat{\tau}$, estimate k_j by

$$\widehat{k}_j = \text{ round } \left(\frac{k_j\tau + \phi + \eta_j - \widehat{\phi}}{\widehat{\tau}}\right), \tag{3}$$

where round$(\cdot)$ denotes rounding to the nearest integer.

We could also work with the set

$$\sigma' = \{K_j\tau + \eta'_j\}_{j=1}^{n-1} \cup \{k_n\tau + \phi + \eta_n - \widehat{\phi}\},$$

where $K_j = k_j - k_{j+1}$ and $\eta'_j = \eta_j - \eta_{j+1}$. Given the estimate $\widehat{\tau}$, estimate k_n by

$$\widehat{k}_n = \text{ round } (\frac{k_n\tau + \phi + \eta_n - \widehat{\phi}}{\widehat{\tau}})$$

and K_j by

$$\widehat{K_j} = \text{ round } (\frac{K_j\tau + \eta'_j}{\widehat{\tau}}).$$

Then, $\widehat{k}_{n-1} = \widehat{K}_{n-1} + \widehat{k}_n$, $\widehat{k}_{n-2} = \widehat{K}_{n-2} + \widehat{k}_{n-1}$, and so on.

3 Relatively Prime Numbers and the Zeta Function

In this section, we present Theorem 3.1, which says that $\gcd(k_1, \ldots, k_n) \longrightarrow 1$ with probability 1 as $n \longrightarrow \infty$. Moreover, this convergence is very quick. We use this to show that

$$\gcd(K_1\tau, \ldots, K_{n-1}\tau) \longrightarrow \tau, \tag{4}$$

with probability 1 as $n \longrightarrow \infty$. We show that the proposed modified Euclidean algorithm yields τ with a high probability for small ($n \approx 10$) to moderate ($n \approx 100$) values of n, depending on the distribution of the η_j's and the percentage of outliers. In the noise–free case, the theory tells us that the algorithm will almost certainly yield τ given only 10 data samples.

The proof is relatively long, and is presented in the Appendix. The development of the theorem involves techniques and results from analytic number theory. Two functions, Riemann's Zeta function $\zeta(n)$ and the Möbius function $\mu(m)$, play key roles. The result is classical for the case $n = 2$, and was proven by Dirichlet in 1849 (see Knuth [19], pp. 324, 337, 595, and Schroeder [30], pp. 48–50). An elegant formulation for $n = 2$ is given in Hardy and Wright [13].

Riemann's Zeta function is defined on the complex half space $\{z \in \mathbf{C} : \Re(z) > 1\}$ by

$$\zeta(z) = \sum_{n=1}^{\infty} n^{-z} . \tag{5}$$

Euler demonstrated the connection of ζ with number theory by showing, in 1736, that

$$\zeta(z) = \prod_{j=1}^{\infty} \frac{1}{1 - (p_j)^{-z}} , \Re(z) > 1 , \tag{6}$$

where $\mathbf{P} = \{p_1, p_2, p_3, \ldots\} = \{2, 3, 5, \ldots\}$ is the set of all prime numbers (see Conway [5]). In the following $P\{\cdot\}$ denotes probability.

Theorem 3.1 *Given n ($n \geq 2$) randomly chosen positive integers $\{k_1, \ldots, k_n\}$,*

$$P\{\gcd(k_1, \ldots, k_n) = 1\} = [\zeta(n)]^{-1} .$$

The proof of Theorem 3.1 follows directly from the following theorem. We let $\mathrm{card}\{\cdot\}$ denote cardinality of the set $\{\cdot\}$, and let $\{1, \ldots, \ell\}^n$ denote the sublattice of positive integers in $\mathbf{R}^n$ with coordinates c such that $1 \leq c \leq \ell$. Therefore, $N_n(\ell) = \mathrm{card}\{(k_1, \ldots, k_n) \in \{1, \ldots, \ell\}^n : \gcd(k_1, \ldots, k_n) = 1\}$ is the number of relatively prime elements in $\{1, \ldots, \ell\}^n$.

Theorem 3.2 *Let*

$$N_n(\ell) = \mathrm{card}\{(k_1, \ldots, k_n) \in \{1, \ldots, \ell\}^n : \gcd(k_1, \ldots, k_n) = 1\} ,$$

For $n \geq 2$, we have that

$$\lim_{\ell \to \infty} \frac{N_n(\ell)}{\ell^n} = [\zeta(n)]^{-1} .$$

Proof : See the Appendix. □

Our original data set contains phase information. We eliminate this by the subtraction of adjacent elements in the set. But then, rather than working with $\{k_1, \ldots, k_n\}$, we are working with $\{(k_1 - k_2), \ldots, (k_{n-1} - k_n)\}$. Therefore, we need the following.

Corollary 3.1 *Let* $\{k_1, \ldots, k_n\}$ *be* n $(n \geq 3)$ *randomly chosen positive integers, with* $k_j > k_{j+1}$, *and let* $K_j = k_j - k_{j+1}$ *for* $j = 1, \ldots, n-1$. *Then*

$$P\{\gcd(K_1, \ldots, K_{n-1}) = 1\} = [\zeta(n-1)]^{-1} .$$

Proof : See the Appendix. □

Various techniques, e.g., Cauchy residue theory, allow us to get a closed form solution of $\zeta(n)$ for the even natural numbers. This is (see Ireland and Rosen [14] and Conway [5])

$$2\zeta(2k) = (-1)^{k+1} \frac{(2\pi)^{2k}}{(2k)!} (2k^{\text{th}} \text{ Bernoulli Number}) . \tag{7}$$

("The result is due to Euler and constitutes one of his most remarkable computations." Ireland and Rosen [14], page 231.) The values $\zeta(2k+1)$ can be estimated numerically (see the CRC tables [3]). Some representative values are shown in Table 1. We can see from these values that $[\zeta(n)]^{-1} \longrightarrow 1$ quickly as n increases. In fact, we can show that this rate of convergence is at least exponential.

Proposition 3.1 *Let* $\omega \in (1, \infty)$. *Then*

$$\lim_{\omega \to \infty} [\zeta(\omega)]^{-1} = 1 , \tag{8}$$

converging to 1 from below faster than $1/(1 - 2^{1-\omega})$.

Proof : See the Appendix. □

Combining Corollary 3.1 with Propositions 2.1 – 3.1 gives the theoretical underpinnings of the algorithm.

Theorem 3.3 *Let* $\{k_1, \ldots, k_n\}$ *be* n $(n \geq 3)$ *randomly chosen positive integers, with* $k_j > k_{j+1}$, *and let* $K_j = k_j - k_{j+1}$ *for* $j = 1, \ldots, n-1$. *Then*

$$\gcd(K_1\tau, \ldots, K_{n-1}\tau) \longrightarrow \tau\,,$$

with probability 1 as $n \longrightarrow \infty$.

We close this section with a heuristic argument for the validity of Theorem 3.1. The argument, although not rigorous, does provide insight into the result. In this discussion, we have not carefully defined what it means for the integers to be chosen at random, and we do not present the estimates needed to justify convergence. These are presented in the proof of Theorem 3.2 in the Appendix.

