Bayesian Deconvolution of Seismic Array Data

By

ERIC ALBERT SUESS

B.A. (University of California, Berkeley) 1991

M.S. (California State University, Hayward) 1993

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Statistics

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Committee in Charge

1998

To

Ms. Amerman, Ms. Carol Suess, and Mr. George.

Mr. Randy Herzberg.

Dr. Seuss.

Acknowledgments

Most importantly, I would like to thank my advisor Professor Robert Shumway for his teaching, guidance, and support. My experience working with Bob has been exciting and truly rewarding.

I would like to thank the members of my dissertation committee: Professors Rahman Azari, Richard Levine, Y.P. Mack, and Jim McClain. The time and input they have given has been invaluable.

I would also like to thank my mentor Professor Bruce Trumbo for his encouragement and advice during my time in graduate school.

Lastly, I want to express my gratitude to Larry Tai and Arthur Gin for their technical computing advice and assistance. Their help made my computing programming efforts more efficient and enjoyable.

Eric Albert Suess
December 1998
Statistics

Bayesian Deconvolution of Seismic Array Data

Abstract

From the time of the earliest nuclear test ban treaties seismic data have played an important role in monitoring for compliance: for example, to distinguish large nuclear explosions from earthquakes. More restrictive recent treaties, such as the 1996 Comprehensive Test Ban Treaty (CTBT), have focused attention on the detection of small nuclear explosions. The problem examined in this dissertation is the use of data from regional seismic arrays to discriminate between small nuclear explosions and commonly occurring ripple-fired mining explosions.

This work presents a time-domain convolution model for seismic data collected by an array of sensors produced by ripple-fired events. The large number of parameters in this model and the existing body of expert knowledge about some of the characteristics of such explosions motivates a Bayesian approach. Gibbs sampling methods, following the univariate work of Rong Chen and Cheng, Chen, and Li (1996), lead to a procedure for deconvolving seismic array data, estimating parameters in the model, and determining the common delay pattern contained in the seismic data as it appears at each sensor. The program that implements the Gibbs sampler in the problem is included in the Appendix.

Results on simulated data demonstrate the ability of the deconvolution method to accurately estimate the common delay structure. Applied to the P_n, P_g, S_n, and L_g-phases of a ripple-fired explosion, the method shows agreement among the four phases of ripple-firing. A comparison of the results from the Bayesian approach with the results from other methods shows comparable delay time estimates with more consistency among the phases. Concluding discussion examines the implications of this seismic deconvolution method for monitoring nuclear testing.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

As a result of the numerous treaties related to nuclear testing, the monitoring of regional seismic activity for possible nuclear testing continues to be important. While there has been extensive research into the use of seismic signals to detect nuclear explosions, distinguishing nuclear from non-nuclear explosions remains a significant problem. This research has been driven primarily by the many nuclear testing agreements that have been signed in the past between the United States, the former U.S.S.R. and other nuclear powers. The main treaties are: (1) the 1963 Limited Nuclear Test Ban Treaty, which prohibits nuclear testing in the atmosphere, in outer space, and underwater, (2) the 1968 Non-Proliferation of Nuclear Weapons Treaty (NPT), an agreement among the nuclear powers of the time to discontinue all nuclear testing, (3) the 1974 Threshold Test Ban Treaty (TTBT), which restricts the testing of underground nuclear weapons to yields no greater than 150 kilotons, (4) the 1976 Peaceful Nuclear Explosions Treaty (PNET), which restricts individual peaceful nuclear explosions to yields no greater than 150 kilotons and to aggregate yields of no greater than 1500 kilotons per year, and (5) the 1996 Comprehensive Test Ban Treaty (CTBT), which effectively bans all nuclear weapons testing [40]. The CTBT has been signed by most of the nations of the world; some of the notable exceptions are India, Pakistan, and North Korea.

Much of the past research related to monitoring seismic activity has focused on distinguishing nuclear explosions from earthquakes. Because underground nuclear tests are conducted closer to the surface than typical earthquakes occur, depth estimates have been used as one method to distinguish between these two types of seismic events. More recently, due to the limitations on the size of permissible nuclear tests, other smaller seismic events, such as industrial mining explosions, have become of interest in the overall discrimination problem. Distinguishing these smaller types of explosions from small nuclear explosions becomes an important issue.

1.2 The Problem

The problem that motivates the research in this dissertation is to distinguish typical ripple-fired mining explosions from low level nuclear explosions. These two types of explosions are typically of the same magnitude. Ripple-fired mining explosions are used to produce maximum rock fracture while minimizing near-surface disturbances and are common seismic events worldwide. This technique of quarry mining involves the use of multiple rows of explosives that are detonated with approximately equal time delays between the rows. For our purposes, we assume all of the devices within each row are detonated simultaneously. Konya and Walter [22] coauthored a book on the subject of industrial mining and Pearson, Strump, and Anderson [27] have created an excellent web-site from their research that contains a multimedia paper containing movie clips and visual representations of three dimensional models of ripple-fired mining explosions.

1.3 The Procedure

The objective of this dissertation is to develop an estimation procedure that estimates delay times in seismic events. The delay structure of a single seismic event is characterized by the size (or amplitude) of the delayed explosions and the delay times between the single explosions. If a delay structure can be estimated, this will be useful in discriminating ripple-fired mining explosions from other, possibly nuclear, seismic events. In addressing this problem, we have developed a model for ripple-fired events and a deconvolution method based on work by Rong Chen, in the univariate case, and work by Cheng, Chen, and Li [7] that is used to estimate parameters in our model related to the delay time structure.

Our method woks by separating, or deconvolving a *pulse sequence* from the *underlying signal-path effects sequences*. The *pulse sequence* is used to model the detonation design of a mining explosion and the *underlying signal-path effects sequences* is used to model the common underlying signal sent by each detonation in a ripple-fired event and the effects of the path on the signal as it traces to the sensors. We approach the problem from a Bayesian perspective. The Gibbs sampler is implemented to produce posterior estimates of the parameters in our model, using the data and available prior information. The prior information we use in our analysis is taken from past research on ripple-fired mining explosion data. From the estimated *pulse sequence* we produce estimates of the delay time structure of the ripple-fired events.

The related problem of estimating depths for discriminating between nuclear explosions and earthquakes provides a basis for our discussion of estimating the delay time structure of ripple-fired mining explosions. One method used to estimate the depth of explosions is the delay time observed between the P-phase and the pP-

reflection of a recorded seismic event. The top panel of Figure 1.2 shows an example of a seismogram from on channel of a seismic event recorded at a regional distance; and the third panel shows an enlargement of the P_n-phase. The $P-n$-phase of a seismic event is a body wave. It is a compressional wave that travels through the crust of the Earth, then travels mostly horizontal across the top of the upper mantle and finally travels through the crust to the seismometer. At regional distances the P_n-phase is usually the first to appear at the sensor recording the seismic event [9].

The P_n-phase with deep explosions has two parts. The first part is the P-phase, which travels directly to the mantle and then to the sensor. The second part is the pP-phase, which travels to the surface first, is inverted, travels to the mantle, and finally travels to the sensor. The estimated time difference between the arrivals, or the delay time, can be used to obtain depth estimates of both single explosions and earthquakes. For discrimination purposes, earthquakes tend to be deeper within the earth than nuclear tests. Therefore, larger delays between the arrivals of the P-phase and pP reflection are associated with earthquakes and shorter delays with nuclear tests. The P-phase and the pP-phases contain the same type of signal, but they arrive at different times. If the amplitude of the P-phase is defined as $a_0 = 1$, then the amplitude and time of the arrival of the pP-phase can be represented as a_j, where j is the delay time and a_j is the amplitude of the delayed signal relative to a_0. Figure 1.1 represents a hypothetical *pulse sequence* of a nuclear explosion or an earthquake. Our current interest in this related problem is the use of delay time structure for discriminating seismic signals of ripple-fired events from nuclear explosions and earthquakes [9]. Figure 1.3 represents a hypothetical *pulse sequence* of a ripple-fired explosion. The difference in these estimated *pulse sequences* for the basis of our deconvolution method.

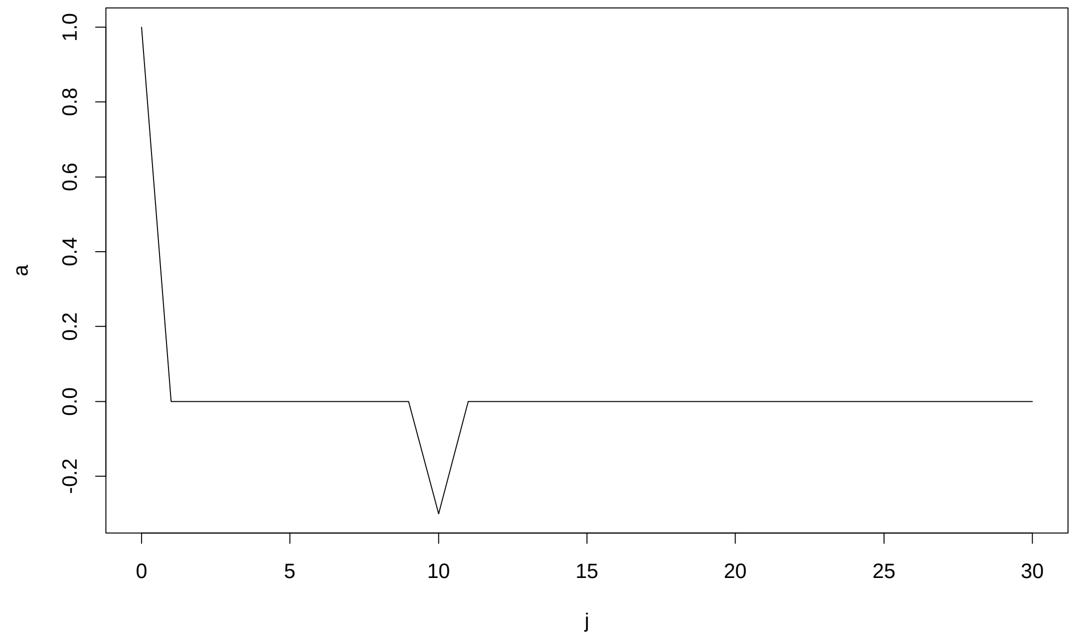


Figure 1.1: Hypothetical pulse sequence of a nuclear explosion or earthquake.

1.4 Literature Review

Estimating the delay structure of ripple-fired explosions has been studied almost exclusively in the frequency domain. Baumgardt and Ziegler [1] considered lining up the log-spectra and cepstra for an array to detect the delays present in ripple-fired events. Hedlin, Minster, and Orcutt [19] proposed graphical techniques for detecting ripple firing that use time-dependent log-spectra and cepstra, and binary sonograms. Shumway, Baumgardt, and Der [30] developed a cepstral F-statistic method for detecting statistically significant delays based on detrended log-spectra. A reference for the cepstrum is by Bogert, Healy, and Tukey [2]. In the time domain, Shumway and MacQuarrie [32] developed an ARIMA search technique for estimating a single delay parameter in a ripple-fired event.

We propose in this dissertation a new approach to identifying ripple-fired mining explosions using Bayesian methods. A convolution model is developed with parameters related to the *pulse sequence* that represents the sub-explosions in a ripple-fired event and the *underlying signal-path effects sequences* contained within the seismic traces recorded by each seismometer in an array. Both the *pulse sequence* and the *underlying signal-path effects sequences* are assumed to be random and this randomness is modeled. We develop a deconvolution method to estimate the relevant parameters. Our approach is similar to the deconvolution methods developed by Cheng, Chen, and Li [7] and also by Cheng and Li [6] in that we use the Gibbs sampler to produce deconvolution estimates of the parameters and we employ the Bernoulli-Gaussian model.

Cheng, Chen, and Li [7] use the Gibbs sampler to estimate parameters in the univariate deconvolution of seismic data collected in reflection seismology experiments using one sensor. In such tests, a signal is sent to probe the earth and the reflectivity sequence is recorded at the surface. Through this signal the structure of the shallow earth is analyzed [4]. The *true reflectivity* is not directly measurable; instead, a smeared version of the reflectivity caused by the different layers in the earth is recorded. The main objective of this type of seismic data analysis is to recover the *true reflectivity*. To this end Cheng, Chen, and Li develop a single series seismic deconvolution method using the Gibbs sampler.

Alternative deconvolution methods have been developed, Hannan and Thompson [18] developed a maximum likelihood approach to detect single delays in one channel seismic signals in the frequency domain. Shumway and Der [31] worked on a maximum likelihood approach to multichannel deconvolution also in the frequency domain. In the time domain, maximum likelihood procedures were developed by Kolmylo and Mendel [23] and by Goutsias and Mendel [17]; in both papers, maximum likelihood deconvolution procedures for one-channel data are presented. Two useful books on the subject of seismic deconvolution were written by Mendel [24] and [25]. Very early minimum entropy approaches have been developed by Donoho [10] and by Wiggins [42]. The application of each of these approaches appears to be limited when applied to multichannel data.

1.5 Outline and The Data

In Chapter 2, a multidimensional convolution model of ripple-fired mining explosions is developed in the time domain and the Bayesian approach, using the Gibbs sampler is applied in the deconvolution problem. In Chapter 3, this method is first tested using simulated data contrived to contain spectral properties similar to real ripple-fired seismic events. Then in Chapter 4, the method is applied to array data from mining explosions recorded by the Arctic Experimental Seismic Station (ARCESS) in northern Norway, that has been previously investigated by Shumway, Baumgardt, and Der [30]. In Chapter 5, we provide conclusions and recommendations for further work.

An example of a typical ripple-fired mining explosion is shown in Figure 1.2. The top panel shows a full seismic recording from one channel of Event 054 sampled at 40 points per second. The four arrival phases are marked, from left to right. The arrival phases are divided into the body waves and surface waves. The body waves include the P_n and P_g-phases, which are compressional waves, and the S_n-phases, which is a shear wave. As described earlier, the P_n-phase travels through the crust and across the top of the mantle. P_g is a wave that usually arrives later than P_n and it travels entirely within the Earth's crust. S_n is a shear wave that travels a path similar to P_n. The S_n-phase is latter arriving because shear waves travel slower than compressional waves. The surface waves include the L_g-phase, which is a shear wave that mostly travels within the Earth's crust. L_g is an important regional wave because it is often the largest wave observed at regional distances. The L_g-phase recorded in the ARCESS data may also include the S_g-phase, which is a shear wave that travels slower than the S_n wave [40]. We apply our method to the P_n-phase because it is the most commonly analyzed phase for seismic discrimination, as it does not contain possible reflections of previously arriving phases. For verification purposes, since the actual delays for these data are not known, we apply our method to the P_g, S_n, and L_g-phases to test our results derived from the P_n-phase. We compare the estimated *pulse sequences* from each phase for similarities in the estimated delay times and the number of delayed detonations detected. The results derived from this procedure are then compared with the results of other methods presented in the literature.

In the next section of this chapter, we begin by developing a one-channel convolution model of a ripple-fire event and a simple contrived example illustrating the deconvolution problem, along with a description of the deconvolution we perform.

1.6 A Simple Contrived Example

To introduce a model for a ripple-fired event, as well as to illustrate what is involved in the deconvolution procedure, we present a simple example using contrived data. This example assumes that seismic activity is monitored by one seismometer, recording seismic waves at 40 points per second. Also, suppose that, unknown to us, the data are generated as a ripple-fired mining explosion containing two delayed detonations, each successively smaller in amplitude than the previous one, with a total duration of 0.5 seconds or 20 points in our data. The example therefore contains information related to the delays used in the blasting design. The first delayed explosion is 75%

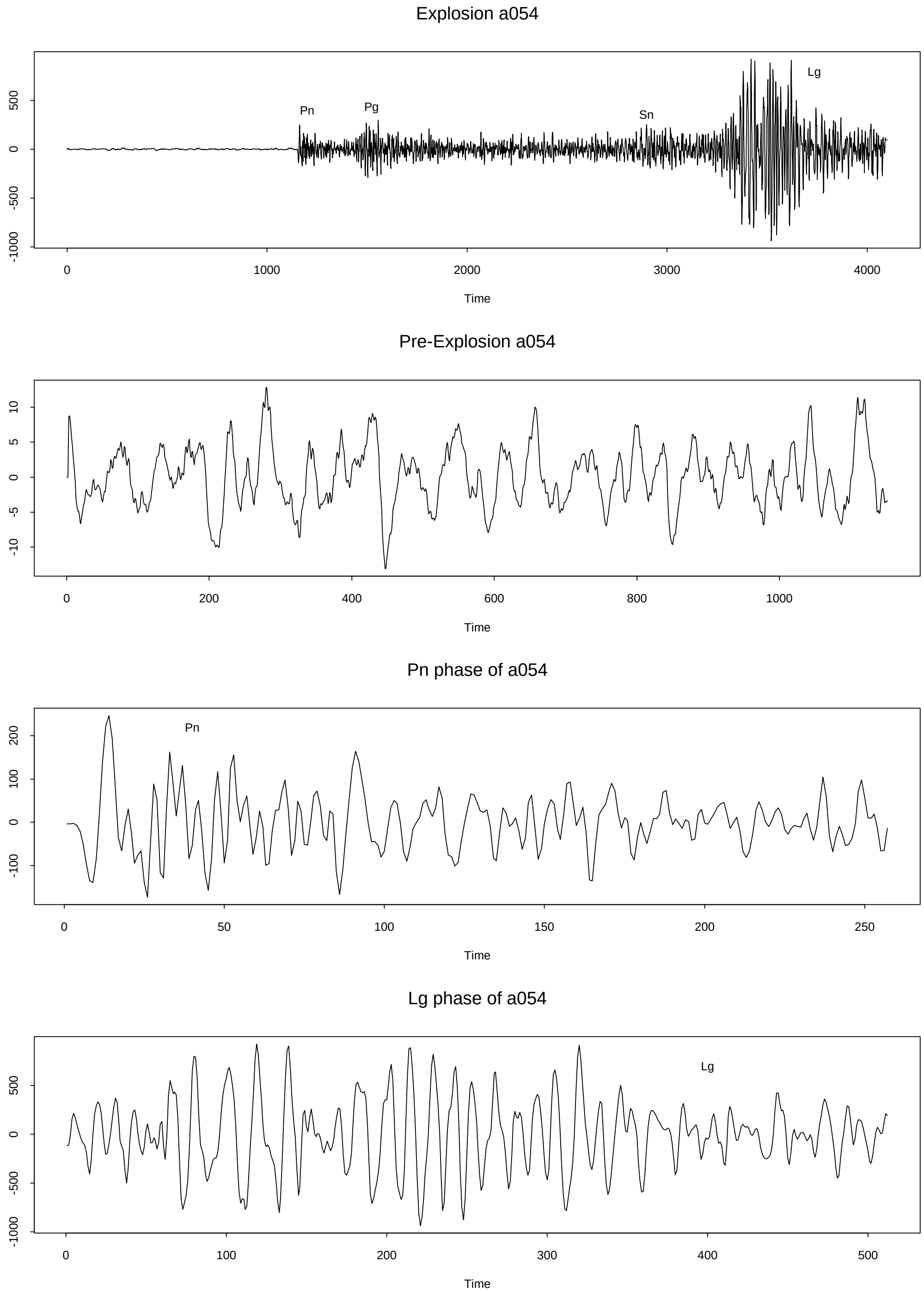


Figure 1.2: First panel (top): A seismogram of Event 054, a known ripple-fired mining explosion. Second panel: Enlargement of the pre-P_n noise. Third panel: Enlargement of the P_n-phase. Fourth panel: Enlargement of the L_g-phase.

of the initial explosion at a delay of 0.25 seconds or 10 points. the second delayed explosion is 50% of the initial explosion at a delay of 0.5 seconds or 20 points. This delay pattern is demonstrated in Figure 1.3. For modeling purposes, it is assumed that each detonation in the ripple-fired events sends the same signal.

The delay structure described above has been modeled in the past using a time domain convolution model such as

$$\begin{aligned} y(t) &= \sum_{j=0}^{m} a_j s(t-j) + \varepsilon(t) \qquad (1.1) \\ &= a(t) * s(t) + \varepsilon(t). \end{aligned}$$

In this model, $y(t)$, $t = 1, ..., n$, is the seismic trace recorded by our seismometer, n is the total number of points recorded, m is the suspected duration of the ripple-fired event, a_j, $j = 0, 1, ..., m$, are the amplitudes at time j for the event (most of the a_j will be zero with the nonzero values occurring at the times of the delayed sub-explosions); $s(t)$, $t = 1, ..., n$, is the underlying signal that is assumed to be the same for each detonation; and $\varepsilon(t)$, $t = 1, ..., n$, is the error term which is assumed to be independent and identically distributed with a $N(0, \sigma^2)$ distribution (* denotes the convolution operator) [23], [7], [24], [30], [34], [35], and [36].

For the contrived example, "Truth" is: $m = 20$, $a_0 = 1$, $a_{10} = 0.75$, $a_{20} = 0.50$, and $\varepsilon(t)$ is distributed *i.i.d.* $N(0, \sigma^2)$, where $\sigma^2 = 1$. For illustrative purposes the *underlying signal* is assumed to be the deterministic signal

$$s(t) = t \exp(-bt) \sin(2\pi f t),$$

where $b = 0.1$, the period is 8 points, $f = 1/8$, and $n = 128$. Therefore, the true

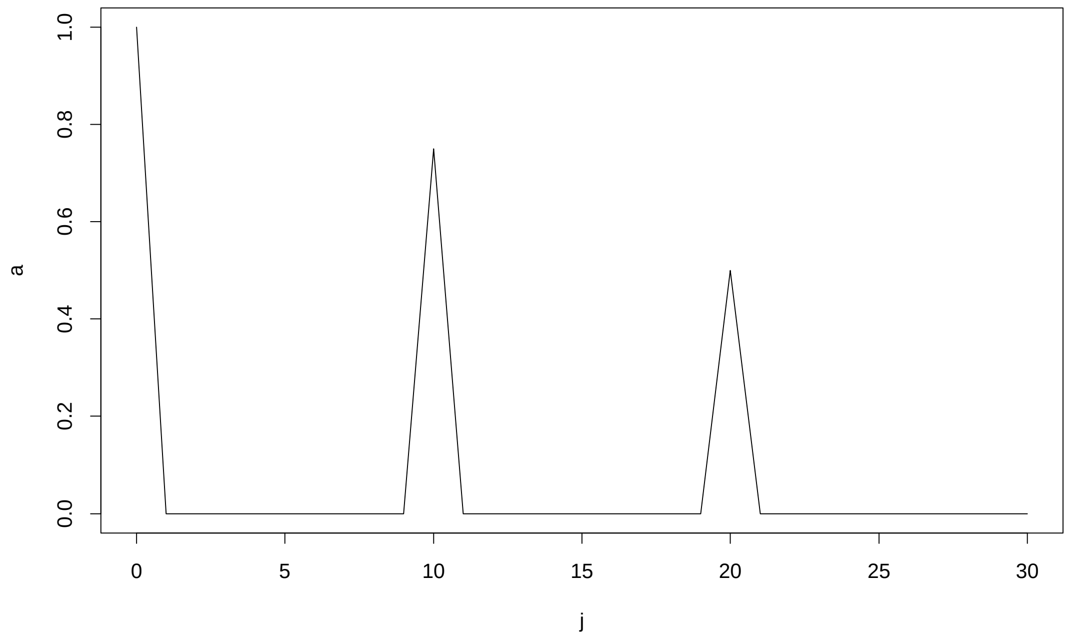


Figure 1.3: Hypothetical pulse sequence of a ripple-fired seismic event.

model is

$$y(t) = s(t) + 0.75s(t-10) + 0.50s(t-20) + \varepsilon(t).$$

The convolution is illustrated in Figure 1.4. The top panel represents the signal sent by the first detonation in the ripple-fired event which is assumed to have amplitude $a_0 = 1$. The second panel shows the 10-point delayed signal sent by the second detonation of amplitude $a_{10} = 0.75$. The third panel has the last signal sent, which has amplitude $a_{20} = 0.50$. The fourth panel contains the convolved data

$$y(t) = a(t) * s(t),$$

and the bottom panel contains $y(t)$ from the convolution 1.1 with the additive error $\varepsilon(t)$ included, where $\sigma^2 = 1$.

The deconvolution problem studied in this dissertation can briefly be described as follows: If a seismic trace is received that is suspected to have been generated by a ripple-fired mining explosion we begin with the bottom panel of Figure 1.4. Our goal is to reproduce the signal $s(t)$ in the first panel as well as the delay structure contained in the a_j, $j = 1, ..., m$, which are given in Figure 1.3. For there to be identifiability between the a_j and $s(t)$, a_0 is fixed to be 1, and the rest of the a_j, $j = 1, ..., m$, are estimated relative to a_0. The problem just described is the univariate version of the multivariate deconvolution problem studied in this dissertation.

The most important of several extensions is the treatment of seismic array data, which are data collected for one event by many seismometer. This is an important extension because the multivariate data will produce better estimates of the parameters related to the *pulse sequence* since all sensors in the array are assumed to have observed the same set of delay times and the same *underlying signal-path effects sequence*.

In the Chapter 3 we present the mathematics needed to perform the Bayesian deconvolution on seismic array data for common delay patterns.

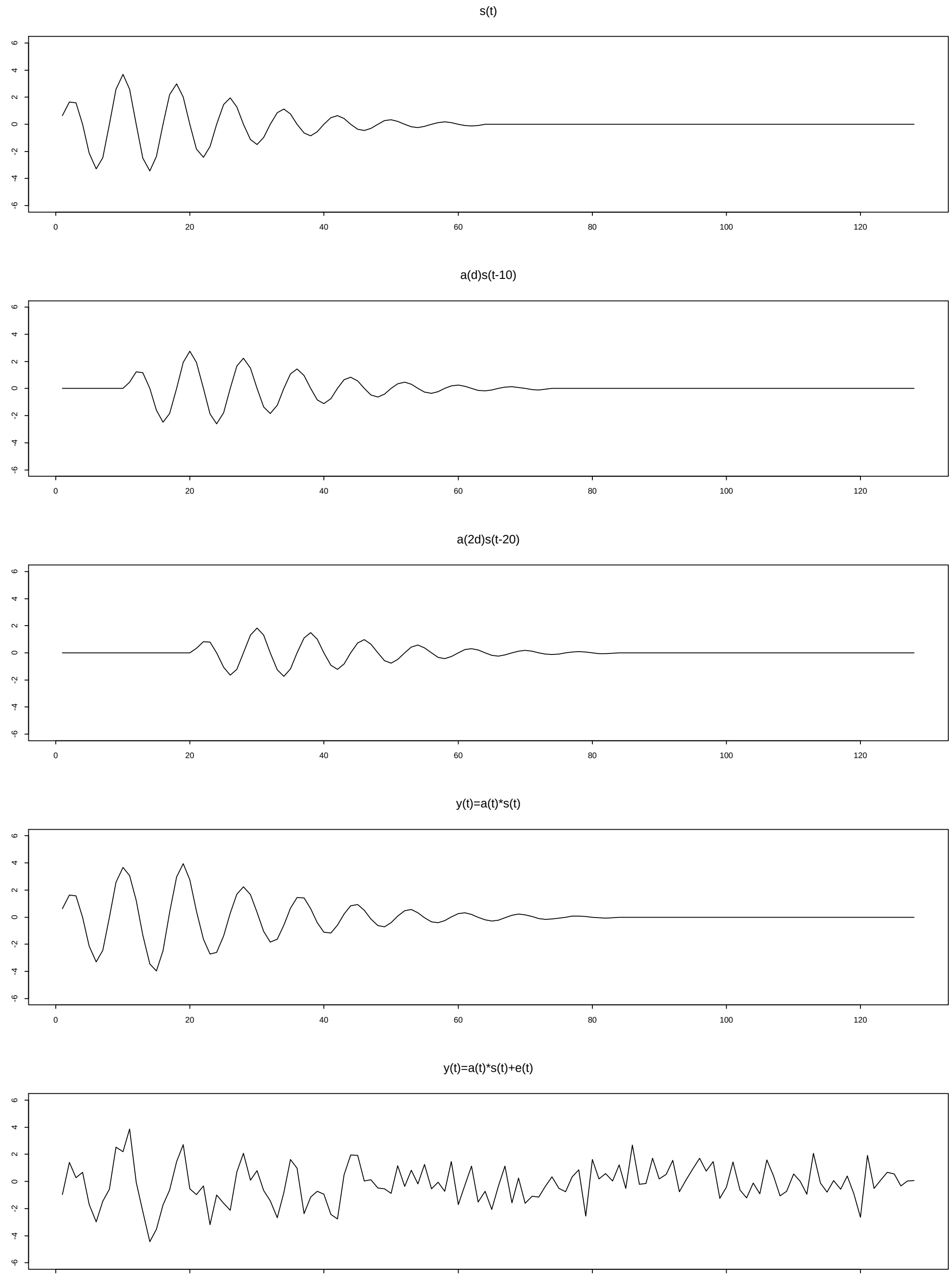


Figure 1.4: A convolution example.

Chapter 2

Bayesian Deconvolution for Common Delay Patterns

In this chapter we give the details of our Bayesian deconvolution method for seismic array data. First, we present the multidimensional version of the convolution model introduced in Chapter 1. Second, we review Bayesian methodology and describe the Gibbs sampler. Third, we describe the prior distributions chosen and the model induced distributions. Next, we produce the conditional marginal posterior distributions needed to implement the Gibbs sampler. Finally, we give an algorithm for running the Gibbs sampler to estimate the parameters in $\mathbf{\Theta}$. Chapters 3 and 4 present results for simulated and real data, respectively.

2.1 The Model

In Chapter 1 we stated a seismic monitoring problem and a relevant model of ripple-fired mining explosions. Here we develop our model for data collected by an array of q-seismometers at regional distances, i.e., within 2000 kilometers of the seismic event. We use regional data because at shorter distances more of the high frequencies produced by the event in question are retained in the seismic traces collected by each seismometer in the array. Regional data is commonly recorded for monitoring purposes and is believed to be necessary for monitoring a Comprehensive Test Ban Treaty (CTBT) [9], [40].

When seismic activity is recorded for monitoring nuclear explosions, it is necessary to have methods capable of identifying common seismic activity. Having methods capable of accurately identifying common seismic activity can reduce the false-alarm rates in the monitoring process. Here we focus on industrial explosions, specifically ripple-fired mining explosions. We model the k^{th} seismic trace $y_k(t)$, $t = 1, ..., n$, recorded by the k^{th} seismometer in an array of q-sensors, that is suspected to have been produced by a ripple-fired mining explosion, using the following multivariate extension of the convolution model presented in Chapter 1. Equation (1.1) can be extended to array data by modeling $y_k(t)$ as

$$y_k(t) = \sum_{j=0}^{m} a_j s_k(t-j) + \varepsilon_k(t). \tag{2.1}$$

We refer to the vector of parameters $\mathbf{a} = [a_1, ..., a_m]'$ as the relative *pulse sequence*. This set of parameters is assumed to reflect the delay pattern used in the design of the ripple-fired event. We also assume that the same set of parameters can be used to model the delay structure in each seismic trace $y_k(t)$ recorded by the array. We define the k^{th} *underlying signal-path effects sequence*, $s_k(t)$, $t = 1, ..., n$, as the combination of the "signal" produced by each sub-explosion in a ripple-fired event along with the "path effects" that are due to the random imperfections in the material the signal travels through on the k^{th} channel to the k^{th} seismometer. We assume that the *pulse sequence* $\mathbf{a}$ is independent of the *underlying signal-path effects sequence* $s_k(t)$, $t = 1, ..., n$, on each channel k, $k = 1, .., q$. This independence follows from a reasonable assumption that the design of the delay pattern used to conduct a ripple-fired mining explosion does not influence the signal produced by each row of explosives. Assuming that the first explosion in the ripple-fired event is used as the reference signal, for which we fix a_0 to be 1, and that the remaining a_j are calculated relative to a_0, thus our model can be written in a more insightful form as

$$y_k(t) = s_k(t) + \sum_{j=1}^{m} a_j s_k(t-j) + \varepsilon_k(t). \tag{2.2}$$

This model assumes that each delayed explosion in a ripple-fired event sends a signal having the same form as the first. The only difference is that its amplitudes a_j may vary. In the univariate case this model has been used by Smith [34] [35] and physically verified by Strump and Reinke [36]. In the multivariate case it has been used in this form by Shumway, Baumgart, and Der [30].

Further modeling assumptions on the *pulse sequence* **a**, and the *underlying signal-path effects sequence* $s_k(t)$, for the k^{th} channel are needed to insure that the parameters in the convolution model are identifiable. To model the pulse sequence we use a variant of the Bernoulli-Gaussian model commonly used to model reflection seismology data. A discussion of the model can be found in Mendel [24]. For the *pulse sequence* **a**, we fix $a_0 = 1$ and model the relative amplitudes as follows: We define X_j, $j = 1, ..., m$, to be a sequence of independent Bernoulli random variables with

$$p(X_j = 1) = 1 - \eta \tag{2.3}$$

and

$$p(X_j = 0) = \eta. \tag{2.4}$$

The nonzero values of X_j index the possible detonation times in the ripple-fired event of duration m, measured in points per second. The parameter η is the probability of observing a zero value and $1 - \eta$ is the probability of observing a nonzero value in the *pulse sequence*. The parameter m is the length of the vector **a**, m is assumed to be known. Next, we define the distribution of each a_j, conditional on X_j, $j = 1, ..., m$.

In ripple-fired events the amplitudes will be positive because mining explosions are usually performed at shallow depths and therefore, the individual detonations will not have delayed reflections off of the surface of the earth, which would produce negative amplitudes. We define a_j, $j = 1, ..., m$, conditional on X_j, $j = 1, ..., m$, as

$$\begin{aligned} p(a_j|X_j = 0) &= 1 \qquad a_j = 0 \\ &= 0 \qquad a_j \neq 0. \end{aligned} \tag{2.5}$$

and

$$p(a_j|X_j = 1) = c^{-1}(\mu_\alpha, \sigma_\alpha^2)\frac{1}{\sqrt{2\pi}\sigma_\alpha}\exp\left\{-\frac{(a_j - \mu_\alpha)^2}{\sigma_\alpha^2}\right\} I(a_j > 0), \tag{2.6}$$

where

$$c^{-1}(\mu_\alpha, \sigma_\alpha^2) = \int_0^\infty \frac{1}{\sqrt{2\pi}\sigma_\alpha}\exp\left\{-\frac{(x - \mu_\alpha)^2}{\sigma_\alpha^2}\right\} dx. \tag{2.7}$$

and

$$\begin{aligned} I(a_j > 0) &= 0 \qquad a_j \leq 0 \\ &= 1 \qquad a_j > 0. \end{aligned}$$

The truncated normal distribution is used because the a_j, the relative amplitude of the delayed detonation at time j, is assumed to be positive and random. Truncating at zero restricts the values of a_j to positive values.

It is reasonable to use the normal distribution to model the variation in the size of a single detonation since the distribution of the size of a single explosion should be centered at a mean value μ_α with a variance σ_α^2. Alternatively, we write the density of the independent a_j, $j = 1, ..., m$, as

$$p(a_j|\eta) = \eta I(a_j = 0) + (1 - \eta)p(a_j|X_j = 1)I(a_j > 0), \tag{2.8}$$

which is a mixture of a Bernoulli distribution and a truncated Gaussian distribution. The mean μ_α is used to model the average size of the delayed explosions in ripple-fired events in the monitoring region and the variance σ_α^2 is used to model the variance of such explosions. So the a_j are independent and identically distributed with a density defined in (2.8). Note that for the Bernoulli sequence X_j, $j = 1, ..., m$, will have n_α 0 values and $(m - n_\alpha)$ unit values, which implies that the *pulse sequence* has $(m - n_\alpha)$ nonzero values or pulses. Note that n_α is random. Define J_α to be the index set that includes the values of j for nonzero values of X_j or thereby the nonzero values of a_j. Hence, we define the density of $\mathbf{a} = [a_1, ..., a_m]'$ as

$$
\begin{aligned}
p(\mathbf{a}|\eta) &= \prod_{j=1}^{m} p(a_j|\eta) \\
&= \prod_{j=1}^{m} [\eta I(a_j = 0) + \\
&\qquad (1-\eta) c^{-1}(\mu_\alpha, \sigma_\alpha^2) \frac{1}{\sqrt{2\pi}\sigma_\alpha} \exp\left\{ -\frac{(a_j - \mu_\alpha)^2}{2\sigma^2} \right\} I(a_j > 0)] \\
&= \eta^{n_\alpha} (1-\eta)^{m-n_\alpha} \prod_{J_\alpha} c^{-1}(\mu_\alpha, \sigma_\alpha^2) \frac{1}{\sqrt{2\pi}\sigma_\alpha} \exp\left\{ -\frac{(a_j - \mu_\alpha)^2}{\sigma_\alpha^2} \right\} \\
&= \eta^{n_\alpha} (1-\eta)^{m-n_\alpha} c^{-n_\alpha}(\mu_\alpha, \sigma_\alpha^2) \left[\frac{1}{\sqrt{2\pi}\sigma_\alpha} \right]^{n_\alpha} \\
&\qquad \times \exp\left\{ -\frac{1}{2\sigma_\alpha^2} \sum_{J_\alpha} (a_j - \mu_\alpha)^2 \right\}. \qquad (2.9)
\end{aligned}
$$

This modeling assumption is based on the Bernoulli-Gaussian model used by Cheng, Chen, and Li [7].

To model the *underlying signal-path effects sequence* $s_k(t)$, in the k^{th} seismic trace $y_k(t)$, $t = 1, ..., n$, we use a low-order autoregressive, $AR(p)$, model

$$\phi_0 s_k(t) + \phi_1 s_k(t-1) + \phi_2 s_k(t-2) + ... + \phi_p s_k(t-p) = e_k(t), \qquad (2.10)$$

where we fix $\phi_0 = 1$ and let $e_k(t)$ be independent and identically distributed, over

t and k, normal with mean 0 and variance σ^2. The use of this model is justified by Dargahi-Noubary [8] and Tjøstheim [38] [39] who gave theoretical and empirical arguments for modeling seismic waveforms using low-order autoregressive models. The *precision* of $s_k(t)$ is defined as $\tau = 1/\sigma^2$, which is used later in our application of the Gibbs sampler. We use the same set of autoregressive coefficients, $\boldsymbol{\phi} = [\phi_1, ..., \phi_p]'$, to model the k^{th} *underlying signal-path effects sequence* $s_k(t)$, since the same event is assumed to generate the trace $y_k(t)$ recorded at the k^{th} seismometer. Hence, we believe the same autoregressive model will be applicable for each *underlying signal-path effect sequence* $s_k(t)$, $k = 1, ..., q$, $t = 1, ..., n$, although the functions generated will be different.

Finally, we assume that the additive observation *noise* term $\varepsilon_k(t)$ in (2.2) is distributed independent and identically, over t and k, normal with mean 0 and variance $c\sigma^2$, where $c > 0$ is fixed. The parameter c can be thought of as the inverse of the signal-to-noise ratio, defined as $SNR = \sigma_e^2/\sigma_\varepsilon^2$, where σ_e^2 is the variance of the *underlying signal-path effects sequence* on each channel and σ_ε^2 is the variance of the *noise* term. Recall that $\sigma_e^2 = \sigma^2$ and $\sigma_\varepsilon^2 = c\sigma^2$, which implies that $SNR = 1/c$. The use of the fixed parameter c alleviates a scaling problem that exists if two separate variance terms are included in our model.

2.2 Bayesian Methodology

In this dissertation the deconvolution of seismic array data for common delay patterns is achieved using Bayesian methods. We believe the Bayesian approach is well suited to this problem for two reasons. First, the parameters in our convolution model all have physical interpretations. This makes the choice of prior distributions and the corresponding choices of hyperparameters possible. The second is that classical approaches to similar deconvolution problems are very difficult due to the fact that deconvolution problems are *underdetermined*, i.e., having more parameters than data points available. Specifically in the model under consideration in (2.1), the parameter set is

$$\begin{aligned} \mathbf{\Theta} &= \{\eta, \mathbf{a}, \boldsymbol{\phi}, \tau, \mathbf{S}\} \qquad (2.11) \\ &= \{\eta, a_1, ..., a_m, \phi_1, ..., \phi_p, \tau, \mathbf{s}_1, ..., \mathbf{s}_q\}. \end{aligned}$$

The set $\mathbf{\Theta}$ contains $1 + m + p + 1 + nq$ elements, where we define

$$\mathbf{a} = [a_1, a_2, ..., a_m]', \qquad (2.12)$$

$$\boldsymbol{\phi} = [\phi_1, \phi_2, ..., \phi_p]', \qquad (2.13)$$

and

$$\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_q]', \qquad (2.14)$$

such that each row of $\mathbf{S}$ is

$$\mathbf{s}_k = [s_k(1), s_k(2), ..., s_k(n)]'. \qquad (2.15)$$

The given data consists of the collection of q seismic traces recorded at q-seismometers for a duration of n points. Here we define

$$\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_q]', \qquad (2.16)$$

such that each row of $\mathbf{Y}$ is

$$\mathbf{y}_k = [y_k(1), y_k(2), ..., y_k(n)]'. \qquad (2.17)$$

Hence in this problem only qn data points are available to estimate all of the parameters in $\mathbf{\Theta}$. Having fewer data points than model parameters makes maximum likelihood estimation difficult. However, using the Bayesian approach and the Gibbs sampler is much less problematic. All of the necessary conditional marginal posterior distributions can be calculated.

In the one-dimensional problem, where seismic data are recorded by one seismometer (or when beamforming is used) the classical approaches that have been considered appear to be very difficult to extend to array data. In the time domain Kormolo and Mendel [23] develop a maximum likelihood deconvolution method. In the frequency domain Hannan and Thomson [18] consider a Maximum Likelihood Estimation (MLE) approach to deconvolving seismic data for a single delay. Neither of these approaches seem ease to generalize to array data although, Shumway and Der [31] derive a frequency domain multiple deconvolution method for multiple events and arrays. Minimum entropy approaches to deconvolution have been considered by Donoho [10] and Wiggins [42] and their methods do not appear applicable in the deconvolution problem we are considering. In contrast the use of the Gibbs sampler by Cheng, Chen, and Li [7] and Cheng and Li [6], in the univariate case, has been the main guide in our Bayesian approach to deconvolving for common delay patterns in seismic array data.

To develop our Bayesian deconvolution method we follow the Bayesian approach to statistical data analysis, which proceeds as follows: First, we develop a model and likelihood

$$p(\mathbf{Y}|\mathbf{\Theta}). \tag{2.18}$$

Second, prior distributions for the parameters are chosen to model the prior uncertainty in individual or groups of parameters in the parameter set $\mathbf{\Theta}$. The overall prior is

$$p(\mathbf{\Theta}). \tag{2.19}$$

And third, the posterior distribution of the parameters in $\mathbf{\Theta}$, given the data $\mathbf{Y}$, is calculated. The posterior distribution is

$$p(\mathbf{\Theta}|\mathbf{Y}), \tag{2.20}$$

which represents the gain in knowledge regarding the parameters, given the data $\mathbf{Y}$. By using Bayes' Rule the posterior distributions can be calculated as

$$\begin{aligned} p(\mathbf{\Theta}|\mathbf{Y}) &= \frac{p(\mathbf{Y}|\mathbf{\Theta})p(\mathbf{\Theta})}{\int p(\mathbf{Y}|\mathbf{\Theta})p(\mathbf{\Theta})d\mathbf{\Theta}} \\ &\propto p(\mathbf{Y}|\mathbf{\Theta})p(\mathbf{\Theta}). \end{aligned} \tag{2.21}$$

Then, to draw inferences about the unknown parameters in our model, we calculate the marginal posteriors. For point estimates we calculate the posterior means, medians, or modes. Finally, for interval estimates we calculate Bayesian posterior intervals.

In our deconvolution problem, $\mathbf{\Theta}$ is a vector of many parameters, and due to our modeling assumptions the marginal posterior distributions are difficult to calculate. However, the conditional marginal distributions are available. From the latter we can implement the Gibbs sampler, a Markov Chain Monte Carlo technique that simulates "random samples" from the conditional marginal distributions, which can be used to calculate posterior estimates of the parameters in our model.

The rest of this chapter is outlined as follows: First in Section 2.3 we describe the Gibbs sampler. Then we proceed, in Section 2.4, to define the parametric forms used as priors for the parameters in our model. In Section 2.5 we define the likelihood function and specify the joint distribution of the data $\mathbf{Y}$ and the set of model parameters $\mathbf{\Theta}$, which are used in Section 2.6 to determine the conditional marginal posterior distributions used in the Gibbs sampler. Last, in Section 2.7 we list the steps used to run the Gibbs sampler for our seismic deconvolution.

2.3 The Gibbs Sampler

The Gibbs sampler is a flexible estimation technique based on sampling. It was first applied by Geman and Geman [15] in image processing and later applied to statistical problems by Gelfand and Smith [11], and Gelfand, Hill, Racine-Poon, and Smith [12] in Bayesian data analysis problems. Other references are Tanner [37], Gelman and Rubin [13], Geyer [14], and Gilk, Richardson, and Spiegelhalter [16]. Some basic introductory papers on the subject are Casella [5] and for a roundtable discussion on the practical use of Markov Chain Monte Carlo methods, see [20]. Gibbs sampling is useful in Bayesian problems where the joint density is complex and the analytic computation of the marginal posterior densities are difficult or impossible. The idea is to break the problem down into a problem of sampling from conditional marginal density functions, from which random samples can easily be generated.

The goal of Gibbs sampling is to generate random samples from conditional marginal posterior distributions. Here by random samples we refer to the Markov chains produced by sampling iteratively from the conditional marginal posterior distributions. It has been proven that these random samples converge to random samples from the joint distribution of interest [11]. The Gibbs sampler can be described as follows. Suppose the parameter set consists of d components, $\mathbf{\Theta} = (\theta_1, ..., \theta_d)'$, where θ_i, $i = 1, ..., d$, may be a scalar or vector parameter. Given a starting point $\mathbf{\Theta}^0 = (\theta_1^{(0)}, ..., \theta_d^{(0)})'$ and the data $\mathbf{Y}$, the algorithm used to produce Markov chains for each parameter θ_i proceeds as follows: At iteration h

1. Sample $\theta_1^{(h+1)}$ from $p(\theta_1|\theta_2^{(h)}, ..., \theta_d^{(h)}, \mathbf{Y})$.

2. Sample $\theta_2^{(h+1)}$ from $p(\theta_2|\theta_1^{(h+1)}, \theta_3^{(h)}..., \theta_d^{(h)}, \mathbf{Y})$.

3. ...

4. Sample $\theta_d^{(h+1)}$ from $p(\theta_d|\theta_1^{(h+1)}, ..., \theta_{d-1}^{(h+1)}, \mathbf{Y})$.

5. set $h = h + 1$ and go to 1.

The sequence $\mathbf{\Theta}^{(0)}, \mathbf{\Theta}^{(1)}, ..., \mathbf{\Theta}^{(t)}, ...$ forms a realization of a Markov chain with transition probability from $\mathbf{\Theta}'$ to $\mathbf{\Theta}$, defined by

$$\begin{aligned} T(\mathbf{\Theta}', \mathbf{\Theta}) &= p(\theta_1|\theta_2', ..., \theta_d', \mathbf{Y}) \\ &\quad \times p(\theta_2|\theta_1', \theta_2', ..., \theta_d', \mathbf{Y}) \times \\ &\quad ... \times p(\theta_d|\theta_1', ..., \theta_{d-1}', \mathbf{Y}). \end{aligned}$$

After an initial "burn in" period the Markov chains are considered random samples from the posterior joint density. These sample points are then used to draw inferences about the parameters. We calculate sample means or medians to estimate the parameters. Density plots of each sample can be used to show the estimated posterior distributions of the individual parameters.

2.4 Prior Distributions

To perform our Bayesian analysis we choose prior distributions for each set of unknown parameters in the parameter set $\mathbf{\Theta}$. We specify priors on η, $\boldsymbol{\phi}$, and τ since they are unknown. We have modeling assumptions that determine the distributions of the random parameters $\mathbf{a}$ and $\mathbf{S}$, so we do not need to choose priors for these parameters; their distributions are specified by our model. Note once again that we fix the signal-to-noise ratio in our model, so no prior is selected on the vector $\mathbf{c}$. We use conjugate priors on our unknown parameters for two reasons. First, we believe our prior knowledge about the unknown parameters can be accurately modeled using such distributions. Second, they facilitate the calculation of the conditional marginal posterior distributions for use in the Gibbs sampler.

The probability η of seeing a zero at any point in the *pulse sequence* a_j, $j = 1, ..., m$, is assumed to have a beta prior, where

$$p(\eta) = \frac{\Gamma(\beta_1 + \beta_2)}{\Gamma(\beta_1)\Gamma(\beta_2)} \eta^{\beta_1 - 1} (1 - \eta)^{\beta_2 - 1}, 0 \leq \eta \leq 1, \tag{2.22}$$

and is zero elsewhere, where $\beta_1 > 0$ and $\beta_2 > 0$ are predetermined hyperparameters. A beta prior is a natural choice as the prior on a probability because it is a flexible distribution that models values between zero and one and can take many different shapes by changing the parameters β_1 and β_2. The hyperparameters β_1 and β_2 can easily be chosen from the expert opinion of mining practitioners about η, the probability of an explosion in a ripple-fired mining explosion at a delay time j. Information can be elicited regarding the mode of the distribution of possible values of η and say the 5^{th} and 95^{th} percentiles. This information should be attainable and will lead to values of the hyperparameters β_1 and β_2. See Figure 2.1 for an example of a possible beta prior on η, with a mode of 0.9.

For the $AR(p)$ coefficients $\boldsymbol{\phi} = [\phi_1, \phi_2, ..., \phi_p]'$, we assume a jointly normal prior distribution of the form

$$p(\boldsymbol{\phi}) = (2\pi)^{-\frac{p}{2}} |\Sigma_0|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\boldsymbol{\phi} - \boldsymbol{\phi}_0)' \Sigma_0^{-1} (\boldsymbol{\phi} - \boldsymbol{\phi}_0)\right\}, \tag{2.23}$$

where $\boldsymbol{\phi}_0$ and Σ_0 are predetermined hyperparameters. The hyperparameters $\boldsymbol{\phi}_0$ and Σ_0 can be chosen from expert opinion or from the literature regarding the mean value of the vector of AR coefficients and their covariances. In our Bayesian analysis we choose our hyperparameters from past research conducted by Tjøstheim [39] related to fitting $AR(p)$ models to single explosion data.

To specify the prior for the common precision of the *underlying signal-path effects*, for each channel k, we assume that τ has a gamma prior of the form

$$p(\tau) = \frac{\gamma_2^{\gamma_1}}{\Gamma(\gamma_1)} \tau^{\gamma_1 - 1} \exp\left\{-\gamma_2 \tau_k\right\}, 0 \leq \tau \leq \infty, \tag{2.24}$$

and zero otherwise, where $\gamma_1 > 0$ and $\gamma_2 > 0$ are predetermined hyperparameters. The use of the gamma prior is conventional for precision parameters. The parameter τ is positive and the gamma density can model all possible values of τ. Since τ is an unobservable parameter in our model, we choose the hyperparameters to reflect, to the best of our knowledge, the likely values of τ. See Figure 2.1 for an example of such a possible gamma prior on τ, with its mode at 1.

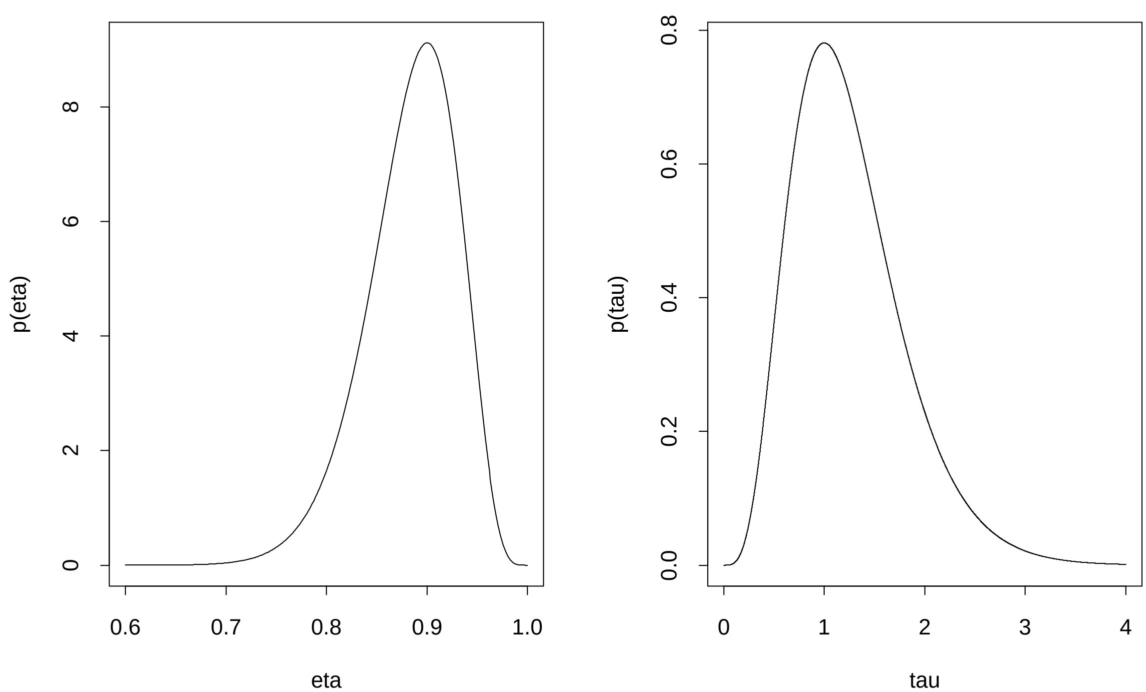


Figure 2.1: $\eta \sim$ Beta(43,5.67) and $\tau \sim$ Gamma(5,4).

The *pulse sequence* $\mathbf{a} = [a_1, ..., a_m]'$ has a Bernoulli-Gaussian distribution because of our specifications. The distribution of $\mathbf{a}$, conditional on η, has the density function given in (2.9), where μ_α and σ_α^2 are the predetermined mean and variance of the nonzero truncated normal distributed values in the Bernoulli-Gaussian distribution and where we assume the amplitudes to be independent. The parameters μ_α and σ_α^2 can be set by estimating the mean and variance of the relative amplitude of the delayed explosions in the ripple-fired mining explosions used in the region that is being monitored.

The distribution of the *underlying signal-path effects* matrix $\mathbf{S}$ is determined by the $AR(p)$ model assumed for $\mathbf{s}_k$ on each channel. To begin let

$$\begin{aligned} \mathbf{s}_k &= [s_k(1), ..., s_k(n)]' \\ &= \begin{bmatrix} \mathbf{s}_k^{(1)} \\ \mathbf{s}_k^{(2)} \end{bmatrix}, \end{aligned} \tag{2.25}$$

where

$$\mathbf{s}_k^{(1)} = [s_k(1), s_k(2), ..., s_k(p)]' \tag{2.26}$$

and

$$\mathbf{s}_k^{(2)} = [s_k(p+1), ..., s_k(n)]'. \tag{2.27}$$

Hence

$$\mathbf{S} = [\mathbf{s}_1, ..., \mathbf{s}_q]' \tag{2.28}$$

$$= [\mathbf{S}^{(1)}, \mathbf{S}^{(2)}], \tag{2.29}$$

where

$$\mathbf{S}^{(1)} = [\mathbf{s}_1^{(1)}, ..., \mathbf{s}_q^{(1)}]' \tag{2.30}$$

and

$$\mathbf{S}^{(2)} = [\mathbf{s}_1^{(2)}, ..., \mathbf{s}_q^{(2)}]'. \tag{2.31}$$

This partitioning is needed since each $s_k(t)$, $t = (p+1), ..., n$, depends on the previous p values, $s_k(t-1), ..., s_k(t-p)$, as can be seen from the definition of the AR model given in (2.10). So for channel k, the conditional distribution of $s_k(t)$, $t = (p+1), ..., n$, is normal because of the normal error term in the $AR(p)$ model used for $s_k(t)$. Then we have

$$p(\mathbf{s}_k^{(2)}|\boldsymbol{\phi}, \tau, \mathbf{s}_k^{(1)}) \propto (\tau)^{(\frac{n-p}{2})} \exp\left\{-\frac{\tau}{2}\sum_{t=p+1}^{n} e_k^2(t)\right\}, \tag{2.32}$$

where

$$\begin{aligned} e_k(t) &= \sum_{i=0}^{p} \phi_i s_k(t-i) \\ &= s_k(t) - \sum_{i=1}^{p} \phi_i s_k(t-i), \end{aligned} \tag{2.33}$$

since $\phi_0 = 1$. The fact that the conditional distribution is proportional to the right hand side of (2.32) follows from a transformation from $\mathbf{s}_k^{(2)}$ to $e_k(p+1), ..., e_k(n)$, where the Jacobian is 1 [29]. We let $\mathbf{s}_k^{(1)}$ have a joint normal distribution

$$p(\mathbf{s}_k^{(1)}|\boldsymbol{\phi}, \tau) \propto N(0, \Sigma_\phi), \tag{2.34}$$

where Σ_ϕ is a function of $\boldsymbol{\phi}$ and τ, depending on the Yule-Walker equations. Therefore, for channel k, the distribution of $\mathbf{s}_k$ is

$$p(\mathbf{s}_k|\boldsymbol{\phi}, \tau) \propto p(\mathbf{s}_k^{(2)}|\boldsymbol{\phi}, \tau, s_k^{(1)})p(\mathbf{s}_k^{(1)}|\boldsymbol{\phi}, \tau). \tag{2.35}$$

For the matrix $\mathbf{S}$

$$p(\mathbf{S}|\boldsymbol{\phi}, \tau) = \prod_{k=1}^{q} p(\mathbf{s}_k|\boldsymbol{\phi}, \tau) \tag{2.36}$$

$$= \prod_{k=1}^{q} p(s_k^{(2)}|\boldsymbol{\phi},\tau,\mathbf{s}_k^{(1)})p(\mathbf{s}_k^{(1)}|\boldsymbol{\phi},\tau) \quad (2.37)$$

$$\propto \prod_{k=1}^{q}(\tau)^{\left(\frac{n-p}{2}\right)} \exp\left\{-\frac{\tau}{2}\sum_{t=p+1}^{n} e_k^2(t)\right\}$$
$$\prod_{k=1}^{q}|\Sigma_\phi|^{-1}\exp\left\{-\frac{1}{2}\mathbf{s}_k^{(1)\prime}\Sigma_\phi^{-1}\mathbf{s}_k^{(1)}\right\}. \quad (2.38)$$

Following the motivating work of Dargahi-Noubary [8] and Tjøstheim [38], we assume that $p = 3$ in our application of our model to real data. We give the form of Σ_ϕ now for completeness, even though it will not be used until later. We model the *underlying signal-path effect sequence* on each channel k with an $AR(3)$ model

$$s_k(t) + \phi_1 s_k(t-1) + \phi_2 s_k(t-2) + \phi_3 s_k(t-3) = e_k(t), \quad (2.39)$$

where $e_k(t) \sim N(0,\sigma^2)$ and $\tau = 1/\sigma^2$. From the Yule-Walker equations, the covariance matrix of $\mathbf{s}_k^{(1)}$ is

$$\Sigma_\phi = \frac{1}{\tau}\begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{12} & \sigma_{22} & \sigma_{23} \\ \sigma_{13} & \sigma_{23} & \sigma_{33} \end{bmatrix},$$

where

$$\begin{aligned}
\sigma_{11} &= Var[s_k(1)] = \phi_1^2 + 1 \\
\sigma_{22} &= Var[s_k(2)] = (\phi_1^2 - \phi_2)^2 + \phi_1^2 + 1 \\
\sigma_{33} &= Var[s_k(3)] = (-\phi_1^3 + 2\phi_1\phi_2 - \phi_3)^2 + (\phi_1^2 - \phi_2)^2 + \phi_1^2 + 1 \\
\sigma_{12} &= Cov[s_k(1), s_k(2)] = -\phi_1(\phi_1^2 - \phi_2) - \phi_1 \\
\sigma_{13} &= Cov[s_k(1), s_k(3)] = -\phi_1(-\phi_1^3 + 2\phi_1\phi_2 - \phi_3) + (\phi_1^2 - \phi_2) \\
\sigma_{23} &= Cov[s_k(2), s_k(3)] = (\phi_1^2 - \phi_2)(-\phi_1^3 + 2\phi_1\phi_2 - \phi_3) - \phi_1(\phi_1^2 - \phi_2) - \phi_1.
\end{aligned}$$

2.5 Overall Prior Distribution, Likelihood, Joint and Posterior Distributions

The overall prior distribution on the parameter set $\mathbf{\Theta}$ is defined using the following independence assumptions. The parameter η, the probability of observing a 0 value in the *pulse sequence* at time j, is assumed to be independent of the rest of the parameters. Conditional on η, $\mathbf{a}$ is assumed independent of the rest of the parameters. The vector of AR coefficients $\boldsymbol{\phi}$ and the precision τ, used to model the *underlying signal-path effects sequence* on channel k are assumed to be independent and each is independent of the rest of the parameters. Conditional on $\boldsymbol{\phi}$ and τ, the *underlying signal-path effects sequence* $\mathbf{s}_k$ is assumed to be independent over k and independent of the rest of the parameters, η and $\mathbf{a}$. Incorporating these independence assumptions into the prior distribution of $\mathbf{\Theta}$, we have

$$\begin{aligned} p(\mathbf{\Theta}) &= p(\eta, \mathbf{a}, \boldsymbol{\phi}, \tau, \mathbf{S}) \\ &= p(\eta)p(\mathbf{a}|\eta)p(\boldsymbol{\phi})p(\tau)p(\mathbf{S}|\boldsymbol{\phi}, \tau) \\ &= p(\eta)p(\boldsymbol{\phi})p(\tau)\prod_{j=1}^{m} p(a_j|\eta)\prod_{k=1}^{q} p(\mathbf{s}_k|\boldsymbol{\phi}, \tau). \end{aligned} \tag{2.40}$$

We specify the conditional likelihood and use it as an approximation of the full likelihood. Due to the dependence of $y_k(t)$ on the past m values of $s_k(t)$, which can be seen in (2.1), and the dependence of $s_k(t)$ on its past p values, which can be seen in (2.10), we define the likelihood conditional on $y_k(1), ..., y_k(l),\ k = 1, ..., q$, where $l = m + p$. We now redefine the data matrix as

$$\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_q], \tag{2.41}$$

where each $\mathbf{y}_k$, $k = 1, ..., q$, starts at $l + 1 = (m + p) + 1$, so

$$\mathbf{y}_k = [y_k(l + 1), ..., y_k(n)]'. \tag{2.42}$$

Using the model assumption that the *noise*, $\varepsilon_k(t)$, is independent and identically distributed normal with mean 0 and variance σ^2, the conditional likelihood is

$$\begin{aligned} p(\mathbf{Y}|\boldsymbol{\Theta}) &= \prod_{k=1}^{q} p(\mathbf{y}_k|\boldsymbol{\Theta}) \\ &= \prod_{k=1}^{q} p(\mathbf{y}_k|\eta, \mathbf{a}, \boldsymbol{\phi}, \tau, \mathbf{S}) \\ &= \prod_{k=1}^{q} (2\pi)^{-(n-l)/2} \left(\frac{\tau}{c}\right)^{(n-l)/2} \exp\left\{-\frac{\tau}{2c}\sum_{t=l+1}^{n} \varepsilon_k^2(t)\right\} \\ &= (2\pi)^{-q(n-l)/2} \left(\frac{\tau}{c}\right)^{q(n-l)/2} \exp\left\{-\frac{\tau}{2c}\sum_{k=1}^{q}\sum_{t=l+1}^{n} \varepsilon_k^2(t)\right\}. \end{aligned} \tag{2.43}$$

Defining

$$S_{\varepsilon\varepsilon} = \sum_{k=1}^{q}\sum_{t=l+1}^{n} \varepsilon_k^2(t) \tag{2.44}$$

and rewriting the likelihood, we have

$$p(\mathbf{Y}|\boldsymbol{\Theta}) = (2\pi)^{-q(n-l)/2} \left(\frac{\tau}{c}\right)^{q(n-l)/2} \exp\left\{-\frac{\tau}{2c}S_{\varepsilon\varepsilon}\right\}. \tag{2.45}$$

For computational purposes, note that

$$\begin{aligned} \varepsilon_k(t) &= y_k(t) - \sum_{j=0}^{m} a_j s_k(t-j) \\ &= y_k(t) - s_k(t) - \sum_{j=1}^{m} a_j s_k(t-j). \end{aligned} \tag{2.46}$$

The joint density of the unknown parameters $\boldsymbol{\Theta}$ and the data $\mathbf{Y}$ is

$$p(\mathbf{Y}, \boldsymbol{\Theta}) = p(\mathbf{Y}|\boldsymbol{\Theta})p(\boldsymbol{\Theta}),$$

where the likelihood is given in (2.43) and the prior is given in (2.40). The joint posterior of the unknown parameters, given the data, can be expressed using Bayes' Rule (2.21) as proportional to the likelihood times the prior, here

$$p(\boldsymbol{\Theta}|\mathbf{Y}) \propto p(\mathbf{Y}|\boldsymbol{\Theta})p(\boldsymbol{\Theta}). \tag{2.47}$$

For our deconvolution, calculations of $p(\eta|\mathbf{Y})$, $p(\mathbf{a}|\mathbf{Y})$, $p(\boldsymbol{\phi}|\mathbf{Y})$, $p(\tau|\mathbf{Y})$, and $p(\mathbf{S}|\mathbf{Y})$ are not easily calculated analytically. So we now proceed to develop the conditional marginal posterior distributions of the parameters in our model, which are used in the Gibbs sampler. It is through the Gibbs sampler that we produce estimates of the parameters in our model.