Given randomly distributed positive integers, by the Law of Large Numbers, about $1/2$ of them are even, $1/3$ of them are multiples of three, and $1/p$ are a multiple of some prime p. Thus, given n independently chosen positive integers,

$$\begin{aligned} P\{p|k_1, p|k_2, \ldots, \text{and } p|k_n\} &= \\ P\{p|k_1\} \cdot P\{p|k_2\} \cdot \ldots \cdot P\{p|k_n\} &= \\ 1/(p) \cdot 1/(p) \cdot \ldots \cdot 1/(p) &= \\ 1/(p)^n\,. & \end{aligned}$$

Therefore,

$$P\{p \not| k_1, p \not| k_2, \ldots, \text{or } p \not| k_n\} = 1 - 1/(p)^n\,. \tag{9}$$

By the Fundamental Theorem of Arithmetic, every integer has a unique representation as a product of primes. Combining that theorem with the definition of gcd, we get

$$P\{\gcd(k_1, \ldots, k_n) = 1\} = \prod_{j=1}^{\infty} 1 - 1/(p_j)^n. \tag{10}$$

where p_j is the j^{th} prime. But, by Euler's formula,

$$\zeta(z) = \prod_{j=1}^{\infty} \frac{1}{1 - (p_j)^{-z}}\,,\ \Re(z) > 1\,. \tag{11}$$

Therefore,

$$P\{\gcd(k_1, \ldots, k_n) = 1\} = 1/(\zeta(n))\,. \tag{12}$$

This completes our heuristic argument.

4 Further Modifications

In this section we consider the effects of noise and outliers on the modified Euclidean algorithm described in Section 2. The modified algorithm replaces division, as required in the standard Euclidean algorithm, with repeated subtraction in order to gain stability with respect to noise. Error analysis of this approach is complicated by the facts that the algorithm is iterative and that the algorithm sorts the data. Therefore, this analysis involves order statistics.

Suppose the pdf of the η_j's is given by $f_\eta(\eta)$, and consider the set of differences obtained in the first iteration, given by

$$y_j = s_j - s_{j+1} = (k_j - k_{j+1})\tau + (\eta_j - \eta_{j+1}). \tag{13}$$

Invoking the zero–mean iid assumption on the η_j's, the pdf of $(\eta_j - \eta_{j+1})$ is given by the convolution $f_\eta(\eta) * f_\eta(\eta)$. So, for example, if $f_\eta(\eta) \sim \mathcal{U}[-\frac{\Delta}{2}, \frac{\Delta}{2}]$ (η is uniformly distributed with parameter Δ) then $f_{y_j}(y) = \text{tri}[y-(k_j-k_{j+1})\tau]$, the triangle function centered at $(k_j-k_{j+1})\tau$. Two points can now be made. First, after the first iteration, the differencing operation has removed the independence of the error terms. Second, the ordering operation makes the nature of the dependence in subsequent iterations difficult to determine. Analysis of order statistics very often rests on an iid assumption, e.g., see Sarhan and Greenberg [28] and Reiss [25]. Without the iid assumption, this analysis leads into many open questions (see Reiss [25]).

In general, beyond the first iteration the pdf of the subsequent error terms becomes asymmetric, even when starting with iid η_j's with symmetric pdf $f_\eta(\eta)$. This occurs due to the reordering before differencing at each iteration, and because after the first iteration the errors are no longer iid. The result is that using the modified Euclidean algorithm can lead to negatively biased estimates of τ after the first iteration due to the skewness of the pdf of the errors. As we will see this can be corrected for by averaging.

In order to illustrate the behavior of the algorithm consider the following example. Let the set S of equation (1) be generated as follows. Let $\tau = 1$, $n = 100$ data samples, the jumps in the k_j's be randomly selected from a discrete uniform distribution on the interval $[1, 10]$, and the noise be iid and uniformly distributed as $f_\eta(\eta) \sim \mathcal{U}[-0.1, 0.1]$. A data set S was generated according to

these parameters and used as input to our algorithm. Consider the results after one iteration, in which the data has been differenced and sorted into descending order, as plotted in Figure 1. The data are clustered into "steps" around integer multiples of $\tau = 1$, as we expect from (13). That the steps are all of the same approximate length is due to the uniform distribution in the jumps of the k_j's in the original data set S. Other distributions will result in different proportions. From (13) and the assumptions on the noise we know that the data has a mean that is an integer multiple of τ given by $(k_j - k_{j+1})\tau$, with noise symmetrically distributed around this mean. This suggests isolating each step and averaging the data within each step to reduce noise effects.

A straightforward method for clustering the data is to employ a gradient operator to determine when a step has occurred. After the first iteration (as in Figure 1) the gradient is estimated, with large gradient values indicating a step or "edge" in the data. We have employed a simple estimator by convolving with an impulse response given by $[-1, 0, 1]$. This operator is well known in signal and image processing, e.g., see Jain [15]. A data–adaptive gradient threshold g_0 is selected as 10% of the maximum gradient value, and data points above this threshold are assumed to correspond to the step edges. After the steps have been isolated the step heights are easily found, and the minimum step height, call it $\tilde{\tau}$, is taken as a coarse estimate of τ. Referring to Figure 1, all of the step heights are approximately equal to τ, again due to the original distribution of the jumps in the k_j's used in generating S. We then use $\tilde{\tau}$ to set two thresholds. The first is $\eta_0 = 0.35\tilde{\tau}$, used to define the neighborhood of zero in which data will be eliminated during each iteration (see Section 2). The second we take to be $y_0 = 0.6\tilde{\tau}$, used to segment the steps at each iteration. The segmentation proceeds by searching for jumps in height greater than y_0, and averaging over each segment. The choices of 0.35 and 0.6 are based on extensive simulation experience (see Section 5), and can be more rigorously justified in specific cases. However, performance is reasonably robust to changes in these weights under the various scenarios considered. The averaging produces significant data reduction, and therefore greatly increases the speed of convergence. The gradient operator is applied only as part of the first iteration, the data reduces rapidly with each iteration and precludes use of the gradient operator except as part of the first iteration.

We summarize the foregoing in the following algorithm statement. Again we assume the data

is initially sorted in descending order. Recall that appending zero in the first step appends the previous minimum.

Modified Euclidean Algorithm (With Averaging)

1.) After the first iteration, append zero.

2.) Form the new set with elements $s_j - s_{j+1}$.

3.) Sort in descending order.

4.) On the first iteration, apply gradient and obtain $\tilde{\tau}$, yielding $\eta_0 \cong 0.35\tilde{\tau}$ and $y_0 \cong 0.6\tilde{\tau}$ (see text).

5.) Average the data over each step, with steps determined by jumps of height y_0.

6.) Eliminate elements in $[0, \eta_0]$ from end of the set.

7.) Algorithm is done if left with a single element. Declare $\widehat{\tau} = s_1$. If not done, go to **(1.)**.

If the data is such that, after the first iteration, there is a relatively large cluster around the step nearest to zero, then we can readily estimate τ by finding this step, averaging only over these data points, and declaring this to be $\widehat{\tau}$. This is the rightmost or lowest step after the first iteration (see Figure 1). Under our assumptions this mean is an unbiased estimate of τ. Accurate estimation of τ from a single iteration assumes that n is large enough, and the span of the original data set S small enough, to yield sufficient data in the neighborhood of τ. This is a function of the distribution of the k_j's.

In our example above the missing observations were modeled by taking the jumps in the k_j's as uniformly distributed on the discrete interval $[1, M]$, with $M = 10$. Thus, for large n, after the first iteration the data will cluster in M steps with an expected value of n/M samples in each step, as in Figure 1. As another model for missing observations we can employ an iid Bernoulli process to determine if an observation is missing or not, where

$$\begin{aligned} P(\text{missing observation}) &= \lambda \\ P(\text{observation occurring}) &= 1 - \lambda. \end{aligned} \tag{14}$$

For example, with $\lambda = 0.6$, we expect 60% of the observations to be missing. Given an element of S, $s_j = k_j\tau + \eta_j + \phi$, then the probability that $s_{j+1} = k_{j+1}\tau + \eta_{j+1} + \phi = (k_j + 1)\tau + \eta_{j+1} + \phi$ is an element of the set S is given by $1 - \lambda$. It follows that, after the first iteration, we expect $(1 - \lambda)(n - 1)$ data samples to be in the lowest step clustered around the true value of τ. Thus, if $n = 101$ and $\lambda = 0.6$, then after the first iteration we expect $(1 - 0.6)100 = 40$ data samples to be clustered in the lowest step around τ; the average over these 40 data samples may then be taken as an estimate of τ.