2.6 The Conditional Marginal Posterior Distributions

In this section we proceed to develop the conditional marginal posterior distributions of the sets of parameters in our model. The following conditional marginal posteriors distributions are calculated: $p(\eta|\mathbf{Y}, rest)$, $p(\mathbf{a}|\mathbf{Y}, rest)$, $p(\boldsymbol{\phi}|\mathbf{Y}, rest)$, $p(\tau|\mathbf{Y}, rest)$, and $p(\mathbf{S}|\mathbf{Y}, rest)$, where *rest* refers to the remaining parameters in $\boldsymbol{\Theta}$ for a specific parameter or set of parameters of interest. For example, for the parameter η the conditional marginal posterior distribution of η, given the data $\mathbf{Y}$, and the remaining parameters, is

$$p(\eta|\mathbf{Y}, rest)$$

where

$$rest = \{\mathbf{a}, \boldsymbol{\phi}, \tau, \mathbf{S}\}.$$

In the final section of this chapter we use the conditional marginal posterior distributions to specify the Gibbs sampling algorithm that we use in our deconvolution problem.

2.6.1 Conditional Marginal Posterior for η

The posterior of η, given the data $\mathbf{Y}$, and the *rest* is calculated as follows. Note that the likelihood $p(\mathbf{Y}|\boldsymbol{\Theta})$, given in (2.43), does not directly depend on η and we use (2.9). We calculate

$$\begin{aligned} p(\eta|\mathbf{Y}, rest) &\propto p(\mathbf{Y}|\boldsymbol{\Theta})p(\boldsymbol{\Theta}) \\ &= p(\mathbf{Y}|\boldsymbol{\Theta})p(\eta)p(\mathbf{a}|\eta)p(\boldsymbol{\phi})p(\tau)p(\mathbf{S}|\boldsymbol{\phi}, \tau) \\ &\propto p(\eta)p(\mathbf{a}|\eta). \end{aligned}$$

By (2.22) and (2.9) it follows that

$$\begin{aligned} p(\eta|\mathbf{Y}, rest) &\propto \eta^{\beta_1-1}(1-\eta)^{\beta_2-1}\eta^{n_\alpha}(1-\eta)^{m-n_\alpha} \\ &= \eta^{n_\alpha+\beta_1-1}(1-\eta)^{m-n_\alpha+\beta_2-1}, \end{aligned} \tag{2.48}$$

where n_α is the number of cases for which $a_i = 0$ in the pulse sequence $\mathbf{a} = [a_1, ..., a_m]'$. The kernel of the density is beta, therefore

$$\eta|\mathbf{Y}, rest \sim beta(\beta_1^*, \beta_2^*), \tag{2.49}$$

with parameters

$$\beta_1^* = n_\alpha + \beta_1 \tag{2.50}$$

and

$$\beta_2^* = m - n_\alpha + \beta_2. \tag{2.51}$$

2.6.2 Conditional Marginal Posterior for a

The conditional marginal posterior of the *pulse sequence* $\mathbf{a}$, given the data $\mathbf{Y}$, and the *rest* of the parameters is calculated as follows. Note that the likelihood $p(\mathbf{Y}|\boldsymbol{\Theta})$

depends directly on $\mathbf{a}$ and that we use (2.9). We calculate

$$\begin{aligned} p(\mathbf{a}|\mathbf{Y}, rest) &\propto p(\mathbf{Y}|\mathbf{\Theta})p(\mathbf{\Theta}) \\ &= p(\mathbf{Y}|\mathbf{\Theta})p(\eta)p(\mathbf{a}|\eta)p(\boldsymbol{\phi})p(\tau)p(\mathbf{S}|\phi,\tau) \\ &\propto p(\mathbf{Y}|\mathbf{\Theta})p(\mathbf{a}|\eta) \\ &= p(\mathbf{Y}|\mathbf{\Theta})\prod_{j=1}^{m} p(a_j|\eta) \\ &= p(\mathbf{Y}|\mathbf{\Theta})p(a_1|\eta) \times ... \times p(a_m|\eta). \end{aligned} \tag{2.52}$$

Since the a_j, $j = 1, ..., m$, are assumed to be independent and identically distributed, with the density given in (2.9), we can calculate the conditional marginals for each a_j, $j \in \{1, ..., m\}$ separately. Thus, it follows that

$$p(a_j|\mathbf{Y}, rest) \propto p(\mathbf{Y}|\mathbf{\Theta})p(a_j|\eta). \tag{2.53}$$

Next we proceed by isolating a_j in the likelihood $p(\mathbf{Y}|\mathbf{\Theta})$. For $j \in \{1, ..., m\}$ we calculate

$$\begin{aligned} p(a_j|\mathbf{Y}, rest) &\propto p(\mathbf{Y}|\mathbf{\Theta})p(a_j|\eta) \\ &\propto \exp\left\{-\frac{\tau}{2c}S_{\varepsilon\varepsilon}\right\}[\eta I(a_j = 0) \\ &\quad +(1-\eta)c^{-1}(\mu_\alpha, \sigma_\alpha^2)\frac{1}{\sqrt{2\pi}\sigma_\alpha}\exp\left\{-\frac{(a_j-\mu_\alpha)^2}{2\sigma_\alpha^2}\right\}I(a_j > 0)], \end{aligned}$$

where we recall from (2.44)

$$S_{\varepsilon\varepsilon} = \sum_{k=1}^{q}\sum_{t=l+1}^{n}\varepsilon_k^2(t) \tag{2.54}$$

and

$$\varepsilon_k(t) = y_k(t) - \sum_{j=0}^{m} a_j s_k(t-j). \tag{2.55}$$

In order to isolate a_j in $S_{\varepsilon\varepsilon}$, let

$$\varepsilon_k(t) = y_k(t) - \sum_{i=0, i\neq j}^{m} a_i s_k(t-i) - a_j s_k(t-j) \tag{2.56}$$

and define

$$\varepsilon_k^*(t[-j]) = y_k(t) - \sum_{i=0, i\neq j}^{m} a_i s_k(t-i). \tag{2.57}$$

Thus

$$\varepsilon_k(t) \quad = \quad \varepsilon_k^*(t[-j]) - a_j s_k(t-j). \tag{2.58}$$

Therefore, substituting (2.58)

$$\begin{aligned}
p(a_j|\mathbf{Y}, rest) \quad \propto \quad & \exp\left\{-\frac{\tau}{2c}\sum_{k=1}^{q}\sum_{t=l+1}^{n}\left[\varepsilon_k^*(t[-j]) - a_j s_k^2(t-j)\right]^2\right\} \\
& \times\left[\eta I(a_j=0) + (1-\eta)c^{-1}(\mu_\alpha, \sigma_\alpha^2)\frac{1}{\sqrt{2\pi}\sigma_\alpha}\exp\left\{-\frac{(a_j-\mu_\alpha)^2}{2\sigma_\alpha^2}\right\}\right] \\
& \times I(a_j > 0) \\
\propto \quad & \eta\exp\left\{-\frac{\tau}{2c}\sum_{k=1}^{q}\sum_{t=l+1}^{n}\left[\varepsilon_k^*(t[-j])\right]^2\right\} I(a_j=0) \\
& +(1-\eta)c^{-1}(\mu_\alpha, \sigma_\alpha^2)\frac{1}{\sqrt{2\pi}\sigma_\alpha} \\
& \times\exp\left\{-\frac{1}{2}\left[\frac{(a_j-\mu_\alpha)^2}{\sigma_\alpha^2} + \frac{\tau}{2c}\sum_{k=1}^{q}\sum_{t=l+1}^{n}\left[\varepsilon_k^*(t[-j]) - a_j s_k(t-j)\right]^2\right]\right\} \\
& \times I(a_j > 0).
\end{aligned}$$

Define

$$K_{\varepsilon\varepsilon}^j = \sum_{k=1}^{q}\sum_{t=l+1}^{n}\left[\varepsilon_k^*(t[-j])\right]^2 \tag{2.59}$$

and note that $K^j_{\varepsilon\varepsilon}$ excludes a_j. Also, define

$$A^j = \frac{(a_j - \mu_\alpha)^2}{\sigma^2_\alpha} + \frac{\tau}{c}\sum_{k=1}^{q}\sum_{t=l+1}^{n}\left[\varepsilon^*_k(t[-j]) - a_j s_k(t-j)\right]^2 \tag{2.60}$$

So

$$\begin{aligned} p(a_j|\mathbf{Y}, rest) \quad &\propto \quad \eta \exp\left\{-\frac{\tau}{2c}K^j_{\varepsilon\varepsilon}\right\} I(a_j = 0) \\ &+(1-\eta)c^{-1}(\mu_\alpha, \sigma^2_\alpha)\frac{1}{\sqrt{2\pi}\sigma_\alpha}\exp\left\{-\frac{1}{2}A^j\right\} I(a_j > 0). \end{aligned} \tag{2.61}$$

By completing the square in the numerator of the exponential of the second term in (2.61) and by manipulating the coefficient of the exponential term, which depends on η, we can determine the conditional marginal distribution of a_j. We begin by examining the exponent

$$\begin{aligned} A^j \quad &= \quad \frac{(a_j - \mu_\alpha)^2}{\sigma^2_\alpha} + \frac{\tau}{c}\sum_{k=1}^{q}\sum_{t=l+1}^{n}\left[\varepsilon^*_k(t[-j]) - a_j s_k(t-j)\right]^2 \\ &= \quad \frac{a_j^2}{\sigma^2_\alpha} - \frac{2a_j\mu_\alpha}{\sigma^2_\alpha} + \frac{\mu^2_\alpha}{\sigma^2_\alpha} + \frac{\tau}{c}\sum_{k=1}^{q}\sum_{t=l+1}^{n}\left[\varepsilon^*_k(t[-j])\right]^2 \\ &\quad -\frac{2\tau}{c}\sum_{k=1}^{q}\sum_{t=l+1}^{n}\varepsilon^*_k(t[-j])a_j s_k(t-j) + \frac{\tau}{c}\sum_{k=1}^{q}\sum_{t=l+1}^{n}a_j^2 s_k^2(t-j). \end{aligned}$$

Define

$$K^j_{\varepsilon s} = \sum_{k=1}^{q}\sum_{t=l+1}^{n}\varepsilon^*_k(t[-j])s_k(t-j), \tag{2.62}$$

$$K^j_{ss} = \sum_{k=1}^{q}\sum_{t=l+1}^{n}s_k^2(t-j), \tag{2.63}$$

and recall that

$$K^j_{\varepsilon\varepsilon} = \sum_{k=1}^{q}\sum_{t=l+1}^{n}\left[\varepsilon^*_k(t[-j])\right]^2. \tag{2.64}$$

So the exponent of the exponential is

$$A^j = \frac{a_j^2}{\sigma_\alpha^2} - \frac{2a_j\mu_\alpha}{\sigma_\alpha^2} + \frac{\mu_\alpha^2}{\sigma_\alpha^2} + \frac{\tau}{c}K_{\varepsilon\varepsilon}^j - \frac{2\tau}{c}K_{\varepsilon s}^j a_j + \frac{\tau}{c}K_{ss}^j a_j^2. \tag{2.65}$$

Simplifying, we get

$$A^j = \left[\frac{1}{\sigma_\alpha^2} + \frac{\tau}{c}K_{ss}^j\right] a_j^2 - 2\left[\frac{\mu_\alpha}{\sigma_\alpha^2} + \frac{\tau}{c}K_{\varepsilon s}^j\right] a_j + \frac{\mu_\alpha^2}{\sigma_\alpha^2} + \frac{\tau}{c}K_{\varepsilon\varepsilon}^j. \tag{2.66}$$

By completing the square, it follows that the exponent is equal to

$$A^j = \frac{a_j^2}{\sigma_{a_j}^2} - \frac{2\mu_{a_j}a_j}{\sigma_{a_j}} + \frac{\mu_{a_j}^2}{\sigma_{a_j}^2} - \frac{\mu_{a_j}^2}{\sigma_{a_j}^2} + \frac{\mu_\alpha^2}{\sigma_\alpha^2} + \frac{\tau}{c}K_{\varepsilon\varepsilon}^j, \tag{2.67}$$

where

$$\frac{a_j^2}{\sigma_{a_j}^2} = \left[\frac{1}{\sigma_\alpha} + \frac{\tau}{c}K_{ss}^j\right] a_j^2, \tag{2.68}$$

which implies

$$\sigma_{a_j}^2 = \left[\frac{1}{\sigma_\alpha^2} + \frac{\tau}{c}K_{ss}^j\right]^{-1}, \tag{2.69}$$

and where

$$\frac{\mu_{a_j}a_j}{\sigma_{a_j}} = \left[\frac{\mu_\alpha}{\sigma_\alpha} + \frac{\tau}{c}k_{ss}^j\right] a_j, \tag{2.70}$$

so that

$$\mu_{a_j} = \sigma_{a_j}^2 \left[\frac{\mu_\alpha}{\sigma_\alpha^2} + \frac{\tau}{c}K_{\varepsilon s}^j\right]. \tag{2.71}$$

Finally, the exponent is

$$A^j = \frac{(a_j - \mu_{a_j})^2}{\sigma_{a_j}} - \left[\frac{\mu_{a_j}^2}{\sigma_{a_j}^2} - \frac{\mu_\alpha^2}{\sigma_\alpha^2}\right] + \frac{\tau}{c}K_{\varepsilon\varepsilon}^j. \tag{2.72}$$

Thus, from (2.68),(2.70), and (2.72), it follows that

$$
\begin{aligned}
p(a_j|\mathbf{Y}, rest) \;\; \propto \;\; & \eta \exp\left\{-\frac{\tau}{2c}K_{\varepsilon\varepsilon}^{j}\right\} I(a_j = 0) \\
& +(1-\eta)c^{-1}(\mu_\alpha, \sigma_\alpha^2) \exp\left\{\frac{1}{2}\left[\frac{\mu_{a_j}^2}{\sigma_{a_j}^2} - \frac{\mu_\alpha^2}{\sigma_\alpha^2}\right]\right\} \\
& \times \exp\left\{-\frac{\tau}{2c}K_{\varepsilon\varepsilon}^{j}\right\} \\
& \times \frac{1}{\sqrt{2\pi}\sigma_\alpha} \exp\left\{-\frac{1}{2}\frac{(a_j - \mu_{a_j})^2}{\sigma_{a_j}^2}\right\} I(a_j > 0). \qquad (2.73)
\end{aligned}
$$

Since $\exp\left\{-\frac{\tau}{2c}K_{\varepsilon\varepsilon}^{j}\right\}$ does not depend on a_j and by inserting the correct constant for the truncated normal distribution of a_j with mean μ_{a_j} and variance $\sigma_{a_j}^2$, we get

$$
\begin{aligned}
p(a_j|\mathbf{Y}, rest) \;\; \propto \;\; & \eta I(a_j = 0) \\
& +(1-\eta)\left[\frac{c(\mu_{a_j}, \sigma_{a_j}^2)}{c(\mu_\alpha, \sigma_\alpha^2)}\right]\left(\frac{\sigma_{a_j}}{\sigma_\alpha}\right) \exp\left\{\frac{1}{2}\left[\frac{\mu_{a_j}^2}{\sigma_{a_j}^2} - \frac{\mu_\alpha^2}{\sigma_\alpha^2}\right]\right\} \\
& \times c^{-1}(\mu_{a_j}, \sigma_{a_j}^2)\frac{1}{\sqrt{2\pi}\sigma_{a_j}} \exp\left\{-\frac{1}{2}\frac{(a_j - \mu_{a_j})^2}{\sigma_{a_j}^2}\right\} I(a_j > 0).
\end{aligned}
$$

Define

$$
d_j = \left[\frac{c(\mu_{a_j}, \sigma_{a_j}^2)}{c(\mu_\alpha, \sigma_\alpha^2)}\right]\left(\frac{\sigma_{a_j}}{\sigma_\alpha}\right) \exp\left\{\frac{1}{2}\left[\frac{\mu_{a_j}^2}{\sigma_{a_j}^2} - \frac{\mu_\alpha^2}{\sigma_\alpha^2}\right]\right\}. \qquad (2.74)
$$

Note that d_j is a constant with respect to a_j. So by manipulating the constants

$$
\begin{aligned}
p(a_j|\mathbf{Y}, rest) \;\; \propto \;\; & \eta_j I(a_j = 0) + (1-\eta_j)c^{-1}(\mu_{a_j}\sigma_{a_j}^2)\frac{1}{\sqrt{2\pi}\sigma_{a_j}} \\
& \times \exp\left\{-\frac{1}{2}\frac{(a_j - \mu_{a_j})^2}{\sigma_{a_j}^2}\right\} I(a_j > 0), \qquad (2.75)
\end{aligned}
$$

where

$$
\eta_j = \frac{\eta}{\eta + (1-\eta)d_j}. \qquad (2.76)
$$

This shows that the conditional marginal posterior distribution of a_j is of the same form as the modeling distribution of a_j. The conditional marginal distribution, given in (2.75), is Bernoulli-Gaussian with a mixture of a Bernoulli with probability η_j of taking the value 0 and has a $1-\eta_j$ probability of taking the truncated normal value with mean μ_{a_j} defined in (2.71) and variance $\sigma^2_{a_j}$ defined in (2.69).

2.6.3 Conditional Marginal Posterior for $\boldsymbol{\phi}$

The conditional marginal posterior of $\boldsymbol{\phi}$ given the data $\mathbf{Y}$ and the *rest* of the parameters is calculated by conditioning on $\mathbf{s}_k^{(2)} = [s_k(p+1), ..., s_k(n)]'$, $k = 1, ..., q$, while ignoring the dependence of $\boldsymbol{\phi}$ on $\mathbf{s}_k = [s_k(1), ..., s_k(p)]'$. This simplifies the derivation and the computing used in the Gibbs sampler algorithm. Also, since p is much smaller than n, the resulting approximation will have a negligible effect in our application. Here

$$\begin{aligned} p(\boldsymbol{\phi}|\mathbf{Y}, rest) &\propto p(\mathbf{Y}|\boldsymbol{\Theta})p(\boldsymbol{\Theta}) \\ &= p(\mathbf{Y}|\boldsymbol{\Theta})p(\eta)p(\mathbf{a}|\eta)p(\boldsymbol{\phi})p(\tau)p(\mathbf{S}|\boldsymbol{\phi}, \tau). \end{aligned}$$

Note by (2.43), that $p(\mathbf{Y}|\boldsymbol{\Theta})$ does not directly depend on $\boldsymbol{\phi}$, so

$$\begin{aligned} p(\boldsymbol{\phi}|\mathbf{Y}, rest) &\propto p(\boldsymbol{\phi})p(\mathbf{S}|\boldsymbol{\phi}, \tau) \\ &= p(\boldsymbol{\phi}) \prod_{k=1}^{m} p(\mathbf{s}_k|\boldsymbol{\phi}, \tau) \\ &= p(\boldsymbol{\phi}) \prod_{k=1}^{m} p(\mathbf{s}_k^{(2)}|\boldsymbol{\phi}, \tau, \mathbf{s}_k^{(1)})p(\mathbf{s}_k^{(1)}|\boldsymbol{\phi}, \tau) \\ &\approx p(\boldsymbol{\phi}) \prod_{k=1}^{q} p(\mathbf{s}_k^{(2)}|\boldsymbol{\phi}, \tau, \mathbf{s}_k^{(1)}) \\ &\propto \exp\left\{-\frac{1}{2}(\boldsymbol{\phi} - \boldsymbol{\phi}_0)'\Sigma^{-1}(\boldsymbol{\phi} - \boldsymbol{\phi}_0)\right\} \prod_{k=1}^{q} \exp\left\{-\frac{\tau}{2}\sum_{t=p+1}^{n} e_k^2(t)\right\}. \end{aligned}$$

The last proportionality follows from 2.23 and 2.32 . So

$$
\begin{aligned}
p(\boldsymbol{\phi}|\mathbf{Y}, rest) &\approx \exp\left\{-\frac{1}{2}\left[\tau\sum_{k=1}^{q}\sum_{t=p+1}^{n} e_k^2(t) + (\boldsymbol{\phi}-\boldsymbol{\phi}_0)'\Sigma_0^{-1}(\boldsymbol{\phi}-\boldsymbol{\phi}_0)\right]\right\} \\
&= \exp\left\{-\frac{1}{2}\left[\tau S_{ee} + (\boldsymbol{\phi}-\boldsymbol{\phi}_0)'\Sigma_0^{-1}(\boldsymbol{\phi}-\boldsymbol{\phi}_0)\right]\right\},
\end{aligned}
$$

where

$$S_{ee} = \sum_{k=1}^{q}\sum_{t=p+1}^{n} e_k^2(t). \tag{2.77}$$

We proceed to complete the square. In order to isolate the parameters $\phi_1, ...\phi_p$ in S_{ee} we define

$$e_k(t) = s_k(t) + \boldsymbol{\phi}'\tilde{\mathbf{s}}_k(t), \tag{2.78}$$

where

$$\tilde{\mathbf{s}}_k(t) = [s_k(t-1), ..., s_k(t-p)]'. \tag{2.79}$$

So

$$
\begin{aligned}
\tau S_{ee} + (\boldsymbol{\phi}-\boldsymbol{\phi}_0)'\Sigma_0^{-1}(\boldsymbol{\phi}-\boldsymbol{\phi}_0) &= \tau\sum_{k=1}^{q}\sum_{t=p+1}^{n} e_k^2(t) + (\boldsymbol{\phi}-\boldsymbol{\phi}_0)'\Sigma_0^{-1}(\boldsymbol{\phi}-\boldsymbol{\phi}_0) \\
&= \tau\sum_{k=1}^{q}\sum_{t=p+1}^{n} [s_k(t) \\
&\qquad +\boldsymbol{\phi}'\tilde{\mathbf{s}}_k(t)]^2 + (\boldsymbol{\phi}-\boldsymbol{\phi}_0)'\Sigma_0^{-1}(\boldsymbol{\phi}-\boldsymbol{\phi}_0) \\
&= \tau\sum_{k=1}^{q}\sum_{t=p+1}^{n} [s_k^2(t) + 2\tilde{\mathbf{s}}_k'(t)\boldsymbol{\phi} + \boldsymbol{\phi}'\tilde{\mathbf{s}}_k(t)\tilde{\mathbf{s}}_k'(t)\boldsymbol{\phi}] \\
&\qquad +\boldsymbol{\phi}'\Sigma_0^{-1}\boldsymbol{\phi} - 2\boldsymbol{\phi}_0'\Sigma_0^{-1}\boldsymbol{\phi} + \boldsymbol{\phi}_0'\Sigma_0^{-1}\boldsymbol{\phi}_0.
\end{aligned}
$$

This simplifies to the quadratic form

$$(\boldsymbol{\phi}-\boldsymbol{\phi}_*)'\Sigma_*^{-1}(\boldsymbol{\phi}-\boldsymbol{\phi}_*) = \boldsymbol{\phi}'\Sigma_*^{-1}\boldsymbol{\phi} - 2\boldsymbol{\phi}'\Sigma_*^{-1}\boldsymbol{\phi}_* + \boldsymbol{\phi}_*'\Sigma_*^{-1}\boldsymbol{\phi}_*, \tag{2.80}$$

where

$$\boldsymbol{\phi}'\Sigma_*^{-1}\boldsymbol{\phi} = \boldsymbol{\phi}'\left(\tau\sum_{k=1}^{q}\sum_{t=l+1}^{n} \tilde{\mathbf{s}}_k(t)\tilde{\mathbf{s}}_k'(t) + \Sigma_0^{-1}\right)\boldsymbol{\phi}, \tag{2.81}$$

which implies

$$\Sigma_*^{-1} = \tau \sum_{k=1}^{q} \sum_{t=l+1}^{n} \tilde{\mathbf{s}}_k(t)\tilde{\mathbf{s}}_k'(t) + \Sigma_0^{-1} \tag{2.82}$$

and

$$\boldsymbol{\phi}'\Sigma_*^{-1}\boldsymbol{\phi}_* = \boldsymbol{\phi}'\left(-\tau \sum_{k=1}^{q} \sum_{t=l+1}^{n} s_k(t)\tilde{\mathbf{s}}_k(t) + \Sigma_0^{-1}\boldsymbol{\phi}_0\right), \tag{2.83}$$

so that

$$\boldsymbol{\phi}_* = \Sigma_*\left[-\tau \sum_{k=1}^{q} \sum_{t=l+1}^{n} s_k(t)\tilde{\mathbf{s}}_k(t) + \Sigma_0^{-1}\boldsymbol{\phi}_0\right]. \tag{2.84}$$

The above quadratic form, in (2.80), gives the kernel of a normal density, hence

$$\boldsymbol{\phi}|\mathbf{Y}, rest \sim N_p(\boldsymbol{\phi}_*, \Sigma_*), \tag{2.85}$$

where $\boldsymbol{\phi}_*$ and Σ_* are given by (2.84) and (2.82), respectively.

2.6.4 Conditional Marginal Posterior for τ

The conditional marginal posterior of τ, given the data $\mathbf{Y}$, and the *rest* of the parameters is calculated using the same approximate conditioning argument as was used for $\boldsymbol{\phi}$. The conditional posterior follows

$$\begin{aligned}
p(\tau|\mathbf{Y}, rest) &\propto p(\mathbf{Y}|\boldsymbol{\Theta})p(\boldsymbol{\Theta}) \\
&= p(\mathbf{Y}|\tau)p(\eta)p(\mathbf{a}|\eta)p(\boldsymbol{\phi})p(\tau)p(\mathbf{S}|\boldsymbol{\phi},\tau) \\
&\propto p(\mathbf{Y}|\boldsymbol{\Theta})p(\tau)\prod_{k=1}^{q} p(\mathbf{s}_k|\boldsymbol{\phi},\tau) \\
&= p(\mathbf{Y}|\boldsymbol{\Theta})p(\tau)\prod_{k=1}^{q} p(\mathbf{s}_k^{(2)}|\boldsymbol{\phi},\tau,\mathbf{s}_k^{(1)})p(\mathbf{s}_k^{(1)}|\boldsymbol{\phi},\tau) \\
&\approx p(\mathbf{Y}|\boldsymbol{\Theta})p(\tau)\prod_{k=1}^{q} p(\mathbf{s}_k^{(2)}|\boldsymbol{\phi},\tau,\mathbf{s}_k^{(1)}) \\
&\propto \tau^{q(n-l)/2}\exp\left\{-\frac{\tau}{2c}\sum_{k=1}^{q}\sum_{t=l+1}^{n}\varepsilon_k^2(t)\right\} \\
&\qquad \times\tau^{\gamma_1-1}\exp\{-\gamma_2\tau\} \\
&\qquad \times\tau^{q(n-p)/2}\exp\left\{-\frac{\tau}{2}\sum_{k=1}^{q}\sum_{t=p+1}^{n}e_k^2(t)\right\}.
\end{aligned} \tag{2.86}$$

The step above (2.86) follows from 2.43, 2.24, and 2.32. Simplifying, we see that

$$p(\tau|\mathbf{Y}, rest) \propto \tau^{\frac{q(2n-l-p)}{2}+\gamma_1-1} \exp\left\{-\left[\frac{1}{2c}\sum_{k=1}^{q}\sum_{t=l+1}^{n}\varepsilon_k^2(t) + \frac{1}{2}\sum_{k=1}^{q}\sum_{t=p+1}^{n}e_k^2(t) + \gamma_2\right]\tau\right\}.$$

Define

$$S_{\varepsilon\varepsilon} = \sum_{k=1}^{q}\sum_{t=l+1}^{n}\varepsilon_k^2(t) \tag{2.87}$$

and

$$S_{ee} = \sum_{k=1}^{q}\sum_{t=p+1}^{n}e_k^2(t). \tag{2.88}$$

Because

$$\begin{aligned}\varepsilon_k(t) &= y_k(t) - \sum_{j=0}^{m} a_j s_k(t-j) \\ &= y_k(t) - s_k(t) - \sum_{j=1}^{m} a_j s_k(t-j) \end{aligned} \tag{2.89}$$

and

$$\begin{aligned}e_k(t) &= \sum_{j=0}^{p} \phi_j s_k(t-j) \\ &= s_k(t) + \sum_{j=1}^{p} \phi_j s_k(t-j). \end{aligned} \tag{2.90}$$

Therefore,

$$p(\tau|\mathbf{Y}, rest) \propto \tau^{q(2n-l-p)+\gamma_2]-1} \exp\left\{-\left[\frac{1}{2c}S_{\varepsilon\varepsilon} + \frac{1}{2}S_{ee} + \gamma_2\right]\tau\right\}. \tag{2.91}$$

This is the kernel of a gamma density, thus

$$\tau|\mathbf{Y}, rest \sim gamma(\gamma_1^*, \gamma_2^*), \tag{2.92}$$

where

$$\gamma_1^* = q(2n-l-p)/2 + \gamma_1 \tag{2.93}$$

and

$$\gamma_2^* = \frac{1}{2c}S_{\varepsilon\varepsilon} + \frac{1}{2}S_{ee} + \gamma_2. \tag{2.94}$$

2.6.5 Conditional Marginal Posterior of S

In this subsection we present the conditional marginal distribution of the matrix of *underlying signal-path effects sequences* $s_k(t),\ t = 1, ..., n,\ k = 1, ..., q$. Recall the notation

$$\mathbf{S} = [\mathbf{S}^{(1)}, \mathbf{S}^{(2)}]. \tag{2.95}$$

In view of (2.38) the conditional marginal posterior of $\mathbf{S}$ becomes

$$\begin{aligned} p(\mathbf{S}|\mathbf{Y}, rest) &\propto p(\mathbf{Y}|\mathbf{\Theta})p(\mathbf{\Theta}) \\ &= p(\mathbf{Y}|\mathbf{\Theta})p(\eta)p(\mathbf{a}|\eta)p(\boldsymbol{\phi})p(\tau)p(\mathbf{S}|\boldsymbol{\phi}, \tau) \\ &\propto p(\mathbf{Y}|\mathbf{\Theta})p(\mathbf{S}|\boldsymbol{\phi}, \tau) \\ &= p(\mathbf{Y}|\mathbf{\Theta}) \prod_{k=1}^{q} p(\mathbf{s}_k^{(2)}|\boldsymbol{\phi}, \tau, \mathbf{s}_k^{(1)})p(\mathbf{s}_k^{(1)}|\boldsymbol{\phi}, \tau) \end{aligned} \tag{2.96}$$

We proceed to calculate the posterior for $\mathbf{S}$ in two pieces. We calculate the posterior for $\mathbf{S}^{(2)}$ conditional on $\mathbf{S}^{(1)}$ and then for $\mathbf{S}^{(1)}$ unconditionally. We calculate the conditional marginal posterior of $s_k(i)$ individually because of the independence over k expressed in 2.96. We begin by calculating the posterior for $s_k(i)$, for $i \in (p+1), ..., n$, $k \in 1, ..., q$, using the likelihood, (2.43) and (2.9):

$$\begin{aligned} p(s_k(i)|\mathbf{Y}, rest) &\propto p(\mathbf{Y}|\mathbf{\Theta})p(s_k(i)|\boldsymbol{\phi}, \tau, s_k^{(1)}, s_k(p+1), ...s_k(i-1)) \\ &\propto \exp\left\{-\frac{\tau}{2c}\sum_{t=l+1}^{n} \varepsilon_k^2(t)\right\} \exp\left\{-\frac{\tau}{2}\sum_{t=p+1}^{n} e_k^2(t)\right\}. \end{aligned} \tag{2.97}$$

Define

$$E_k = \sum_{t=l+1}^{n} \varepsilon_k^2(t) \tag{2.98}$$

and

$$F_k = \sum_{t=p+1}^{n} e_k^2(t). \tag{2.99}$$

So

$$p(s_k(i)|\mathbf{Y}, rest) \propto \exp\left\{-\frac{\tau}{2c}E_k\right\}\exp\left\{-\frac{\tau}{2}F_k\right\}. \tag{2.100}$$

Note that $\varepsilon_k(t)$ depends on the past m points of the k^{th} *underlying signal-path effects sequence*, here $s_k(t-1), ..., s_k(t-m)$. Also note that each $s_k(i)$, $i \in \{(t-1), ..., (t-m)\}$ depends on its previous p points. Thus to calculate $\varepsilon_k(t)$ we need the past $l = m + p$ points of the *underlying signal-path effects sequence*.

Now we need to isolate the values of $\varepsilon_k(t)$ in E_k and $e_k(t)$ in F_k that depend on $s_k(i)$. Also, recall that t is assumed to start at $(l+1)$ and ends at n. First, in order to find the values of t where E_k depends on $s_k(i)$, we write

$$\begin{aligned}\varepsilon_k(t) &= y_k(t) - \sum_{j=0}^{m} a_j s_k(t-j) \\ &= y_k(t) - s_k(t) - a_1 s_k(t-1) - ... - a_m s_k(t-m)\end{aligned}$$

and note that for t between i and $i+m$, $\varepsilon_k(t)$ depends on $s_k(i)$. Thus the part of E_k that depends on $s_k(i)$, defined as E_k', is

$$E_k' = \sum_{t=max(i,(l+1))}^{min((i+m),n)} \varepsilon_k^2(t) \tag{2.101}$$

and we let

$$\sum^{1} \equiv \sum_{t=max(i,(l+1))}^{min((i+m),n)}. \tag{2.102}$$

Similarly, to find the values of t where F_k depends on $s_k(i)$, we write

$$\begin{aligned}e_k(t) &= \sum_{j=0}^{p} \phi_j s_k(t-j) \\ &= s_k(t) + \phi_1 s_k(t-1) + ... + \phi_p s_k(t-p)\end{aligned}$$

and note that for t between i and $i+p$, $e_k(t)$ depends on $s_k(i)$. Here we note that t is assumed to be between $(p+1)$ and n. Thus the part of F_k that depends on $s_k(i)$, defined as F_k', is

$$F_k' = \sum_{t=i}^{min((i+p),n)} e_k^2(t) \tag{2.103}$$

and we define

$$\sum^{2} \equiv \sum_{t=i}^{min((i+p),n)} . \tag{2.104}$$

So now

$$p(s_k(i)|\mathbf{Y}, rest) \propto \exp\left\{-\frac{\tau}{2c}E'_k\right\}\exp\left\{-\frac{\tau}{2}F'_k\right\}. \tag{2.105}$$

Second, to isolate $s_k(i)$ in both sums E'_k and F'_k we define

$$\varepsilon'_k(t[-i]) = y_k(t) - \sum_{j=0, j\neq t-i}^{m} a_j s_k(t-j) \tag{2.106}$$

so that

$$\varepsilon_k(t) = \varepsilon'_k(t[-i]) - a_{t-i}s_k(i) \tag{2.107}$$

and

$$e'_k(t[-i]) = \sum_{j=0, j\neq t-i} \phi_j s_k(t-j) \tag{2.108}$$

so that

$$e_k(t) = e'_k(t[-i]) + \phi_{t-i}s_k(i). \tag{2.109}$$

Thus

$$\begin{aligned} p(s_k(i)|\mathbf{Y}, rest) &\propto \exp\left\{-\frac{\tau}{2c}\sum^{1}[\varepsilon'_k(t[-i]) - a_{t-i}s_k(i)]^2\right\} \\ &\quad \times \exp\left\{-\frac{\tau}{2}\sum^{2}[e'_k(t[-i]) + \phi_{t-i}s_k(i)]^2\right\} \\ &= \exp\left\{-\frac{1}{2}\left[\frac{\tau}{c}\sum^{1}[\varepsilon'_k(t[-i]) - a_{t-i}s_k(i)]^2\right.\right. \\ &\quad \left.\left. +\tau\sum^{2}[e'_k(t[-i]) + \phi_{t-i}s_k(i)]^2\right]\right\}. \end{aligned}$$

To complete the square we expand the exponent:

$$\frac{\tau}{c}\sum^{1}[\varepsilon'_k(t[-i]) - a_{t-i}s_k(i)]^2 + \tau\sum^{2}[e'_k(t[-i]) + \phi_{t-i}s_k(i)]^2$$

$$= \quad \frac{\tau}{c}\sum^{1}[\varepsilon_k'(t[-i])]^2 - \frac{2\tau}{c}\sum^{1}\varepsilon_k'(t[-i])a_{t-i}s_k(i) + \frac{\tau}{c}\sum^{1}a_{t-i}^2 s_k^2(i)$$
$$+\tau\sum^{2}[e_k'(t[-i])]^2 + 2\tau\sum^{2}e_k'(t[-i])\phi_{t-i}s_k(i) + \tau\sum^{2}\phi_{t-i}^2 s_k^2(i).$$

This simplifies to the form

$$\frac{[s_k(i)-\mu_{s_k(i)}]^2}{\sigma_{s_k(i)}^2} = \frac{s_k^2(i)}{\sigma_{s_k(i)}^2} - \frac{2\mu_{s_k(i)}s_k(i)}{\sigma_{s_k(i)}^2} + \frac{\mu_{s_k(i)}^2}{\sigma_{s_k(i)^2}}, \tag{2.110}$$

where

$$\frac{s_k^2(i)}{\sigma_{s_k(i)}^2} = \left[\frac{\tau}{c}\sum^{1}a_{t-i}^2 + \tau\sum^{2}\phi_{t-i}^2\right]s_k^2(i), \tag{2.111}$$

which implies

$$\sigma_{s_k(i)}^2 = \left[\frac{\tau}{c}\sum^{1}a_{t-i}^2 + \tau\sum^{2}\phi_{t-i}^2\right]^{-1}, \tag{2.112}$$

and

$$\frac{\mu_{s_k(i)}s_k(i)}{\sigma_{s_k(i)}^2} = \left[-\frac{\tau}{c}\sum^{1}\varepsilon_k'(t[-i])a_{t-i} + \tau\sum^{2}e_k'(t[-i])\phi_{t-i}\right]s_k(i), \tag{2.113}$$

so that

$$\mu_{s_k(i)} = \sigma_{s_k(i)}^2\left[-\frac{\tau}{c}\sum^{1}\varepsilon_k'(t[-i])a_{t-i} + \tau\sum^{2}e_k'(t[-i])\phi_{t-i}\right]. \tag{2.114}$$

Hence, for $i \in \{(p+1), ..., n\}$ and $k \in \{1, ..., q\}$

$$p(s_k(i)|\mathbf{Y}rest) \propto \exp\left\{-\frac{1}{2}\frac{[s_k(i)-\mu_{s_k(i)}]^2}{\sigma_{s_k(i)}^2}\right\}. \tag{2.115}$$

This is the kernel of a normal density, therefore

$$s_k(i)|\mathbf{Y}, rest \sim N(\mu_{s_k(i)}, \sigma_{s_k(i)}^2), \tag{2.116}$$

for $i \in \{(p+1), ..., n\}$ and $k \in \{1, ..., q\}$.

Now for $i \in \{1, ..., p\}$ we find the posterior of $\mathbf{S}^{(1)}$. We begin by using the independence in (2.96). And note that the likelihood $p(\mathbf{Y}|\mathbf{\Theta})$ does not directly depend on $\mathbf{s}_k^{(1)}$ because of our assumption that the data starts at $(l+1)$.

$$\begin{aligned} p(\mathbf{s}_k^{(1)}|\boldsymbol{\phi},\tau) &\propto p(\mathbf{Y}|\boldsymbol{\Theta})p(\eta)p(\mathbf{a}|\eta)p(\boldsymbol{\phi})p(\tau)p(\mathbf{s}_k^{(2)}|\boldsymbol{\phi},\tau,\mathbf{s}_k^{(1)})p(\mathbf{s}_k^{(1)}|\boldsymbol{\phi},\tau) \\ &\propto p(\mathbf{s}_k^{(2)}|\boldsymbol{\phi},\tau,\mathbf{s}_k^{(1)})p(\mathbf{s}_k^{(1)}|\boldsymbol{\phi},\tau) \\ &\propto \exp\left\{-\frac{\tau}{2c}\sum_{t=l+1}^{n}\varepsilon_k^2(t)\right\}\exp\left\{-\frac{1}{2}\tilde{\mathbf{s}}_k'(p+1)\Sigma_0\mathbf{s}_k'(p+1)\right\}, \end{aligned}$$

which follows from (2.38). Since, from (2.79) we have

$$\tilde{\mathbf{s}}_k(t) = [s_k(t-1), ..., s_k(t-p)]' \tag{2.117}$$

and for $t = p+1$ we have

$$\tilde{\mathbf{s}}_k(p+1) = [s_k(p), ..., s_k(1)]'. \tag{2.118}$$

Thus $\tilde{\mathbf{s}}_k(p+1)$ is the vector $\mathbf{s}_k^{(1)}$, from (2.26), in reverse order. So the prior on $\tilde{\mathbf{s}}_k(p+1)$ is the same as that given in (2.34)

$$N_p(0, \Sigma_\phi) \tag{2.119}$$

where, now for $p = 3$,

$$\Sigma_\phi = \frac{1}{\tau}\begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{12} & \gamma_{22} & \gamma_{23} \\ \gamma_{13} & \gamma_{23} & \gamma_{33} \end{bmatrix} \tag{2.120}$$

and $\gamma_{11} = \sigma_{33}$, $\gamma_{22} = \sigma_{22}$, $\gamma_{33} = \sigma_{11}$, $\gamma_{12} = \sigma_{23}$, $\gamma_{13} = \sigma_{13}$, and $\gamma_{23} = \sigma_{12}$.

We now proceed to isolate the values of $s_k(t)$, $t = 1, ..., p$, in $p(\mathbf{s}_k^{(2)}|\mathbf{Y}, \boldsymbol{\Theta})$. Note that

$$\begin{aligned} \sum_{t=p+1}^{n} e_k^2(t) &= \sum_{t=p+1}^{n}\left[\sum_{j=0}^{p}\phi_j s_k(t-j)\right]^2 \\ &= \sum_{t=p+1}^{n}[s_k(t) + \phi_1 s_k(t-1) + ... + \phi_p s_k(t-p)]^2 \end{aligned}$$

$$\begin{aligned}
&= [s_k(p+1) + \phi_1 s_k(p) + ... + \phi_p s_k(1)]^2 \\
&\quad + [s_k(p+2) + \phi_1 s_k(p+1) + ... + \phi_p s_k(2)]^2 \\
&\quad +... \\
&\quad + [s_k(p+p) + \phi_1 s_k(p) + ... + \phi_p s_k(p)]^2 \\
&\quad \\
&\quad +[Remaining\ Terms] \\
&= e_k(p+1) + e_k(p+2) + ... + e_k(p+p) + [Remaining\ Terms],
\end{aligned}$$

where the "*Remaining Terms*" do not depend on values in the vector $\mathbf{s}_k(p+1)$. Because only

$$\sum_{t=p+1}^{p+p} e_k^2(t) \tag{2.121}$$

involves $s_k(1), ..., s_k(p)$, so that the important part of

$$\sum_{t=p+1}^{n} e_k^2(t) \tag{2.122}$$

is

$$\sum_{t=1}^{p} e_k^2(t+p). \tag{2.123}$$

Therefore

$$\begin{aligned}
p(\mathbf{s}_k^{(1)}|\boldsymbol{\phi}, \tau) &\propto \exp\left\{-\frac{\tau}{2c}\sum_{t=p+1}^{n} e_k^2(t)\right\} \exp\left\{-\frac{1}{2}\tilde{\mathbf{s}}_k'(p+1)\Sigma_\phi^{-1}\tilde{\mathbf{s}}_k(p+1)\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left[\frac{\tau}{c}\sum_{t=1}^{p} e_k^2(t+p) + \tilde{\mathbf{s}}_k'(p+1)\Sigma_\phi^{-1}\tilde{\mathbf{s}}_k(p+1)\right]\right\}.
\end{aligned}$$

Define $\Phi_t = [\phi_t, ..., \phi_p, 0, ..., 0]'$ for $t = 1, ..., p$. Then

$$\begin{aligned}
\Phi_1 &= [\phi_1, ..., \phi_p]' = \boldsymbol{\phi}, \\
\Phi_2 &= [\phi_2, ..., \phi_p, 0]',
\end{aligned}$$

$$\begin{aligned}\Phi_3 &= [\phi_3, ..., \phi_p, 0, 0]', \\ &\vdots \\ \Phi_p &= [\phi_p, 0, ..., 0]'.\end{aligned}$$

We also define

$$c_k(t) = s_k(p+t) + \phi_1 s_k(p+t-1) + ... + \phi_{t-1} s_k(p+1), \tag{2.124}$$

for $t \in \{1, ..., p\}$. Then

$$\begin{aligned}c_k(1) &= s_k(p+1) \\ c_k(2) &= s_k(p+2) + \phi_1 s_k(p+1) \\ &\vdots \\ c_k(p) &= s_k(2p) + \phi_1 s_k(2p-1) + ... + \phi_{t-1} s_k(p+1).\end{aligned}$$

Note that $c_k(t)$ does not depend on $s_k(t)$, $t \in \{1, ..., p\}$. Then

$$e_k(p+t) = c_k(t) + \tilde{\mathbf{s}}_k'(p+1)\Phi_t, \tag{2.125}$$

and so

$$p(\mathbf{s}_k^{(1)}|\boldsymbol{\phi}, \tau) \propto \exp\left\{-\frac{1}{2}\left[\frac{\tau}{c}\sum_{t=1}^{p}\left[c_k(t) + \tilde{\mathbf{s}}_k'(p+1)\Phi_t\right]^2 + \tilde{\mathbf{s}}_k'(p+1)\Sigma_\phi^{-1}\tilde{\mathbf{s}}_k(p+1)\right]\right\}. \tag{2.126}$$

To complete the square we expand the exponent.

$$\frac{\tau}{c}\sum_{t=1}^{p} c_k^2(t) + \frac{2\tau}{c}\sum_{t=1}^{p} c_k(t)\tilde{\mathbf{s}}_k'(p+1)\Phi_t + \tilde{\mathbf{s}}_k'(p+1)\left[\frac{\tau}{c}\sum_{t=1}^{p}\Phi_t\Phi_t'\right]\tilde{\mathbf{s}}_k(p+1)$$

$$+\tilde{\mathbf{s}}_k'(p+1)\Sigma_\phi^{-1}\tilde{\mathbf{s}}_k(p+1) \tag{2.127}$$

This simplifies to the following quadratic form

$$\left[\tilde{\mathbf{s}}_k(p+1) - \mu_{\tilde{\mathbf{s}}_k(p+1)}\right]'\Sigma_{\tilde{\mathbf{s}}_k(p+1)}^{-1}\left[\tilde{\mathbf{s}}_k(p+1) - \mu_{\tilde{\mathbf{s}}_k(p+1)}\right]$$

$$= \tilde{\mathbf{s}}_k'(p+1)\Sigma^{-1}_{\tilde{\mathbf{s}}_k(p+1)}\tilde{\mathbf{s}}_k(p+1) - 2\tilde{\mathbf{s}}_k'(p+1)\Sigma^{-1}_{\tilde{\mathbf{s}}_k(p+1)}\mu_{\tilde{\mathbf{s}}_k(p+1)}$$
$$+\mu'_{\tilde{\mathbf{s}}_k(p+1)}\Sigma^{-1}_{\tilde{\mathbf{s}}_k(p+1)}\mu_{\tilde{\mathbf{s}}_k(p+1)},$$

where from (2.127)

$$\tilde{\mathbf{s}}_k'(p+1)\Sigma^{-1}_{\tilde{\mathbf{s}}_k(p+1)}\tilde{\mathbf{s}}_k(p+1) = \tilde{\mathbf{s}}_k'(p+1)\left[\frac{\tau}{c}\sum_{t=1}^{p}\Phi_t\Phi_t' + \Sigma_\phi^{-1}\right]\tilde{\mathbf{s}}_k(p+1), \tag{2.128}$$

which implies

$$\Sigma^{-1}_{\tilde{\mathbf{s}}_k'(p+1)} = \left[\frac{\tau}{c}\sum_{t=1}^{p}\Phi_t\Phi_t + \Sigma_\phi^{-1}\right], \tag{2.129}$$

and where from (2.127)

$$\tilde{\mathbf{s}}_k'(p+1)\Sigma^{-1}_{\tilde{\mathbf{s}}_k(p+1)}\mu_{\tilde{\mathbf{s}}_k(p+1)} = \tilde{\mathbf{s}}_k'(p+1)\left[-\frac{\tau}{c}\sum_{t=1}^{p}c_k(t)\Phi_t\right], \tag{2.130}$$

which implies

$$\mu_{\tilde{\mathbf{s}}_k(p+1)} = \Sigma_{\tilde{\mathbf{s}}_k(p+1)}\left[-\frac{\tau}{c}\sum_{t=1}^{p}c_k(t)\Phi_t\right]. \tag{2.131}$$

Therefore, we have the kernel of a normal density and hence

$$\tilde{\mathbf{s}}_k(p+1) \sim N_p(\mu_{\tilde{\mathbf{s}}_k(p+1)}, \Sigma_{\tilde{\mathbf{s}}_k'(p+1)}). \tag{2.132}$$

2.7 The Steps to Perform the Gibbs Sampler

Using the conditional marginal posterior distributions developed in this chapter, the Gibbs sampler proceeds iteratively as follows: given the initial values

$$\left\{\eta^{(0)}, \mathbf{a}^{(0)}, \boldsymbol{\phi}^{(0)}\tau^{(0)}, \mathbf{S}^{(0)}\right\}$$

and values for the collection of predetermined hyperparamters

$$\{\beta_1, \beta_2, \mu_\alpha, \sigma_\alpha^2, \boldsymbol{\phi}_0, \Sigma_0, \gamma_1\gamma_2\}, \tag{2.133}$$

which reflects our prior knowledge, and then for $h = 1$ to the total number of repetitions:

1. Sample $\eta^{(h)}$ from

$$p(\eta|\mathbf{Y}, rest) \sim beta(\beta_1^*, \beta_2^*),$$

where β_1^* and β_2^* are defined in (2.50) and (2.51), respectively.

2. Sample the elements of $\mathbf{a}^{(h)}$ sequentially, for $j = 1, ..., m$ sample a_j from

$$\begin{aligned} p(a_j|\mathbf{Y}, rest) \quad \sim \quad & \eta_j I(a_j = 0) + (1 - \eta_j)c^{-1}(\mu_{a_j}, \sigma_{a_j}^2)\frac{1}{\sqrt{2\pi}\sigma_{a_j}} \\ & \times \exp\left\{-\frac{1}{2}\frac{(a_j - \mu_\alpha)^2}{\sigma_{a_j}^2}\right\} I(a_j > 0), \end{aligned}$$

where μ_{a_j} and $\sigma_{a_j}^2$ are defined in (2.71) and (2.69), respectively. And η_j is defined in (2.76).

3. Sample $\boldsymbol{\phi}^{(h)}$ from

$$p(\boldsymbol{\phi}|\mathbf{Y}, rest) \sim N_p(\boldsymbol{\phi}_*, \Sigma_*),$$

where $\boldsymbol{\phi}_*$ and Σ_* are defined in (2.84) and (2.82).

4. Sample $\tau^{(h)}$ from

$$p(\tau|\mathbf{Y}, rest) \sim gamma(\gamma_1^*, \gamma_2^*),$$

where γ_1^* and γ_2^* are defined in (2.93) and (2.94).

5. sample $\mathbf{S}^{(h)}$ in two pieces

   (a) Sample sequentially $[s_k(i)]^{(h)}$ for $k = 1, ..., q$ and $i = (p + 1), ..., n$ from

   $$p(s_k(i)|\mathbf{Y}, rest) \sim N(\mu_{s_k(i)}, \sigma_{s_k(i)}^2),$$

   where $\mu_{s_k(i)}$ and $\sigma_{s_k(i)}^2$ are defined in (2.114) and (2.112), respectively.

(b) Sample $[\tilde{\mathbf{s}}_k(p+1)]^{(h)} = [s_k^{(h)}(p), ..., s_k^{(h)}(1)]'$ from

$$p(\tilde{\mathbf{s}}_k(p+1)|\mathbf{Y}, rest) \sim N_p(\mu_{\tilde{s}_k(p+1)}, \Sigma_{\tilde{s}_k(p+1)}),$$

where $\mu_{\tilde{s}_k(p+1)}$ and $\Sigma_{\tilde{s}_k(p+1)}$ are defined in (2.131) and (2.129).

6. Set $h = h + 1$ and go to Step 1.

In Chapter 3 we present simulation results related to implementing the Gibbs sampler in our deconvolution problem. The lessons learned from our simulations are used in Chapter 4 to implement our method on real seismic data from a ripple-fired mining explosions.

Chapter 3

Simulation Results

In this Chapter we present the results produced by our software that implements the Bayesian deconvolution method described in Chapter 2 on simulated data. We have written a Fortran 90 program that runs the Gibbs sampler presented in Section 2.8 on a set of array data (see Appendix). The software uses subroutines from Numerical Recipes in Fortran 77 by Press et.al. [28] to generate $uniform(0,1)$, $gamma(a,b)$, and $normal(0,1)$ random deviates. The subroutine used to generate gamma random deviates is from Statistical Computing by Kennedy and Gentle [21]. And the subroutine used to calculate cumulative normal probabilities is due to Alan Miller [26].

This Chapter is organized into three sections. First, in Section 3.1 we describe our method of simulating realistic array data of a ripple-fired event. The spectral characteristics of the simulated data we generate are examined to verify their similarity to real data. Second, Section 3.2 describes how we implement our Bayesian deconvolution method. We specify the hyperparameters of our priors, our choice of the parameters c and m, and the starting values for the Gibbs sampler. Lastly, we present, in Section 3.3, the results of a typical run of our deconvolution program. Our software produces Markov chains for each parameter in the parameter set $\mathbf{\Theta}$. The values of the Markov chains after "burn in" through the total number of repetitions are saved. We calculate posterior estimates of the parameters along with density estimates of the posterior distributions using S-Plus. We consider the goodness-of-fit of our model to the simulated data by comparing our estimates of the *pulse sequence* and the *underlying signal-path effects sequences* with their true values. We also present graphs of the Markov chains that are used to monitor convergence of the Gibbs sampler. Lastly, we present the results of one run of our program on simulated data having no delayed signals.

3.1 Simulated Data Containing a Common Delay Structure

To produce simulated data containing a common delay structure we use the convolution model

$$y_k(t) = \sum_{j=0}^{m} a_j s_k(t-j) + \varepsilon_k(t), \tag{3.1}$$

where $k = 1, ..., q$, $t = 1, ..., n$, and $\varepsilon_k(t)$ are independent and identically distributed normal with mean 0 and variance $c\sigma^2$. We begin by simulating the *underlying signal-path effects sequences* $\mathbf{s}_k = [s_k(1), ..., s_k(n)]'$, $k = 1, ..., q = 5$, of length $n = 256$ points. We use an autoregressive model of order 3 with coefficients from Dargahi-Noubary's k_0-model of a single explosion [8]. The model we use to produce these simulated data, $\mathbf{s}_k$, of a single event is

$$s_k(t) + \phi_1 s_k(t-1) + \phi_2 s_k(t-2) + ... + \phi_p s_k(t-p) = e_k(t), \tag{3.2}$$

where the error term is

$$e_k(t) \sim N(0, \sigma^2)$$

with $\sigma^2 = 1$. The autoregressive coefficients are calculated using the following analytical model, where ϕ_i, $i = 1, 2, 3$, are each a function of a single parameter k_0:

$$\phi_i = (-1)^i \begin{pmatrix} p \\ i \end{pmatrix} \exp(-ik_0). \tag{3.3}$$

Taking $k_0 = 0.6$, we obtain the coefficients

$$\phi_1 = \phi_1(k_0) = -3\exp(-0.6) = -1.646, \tag{3.4}$$

$$\phi_2 = \phi_2(k_0) = 3\exp(-2 * 0.6) = 0.904, \tag{3.5}$$

and

$$\phi_3 = \phi_3(k_0) = -\exp(-3 * 0.6) = -0.165. \tag{3.6}$$

Using the model in (3.2) and the values of the autoregresive coefficients for k_0, we produce the $q = 5$ simulated sequences presented in Figure 3.2. Note that these simulated data contain low frequencies that are not characteristic of real data. (see panels three and four of Figure 1.2).

To produce simulated data $\mathbf{y}_k$ containing a common delay structure we convolved the simulated *underlying signal-path effects sequences*, presented in Figure 3.2 with a *pulse sequence* $\mathbf{a}$. The *pulse sequence* we use is of length $m = 30$ having nonzero values $a_7 = 0.8$, $a_{14} = 0.7$, $a_{21} = 0.6$, and $a_{28} = 0.5$. Equal delays are used because ripple-fired mining explosions are often designed using equal delay times between subexplosions. The additive error $\varepsilon_k(t)$ according to the model has variance $c\sigma^2$ and we use $c = 0.01$; this value of c corresponds to a $SNR = 100$. Thus the simulated *underlying signal-path effects sequence* produced has 100 times the variance as the additive (background) noise. This produces the simulated data presented in Figure 3.4. We note that the low frequencies contained in the simulated *underlying signal-path effects sequences* are enhanced by the convolution process. Next we filter the simulated *underlying signal-path effects sequence* with a common filter before convolving them with the *pulse sequence* $\mathbf{a}$. By first removing the low frequencies in $\mathbf{s}_k$, $k = 1, ..., q$, we produced reasonable simulated data.

We use a 32-point two-sided filter to remove the low frequencies in the simulated *underlying signal-path effects sequences* $\mathbf{s}_k$. This results in realistic appearing simulated data $\mathbf{y}_k$, $k = 1, ..., q$, that contains the appropriate spectral characteristics. We stabilize $\mathbf{s}_k$, $k = 1, ..., q$, shown in Figure 3.2, using a filter $\{b_u : u = -16, ..., 16\}$, where

$$\tilde{s}_k(t) = \sum_{u-16}^{16} b_u s_k(t-u)$$

with the condition

$$\sum_{u=-16}^{16} b_u^2 = 1.$$

The filter we use is designed to remove the frequencies υ below 0.05/point in $\mathbf{s}_k$, $k = 1, ..., q$. The filter is shown in Figure 3.1.

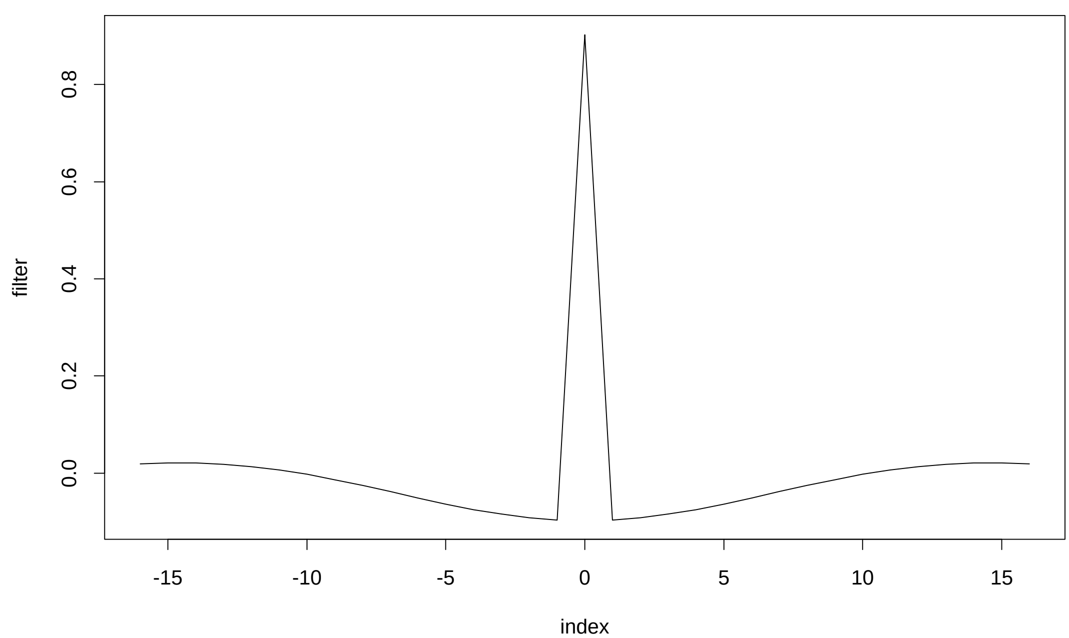


Figure 3.1: 32-point filter.

In Figure 3.6 we present the filtered simulated *underlying signal-path effects sequence* $\tilde{\mathbf{s}}_k$, $k = 1, ..., q = 5$. Figure 3.7 shows the spectrum of each $\tilde{\mathbf{s}}_k$. We note that the filtered data indeed lacks the obvious low frequencies. In Figure 3.8 we present the simulated data $\mathbf{y}_k$, $k = 1, ..., q$, generated using the filtered simulated *underlying signal-path effects sequences* and the *pulse sequence* $\mathbf{a}$, described above. In Figure 3.9 we present the spectra of the simulated data $\mathbf{y}_k$. From the spectra we see that each $\mathbf{y}_k$, $k = 1, ..., q$, has the spectral modulation that is characteristic of seismic data generated by ripple-fired mining explosions [1], [19], [34], and [35]. The term modulation refers to the pattern of peaks and valleys seen in the spectra of each series $\mathbf{y}_k$. The data presented in Figure 3.8 are the data we use in the next section to show the results of our Bayesian deconvolution method using the Gibbs sampler.

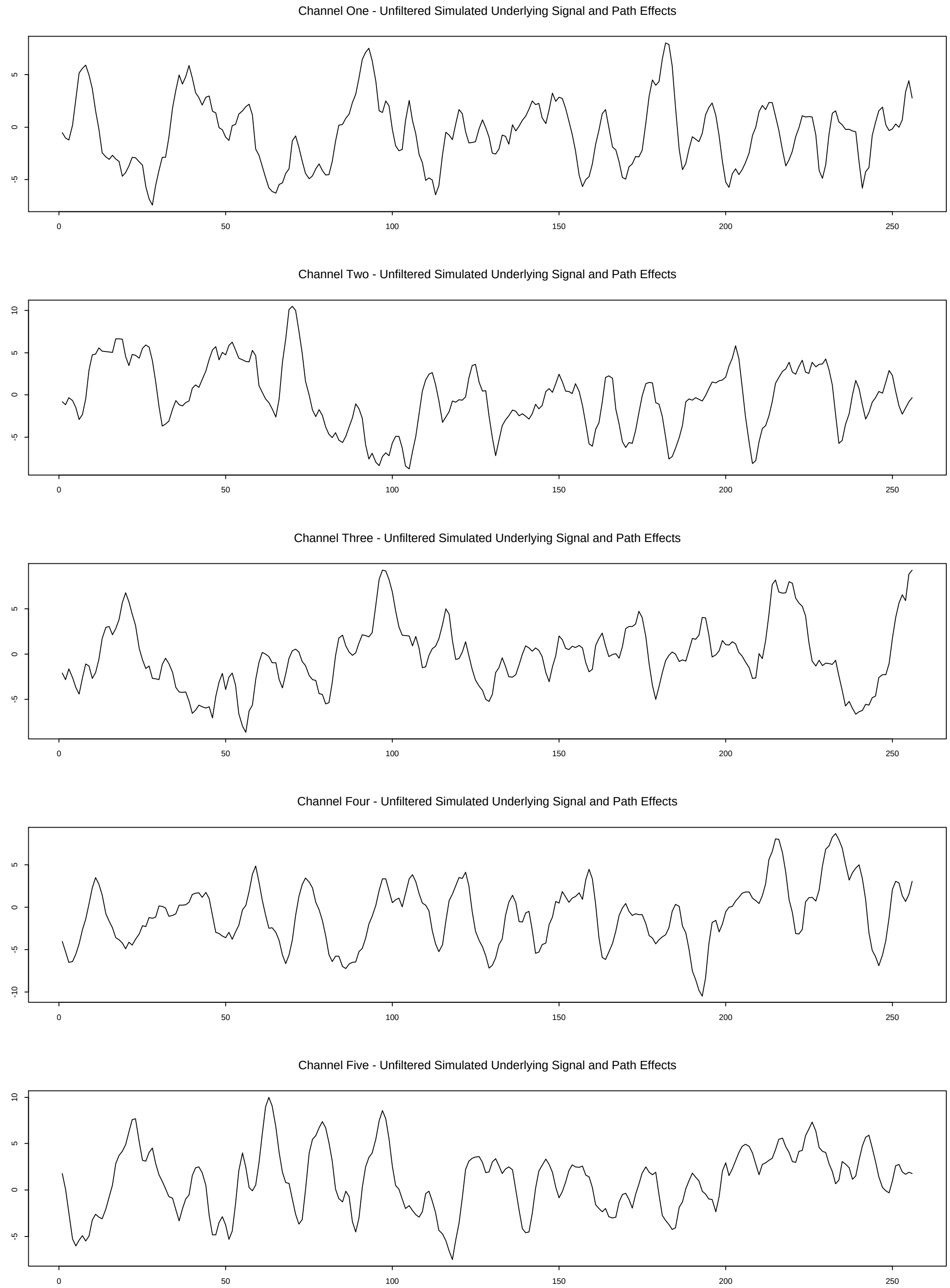


Figure 3.2: Unfiltered simulated underlying signals.