The single iteration algorithm is a simple modification of the preceding multi–iteration version. After the (first) difference and sort operations the gradient is applied and the lowest step isolated. The average over this step is then taken as our estimate of τ. We summarize the single iteration form of the algorithm as follows.

Modified Euclidean Algorithm (Single Iteration)

1.) Given the set S, form the new set with elements $s_j - s_{j+1}$.

2.) Sort the new set in descending order.

3.) Apply gradient operator and obtain $\tilde{\tau}$, with $y_0 \cong 0.6\tilde{\tau}$.

4.) Obtain $\widehat{\tau}$ by averaging the data over the lowest step, isolating this step based on the lowest two jumps of height y_0.

Next we consider the effect of arbitrary outliers. These are, in general, quite harmful to the estimation of the gcd. This is easily seen in the noise–free case because the gcd of a contaminated set may be arbitrarily different from the gcd of the uncontaminated set. Testing of the algorithms presented thus far shows sensitivity to the presence of even a single outlier. This is because the outliers will not necessarily fall into the step–like clusters we expect.

A more robust version of the algorithm can be obtained with the introduction of a step–width threshold, x_0. Let us concentrate on the single iteration approach. After application of the gradient operator as before, we avoid false edges and clusters occurring below the true lowest step by requiring the step–width (the number of data samples in a particular step) to be greater than x_0.

The choice of x_0 is dictated by two considerations, the distribution of the outliers and the expected number of data samples in the lowest step after the first iteration. This represents a tradeoff. For example, consider again the Bernoulli model for missing observations. Suppose $\lambda = 0.25$ and $n = 101$. We therefore expect 75 data samples in the lowest step, allowing us to choose $x_0 = 50$, say. The single iteration algorithm employing x_0 will therefore search for the lowest step in the data that contains at least $x_0 = 50$ data samples. Depending on the distribution and number of outliers it may be extremely unlikely that a step will occur that simultaneously has a mean less than τ and has more than $x_0 = 50$ data values. The drawback to employing x_0 is that setting its value requires some *a priori* knowledge or guesswork and is not easily set adaptively. The advantage is that for reasonable scenarios its use makes the algorithm very robust to the presence of even large percentages of outliers, as illustrated in the simulation results of the next section. This robust single iteration algorithm is summarized as follows.

Robust Single Iteration Algorithm

1.) Given the set S, form the new set with elements $s_j - s_{j+1}$.

2.) Sort the new set in descending order.

3.) Apply gradient operator and obtain $\tilde{\tau}$, with $y_0 \cong 0.6\tilde{\tau}$.

4.) Obtain $\widehat{\tau}$ by averaging the data over the lowest step, isolating the step based on the lowest two jumps of height y_0 with step–width greater than x_0.

Further discussion of the various algorithms and their performance is presented at the end of the next section.

Remarks :

1. The computational load of the robust single iteration algorithm can be approximated as follows. The differencing operation of **1.)** requires $n-1$ additions, a fast sort can be accomplished in about $n \log n$ operations [20], and convolution with the gradient operator $[-1, 0, 1]$ can be accomplished with n additions and requires no multiplies. Finally, **4.)** requires a few

differences and comparisons. Thus, the total computational load is roughly $2n$ additions plus $n \log n$ operations for the sort.

2. Fogel and Gavish have proposed the following estimate for τ [9],

$$\widehat{\tau} = \frac{1}{\arg\max_f \left| \sum_{j=1}^{n} e^{2\pi i f s_j} \right|}. \tag{15}$$

The denominator arises from spectrum analysis of a delta-train function with unit impulses at the occurrence times (see also Bartlett [1]). This algorithm has a simple program statement and requires no tuning parameters other than choice of f. In our tests this algorithm performed well given that the percentage of missing observations was not too large and given some *a priori* knowledge of a range of f over which to search. The algorithm requires fine steps in f when searching due to the convergence properties of the sum of exponentials. Without this fine search it is possible to miss $f = 1/\tau$ altogether. This is an important issue because the number of frequency bins in the search affects the computational load.

Ignoring computation of the exponential function, directly computing the denominator of (15) requires $2nF$ complex multiplies, where F is the number of frequency bins. Given that a fine frequency sampling is required, the computational load can grow significantly without some *a priori* knowledge of where to search. It is possible to compute the denominator using an FFT. However, the input to the FFT is a delta–train (of mostly zeros) with length determined by the time span of the set S and the system clock rate. Thus, the FFT is not of length n, but rather some significantly larger value.

Use of (15) can be contrasted with the proposed modified Euclidean algorithms that are independent of choice of τ. As shown above in Remark 1, the robust single iteration algorithm has a relatively small computational load. In addition, our algorithms can yield estimates of τ despite a very high percentage of missing observations. These remarks suggest that one possible approach to estimation of τ is to use the modified Euclidean algorithm to obtain an estimate and then fine tune this estimate using another algorithm, e.g., (15).

5 Simulation Results and Discussion

5.1 Simulation Results

In this section we present simulation results and discuss performance of the proposed algorithms. All estimates and their standard deviations are based on averaging over 100 Monte–Carlo runs. The number of data samples is given by n (see equation (1)). Estimates of τ are labeled $\widehat{\tau}$, with experimental standard deviations in parentheses. Without loss of generality, we take $\tau = 1$ in all experiments. This choice is arbitrary, and any real positive number will yield similar results. Choice of initial phase ϕ is also arbitrary as the algorithms are independent of its choice due to the ordering and differencing operations. The noise values η_j are modeled as uniformly distributed with pdf $f_\eta(\eta) \sim \mathcal{U}[-\frac{\Delta}{2}, \frac{\Delta}{2}]$. So, for example, $\Delta = 10^{-1}$ implies random phase jitter that is $\pm 5\%$ of the period $\tau = 1$.

(1) *Noise–free estimation.*

In this example we examine the effects of changing n and the percentage of missing observations on the modified Euclidean algorithm of Section 2. The data are noise–free, i.e., $\eta_j = 0$ for all j. In this case the algorithm converges to the exact value of $\tau = 1$ with standard deviation equal to zero, or in some cases for small n to some multiple of τ. The jumps in the k_j's were modeled as uniformly distributed on the (discrete) interval $[1, M]$. Results are shown in Table 2 where $\%miss$ denotes the experimentally determined average percentage of missing observations, and *iter* is the average number of iterations required to converge.

The top half of Table 2 illustrates the effect of changing M, and therefore changing the percentage of missing observations. Given insufficient data the algorithm will converge to a multiple of τ. Columns labeled τ, 2τ, etc., indicate the percentage of runs that converged to these values. The algorithm is able to choose τ correctly based on $n = 10$ data samples, even with 99.998% of the possible observations missing. Convergence in the noise–free case depends on n but is independent of M, as implied by the analysis of Section 3. The bottom half of Table 2 illustrates the effect of changing n for M fixed. Reliable results are achieved for $n \geq 10$.

(2) *Noisy data and the multiple–iteration algorithm.*

In this example we implement the modified Euclidean algorithm (with averaging) of Section 4. Results are shown in Table 3. The missing observations were modeled using a Bernoulli process with parameter λ, Δ is the noise parameter, and *iter* is the mean number of iterations required to converge. Note that λ corresponds to the expected percentage of missing observations (see (14) and discussion).