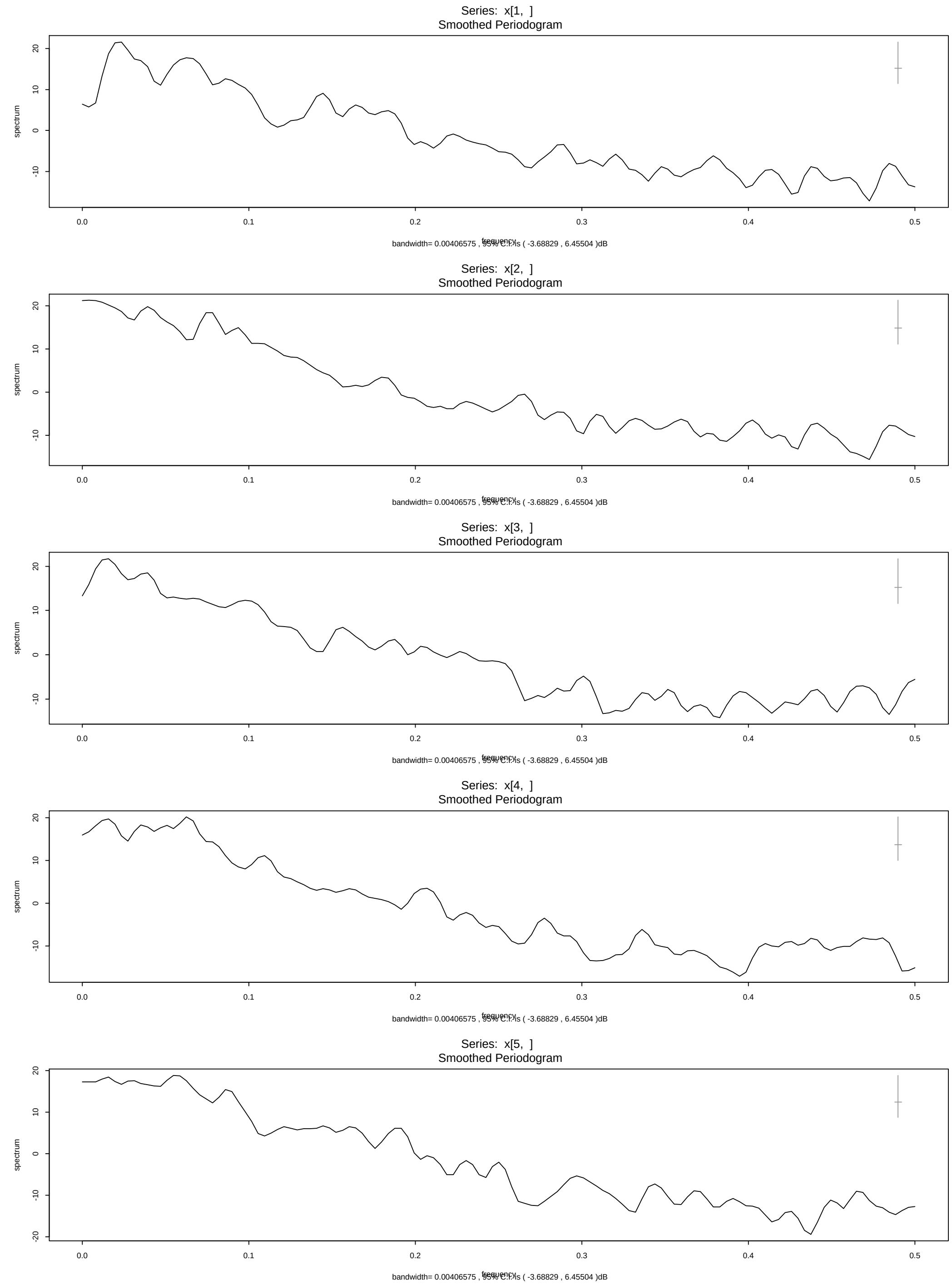


Figure 3.3: Spectra of the unfiltered simulated underlying signals.

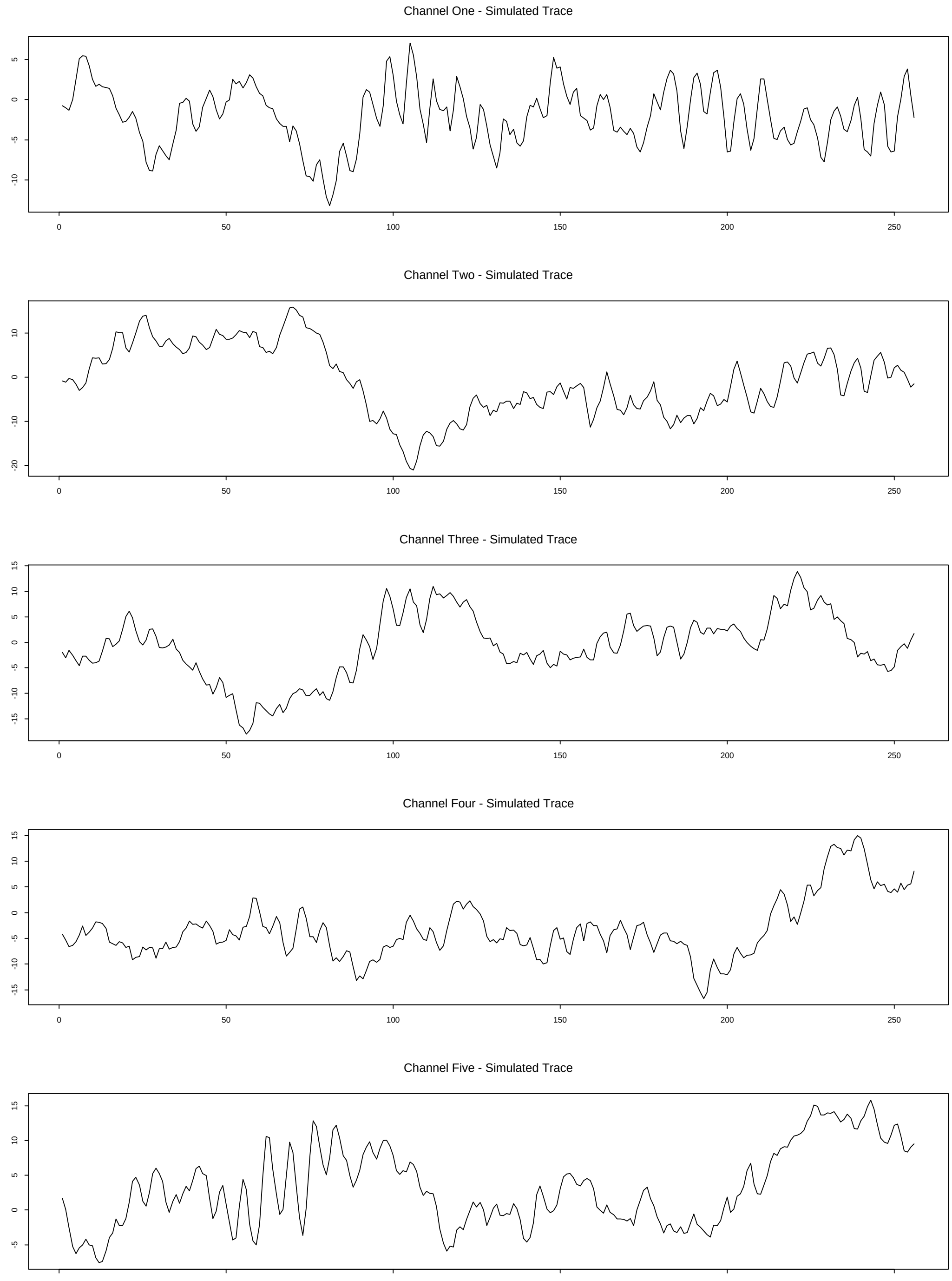


Figure 3.4: Simulated ripple-fired data traces, using unfiltered signals. Note that low frequencies are contained in these series.

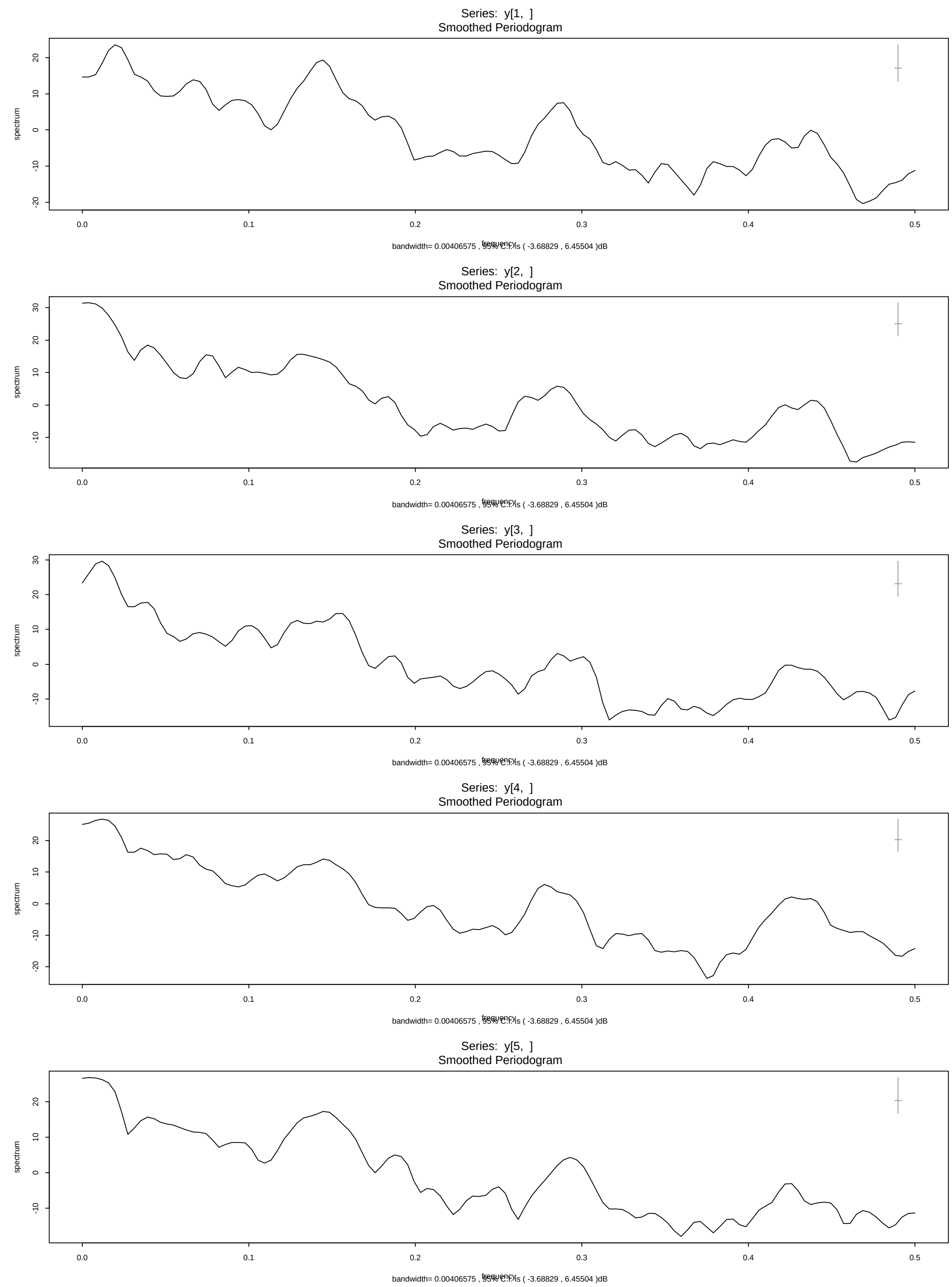


Figure 3.5: Spectra of the simulated ripple-fired traces. Note modulation in the spectra.

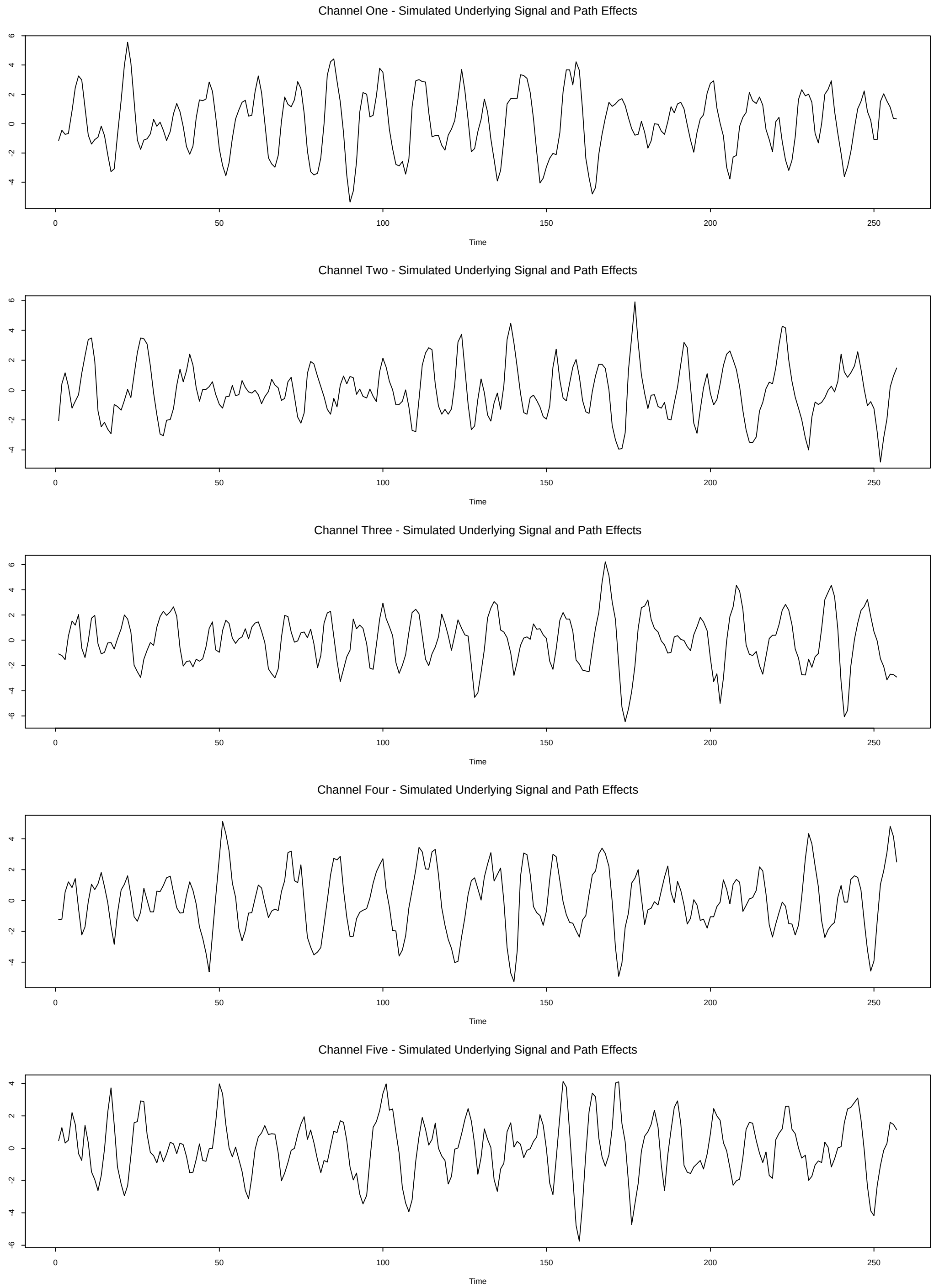


Figure 3.6: Filtered simulated underlying signals.

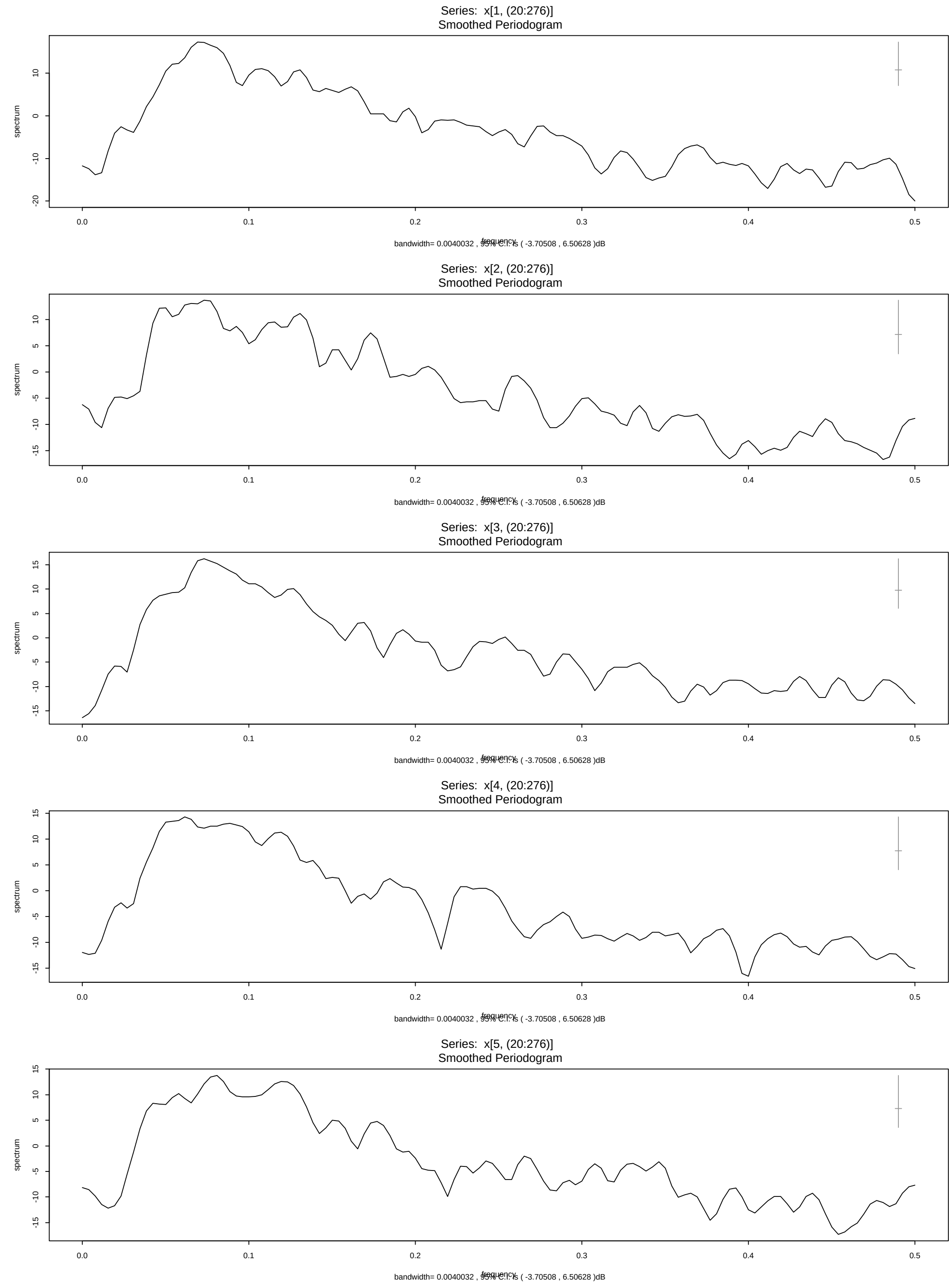


Figure 3.7: Spectra of the filtered simulated underlying signals.

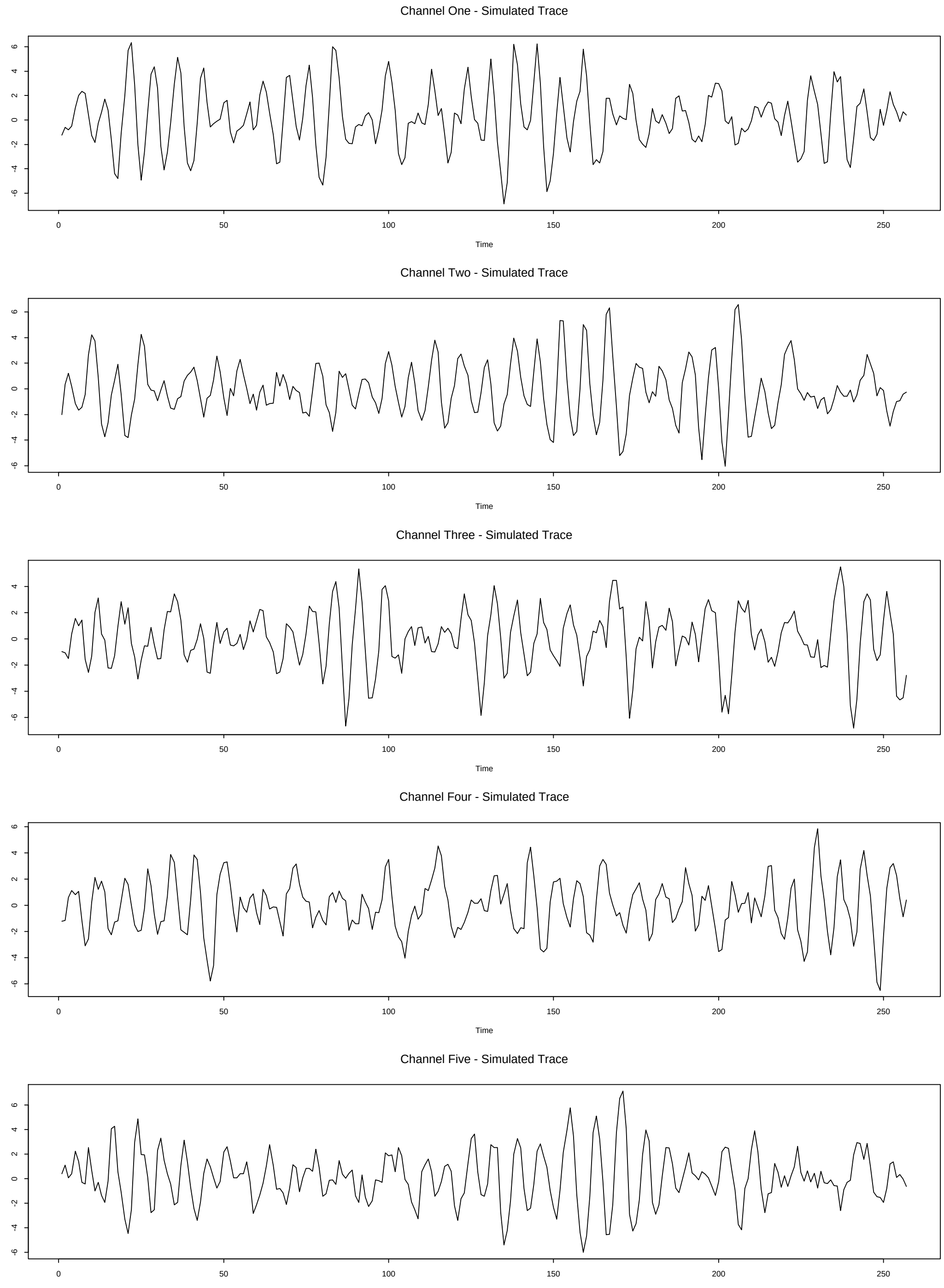


Figure 3.8: Simulated ripple-fired data traces, using the filtered signals. Note these series do not contain obvious low frequencies.

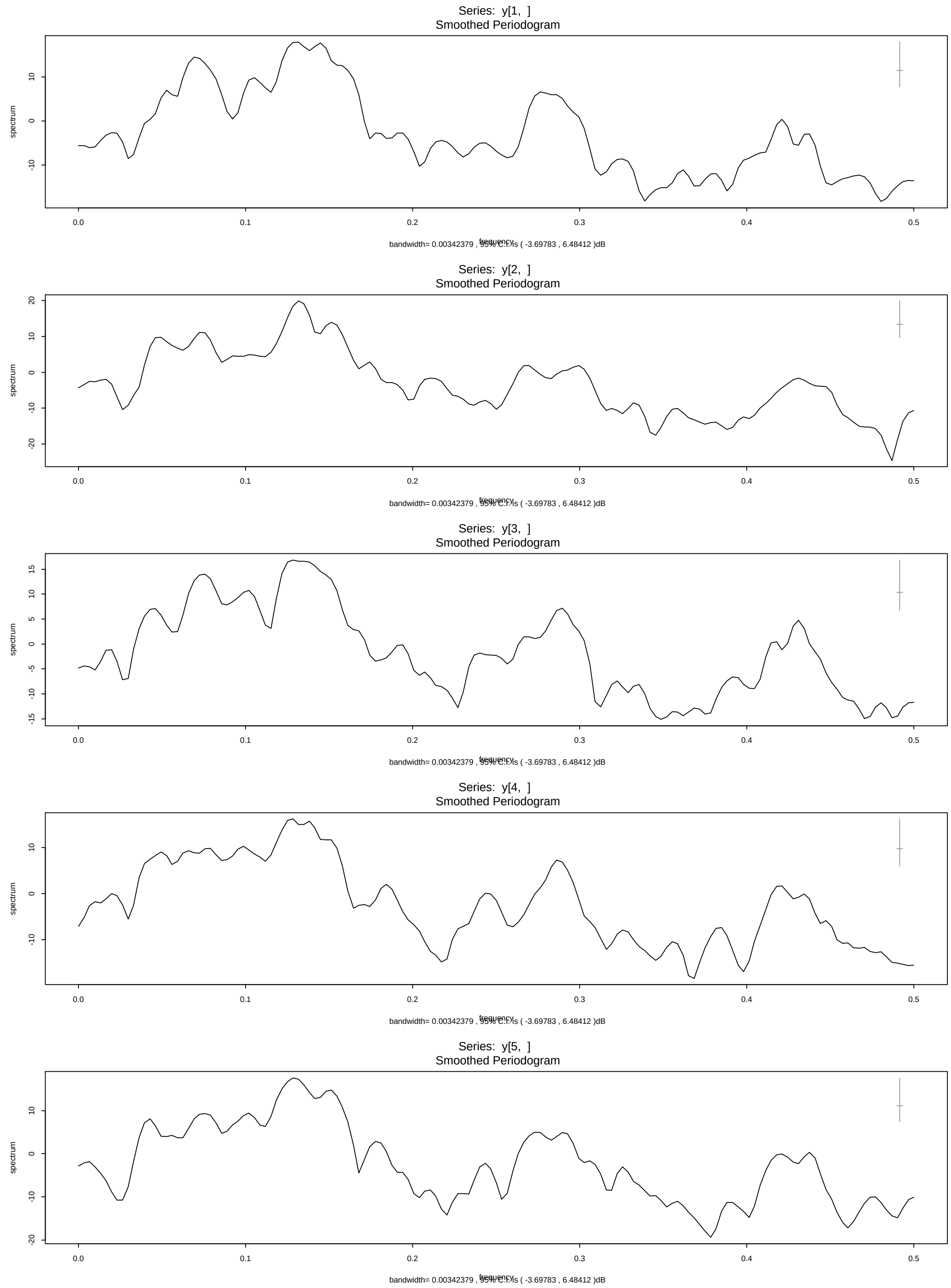


Figure 3.9: Spectra of the simulated ripple-fired data traces. Note the modulation in the spectra.

3.2 Implementation of the Gibbs Sampler

In this section we implement the Gibbs sampler in the deconvolution problem using the simulated data described in Section 3.1. Suppose we are presented with the seismic data in Figure 3.8 and suspect it was produced by a ripple-fired seismic event. To determine if there is evidence in the data for ripple-firing, we apply our Bayesian deconvolution method. Recall the convolution model:

$$y_k(t) = s_k(t) + \sum_{j=1}^{m} a_j s_k(t-j) + \varepsilon_k(t), \tag{3.7}$$

where both the *pulse sequence* $\mathbf{a}$ and the *underlying signal-path effects sequences* $\mathbf{s}_k$, $k = 1, ..., q = 5$, are unknown and assumed to be random. Now to approach the problem of deconvolving $\mathbf{a}$ and $\mathbf{s}_k$, $k = 1, ..., q$, from a Bayesian perspective, we develop prior distributions and produce posterior estimates based on the posterior distributions. In the end, we examine the estimated *pulse sequence* $\hat{\mathbf{a}}$. If $\hat{\mathbf{a}}$ has multiple nonzero values then we have evidence that the event in question was produced by ripple-fired mining explosion and not a nuclear explosion. If $\hat{\mathbf{a}}$ contains all 0's, then the event in question was produced by a single explosion, i.e., possibly a low level nuclear explosion or earthquake, and in this latter case we will need to apply alternative methods to determine the cause of the event.

We begin by describing how the hyperparameters used in the prior distributions are determined for η, $\boldsymbol{\phi}$, and τ. The distribution we use to model the prior uncertainty in η, the probability of seeing a 0 in the *pulse sequence* is beta, given in (2.22). To choose our prior on η for the simulation, suppose η is most likely 0.9 and that we are 95% certain η is greater than 0.80. Restated, we choose β_1 and β_2 such that the mode of the beta prior is 0.9 and that there is a 95% probability η is greater than 0.80. This specification leads to $\beta_1 = 43$ and $\beta_2 = 17/3$, which is presented in

Figure 2.1.

For the joint prior distribution of the autoregressive coefficients $\boldsymbol{\phi}$ we use a p-variate normal given in (2.23). We choose our prior mean $\boldsymbol{\phi}_0$ for the simulation to be $\mathbf{0}$ and the covariance $\Sigma_\phi = I\sigma_\phi^2$, where $\sigma_\phi^2 = 0.5^2$. This assumes that our prior guesses for ϕ_1, ϕ_2, and ϕ_3 are independent and have the same uncertainty. This prior distribution results from a specification that ϕ_i, $i = 1, 2, 3$, each have a mean value of 0 and that we are 95% sure each ϕ_i falls between -1 and 1.

For the precision, $\tau = 1/\sigma^2$, of the *underlying signal-path effects sequences* $\mathbf{s}_k$, $k = 1, ..., q$, we choose the following prior distribution. Suppose the value of τ is approximately 1 and is most likely less than 3. To reflect this we choose $\gamma_1 = 5$ and $\gamma_2 = 4$ for our gamma prior, which is presented in Figure 2.23.

For the hyperparameters that determine the truncated normal distribution of the nonzero values in the Bernoulli-Gausssian model of the *pulse sequence* $\mathbf{a}$, we choose $\mu_\alpha = 0.7$ and $\sigma_\alpha^2 = 0.15^2$. This prior specification is derived from the assumption that the mean value of a_j is 0.7 and that we are 95% certain the nonzero values a_j are between 0.5 and 1.

Finally, we specify the fixed parameters c and m in the model. For the simulation results presented here we used $c = 0.01$ and $m = 30$. The values of c and m assumed for these fixed parameters are the values used to generate the simulated data. This is the ideal situation and for presenting the results that show the capability of this method we believe this illustrates its full ability.