The top half of Table 3 shows that estimates of τ degrade as the noise increases, as we would reasonably expect. The bottom half of Table 3 shows the effects of increasing λ, hence increasing the percentage of missing observations. The performance is essentially unchanged with $0.6 \leq \lambda \leq 0.9$. The last entry in Table 3 shows accurate estimation of τ from 100 data samples with $\pm 5\%$ phase jitter and 90% of the observations missing. It is possible to select $\lambda > 0.9$. However, estimation of $\tilde{\tau}$ becomes less reliable. For example, with $\lambda = 0.95$, 3 out of 100 trials resulted in poor estimates of $\tilde{\tau}$ that in turn resulted in poor estimates of τ, while the other 97 trials produced estimates close to $\tau = 1$.

(3) *Noisy data and the single iteration algorithm.*

Next we repeat Example 2 but now employ the single iteration algorithm of Section 4. Results are shown in Table 4. Here, *stepwidth* is the number of data samples isolated in the lowest step and whose mean resulted in $\widehat{\tau}$. The top half of the table shows the effect of increasing the noise. As we expect the estimate $\widehat{\tau}$ degrades for higher noise. Note that the value of *stepwidth* reveals that the algorithm is able to capture the approximately 50 data samples expected in the lowest step (due to $\lambda = 0.5$).

The bottom half of Table 4 depicts the effects of increasing λ. Here, estimates of τ degrade slightly as λ increases. The degradation is due to the decreasing value of *stepwidth*. As λ increases toward 1.0 fewer data samples are available in the lowest step, hence the mean of these samples becomes a less accurate estimate of τ. This effect can be counterbalanced by increasing n since the value of *stepwidth* is approximately $(1 - \lambda)n$ data samples.

(4) *Noisy data with outliers.*

Next we test the robust single iteration algorithm of Section 4 by adding outliers, with results

shown in Table 5. Here, $\%outliers$ indicates the percentage of the $n = 100$ data samples that are outliers. These were generated by choosing them to be uniformly distributed over the span of the set S. So, e.g., with $n = 100$ and $\%outliers = .15$, the data set was generated with 85 data samples and then 15 outliers were inserted for each Monte Carlo run, maintaining the total number of samples at $n = 100$.

Also shown in Table 5 is the step–width threshold x_0. The last column, labeled $\%runs$, indicates the number of Monte Carlo runs out of 100 that yielded estimates of τ. Failure of the algorithm to produce an estimate occurs when there is no step that exceeds the step–width threshold x_0. Note the last entry in Table 5 indicating successful estimation of τ from $n = 100$ data samples with 80% missing observations and 10 of the 100 data samples being arbitrary outliers.

The results of Table 5 are somewhat distorted by our method of generating outliers. For small λ (fewer missing observations) the total time span of the original data set S is considerably smaller than for larger values of λ. Thus, for small λ, the outliers are more closely spaced in time in the original data set S, and therefore more likely to result in a false step with mean less than τ. Counterbalancing this effect requires a larger step–width threshold x_0. The values for x_0 were chosen to avoid selecting such false steps with mean less than the true value of $\tau = 1$. Hence the relatively smaller value of $\%runs$ for small λ as shown in the Table.

5.2 Discussion

Testing of the modified Euclidean algorithm of Section 2 in the noise–free case (Example 1 above) reveals excellent performance with very few data samples, as theoretically predicted in Section 3. This performance is independent of the percentage of missing observations but the repeated subtraction can lead to very many iterations for extreme cases.

Comparison of the results of Examples 2 and 3 reveals that, in general, the single iteration algorithm outperforms the multi–iteration algorithm by a slight margin. This somewhat counter-intuitive result occurs due to the averaging over each step at each iteration. Some step–widths will be small, such that the resulting average over a particular step may have a high variance due to the lack of data samples. Whereas, in the single iteration algorithm only a single average is

computed over the lowest step. Our results indicate that, if the pattern of missing observations permits, the single iteration algorithm is likely to be preferable to the multi–iteration version. The single iteration version requires less computation and requires only two parameters, the gradient threshold g_0 and the step–height threshold y_0. Our experiments indicate very good robustness to choice of g_0. Choice of y_0 depends on the coarse estimate of τ we called $\tilde{\tau}$, which depends on getting sufficient data in the lowest step after the difference and sort operations, that in turn depends on the distribution of the missing observations in the original data set S.

The single iteration algorithm can also be made robust with the addition of one more parameter, the step–width threshold x_0. Proper choice of x_0 requires some knowledge of the distribution of the missing observations. However, proper choice can insure very robust performance with heavily contaminated data. Obtaining robust behavior with respect to outliers requires choices that are not likely to be optimal in all scenarios. Our experimental choices are based on experience and the scenarios considered.

Finally we note experimentation with using the median rather than the mean when averaging over steps, in an attempt to further combat outliers. While the median did produce slightly better results in some cases, the level of improvement was very slight.

6 Final Remarks

We have developed modifications of the Euclidean algorithm that are based on subtraction rather than division, and applied these algorithms to determine the period from a sparse set of noisy measurements. These results are based on the model given by (1), assuming a single period τ. The algorithms are computationally straightforward and require no *a priori* information. In the noise–free case our algorithm is equivalent to the Euclidean algorithm. It converges with very high probability given $n = 10$ or more data samples. Modifications of the original algorithm work well even in the presence of noise and outliers.

It is natural to consider an extension to the case of multiple τ's, say in a set $\{\tau_j\}_{j=1}^{n}$. We ask what number τ will be produced by the algorithm. This divides into two cases. The first

case considers $\{\tau_j\}$ such that the τ_j's are all rational multiples of each other. The algorithm will produce a number τ such that for each τ_j, there exists a positive integer c_j such that $c_j\tau = \tau_j$. The number τ is just $\gcd(\tau_1, \ldots, \tau_n)$. (This more general notion of gcd is consistent with Euclid's original algorithm, e.g., see Knuth [19], pp. 317–320. Also see Proposition 2.1.) For example, if $\tau_1 = \frac{1}{2}$, $\tau_2 = \frac{3}{4}$, and $\tau_3 = \frac{7}{4}$, then $\tau = \frac{1}{4}$. This result is easily seen, for it is just as if we applied the algorithm directly to the set $\{\tau_j\}$.

The second case considers $\{\tau_j\}$ such that at least two of the τ_j's are not rational multiples of each other, for example, $\tau_1 = \frac{1}{2}$, $\tau_2 = \sqrt{2}$. The values of the τ_j's are incommensurable, that is, there is no fundamental unit in which all of the τ_j's can be expressed as multiples. This follows from the fact that given a fixed irrational γ and any $\eta > 0$, there exists positive integers m and n such that $|m - n\gamma| < \eta$. (Also see Weyl's Equidistribution Theorem, Dym and McKean [7], pp. 54–56.) In this case, the algorithm will in theory produce $\tau = 0$, although this is sensitive to the noise thresholding and truncation error.

Finally, we discuss a least squares procedure for fine tuning our estimate of τ. Given the original data set S, estimate τ and ϕ. Given $\widehat{\tau}$ and $\widehat{\phi}$, get the set $\{\widehat{k}_j\}$ via equation (3) (see the remark at the end of Section 2). This last step is sensitive to the quality of the estimates $\widehat{\tau}$ and $\widehat{\phi}$, and to the distribution of the η_j's. Having estimated the set $\{\widehat{k}_j\}$, we then form the function

$$F(\tau) = \sum_j (s_j - \widehat{\phi} - \widehat{k}_j\tau)^2, \tag{16}$$

for $\tau > 0$, and solve for τ such that

$$\frac{\partial}{\partial\tau}F(\tau) = -2\sum_j \widehat{k}_j(s_j - \widehat{\phi} - \widehat{k}_j\tau) = 0\,. \tag{17}$$

Since $\partial^2/\partial\tau^2 F(\tau) = 2\sum_j(\widehat{k}_j)^2 > 0$, this value of τ will minimize $F(\tau)$.

7 Appendix: Proofs

Proof of Proposition 2.2 :

Let α be a positive integer such that $\alpha | k_j$ for $j = 1, \ldots, n$. Then, $\alpha | (k_j - k_{j+1})$ for $j = 1, \ldots, n-1$, and $\alpha | k_n$.

Conversely, assume β is a positive integer such that $\beta | (k_j - k_{j+1})$ for $j = 1, \ldots, n-1$, and $\beta | k_n$. Therefore, there exists positive integers c and d such that $c\beta = k_n$ and $d\beta = (k_{n-1} - k_n)$. Thus, $d\beta + k_n = (d + c)\beta = k_{n-1}$, and so $\beta | k_{n-1}$. By complete induction, $\beta | k_j$ for $j = 1, \ldots, n$.

Therefore, since the sets $\{k_j\}$ and $\{(k_j - k_{j+1})\} \cup \{k_n\}$ have the same divisors, their gcd's are equal. $\square$

Remark : *A proof of* $\gcd(k_1, k_2) = \gcd((k_1 - k_2), k_2)$ *with an outline of the algorithm.*

We may represent the Euclidean algorithm applied to k_1, k_2, for $k_1 > k_2$, as follows.

$$\begin{aligned} k_1 &= k_2 \cdot q_1 + r_1 \quad : \quad 0 < r_1 < k_2 \\ k_2 &= r_1 \cdot q_2 + r_2 \quad : \quad 0 < r_2 < r_1 \\ &\vdots \qquad\quad \vdots \\ r_{k-2} &= r_{k-1} \cdot q_k + r_k \quad : \quad 0 < r_k < r_{k-1} \\ r_{k-1} &= r_k \cdot q_k \,. \end{aligned} \tag{18}$$

The procedure terminates when $r_{k+1} = 0$. This gives $\gcd(k_1, k_2) = r_k$. Since $q_k \geq 1$,

$$k_1 - k_2 = k_2 \cdot (q_1 - 1) + r_1 \,. \tag{19}$$

We may replace the top line of (18) with (19), yielding the same result. Thus,

$$\gcd(k_1, k_2) = \gcd((k_1 - k_2), k_2) \,. \tag{20}$$

Therefore, given two positive integers k_1 and k_2, $k_1 > k_2$, we can reformulate the Euclidean algorithm using subtraction rather than division. This is stated in pseudocode as follows.

```
FUNCTION gcd(k_1, k_2)
Begin
    Do while k_2 ≠ 0
        k_1 = k_1 − k_2
        If k_1 < k_2, SWAP(k_1, k_2)
    Enddo
gcd(k_1, k_2) = k_1
End
```

Proof of Proposition 2.3 :

Let α be a positive integer such that $\alpha|(k_j - k_{j+1})$ for $j = 1, \ldots, n-1$. Therefore, $\alpha|(k_{n-1} - k_n)$ and $\alpha|(k_{n-2} - k_{n-1})$, and so there exist m_1, m_2 such that $k_{n-1} = m_1\alpha + k_n$ and $k_{n-2} = m_2\alpha + k_{n-1}$. Substituting, we get that $k_{n-2} = (m_2 + m_1)\alpha + k_n$, and so $\alpha|(k_{n-2} - k_n)$. By complete induction, we get that $\alpha|(k_j - k_n)$, for $j = 1, \ldots, n-1$.

Conversely, assume β is a positive integer such that $\beta|(k_j - k_n)$ for $j = 1, \ldots, n-1$. Thus $\beta|(k_j - k_n)$ and $\beta|(k_{j+1} - k_n)$, and so there exists m_j, m_{j+1} such that $k_j = m_j\beta + k_n$ and $k_{j+1} = m_{j+1}\beta + k_n$. Thus, $k_j - k_{j+1} = (m_j - m_{j+1})\beta$, and so $\beta|(k_j - k_{j+1})$.

Therefore, since the sets $\{(k_j - k_{j+1})\}$ and $\{(k_j - k_n)\}$ have the same divisors, their gcd's are equal. □

Proof of Theorem 3.2 :

Let $\lfloor x \rfloor$ denote the floor function of x, namely

$$\lfloor x \rfloor = \max_{k \le x} \{k : k \in \mathbf{Z}\}. \tag{21}$$

Recall that $N_n(\ell) = \text{card}\{(k_1, \ldots, k_n) \in \{1, \ldots, \ell\}^n : \gcd(k_1, \ldots, k_n) = 1\}$ is the number of relatively prime elements in $\{1, \ldots, \ell\}^n$. We claim that

$$N_n(\ell) = \ell^n - \sum_{p_i} \left(\lfloor \frac{\ell}{p_i} \rfloor \right)^n + \sum_{p_i < p_j} \left(\lfloor \frac{\ell}{p_i \cdot p_j} \rfloor \right)^n - \sum_{p_i < p_j < p_k} \left(\lfloor \frac{\ell}{p_i \cdot p_j \cdot p_k} \rfloor \right)^n + \ldots . \tag{22}$$

This is seen as follows. Choose a prime number p_i. The number of integers in $\{1, \ldots, \ell\}$ such that p_i divides an element of that set is $\lfloor \frac{\ell}{p_i} \rfloor$. (Note that is is possible to have $p_i > \ell$, because $\lfloor \frac{\ell}{p_i} \rfloor = 0$.) Therefore, the number of n–tuples $(k_1, \ldots, k_n)$ contained in the lattice $\{1, \ldots, \ell\}^n$ such

that p_i divides every integer in the n–tuple is

$$\left(\lfloor\frac{\ell}{p_i}\rfloor\right)^n. \tag{23}$$

Next, if $p_i \cdot p_j$ divides an integer k, then $p_i|k$ and $p_j|k$. Therefore, the number of n–tuples $(k_1,\ldots,k_n)$ contained in the lattice $\{1,\ldots,\ell\}^n$ such that p_i and p_j both divide every integer in the n–tuple is

$$\left(\lfloor\frac{\ell}{p_i}\rfloor\right)^n + \left(\lfloor\frac{\ell}{p_j}\rfloor\right)^n - \left(\lfloor\frac{\ell}{p_i \cdot p_j}\rfloor\right)^n, \tag{24}$$

where the last term is subtracted so that we do not count the same numbers twice (in a simple application of the inclusion–exclusion principle).