The next step in our Bayesian analysis is to run the Gibbs Sampling program to produce the simulated Markov chains for the parameters in the model. First, we choose the starting values

$$\boldsymbol{\Theta}^{(0)} = \{\eta^{(0)}, \mathbf{a}^{(0)}, \boldsymbol{\phi}^{(0)}, \tau^{(0)}, \mathbf{S}^{(0)}\}, \tag{3.8}$$

(1) for $\eta^{(0)}$ we start with the prior mode 0.9, (2) for the *pulse sequence* $\mathbf{a}^{(0)} = \mathbf{0}$, which indicates no delays are suspected to start, (3) for $\boldsymbol{\phi}^{(0)}$ we let $\boldsymbol{\phi}^{(0)} = \mathbf{0}$, the prior mean (and mode), (4) for $\tau^{(0)}$ we start with $\tau^{(0)} = 1$ again the prior mode, and (5) for $\mathbf{S}^{(0)}$ we start with $\mathbf{S}^{(0)} = \mathbf{0}$ since we do not know anything beforehand about the *underlying signal-path effects sequences*.

Second, the choice of how long to "burn in" the Markov chains and the total number of repetitions we generate was decided upon by examining the chains for convergence. The determination that the chains have converged is difficult due to the large number of parameters and the observed slow mixing of the Markov chains produced by the Gibbs sampler for the sets of parameters $\mathbf{a}$ and $\mathbf{S}$. To determine convergence we examine the m chains produced for the *pulse sequence* $\mathbf{a}$. We stop when the chains appear to stabilize for 5,000 iterations. After much experimentation, we found that running the Gibbs sampler a total of 40,000 repetitions and using the last 5,000 iterations produces reasonably stable chains for the *pulse sequences*. For the remaining parameters, in the cases we have considered with simulated data, a "burn in" of 35,000 repetitions is considered sufficient for convergence. To test convergence we also ran the program three additional times with different seeds for the random number generator. Two of the three runs produced very similar delay structures.

Last, we calculate posterior estimates of the parameters by calculating the posterior means. Point estimates are produced by taking the means of the Markov chains for all parameters after the "burn in" period. We choose to use the posterior mean because it gives essentially the same answers as the posterior median or mode in all cases we have examined. Density estimates are calculated using the last 5,000 iterations of the Markov chains. Interval estimates are also calculated using the 0.025 and 0.975 quantiles of the Markov chains. In the next section we present the results of the deconvolution of our simulated data.

3.3 Simulation Results

3.3.1 Results from Simulated Data Containing Common Delay Structure

We now present the results from one run of our Gibbs sampling program on the simulated data constructed with a common delay pattern. In Figures 3.10, 3.11, and 3.12 we present the "Truth" for the simulated data we analyze. In Figure 3.10 we re-present the simulated traces, $\mathbf{y}_k$, recorded by $q = 5$ sensors. In Figure 3.11 we re-present the "True" *underlying signal-path effects sequences.* In the top panel of Figure 3.12 we present the "True" *pulse sequence* that contains the information regarding the delay structure contained in the simulated data $\mathbf{y}_k$, $k = 1, ..., q$, specifically with delay times at 7, 14, 21, and 28 points. Note that for real data these will not be available, however here we will use them to check the ability of our deconvolution method to reproduce the delay structure contained in the *pulse sequence* and the *underlying signal-path effects sequences* on each of the k channels.

The estimated values of the parameters of interest are presented in Table 3.1. The estimates of the nonzero values of a_j, the estimates of η, $\boldsymbol{\phi}$, τ, and also σ^2 are presented. The means are calculated as the point estimates of the deconvolved parameters in the model and the medians are calculated for comparison. The standard deviations are calculated and the 95% posterior intervals are calculated using the 0.025 and 0.975 quartiles. In the second panel of Figure 3.12, a graphical display of the estimates of the *pulse sequence* $\hat{\mathbf{a}}$ is given. The most important observation we make about the estimated values of $\mathbf{a}$ is that the delay times are reproduced; there are nonzero values at time 7, 14, 21, and 28 points. These estimates are systematically smaller than the true values which can be seen most clearly in Table 3.1 in the interval estimates. The estimates of the *pulse sequence* are influenced by the estimate of τ, which is too large here.

To determine the goodness-of-fit of the model to the simulated data, in Figure 3.13 we present the "True" simulated *underlying signal-path effects sequences* and the estimates $\hat{\mathbf{s}}_k$ shifted one point to the left. All of the deconvolved estimates of the *underlying signal-path effects sequences* $\mathbf{s}_k$, $k = 1, ..., q$, are very close to the "Truth." This result, along with the closeness of the estimates to $\mathbf{a}$, leads us to believe the deconvolution has been successful.

In Figures 3.14 and 3.15 we make crude estimates of the simulated traces $\mathbf{y}_k$ and of the additive noise $\boldsymbol{\varepsilon}_k = [\varepsilon_k(1), ..., \varepsilon_k(n)]'$. In Figure 3.14 we present the simulated traces $\mathbf{y}_k$ and the crude estimates,

$$\hat{y}_k(t) = \sum_{j=0}^{m} \hat{a}_j \hat{s}_k(t-j),$$

shifted one point to the left. The fitted values are quite close in the range $l = m+p = 33$ to $n = 256$, which is the range we use to calculate the likelihood. In Figure 3.15 we present crude estimates of the additive *noise* $\boldsymbol{\varepsilon}_k(t)$, $k = 1, ..., q$, for the points $l = m+p = 33$ to $n = 256$. We calculate the estimated error terms by

$$\hat{\varepsilon}_k(t) = y_k(t) - \hat{y}_k(t)$$

for $k = 1, ...q$, and $t = 1, ...n$. The estimated error terms appear reasonable. Overall, the fit of the model appears encouraging.

The remaining discussion in this section relates to the remaining figures. First, we present the graphs for the *pulse sequence* $\mathbf{a}$. Figure 3.16 contains plots of the final 5000 repetitions of the nonzero values of $\mathbf{a}$. The chains are for a_7, a_{14}, a_{21}, and a_{28}. All of the chains appear to be stable, which is further supported by Figure 3.17.

In Figure 3.17 we present, down the columns, histograms of every 1000 iterations after "burn in" for the Markov chains of a_7, a_{14}, a_{21}, and a_{28}. For each column the histograms are centered around the same values. Figure 3.18 presents posterior density estimates of each a_j, $j = 1, ..., m = 30$. From this Figure the times of the nonzero values of a_j are made clear.

Figures 3.22 and 3.24 present the same set of graphs for the parameters η and τ and in Figures 3.19 and 3.21 similar graphs are presented for $\boldsymbol{\phi}$. Overall, these pictures show the convergence of the Markov chains for these parameters.

In conclusion, three main lessons were learned through the development of the Gibbs sampling program using simulated data. First, due to the slow mixing of the Markov chains for $\mathbf{a}$ and $\mathbf{S}$, many repetitions are needed along with a long "burn in" period. Second, the estimates of the nonzero *pulse sequence* values tend to be underestimated, however, the delay times are identified accurately. Last, the *underlying signal-path effects sequences* are estimated very well and the estimated *additive noise* is small. The accuracy in estimating the *pulse sequence* and the *underlying signal-path effect sequences* demonstrates the ability of our deconvolution methods to identify ripple-firing.

3.3.2 Results from Simulated Data Containing No Delays

Here we present results for simulated data with no delayed signals. To ensure that the estimation procedure behaves correctly when no delays are present in the simulated data, we created a data set $\mathbf{Y}$ based on the filtered simulated *underlying signal-path effect sequences* presented earlier. In Figure 3.25 we present the data which looks like Figure 3.11 with small error terms added. The top panel of Figure 3.26 shows the "True" *pulse sequence* that contains all zero values for $j = 1, ..., m = 30$. In

posterior	mean	median	s.d.	$(p_{0.025}, p_{0.975})$	"Truth"
a(7)	0.628	0.623	0.020	(0.602, 0.665)	0.80
a(14)	0.576	0.574	0.029	(0.536, 0.635)	0.70
a(21)	0.495	0.495	0.035	(0.443, 0.567)	0.60
a(28)	0.415	0.418	0.021	(0.378, 0.444)	0.50
ϕ_1	-1.199	-1.199	0.030	(-1.248, -1.148)	-1.646
ϕ_2	0.534	0.534	0.044	(0.461, 0.606)	0.904
ϕ_3	0.090	0.090	0.030	(0.040, 0.138)	-0.165
η	0.878	0.881	0.037	(0.812, 0.932)	0.867
τ	1.232	1.231	0.051	(1.151, 1.319)	1.000
σ^2	0.813	0.813	0.034	(0.758, 0.869)	1.000

Table 3.1: Posterior estimates of the relevant parameters for the simulated data.

the bottom panel of Figure 3.26 the estimated *pulse sequence*, $\hat{\mathbf{a}}$, has only one small nonzero value at a delay of 16 points. This run of the program shows our method does not produce nonzero delays when they are not present is the simulated data.

In Chapter 4 we present results from runs of our Gibbs Sampling program on real seismic data that resulted from a known ripple-fired mining explosion. We follow the same procedure with the real data as the one we developed using the simulated data in this chapter.

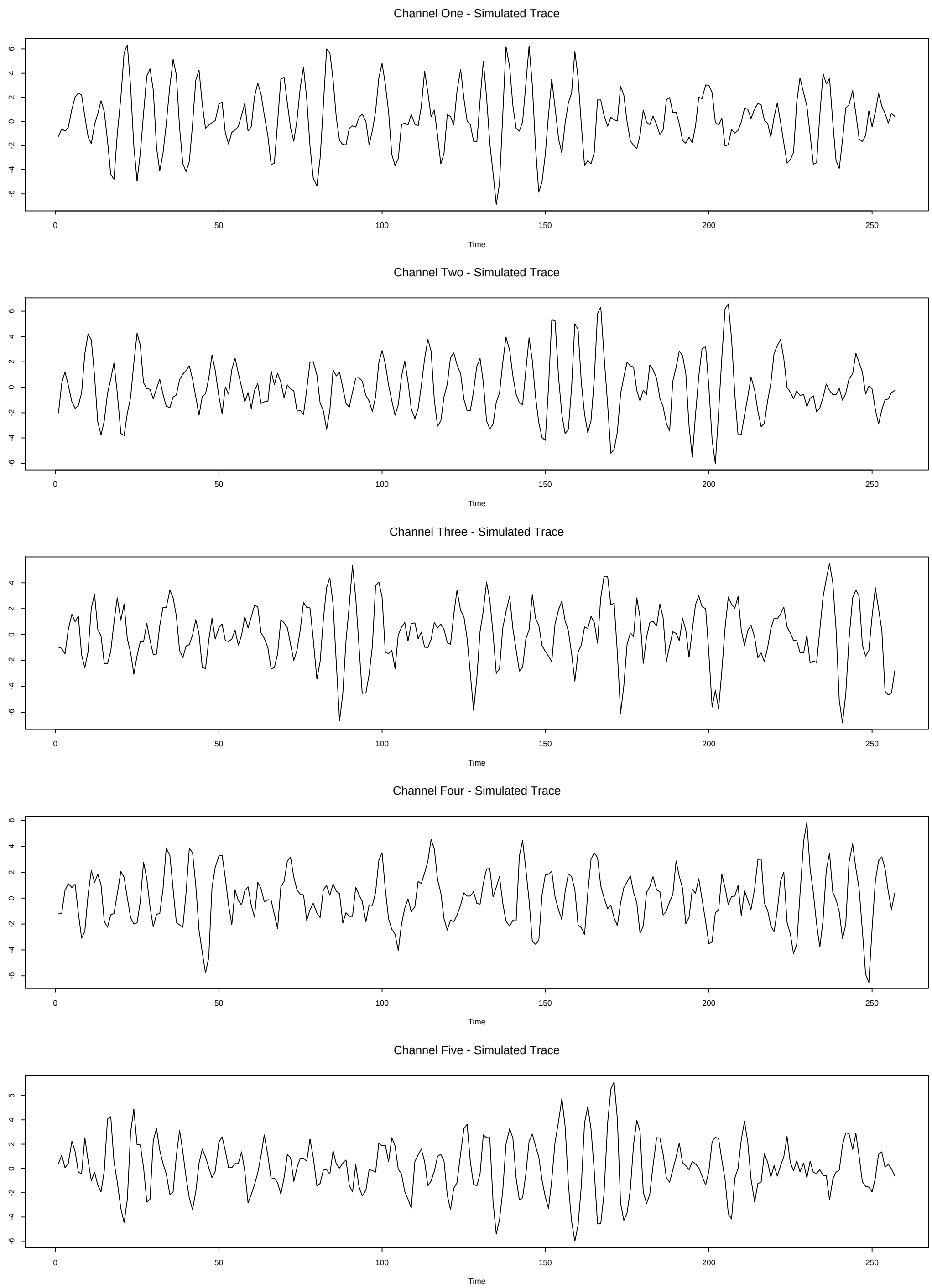


Figure 3.10: Simulated array data $\mathbf{y}_k$, $k = 1, ..., q$.

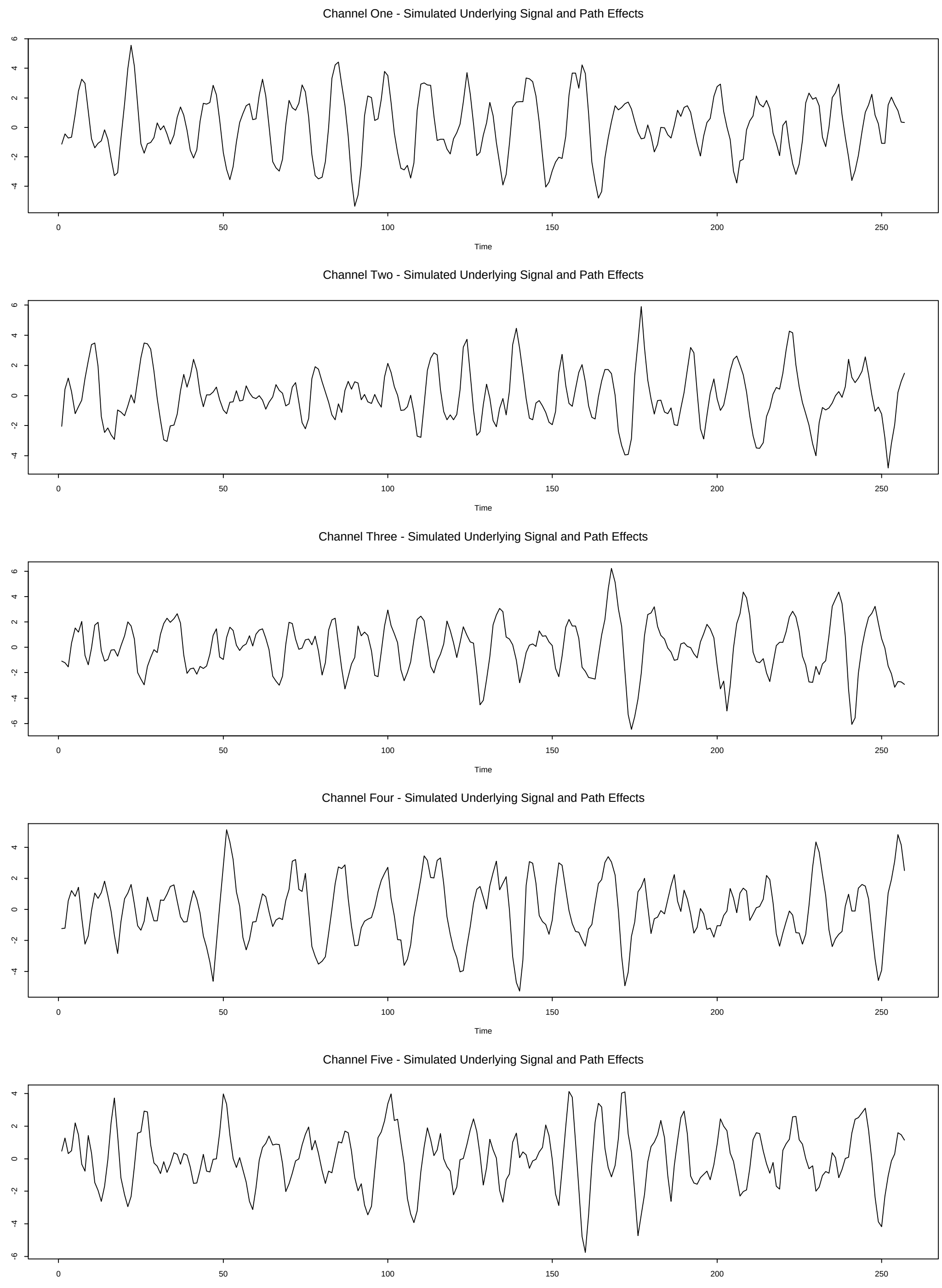
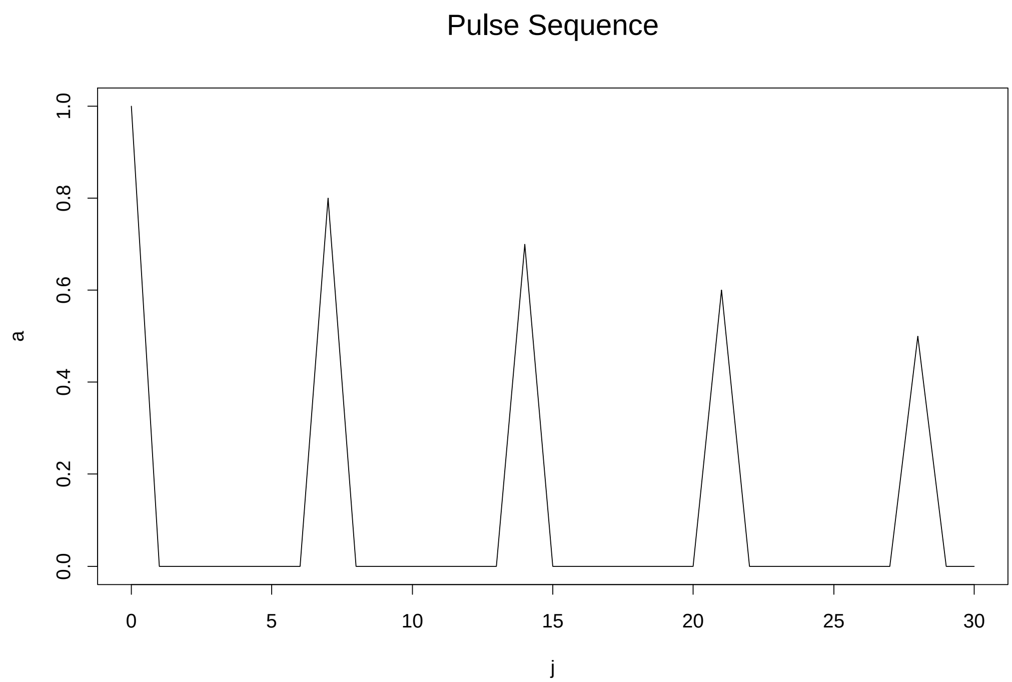


Figure 3.11: Simulated *underlying signal-path effects* $\mathbf{s}_k$, $k = 1, ..., q$.



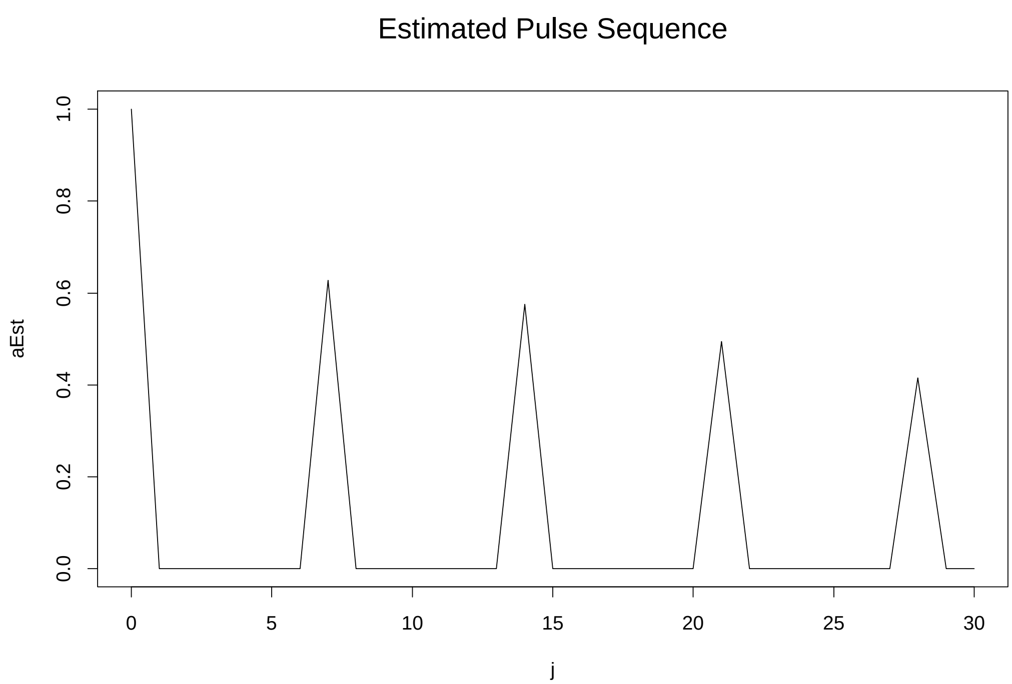


Figure 3.12: Top: True *pulse sequence*, **a**. Bottom: Estimates.

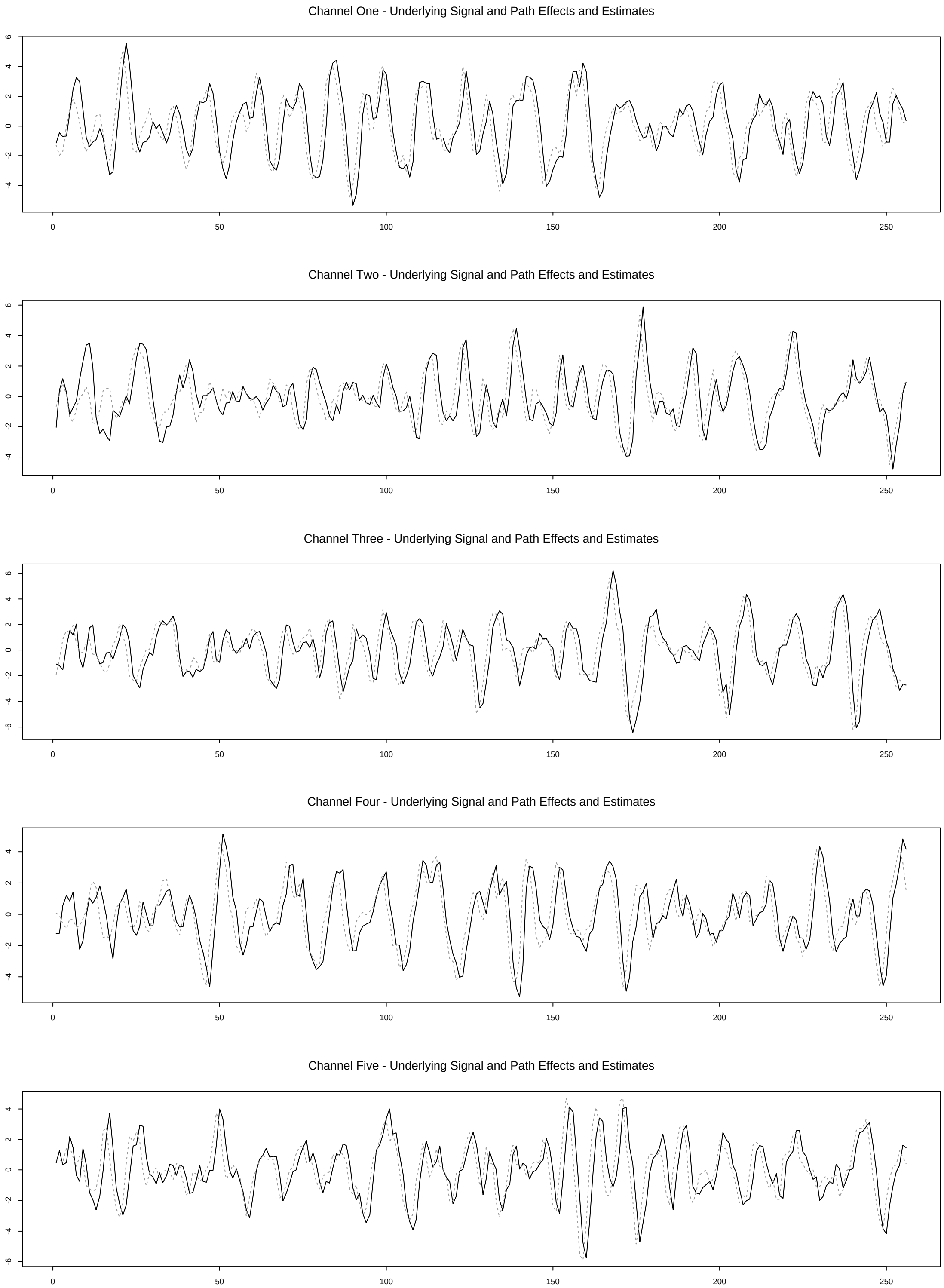


Figure 3.13: Solid: Simulated *underlying signals-path effects sequence* $\mathbf{s}_k$, $k = 1, .., q$. Dots: Estimates $\hat{\mathbf{s}}_k$ shifted one point to the left.

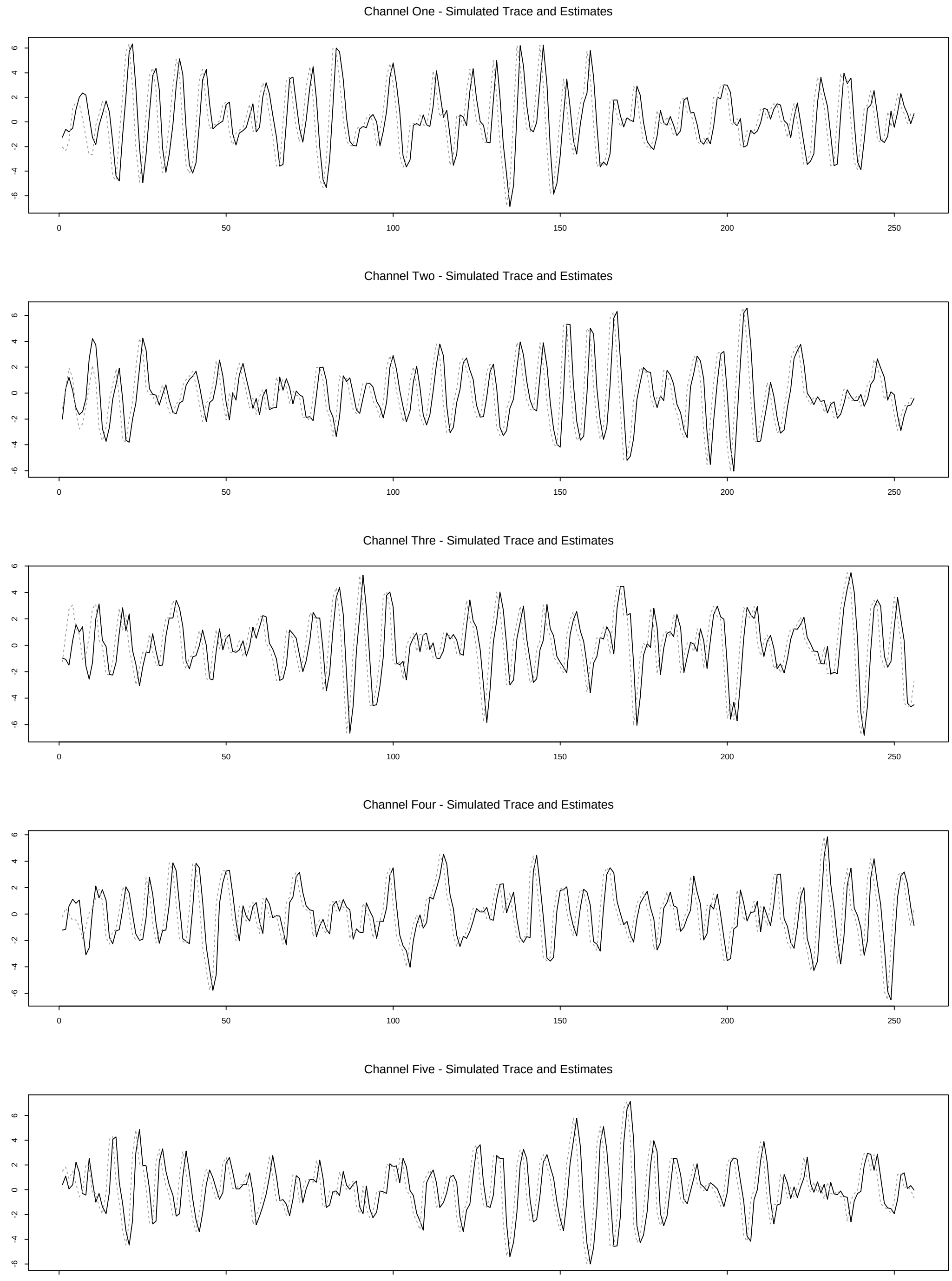


Figure 3.14: Solid: Simulated trace $\mathbf{y}_k$, $k = 1, ..., q$. Dots: Estimates, $\hat{\mathbf{y}}_k$, shifted one point to the left.

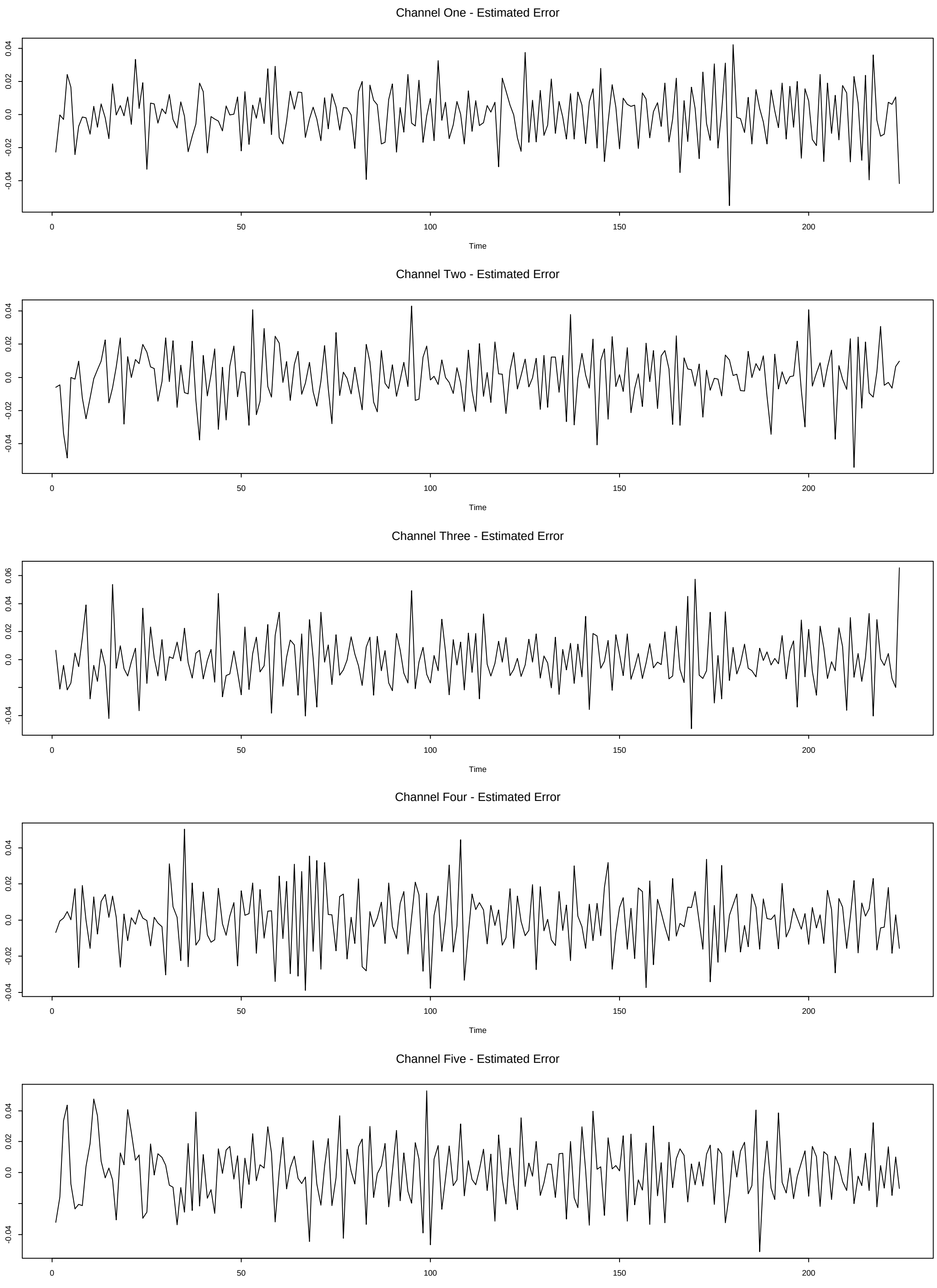


Figure 3.15: Estimated additive noise on each channel, $\boldsymbol{\varepsilon}_k(t)$, $t = (m-m), ..., (n-m)$.