Continuing in this fashion, for three integers, say $p_i < p_j < p_k$, the number of n–tuples $(k_1,\ldots,k_n)$ contained in the lattice $\{1,\ldots,\ell\}^n$ such that p_i, p_j, and p_k all divide every integer in the n–tuple is given by the inclusion–exclusion principle as

$$\left[\left(\lfloor\frac{\ell}{p_i}\rfloor\right)^n + \left(\lfloor\frac{\ell}{p_j}\rfloor\right)^n + \left(\lfloor\frac{\ell}{p_k}\rfloor\right)^n\right] - \left[\left[\left(\lfloor\frac{\ell}{p_i \cdot p_j}\rfloor\right)^n - \left(\lfloor\frac{\ell}{p_j \cdot p_k}\rfloor\right)^n\right] - \left(\lfloor\frac{\ell}{p_j \cdot p_j \cdot p_k}\rfloor\right)^n\right]. \tag{25}$$

We can therefore see by induction that the number of n–tuples $(k_1,\ldots,k_n)$ contained in the lattice $\{1,\ldots,\ell\}^n$ such that $p_i, p_j, p_k, \ldots$ all divide every integer in the n–tuple is given by the inclusion–exclusion principle as

$$\sum_{p_i}\left(\lfloor\frac{\ell}{p_i}\rfloor\right)^n - \sum_{p_i<p_j}\left(\lfloor\frac{\ell}{p_i \cdot p_j}\rfloor\right)^n + \sum_{p_i<p_j<p_k}\left(\lfloor\frac{\ell}{p_i \cdot p_j \cdot p_k}\rfloor\right)^n - \ldots. \tag{26}$$

But this counts the complement of $N_n(\ell)$ in the lattice $\{1,\ldots,\ell\}^n$. Therefore,

$$N_n(\ell) = \ell^n - \sum_{p_i}\left(\lfloor\frac{\ell}{p_i}\rfloor\right)^n + \sum_{p_i<p_j}\left(\lfloor\frac{\ell}{p_i \cdot p_j}\rfloor\right)^n - \sum_{p_i<p_j<p_k}\left(\lfloor\frac{\ell}{p_i \cdot p_j \cdot p_k}\rfloor\right)^n + \ldots. \tag{27}$$

Next, we observe that

$$\begin{aligned}
& \frac{1}{\ell^n}\sum_{p_i<p_j<\ldots<p_k}\left(\lfloor\frac{\ell}{p_i \cdot p_j \cdot \ldots \cdot p_k}\rfloor\right)^n \\
\leq\ & \frac{1}{\ell^n}\sum_{p_i<p_j<\ldots<p_k\leq\ell}\left(\frac{\ell}{p_i \cdot p_j \cdot \ldots \cdot p_k}\right)^n \\
=\ & \sum_{p_i<p_j<\ldots<p_k\leq\ell}\left(\frac{1}{p_i \cdot p_j \cdot \ldots \cdot p_k}\right)^n
\end{aligned}$$

$$
\begin{aligned}
&= \left(\sum_{p\le\ell}\frac{1}{p^n}\right)^k \\
&\le \left(\sum_{p \text{ prime}}\frac{1}{p^n}\right)^k \\
&\le \left(\sum_{j}\frac{1}{j^n}\right)^k ,
\end{aligned}
$$

where the last sum over $j \in \mathbf{N} \setminus \{1\}$. Since $n \ge 2$, this series is convergent. Therefore, each term in the expansion of $\frac{N_n(\ell)}{\ell^n}$ is convergent.

We recall the Möbius function μ, which is defined as follows.

$$
\begin{aligned}
\mu(1) &= 1, \\
\mu(m) &= \left\{ \begin{array}{ll} 0 & \text{if } m \text{ is divisible by the square of a prime,} \\ (-1)^r & \text{if } m = p_1 \cdot p_2 \cdot \ldots \cdot p_r, \text{ where } p_1, p_2, \ldots, p_r \text{ are all distinct primes.} \end{array} \right\}
\end{aligned} \tag{28}
$$

Euler showed that

$$
\begin{aligned}
& 1 - \sum_{p_i}\frac{1}{p_i^n} + \sum_{p_i<p_j}\frac{1}{(p_i\cdot p_j)^n} - \sum_{p_i<p_j<p_k}\frac{1}{(p_i\cdot p_j\cdot p_k)^n} + \ldots \\
= & \sum_m \frac{\mu(m)}{m^n} = [\zeta(n)]^{-1} ,
\end{aligned}
$$

where the last sum is over $m \in \mathbf{N}$. For $n \ge 2$, $\sum_m \frac{\mu(m)}{m^n}$, $\sum_m \frac{1}{m^n}$ are absolutely convergent. Thus, the equality $\sum_m \frac{\mu(m)}{m^n} = [\zeta(n)]^{-1}$ follows because for $j, k, m, n \in \mathbf{N}$,

$$
\sum_m \frac{1}{m^n}\sum_j \frac{\mu(j)}{j^n} = \sum_{m,j}\frac{\mu(j)}{(mj)^n} = \sum_k \frac{1}{k^n}\sum_{d|k}\mu(d) = 1 ,
$$

where we have used the fact that both series in the first term converge absolutely and thus can be multiplied together by adding all possible products of terms and rearranging in any order (see Ireland and Rosen [14] and Leveque [21]).

Let

$$
M_k = \left(\sum_j \frac{1}{j^n}\right)^k , \tag{29}
$$

with sum over $j \in \mathbf{N} \setminus \{1\}$. Since $n \ge 2$ and the sum is over $j \in \mathbf{N} \setminus \{1\}$,

$$
0 < \sum_j \frac{1}{j^n} \le \left(\frac{\pi^2}{6} - 1\right) < 1 . \tag{30}
$$

Since the k^{th} term in the expansion of $\frac{N_n(\ell)}{\ell^n}$ is dominated by M_k, and $\sum_k M_k$ is convergent, we may apply the Weierstrass M test, and evaluate term–by–term. We use the fact that $x - 1 \leq \lfloor x \rfloor \leq x$ for all x, and so $\lim_{x\to\infty} \frac{\lfloor x \rfloor}{x} = 1$. Therefore, we have

$$\begin{aligned}
&\lim_{\ell\to\infty} \frac{N_n(\ell)}{\ell^n} \\
&= \lim_{\ell\to\infty} \frac{1}{\ell^n} \left(\ell^n - \sum_{p_i} \left(\lfloor \frac{\ell}{p_i} \rfloor \right)^n + \sum_{p_i<p_j} \left(\lfloor \frac{\ell}{p_i \cdot p_j} \rfloor \right)^n - \sum_{p_i<p_j<p_k} \left(\lfloor \frac{\ell}{p_i \cdot p_j \cdot p_k} \rfloor \right)^n + \dots \right) \\
&= 1 - \sum_{p_i} \frac{1}{p_i^n} + \sum_{p_i<p_j} \frac{1}{(p_i \cdot p_j)^n} - \sum_{p_i<p_j<p_k} \frac{1}{(p_i \cdot p_j \cdot p_k)^n} + \dots \\
&= \sum_m \frac{\mu(m)}{m^n} \\
&= [\zeta(n)]^{-1} ,
\end{aligned}$$

where the last sum is over $m \in \mathbf{N}$. This completes the proof of the theorem. □

Proof of Corollary 3.1 :

By Proposition 2.3,

$$\gcd((k_1 - k_2), (k_2 - k_3), \dots, (k_{n-1} - k_n)) = \gcd((k_1 - k_n), (k_2 - k_n), \dots, (k_{n-1} - k_n)) .$$

Thus, computing the gcd of the left hand side is the same as computing the gcd of the set $\{k_1, k_2, \dots, k_{n-1}\}$ shifted by k_n. Now, since $k_j - k_{j+1} \geq 1$, we see that we can repeat the analysis of Theorem 3.2 in the $(n-1)$–dimensional sublattice of $\mathbf{N}^n$ consisting of the of $(n-1)$–tuples of the form

$$\{(m + k_n), \dots, (m + k_n)\}, 1 \leq m \leq \ell .$$

In this sublattice, for sufficiently large ℓ, the number of elements that are divisible by some given prime p is essentially equal to the number of elements in $\{1, \dots, \ell\}^{n-1}$ divisible by p. Moreover, if we denote the number of relatively prime $(n-1)$–tuples in this shifted lattice by $N_{n-1}(\ell_s)$, we have that

$$\lim_{\ell\to\infty} \frac{N_{n-1}(\ell_s)}{\ell^{n-1}} = \lim_{\ell\to\infty} \frac{N_{n-1}(\ell)}{\ell^{n-1}} .$$

(Asymptotically, the fact that the sublattice is shifted by some finite fixed amount will not matter.)