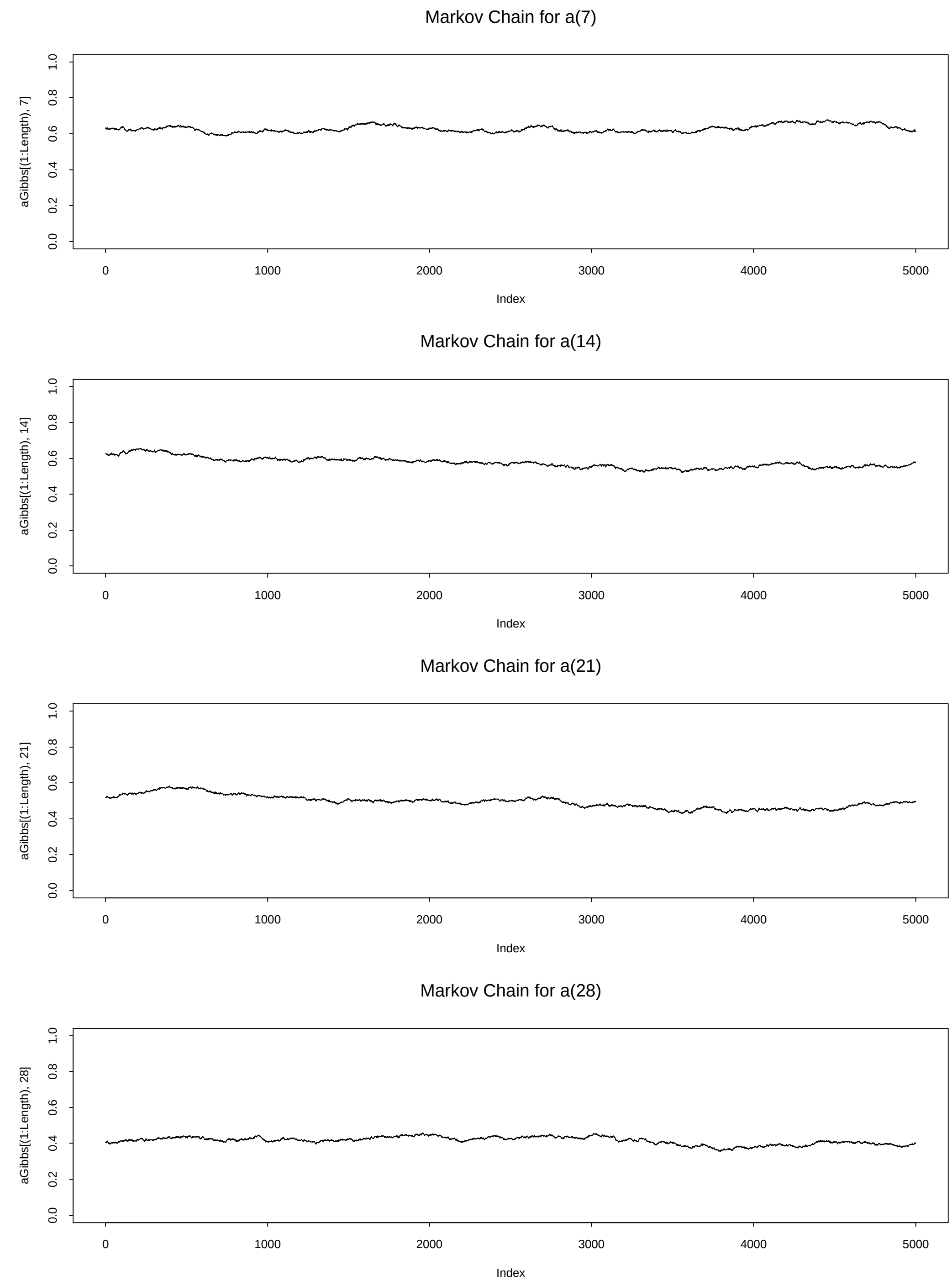


Figure 3.16: Last 5,000 iterations in the Markov chains produced for a_7, a_{14}, a_{21}, and a_{28}.

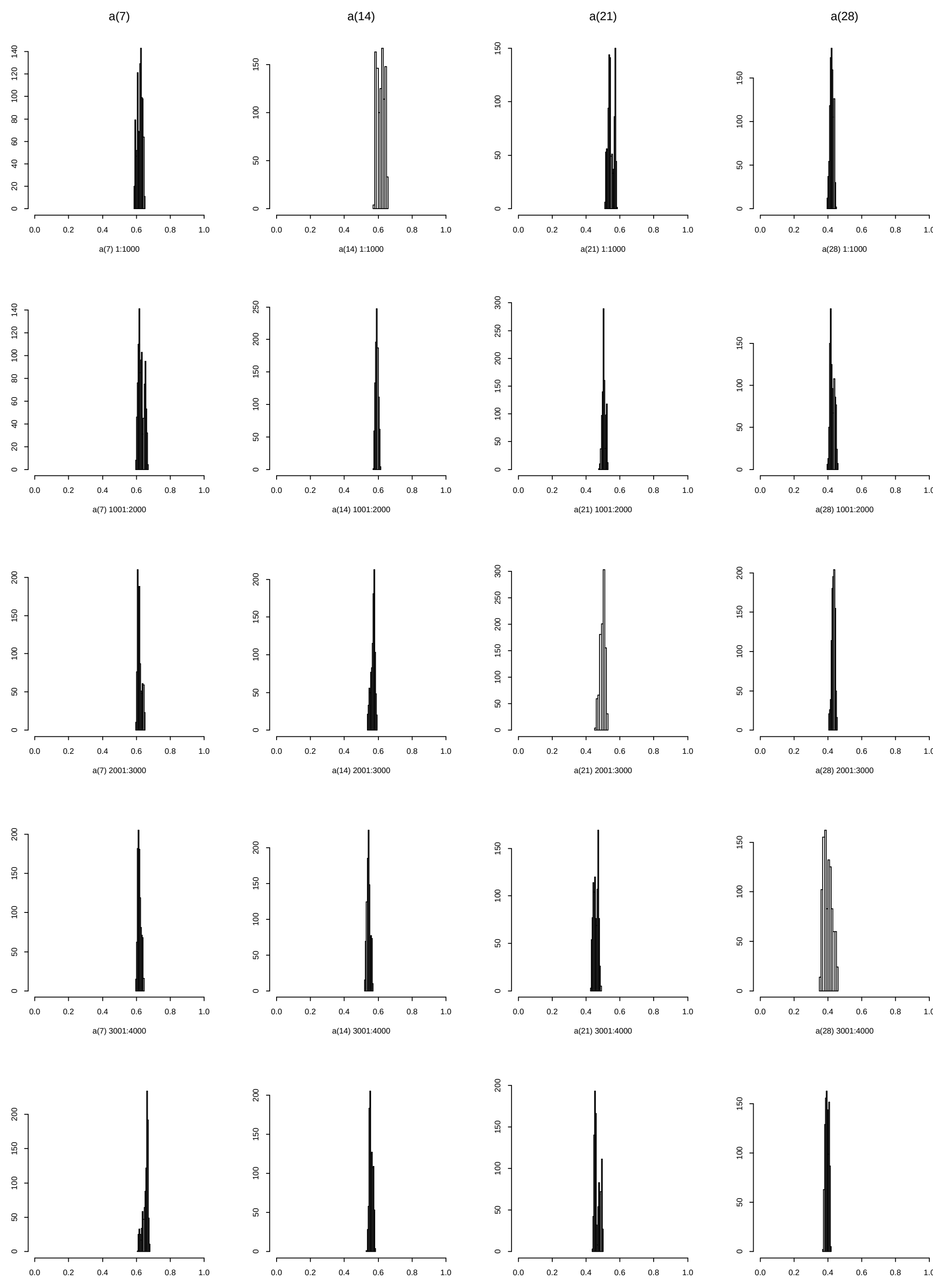


Figure 3.17: Histograms of every 1,000 iterations, after "burn in," for the nonzero values of **a**. The columns show the stability of the Markov chains for a_7, a_{14}, a_{21}, and a_{28}.

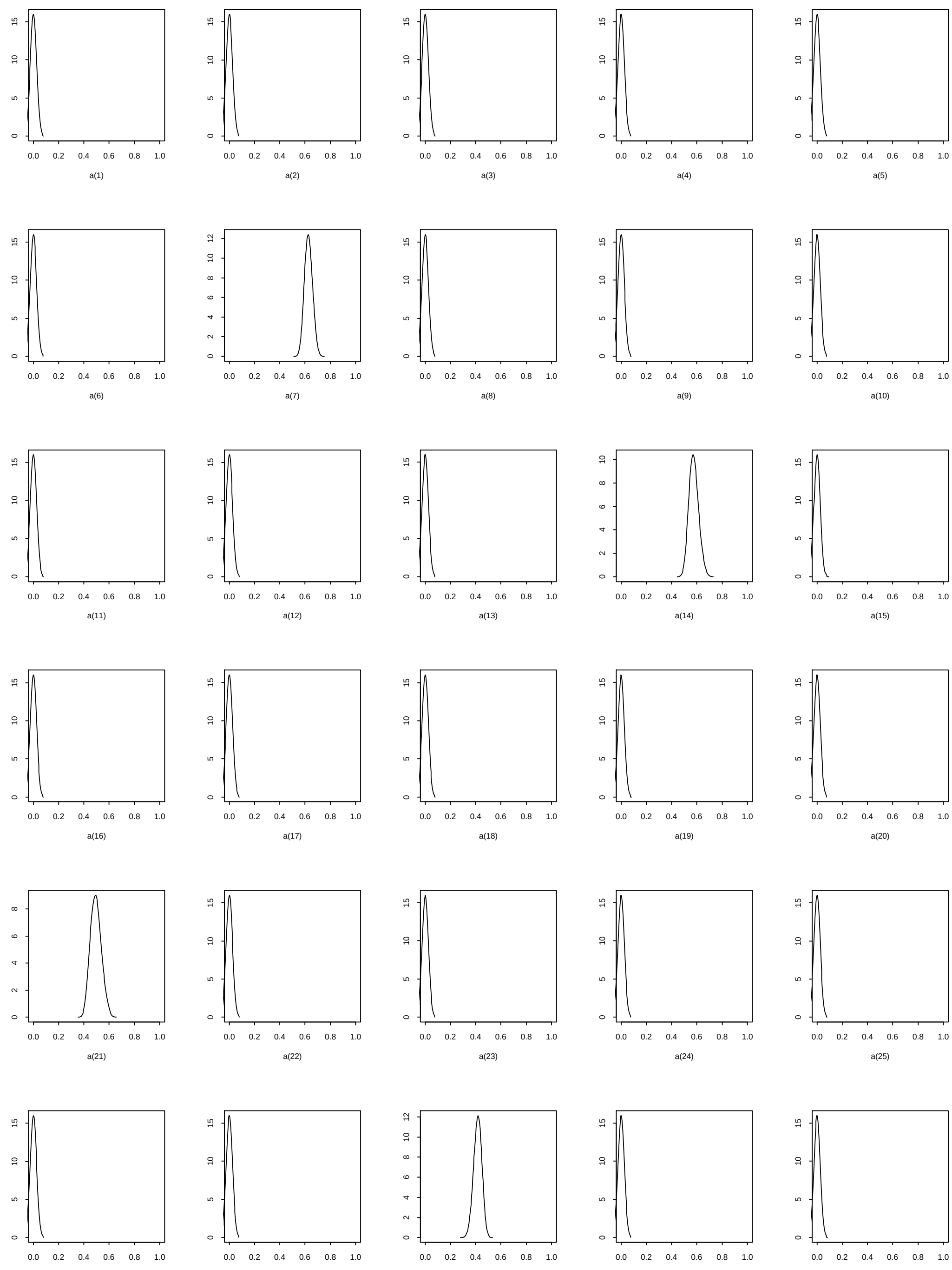


Figure 3.18: Posteriors of a_j, $j = 1, ..., m$.

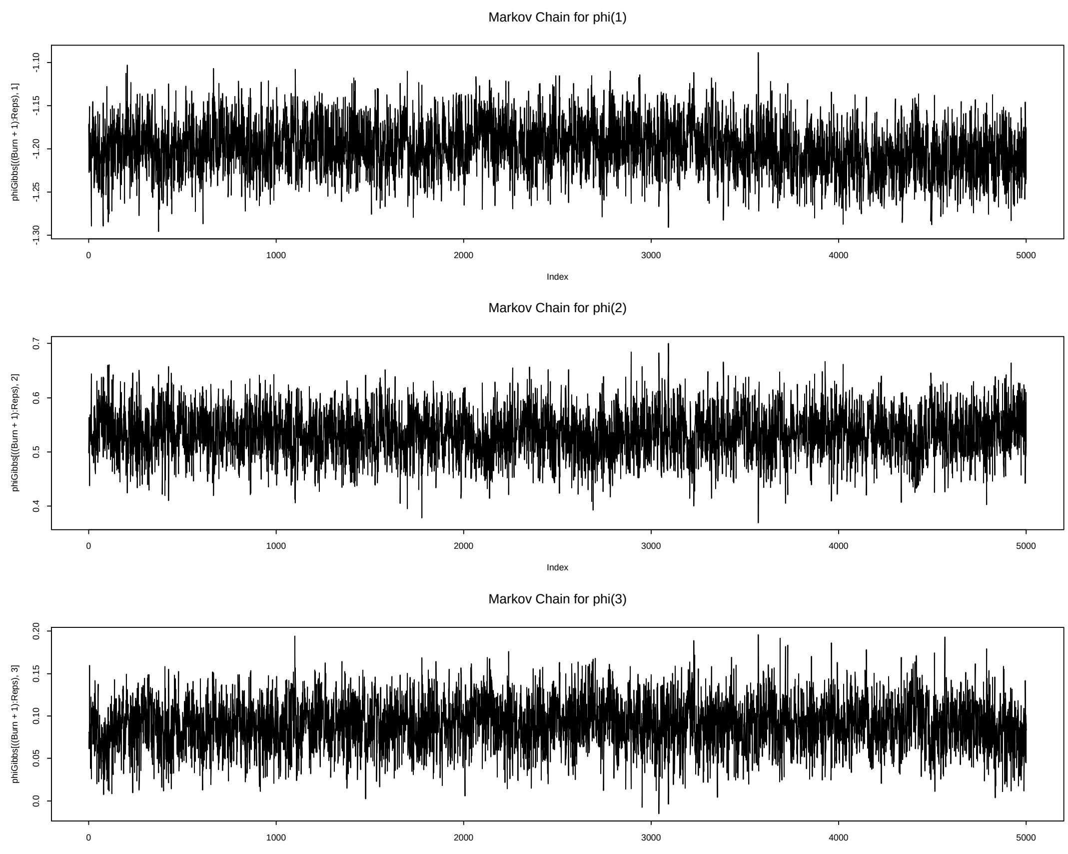


Figure 3.19: Last 5,000 iterations in the Markov chain produced for ϕ_1, ϕ_2, and ϕ_3.

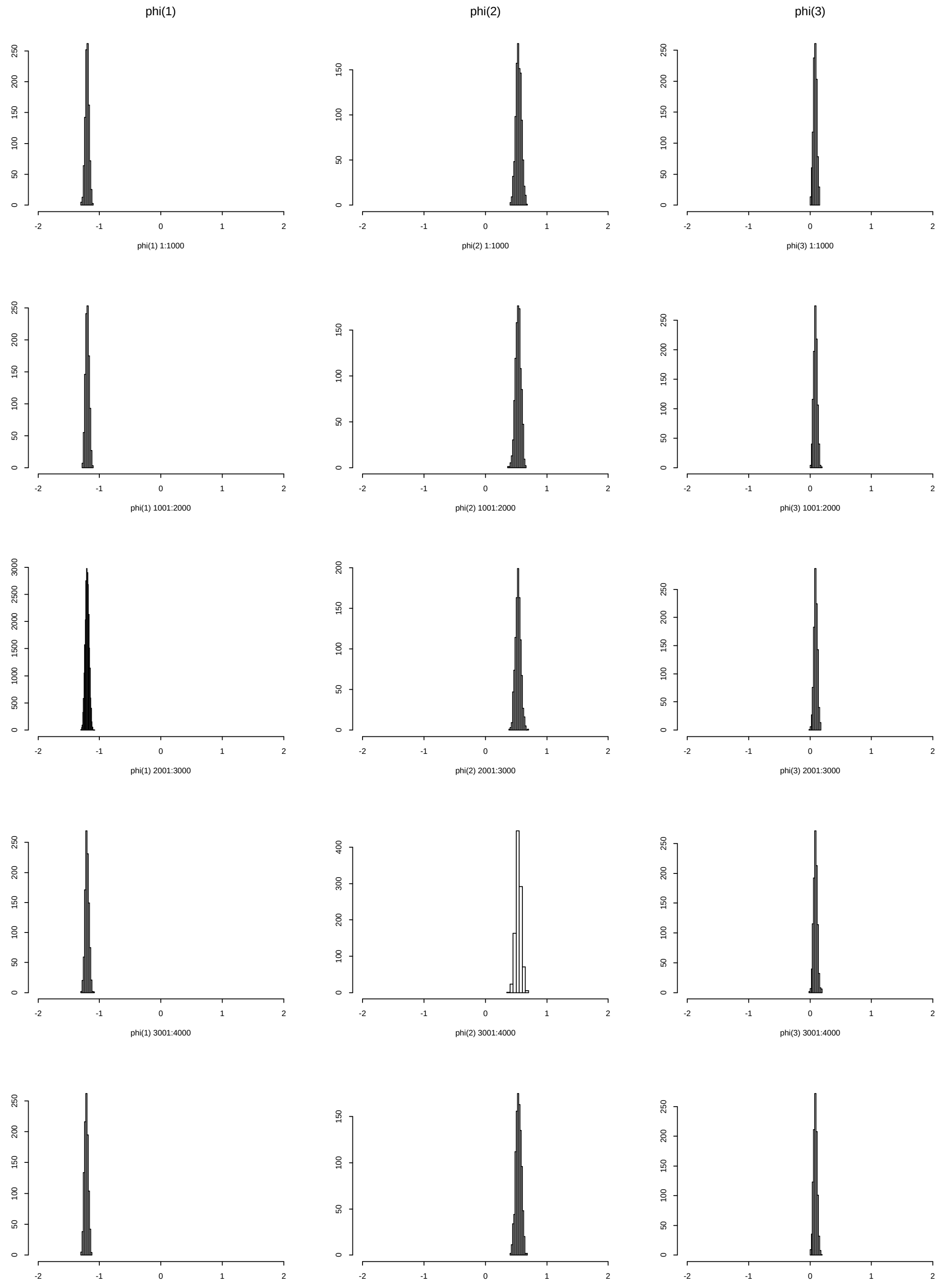


Figure 3.20: Histograms of every 1,000 iterations, after "burn in," for $\boldsymbol{\phi}$. The columns show the stability of the Markov chains for ϕ_1, ϕ_2, and ϕ_3.

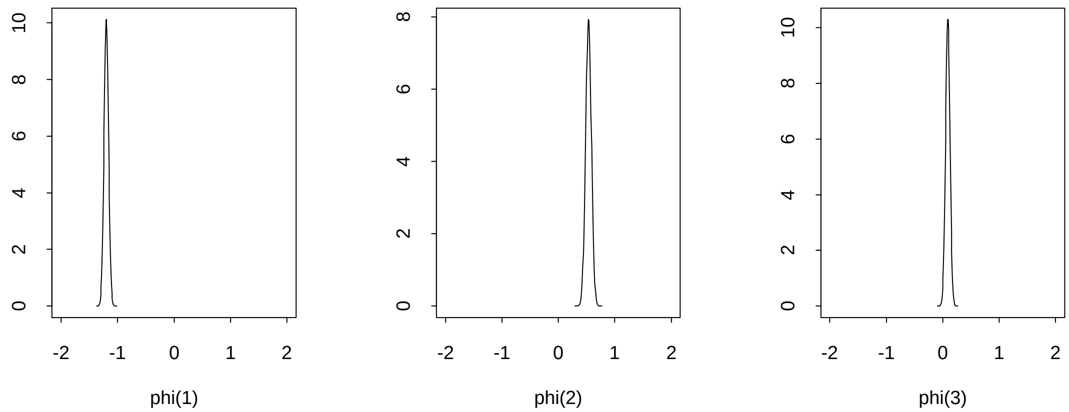


Figure 3.21: Posterior of ϕ_1, ϕ_2, and ϕ_3.

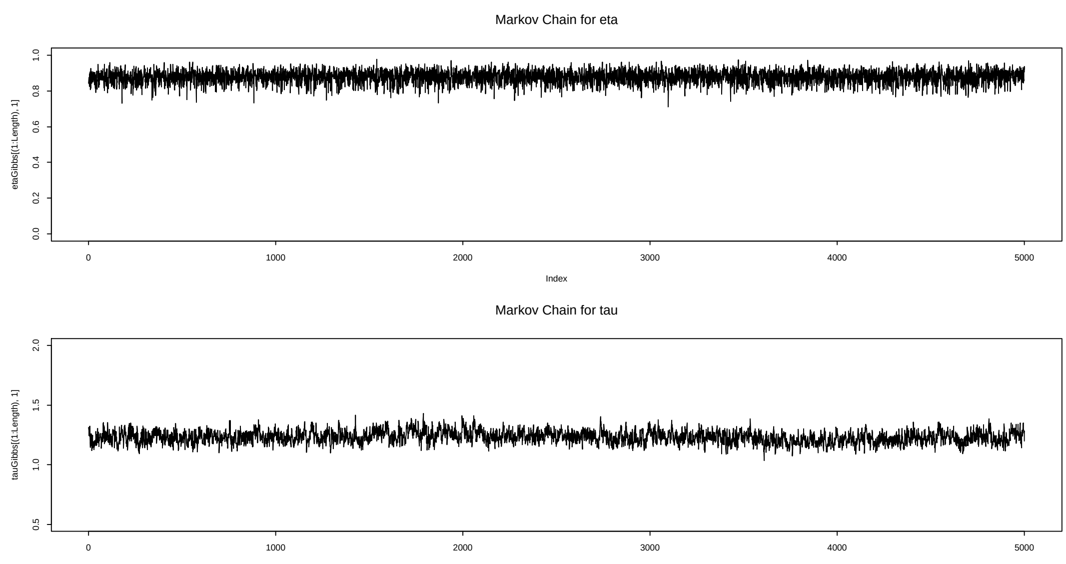


Figure 3.22: Last 5,000 iterations of the Markov chains for η and τ.

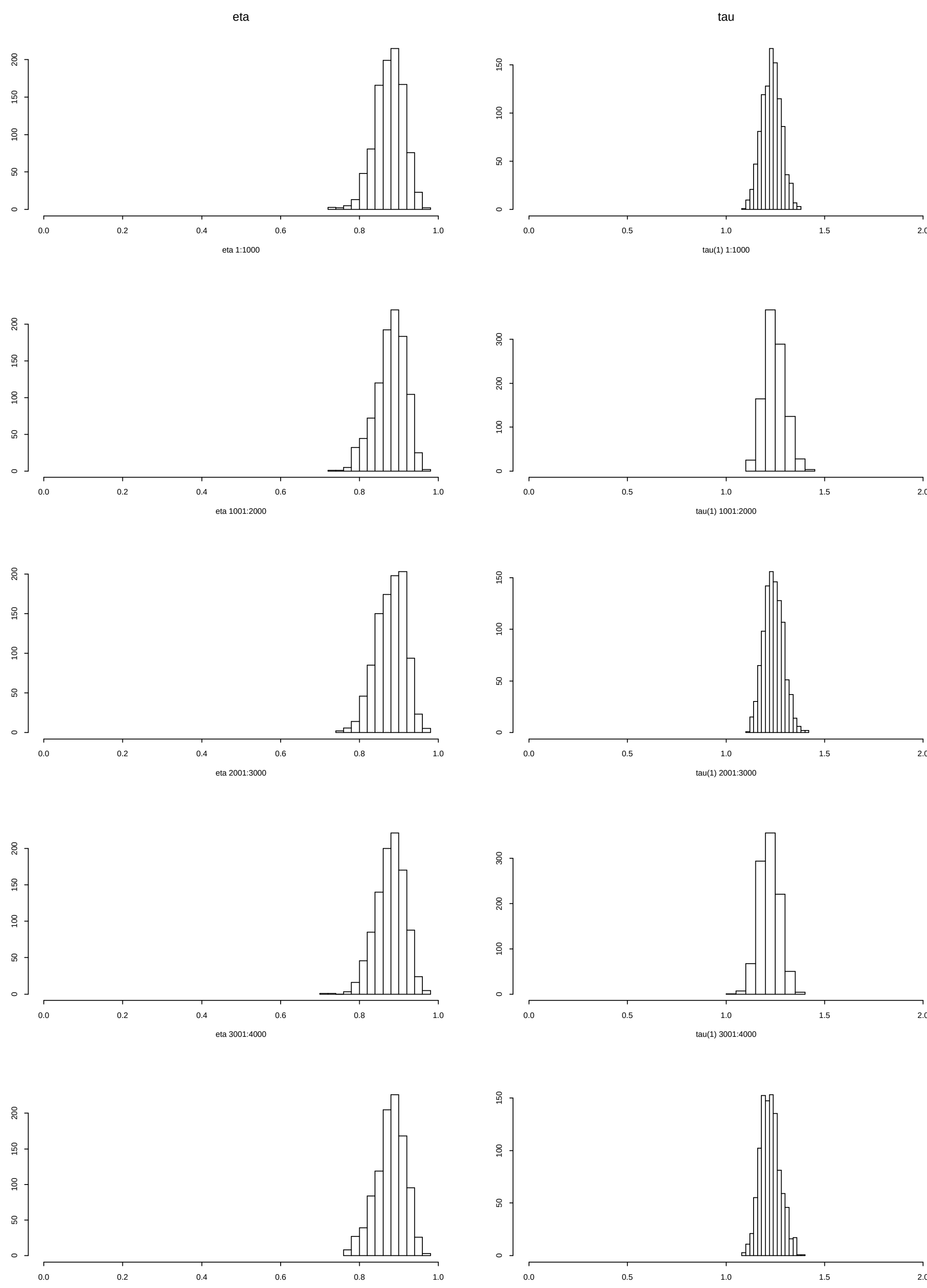


Figure 3.23: Histograms of every 1,000 iterations, after “burn in,” for η and τ.

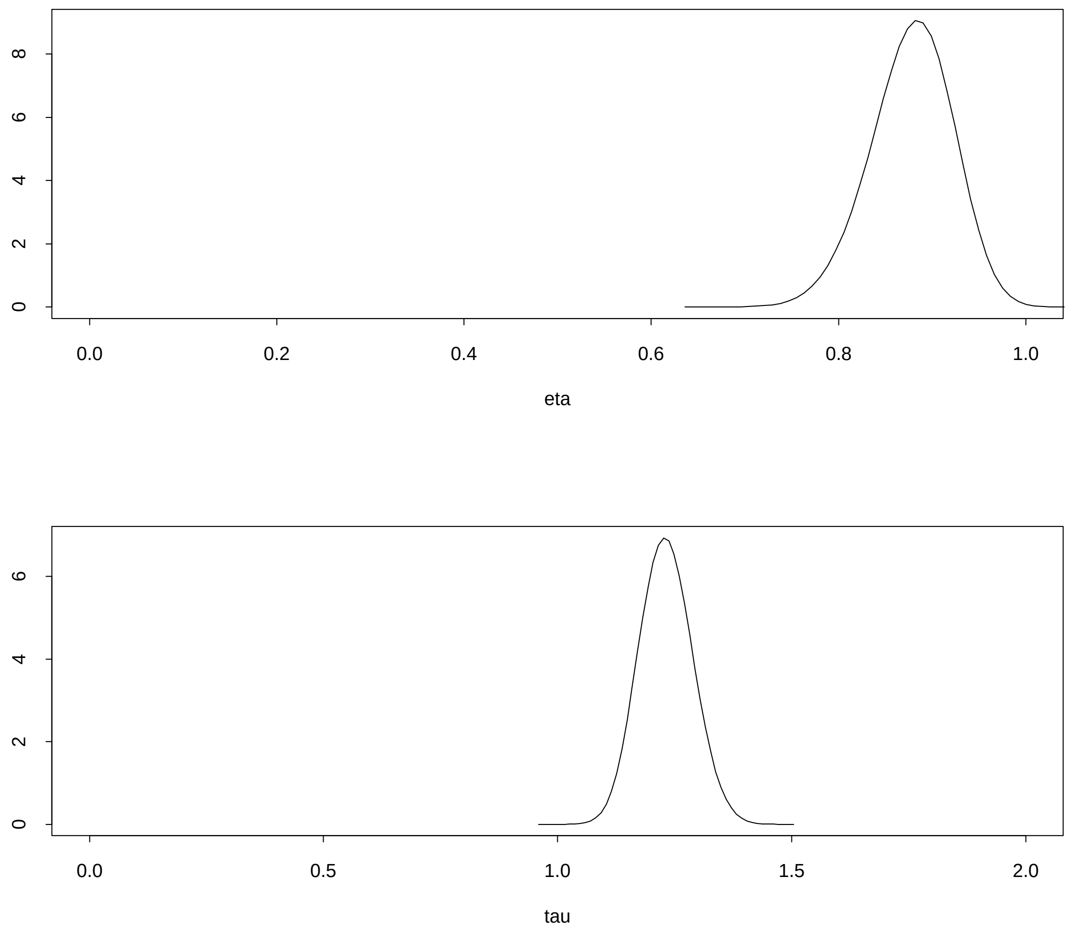


Figure 3.24: Posteriors of η and τ.

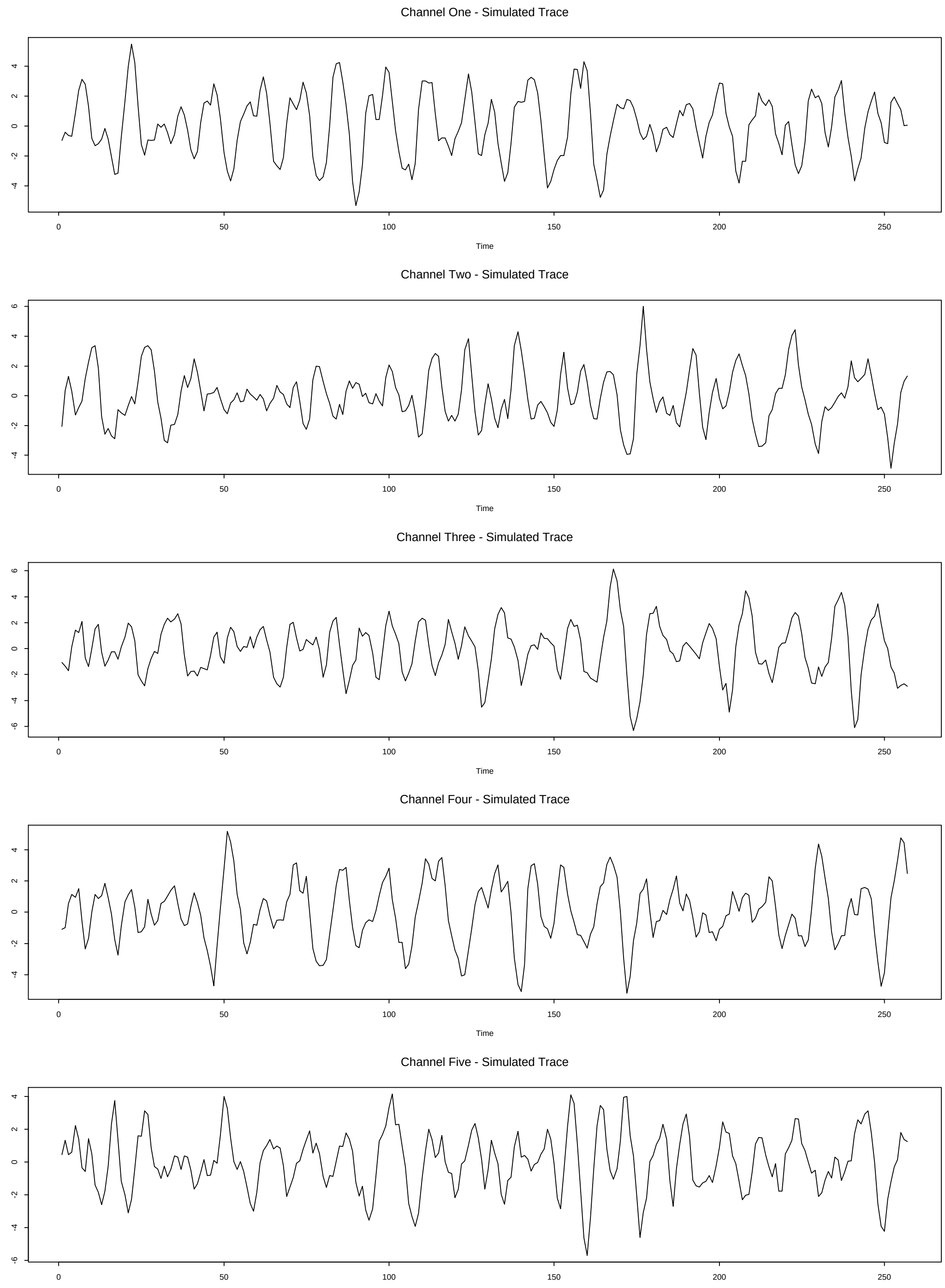


Figure 3.25: Simulated data with no delayed signals.