But then, by Theorem 3.2,

$$\lim_{\ell \to \infty} \frac{N_{n-1}(\ell)}{\ell^{n-1}} = [\zeta(n-1)]^{-1} \ . \ \square$$

Proof of Proposition 3.1 :

Since

$$\zeta(\omega) = \sum_{n=1}^{\infty} n^{-\omega}$$

and $\omega > 1$,

$$\begin{aligned} & 1 \leq \zeta(\omega) = 1 + \frac{1}{2^\omega} + \frac{1}{3^\omega} + \frac{1}{4^\omega} + \frac{1}{5^\omega} + \dots \\ \leq \ & 1 + \frac{1}{2^\omega} + \frac{1}{2^\omega} + \underbrace{\frac{1}{4^\omega} + \dots + \frac{1}{4^\omega}}_{4-\text{times}} + \underbrace{\frac{1}{8^\omega} + \dots + \frac{1}{8^\omega}}_{8-\text{times}} + \dots \\ = \ & \sum_{k=0}^{\infty} \left(\frac{2}{2^\omega}\right)^k = \frac{1}{1 - \frac{2}{2^\omega}} = \frac{1}{1 - 2^{1-\omega}} \ . \end{aligned}$$

As $\omega \longrightarrow \infty$, $(1 - 2^{1-\omega}) \longrightarrow 1^+$. Therefore, by the Squeezing Theorem,

$$[\zeta(\omega)]^{-1} \longrightarrow 1^- \text{ as } \omega \longrightarrow \infty \ . \ \square$$

Acknowledgments : The authors would like to express their sincere thanks to Professor Lawrence Washington of the University of Maryland, Professor David A. Joyner of the U. S. Naval Academy, Professor Lawrence Crone of American University, and Mr. Andree N. Filipov of the Army Research Laboratory for many insightful discussions. We also would like to thank the reviewers for several perceptive comments that led to improvements in the paper.

References

[1] Bartlett, M. S., "The spectral analysis of point processes," *Journal of the Royal Statistical Society B*, Vol. 25, No. 2, pp. 264–280, 1970.

[2] Bloomfield, P., "Spectral analysis with randomly missing observations," *Journal of the Royal Statistical Society B*, Vol. 32, No. 3, pp. 369–380, 1970.

[3] Beyer, W. H., editor, *CRC Standard Mathematical Tables and Formulae (29^{th} Edition)*, CRC Press, Boca Raton, Florida, 1981.

[4] Casey, S. D., and Sadler, B. M., "A modified Euclidean algorithm for isolating periodicities from a sparse set of noisy measurements," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '95)*, Vol. 3, pp. 1764–1767, 1995.

[5] Conway, J. B., *Functions of One Complex Variable (Second Edition)*, Springer–Verlag, New York, 1978.

[6] Dandawate, A. V., and Giannakis, G. B., "Nonparametric cyclic-polyspectral analysis of AM signals and processes with missing observations," *IEEE Transactions on Information Theory,* Vol. 39, No. 6, pp. 1864–1876, 1993.

[7] Dym, H., and McKean, H. P., *Fourier Series and Integrals*, Academic Press, Orlando, Florida, 1972.

[8] Edwards, P. J., "Comments on 'A zero crossing–based spectrum analyzer'," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 37, No. 7, pp. 1143–1144, 1989.

[9] Fogel, E., and Gavish M., "Parameter estimation of quasi–periodic sequences," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '88)*, Vol. 4, pp. 2348–2351, 1988.

[10] Fogel, E., and Gavish M., "Performance evaluation of zero–crossing–based bit synchronizers," *IEEE Transactions on Communications*, Vol. 37, No. 6, pp. 663–665, 1989.

[11] Giannakis, G. B., and Zhou, G., "Parameter estimation of cyclostationary AM time series with application to missing observations," *IEEE Transactions on Signal Processing*, Vol. 42, No. 9, pp. 2408–2419, 1994.

[12] Gray, D., Slocumb, B., and Elton, S., "Parameter estimation for periodic discrete event processes," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '94)*, Vol. 4, pp. 93–96, 1994.

[13] Hardy, G. H., and Wright, E. M., *An Introduction to the Theory of Numbers*, Oxford University Press, New York, 1982.

[14] Ireland, K., and Rosen, M., *A Classical Introduction to Modern Number Theory*, Springer–Verlag, New York, 1982.

[15] Jain, A. K., *Fundamentals of Digital Image Processing,* Prentice–Hall, New Jersey, 1989.

[16] Jones, R. H., "Spectral analysis with regularly missed observations," *Annals of Mathematical Statistics*, Vol. 33, pp. 455–461, 1962.

[17] Kay, S. M., and Sudhaker, R., "A zero crossing–based spectrum analyzer," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 34, No. 1, pp. 96–104, 1986.

[18] Kedem, B., *Time Series Analysis by Higher Order Crossings*, IEEE Press, New York, 1994.

[19] Knuth, D. E., *The Art of Computer Programming, Volume 2: Seminumerical Algorithms (Second Edition)*, Addison–Wesley, Reading, Massachusetts, 1981.

[20] Knuth, D. E., *The Art of Computer Programming, Volume 3: Sorting and Searching*, Addison–Wesley, Reading, Massachusetts, 1973.

[21] Leveque, W. J., *Topics in Number Theory, Volumes 1 and 2*, Addison–Wesley, Reading, Massachusetts, 1956.

[22] Parzen, E., "On spectral analysis with missing observations and amplitude modulation," *Sankhya*, Ser. A, Vol. 25, pp. 383–392, 1963.

[23] Papoulis, A., *Signal Analysis*, McGraw–Hill, New York, 1977.

[24] Pasupathy, S., "Pi, primes, and probability," *IEEE Communications Magazine (Light Traffic)*, Vol. 24, No. 3, pp. 61–62, 1987.

[25] Reiss, R. –D., *Approximate Distributions of Order Statistics*, Springer–Verlag, New York, 1989.

[26] Rosen, K. H., *Elementary Number Theory and Its Applications*, Addison–Wesley, Reading, Massachusetts, 1984.

[27] Rosen, Y. and Porat, B., "Optimal ARMA parameter estimation based on the sample covariances for data with missing observations," *IEEE Transactions on Information Theory*, Vol. 35, No. 2, pp. 342–349, 1989.

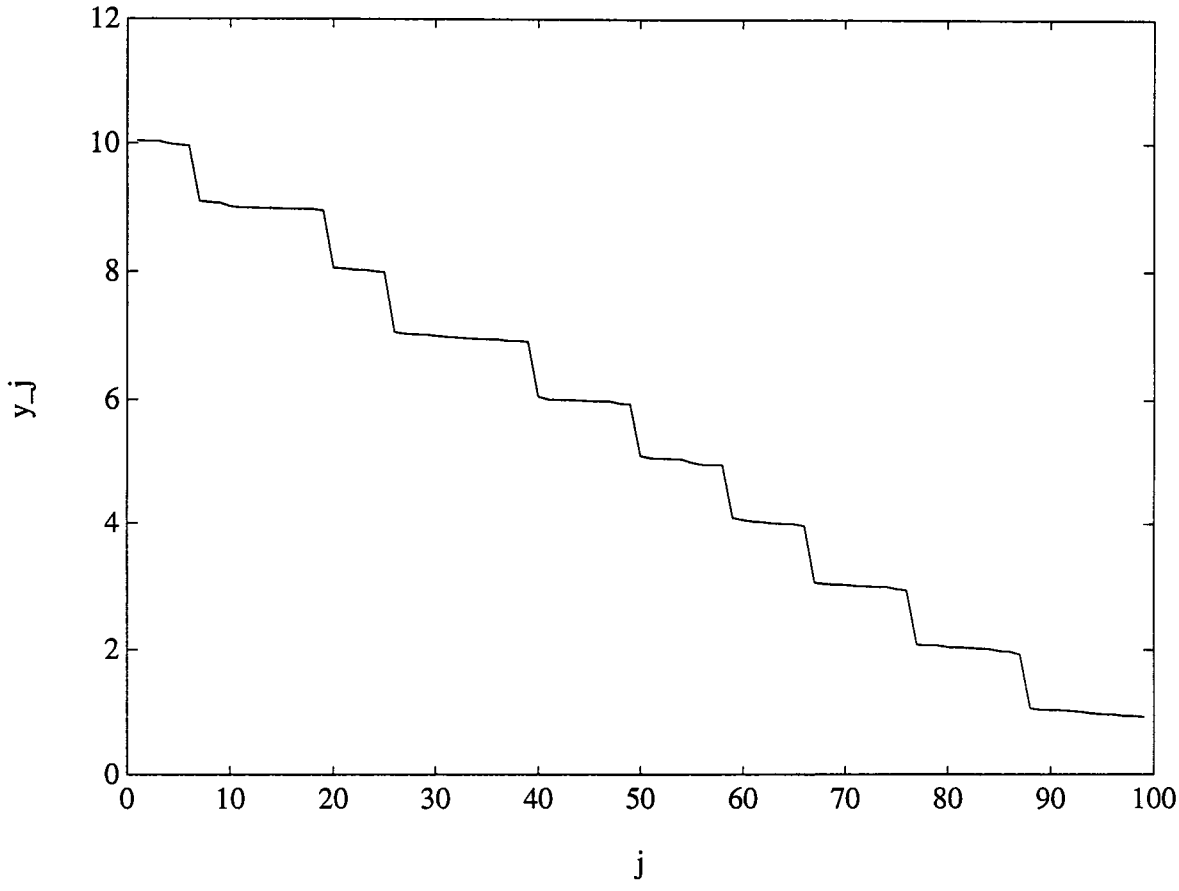
[28] Sarhan, A. E., and Greenberg, B. G., eds., *Contributions to Order Statistics*, John Wiley, New York, 1962.

[29] Scheinok, P. A., "Spectral analysis with randomly missed observations: the binomial case," *Annals of Mathematical Statistics*, Vol. 36, pp. 971–977, 1965.

[30] Schroeder, M. R., *Number Theory in Science and Communication (Second Edition)*, Springer–Verlag, Berlin, 1986.

[31] Wiley, R. G., *Electronic Intelligence: The Analysis of Radar Signals (Second Edition)*, Artech House, Norwood, Massachusetts, 1993.

Figure 1



1. Plot of example data set after one iteration of the modified Euclidean algorithm of Section 2. The data is sorted in descending order into steps centered around multiples of τ ($\tau = 1$ in this example). The stepwidths are a function of the distribution of the k_j's in the original data set S. The lowest (rightmost) step is centered around the true value of $\tau = 1$.

Table 1: Some values of the Zeta function $\zeta(n)$ and $1/\zeta(n)$.

n	$\zeta(n)$	$1/\zeta(n)$
2	1.64493	0.6079
4	1.08232	0.9239
6	1.01734	0.9830
8	1.00407	0.9959
10	1.00099	0.9990
12	1.00024	0.9998
16	1.00001	1.0000

Table 2: Results from simulation Example 1, noise-free estimation of τ with the modified Euclidean algorithm.

n	M	$\%miss$	$iter$	τ	2τ	3τ	$> 3\tau$
10	10^1	81.69	3.3	100%	0	0	0
10	10^2	97.92	10.5	100	0	0	0
10	10^3	99.80	46.5	100	0	0	0
10	10^4	99.98	316.2	100	0	0	0
10	10^5	99.998	2638.7	100	0	0	0
4	10^2	97.84	15.2	82%	12	4	2
6	10^2	97.81	14.2	97	3	0	0
8	10^2	97.96	10.2	98	1	1	0
10	10^2	97.95	10.2	99	1	0	0
12	10^2	97.95	8.6	100	0	0	0
14	10^2	97.97	7.4	100	0	0	0

Table 3: Example 2 results, estimation of τ from noisy measurements using the modified Euclidean algorithm (with averaging).

n	λ	Δ	$\widehat{\tau}$ (std)	$iter$ (std)
100	.5	10^{-3}	1.0000 (.0001)	3.5 (.6)
100	.5	10^{-2}	1.0000 (.0014)	3.5 (.5)
100	.5	10^{-1}	1.0002 (.0169)	3.5 (.6)
100	.5	2×10^{-1}	1.0032 (.0212)	3.5 (.5)
100	.6	10^{-1}	0.9998 (.0158)	3.7 (.6)
100	.7	10^{-1}	0.9997 (.0200)	4.0 (.6)
100	.8	10^{-1}	1.0002 (.0202)	3.9 (.4)
100	.9	10^{-1}	1.0038 (.0213)	4.2 (.6)

Table 4: Example 3 results, estimation of τ from noisy measurements using the single iteration algorithm.

n	λ	Δ	$\widehat{\tau}$ (std)	*stepwidth* (std)
100	.5	10^{-3}	1.0000 (.00001)	49.0 (4.6)
100	.5	10^{-2}	1.0000 (.0004)	48.3 (4.5)
100	.5	10^{-1}	0.9994 (.0043)	49.2 (5.0)
100	.5	2×10^{-1}	0.9959 (.0104)	47.1 (5.2)
100	.6	10^{-1}	1.0000 (.0056)	38.5 (4.9)
100	.7	10^{-1}	1.0009 (.0061)	28.3 (5.0)
100	.8	10^{-1}	0.9986 (.0089)	18.6 (3.9)
100	.9	10^{-1}	0.9988 (.0141)	8.1 (2.7)

Table 5: Example 4 results, for noisy sparse data with outliers, using the robust single iteration algorithm.

n	λ	Δ	*%outliers*	$\widehat{\tau}$ (std)	x_0	*%runs*
100	.05	10^{-1}	0.15	0.9983 (.0035)	35	93
100	.05	10^{-1}	0.25	0.9973 (.0054)	35	86
100	.05	10^{-1}	0.35	0.9928 (.0099)	35	52
100	.10	10^{-1}	0.05	0.9994 (.0027)	25	100
100	.10	10^{-1}	0.15	0.9991 (.0036)	25	99
100	.10	10^{-1}	0.25	0.9975 (.0062)	25	98
100	.10	10^{-1}	0.35	0.9958 (.0105)	25	90
100	.10	10^{-1}	0.45	0.9839 (.0149)	20	16
100	.25	10^{-1}	0.05	1.0006 (.0032)	15	100
100	.25	10^{-1}	0.15	0.9987 (.0037)	15	100
100	.25	10^{-1}	0.25	0.9987 (.0052)	15	100
100	.25	10^{-1}	0.35	0.9953 (.0095)	15	93
100	.50	10^{-1}	0.05	1.0014 (.0048)	10	100
100	.50	10^{-1}	0.15	1.0004 (.0069)	10	100
100	.50	10^{-1}	0.25	1.0003 (.0071)	15	100
100	.50	10^{-1}	0.35	1.0004 (.0110)	15	98
100	.75	10^{-1}	0.05	1.0043 (.0092)	5	100
100	.75	10^{-1}	0.10	1.0036 (.0344)	10	100
100	.75	10^{-1}	0.15	1.0064 (.0189)	10	100
100	.75	10^{-1}	0.20	1.0004 (.0211)	10	100
100	.80	10^{-1}	0.05	1.0038 (.0103)	5	100
100	.80	10^{-1}	0.10	1.0061 (.0143)	5	100