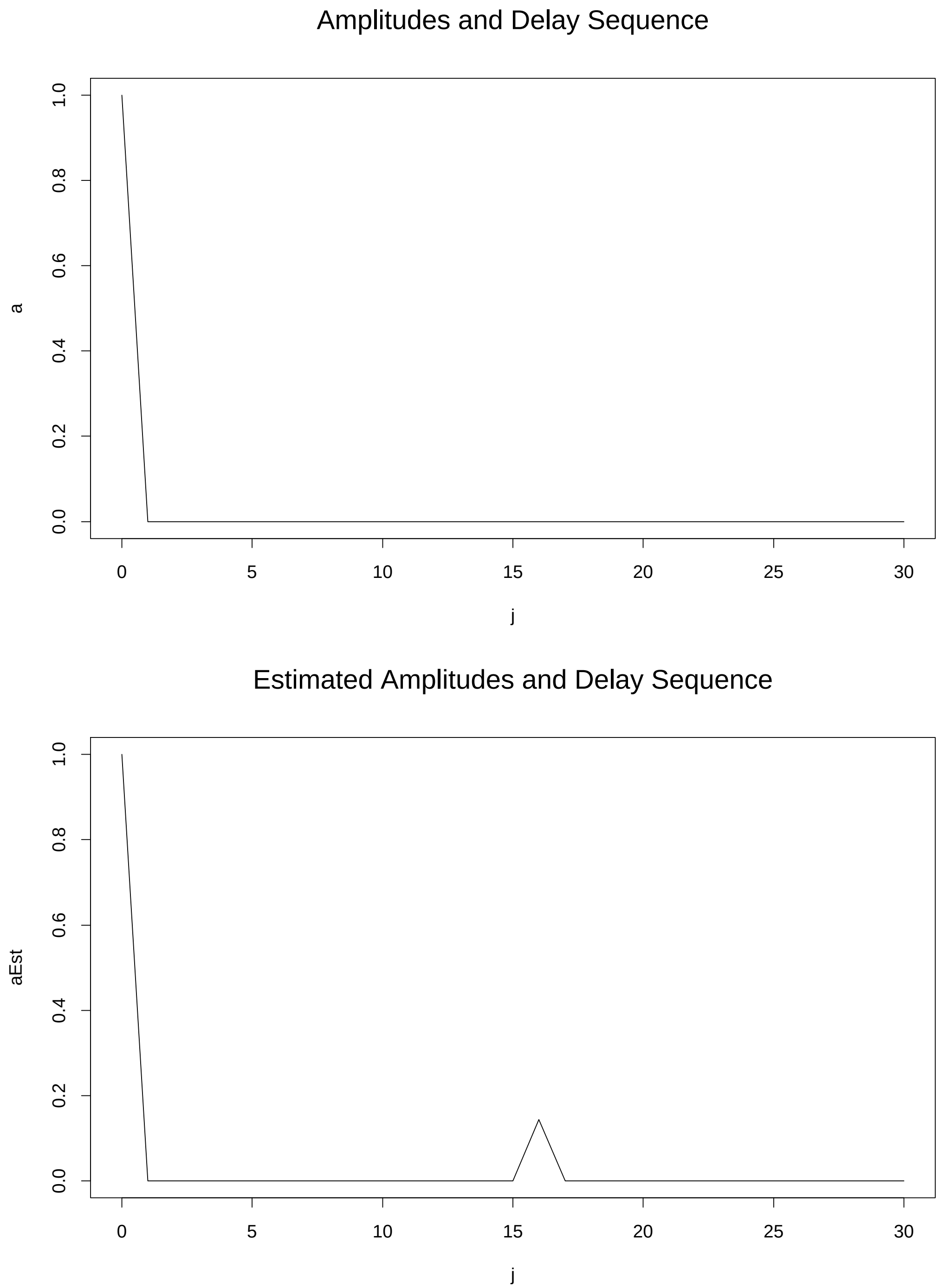


Figure 3.26: Top: “True” *pulse sequence*, **a**, containing no delays. Bottom: Estimates.

Chapter 4

Results for the Real Data

We present in this chapter the results produced by applying our Bayesian deconvolution technique to real data. The array data we analyze is Event 054, a mine blast recorded at the Arctic Experimental Seismic Station (ARCESS) in northern Norway. The seismic source of this data is known to be a ripple-fired mining explosion that was recorded at a regional distance with a sampling rate of 40 points per second.

We begin in Section 4.1 by giving our method of coding the array data for use with our Gibbs sampling program. In Section 4.2 we describe our choice of the hyperparameters for the prior distributions. In Section 4.3 we present our Bayesian estimates of the *pulse sequence* $\hat{\mathbf{a}}$, and hence, the delay structure used in the blasting design of Event 054. The estimates $\hat{\mathbf{a}}$ are calculated using all four phases P_n, P_g, S_n, and L_g, and are compared for consistency. Section 4.4 concludes this chapter with a comparison of our Bayesian deconvolution method with the results from other methods of detecting the delay structure in Event 054.

4.1 Coding of the Data

When developing our Gibbs sampling program with simulated data we found that for computational purposes our program works with data having a limited vertical scale.

For the simulated data we successfully used our program on simulated data with a vertical scale within the range (-6, 6). When applying the Gibbs sampling program to Event 054 we first scaled the data of each phase so the vertical scales were also within this range. Scaling the real data is necessary to avoid overflow errors. The raw data for the P_n, P_g, and S_n-phases of Event 054 have vertical scales in the approximate range (-100, 100) and the L_g-phase is approximately in the range (-400, 400).

For each phase we scaled the 5 series by changing the vertical scales of each series while retaining the relative differences among the series. The maximum in absolute value of all the series was found and the values of each series were scaled by dividing by this value. Finally, each series was multiplied by the constant 6 to put the data on approximately the same scale as was used with the simulated data. These ranges lead to computational overflow error, therefore we scaled our data to avoid this problem.

4.2 Choice of Hyperparameters for the Prior Distributions

In this section we present our choice of hyperparameters for the prior distributions on the parameters in $\boldsymbol{\Theta}$. We use the same values for the analysis of each phase of Event 054 since our prior knowledge is essentially the same for each phase. For η, the probability of observing the value 0 at time j in the *pulse sequence* $\mathbf{a}$, we model our uncertainty in accordance to the results of the cepstral F method of Shumway, Baumgart, and Der [30]. From their results we take our beta prior for η to have a mode of 0.9 and we assume that we are 95% certain η is greater than 0.8. This choice reflects their results in that approximately one delay is estimated every 10 points. The hyperparameters we use to reflect this prior knowledge are $\beta_1 = 43$ and

$\beta_2 = 17/3$, this beta prior is presented in Figure 2.1.

For the autoregressive coefficients $\boldsymbol{\phi}$ we model our uncertainty in light of the efforts of Tjøstheim who fit autoregressive order 3 models to single explosion series [38]. From his results we calculated the mean value and the covariance matrix of the estimates of ϕ_1, ϕ_2, and ϕ_3 from the 40 series he fitted. The mean value we use for our multivariate normal prior on $\boldsymbol{\phi}$ is $\boldsymbol{\phi}_0 = [-0.9, 0.7, -0.2]'$ and the covariance matrix is

$$\Sigma_0 = \begin{bmatrix} (0.45)^2 & -(0.25)^2 & (0.15)^2 \\ -(0.25)^2 & (0.30)^2 & -(0.15)^2 \\ (0.15)^2 & -(0.15)^2 & -(0.15)^2 \end{bmatrix}. \tag{4.1}$$

These values are derived from Tjøstheim's estimates by rounding up and slightly increasing the values in the covariance matrix. The increase in the covariances was made with those involving ϕ_3. The sample covariances seemed too small in our opinion. Also, slightly larger covariances helped facilitate the mixing of the Markov chains produced for $\boldsymbol{\phi}$ in the Gibbs sampler.

The hyperparameters we choose for the gamma prior distribution of the precision of the *underlying signal-path effects sequence* $\mathbf{s}_k$ on each channel k are $\gamma_1 = 5$ and $\gamma_2 = 4$. This prior was chosen to reflect our knowledge that τ should be approximately 1. Due to the method we used to scale our data and the belief that the event in question contains 3 or 4 delays. This assumption comes from the cepstral F results of Shumway, Baumgart, and Der [30], for most of the events they analyzed their method found 3 or 4 delays.

The hyperparameters that determine the normal distribution used to model the independent nonzero elements of the *pulse sequence* a_j are chosen to reflect blasting designs that produce *pulse sequences* similar to those presented by Smith [34]. The mean value we use is $\mu_\alpha = 0.7$ and the variance used is $\sigma_\alpha^2 = 0.15^2$. This choice reflects our belief that the nonzero values of a_j are on average 0.7 and that we are

95% certain these values are between 0.5 and 1 for our model.

4.3 Choice of the Fixed Model Parameters

The fixed model parameters in our model are the length m of the *pulse sequence* $\mathbf{a}$ and the parameter c, where $c\sigma^2$ is the variance of the additive noise $\varepsilon_k(t)$. For the P_n, P_g, and S_n-phases we fix $m = 40$ and for the L_g-phase we fix $m = 50$, since the L_g-phase data we analyze is twice as long as the others. For the P_n phase we analyze $n = 220$ points recorded at $q = 5$ sensors. For the P_g, and S_n-phases we analyze $n = 256$ points recorded by each sensor. For the L_g-phases we analyze $n = 512$ points recorded by each sensor. We fix $c = 0.01$ for the P_n, P_g, and S_n-phases and for the L_g-phase we fix $c = 0.003$. These values of c were selected from the variances of the pre-P_n noise and the actual data.

4.4 Results for Event 054

In this section we present the results of applying our Bayesian deconvolution method to the seismic array data from Event 054. When running our Gibbs sampling program on each phase of Event 054 we produced estimates of all the parameters in $\mathbf{\Theta}$, and thereby, produced estimates of the pulse sequence from which we derive estimates of the delay times used in the blasting design. In the final section of this chapter we compare the estimates of the delay times from each phases with comparable estimates produced by other methods that have been applied to Event 054.

4.4.1 P_n-phase

Here we present the estimates produced by our Bayesian deconvolution program for the P_n-phase of Event 054. The data we analyze includes 220 points of the P_n-phase from each of the five recordings made with a seismic array, see Figure 4.1. The estimates we produce are based on a typical run of our Gibbs sampling program using a "burn in" of 35,000 iterations and total of 40,000 iterations. The estimates are calculated by computing the means of the Markov chains after "burn in" for each parameter. We use the notation $\hat{\theta}$ to represent the estimate of the parameter θ. In Table **??** we present the posterior estimates of the relevant parameters. The most important estimates are the estimated nonzero peak values of $\hat{a}_j$, $j = 1, ..., m$. The peak values of $\hat{a}_j$ for the P_n-phase occur at times 4, 8, 12, 14, 18, 25, 28, and 35. These are the times where there are peaks in the estimated *pulse sequence* $\hat{a}_j$ as seen in Figure 4.2. This delay structure, containing multiple delays, indicates that the P_n-phase of Event 054 was generated by ripple-firing. For completeness we present our deconvolved estimates of the *underlying signal-path effects sequence* $\hat{\mathbf{s}}_k$, $k = 1, ..., q$. Figure 4.3 shows a graph for each $\hat{\mathbf{s}}_k$.

Table **??** presents estimates of the means, medians, standard deviations, and 95% posterior intervals for the relevant parameters. The posterior estimate of $\sigma^2 = 1/\tau$ is also calculated for ease of interpretation. To estimate σ^2 we calculate the mean of the inverted values of the Markov chain related to τ, i.e., after "burn in" we calculate the mean of $1/\tau^{35,000}, ..., 1/\tau^{40,000}$.

In order to assess the goodness-of-fit or our model we examine the fitted values $\hat{y}_k(t)$, $t = 1, ..., n$, for each channel $k = 1, ..., q$, where

$$\hat{y}_k(t) = \hat{s}_k(t) + \sum_{j=1}^{m} \hat{a}_j \hat{s}_k(t).$$

In Figure 4.4 we plot the fits $\hat{y}_k(t)$, shifted one point to the left as dots, along with the true data $y_k(t)$, as solid lines. We note that after $l = m + p = 43$ the fits are very close the actual data. Recall we assume that the likelihood is calculated over the range $(l + 1)$ to n, thus the first l points are not expected to fit well.

To determine convergence of our Gibbs sampler we examine the Markov chains for the peak values of a_j, $j = 1, ..., m$. In Figure 4.5 we plot the 5,000 iterations used to calculate the estimates of the parameters. The Markov chains, from our experience, appear to have converged. In Figure 4.7 we plot the 40 density estimates of the parameters a_j, $j = 1, ..., m = 40$. This plot is useful for interpreting the interval estimates of a_j. Finally, in Figure 4.8 and Figure 4.9 we present the Markov chains used to calculate the estimates of $\boldsymbol{\phi}$, η, and τ. And Figure 4.10 presents the estimated distributions of these parameters.

4.4.2 P_g, S_n, and L_g-phases

In this section we present the results of running the Gibbs sampling program on the latter arriving P_g, S_n, and L_g-phases of Event 054. Tables 4.2, 4.3, and 4.4 give the estimates of the relevant parameters for each phase respectively. The data sets we analyze for the P_g and S_n-phases have $n = 256$ points from each of the $q = 5$ recordings made. We run the programs a total of 40,000 iterations, using $m = 40$, $c = 0.01$, and the same hyperparameters for the priors as those used to analyze the P_n-phase. For the L_g-phase we use $n = 512$ points from each recording, $m = 50$, $c = 0.003$, and run the program 50,000 iterations. The reason we run the program longer is due to the large increase in the parameters in $\boldsymbol{\Theta}$, since n, the sample size for each recording of the L_g-phase is twice as large as for the other phases. The same hyperparameters are again used in the priors to analyze the L_g-phase.

For each run we checked the Markov chains of the parameters for convergence and plotted the posterior distributions. The plots for the other phases were similar to those produced for the P_n phase. Figure A.1 shows the plots of the $q = 5$ recordings of the P_g-phase of Event 054. In Figure A.2 we plot the estimates of the *pulse sequence*

posterior	mean	median	s.d.	$(p_{0.025}, p_{0.975})$
a_4	1.187	1.189	0.031	(1.139,1.235)
a_8	0.502	0.500	0.044	(0.431,0.570)
a_{12}	0.541	0.539	0.031	(0.490,0.595)
a_{14}	0.633	0.622	0.037	(0.583,0.701)
a_{18}	0.565	0.562	0.030	(0.376,0.529)
a_{25}	1.023	1.022	0.059	(0.920,1.129)
a_{28}	0.400	0.403	0.033	(0.346,0.500)
a_{35}	0.505	0.490	0.047	(0.450,0.598)
η	0.654	0.656	0.050	(0.568,0.734)
ϕ_1	-0.961	-0.961	0.036	(-1.020,-0.900)
ϕ_2	0.940	0.940	0.036	(0.879,0.997)
ϕ_3	-0.362	-0.363	0.035	(-0.418,-0.304)
τ	8.097	8.082	0.440	(7.397,8.860)
σ^2	0.124	0.124	0.007	(0.1129,0.135)

Table 4.1: P_n-phase Event 054: Posterior estimates of the relevant parameters.

produced by running our Gibbs sampling program on the P_g data. Figures A.3 and A.4 plot the same pictures for the S_n-phase. And Figures **??** and **??** are for the L_g-phase.

4.5 Comparison of the Delay Time Estimates

Figure 4.11 plots the estimated *pulse sequence* $\hat{a}$ for each of the phases on the same page, we use only the beginning 40 points of $\hat{a}$ for the L_g-phase are plotted so the pictures have the same horizontal scales. The most important observation to make about these pictures is the similar structures and the fact that approximately the same number of peaks are estimated. Pictures like this can be used for monitoring seismic events for ripple-firing. All of the phases exhibit nonzero amplitudes, which indicates delayed signals in Event 054. Based on these comparisons, we believe the Bayesian deconvolution method detects comparable delay time estimates when analyzing ripple-fired seismic data.

posterior	mean	median	s.d.	$(p_{0.025}, p_{0.975})$
a_1	1.733	1.736	0.033	(1.679, 1.782)
a_5	0.766	0.757	0.044	(0.701, 0.840)
a_9	0.369	0.359	0.050	(0.290, 0.465)
a_{12}	0.564	0.549	0.068	(0.449, 0.715)
a_{15}	0.313	0.295	0.072	(0.210, 0.442)
a_{18}	0.307	0.315	0.051	(0.215, 0.374)
a_{22}	0.497	0.503	0.029	(0.440, 0.532)
a_{24}	0.472	0.458	0.045	(0.413, 0.547)
a_{28}	0.306	0.308	0.026	(0.255, 0.344)
η	0.767	0.769	0.045	(0.688, 0.837)
ϕ_1	-1.204	-1.202	0.052	(-1.292, -1.119)
ϕ_2	0.950	0.948	0.057	(0.855, 1.046)
ϕ_3	-0.280	-0.281	0.036	(-0.339, -0.219)
τ	4.935	4.881	0.359	(4.422, 5.606)
σ^2	0.124	0.124	0.007	(0.113, 0.135)

Table 4.2: P_g-phase Event 054: Posterior estimates of the relevant parameters.

posterior	mean	median	s.d.	$(p_{0.025}, p_{0.975})$
a_1	1.433	1.433	0.029	(1.392, 1.479)
a_5	0.775	0.780	0.021	(0.738, 0.805)
a_9	0.559	0.558	0.018	(0.533, 0.590)
a_{12}	0.986	0.987	0.020	(0.946, 1.015)
a_{15}	0.562	0.560	0.034	(0.510, 0.617)
a_{19}	0.421	0.421	0.025	(0.372, 0.466)
a_{22}	0.466	0.463	0.019	(0.442, 0.504)
a_{24}	0.327	0.328	0.017	(0.297, 0.353)
η	0.744	0.746	0.046	(0.666, 0.817)
ϕ_1	-1.645	-1.645	0.031	(-1.695, -1.594)
ϕ_2	1.249	1.249	0.046	(1.172, 1.324)
ϕ_3	-0.313	-0.313	0.030	(-0.363, -0.262)
τ	4.980	4.980	0.285	(4.507, 5.456)
σ^2	0.124	0.124	0.007	(0.113,0.135)

Table 4.3: S_n-phase Event 054: Posterior estimates of the relevant parameters.

posterior	mean	median	s.d.	$(p_{0.025}, p_{0.975})$
a_3	0.608	0.609	0.007	(0.597,0.619)
a_8	0.209	0.206	0.007	(0.199,0.220)
a_{13}	0.423	0.424	0.006	(0.411,0.432)
a_{16}	0.869	0.868	0.007	(0.858,0.883)
a_{22}	0.187	0.186	0.009	(0.173,0.200)
a_{27}	0.860	0.859	0.010	(0.844,0.881)
a_{31}	0.270	0.274	0.021	(0.234,0.297)
a_{46}	0.582	0.582	0.008	(0.567,0.596)
a_{49}	0.392	0.391	0.010	(0.376,0.412)
η	0.730	0.731	0.044	(0.655,0.799)
ϕ_1	-1.824	-1.824	0.017	(-1.852,-1.796)
ϕ_2	1.569	1.569	0.025	(1.527,1.611)
ϕ_3	-0.531	-0.531	0.017	(-0.558,-0.503)
τ	10.209	10.204	0.310	(9.707,10.720)
σ^2	0.098	0.098	0.003	(0.093,0.103)

Table 4.4: L_g-phase Event 054: Posterior estimates of the relevant parameters.

P_n-phase Event 054	delays
Bayesian Deconvolution	4, 8, 12, 14, 18, 25, 28, 35
Cepstral F	4, 21,33
Frequency Domain Deconvolution	4, 14, 21, 33
ARIMA Search	4
P_g-phase Event 054	delays
Bayesian Deconvolution	1, 5, 9, 12, 15, 18, 22, 24, 28
Cepstral F	2, 11, 20
Frequency Domain Deconvolution	5, 12, 21, 26
ARIMA Search	
S_n-phase Event 054	delays
Bayesian Deconvolution	1, 5, 9, 12, 15, 19, 22, 24
Cepstral F	2, 13, 21
Frequency Domain Deconvolution	4, 11, 16, 21, 26
ARIMA Search	4
L_g-phase Event 054	delays
Bayesian Deconvolution	3, 8, 13, 16, 22, 27, 31, 46
Cepstral F	3
Frequency Domain Deconvolution	5, 15, 28
ARIMA Search	4

Table 4.5: Comparison of posterior estimates of the delays form different methods for each pahse.

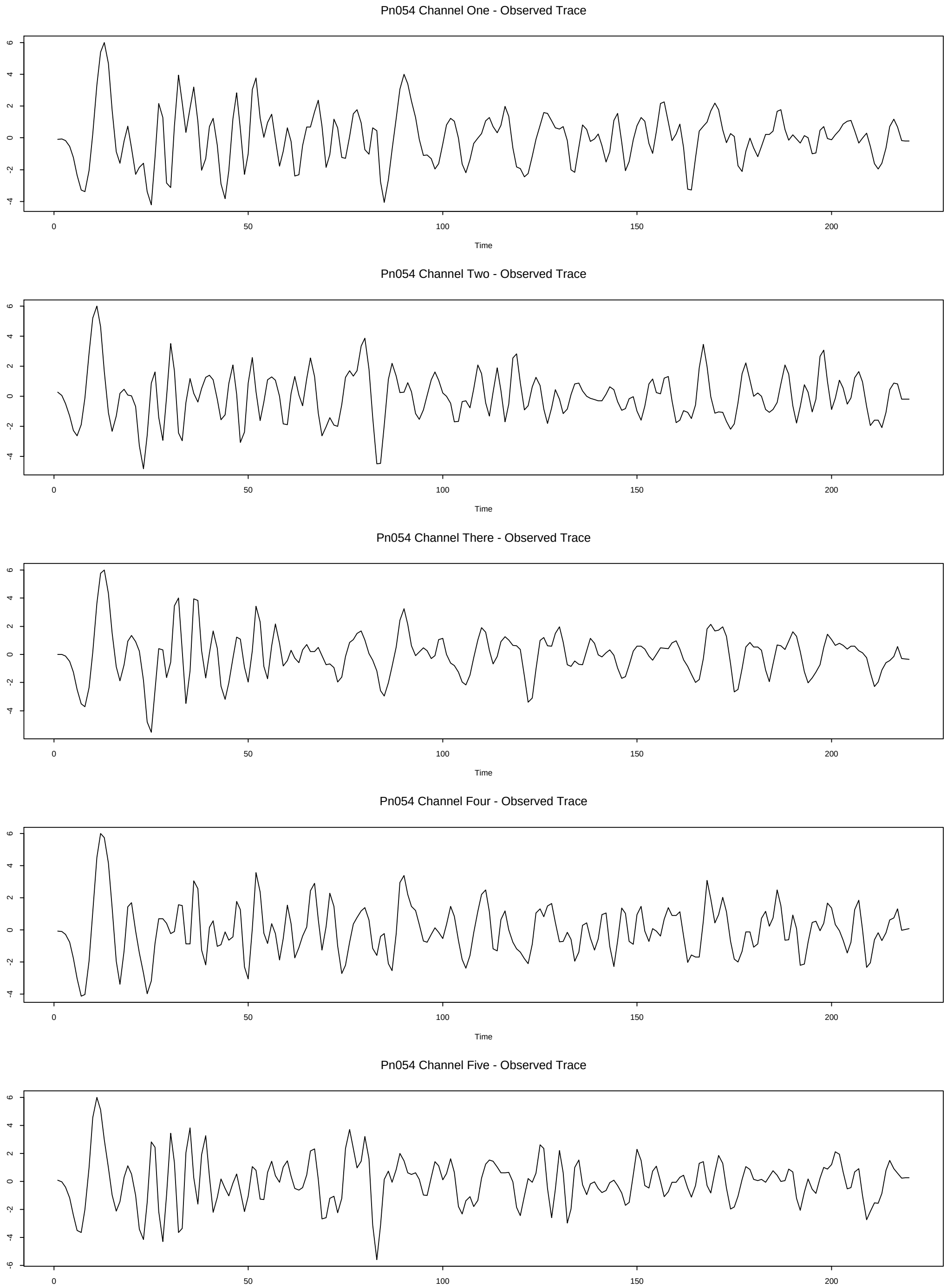


Figure 4.1: P_n-phase Event 054: Scaled data, $\mathbf{y}_k$, $k = 1, ..., q$, where $q = 5$ and $n = 220$.

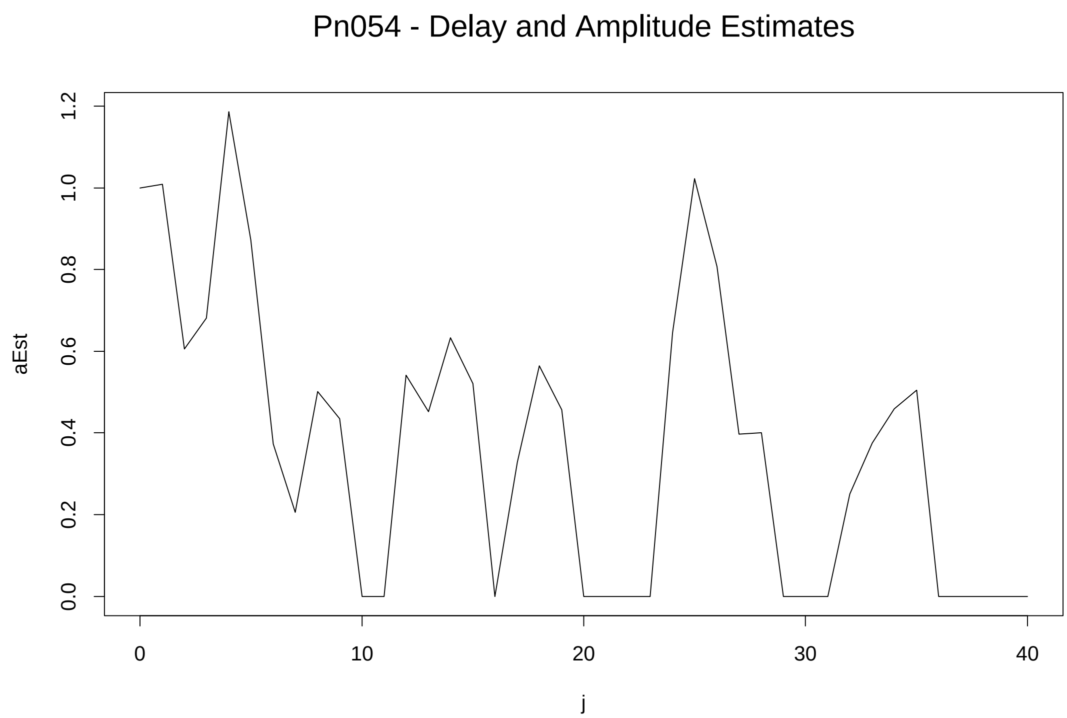


Figure 4.2: P_n-phase Event 054: Posterior estimates of the *pulse sequence*, $\hat{a}_j$, $j = 1, ..., m$, where $m = 40$.

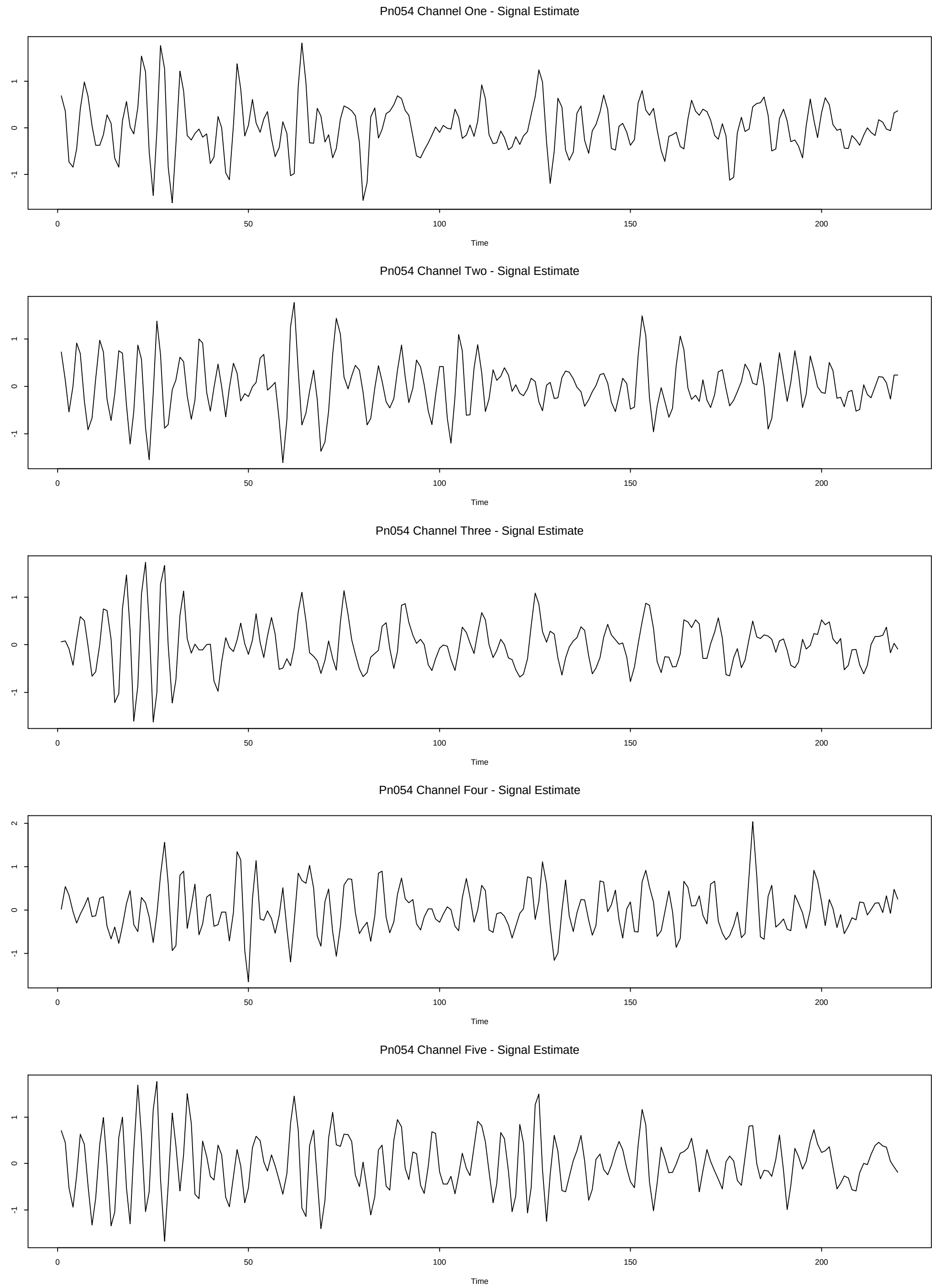


Figure 4.3: P_n-phase Event 054: Posterior estimates of the *underlying signal-path effects*, $\hat{\mathbf{s}}_k$, on each channel.

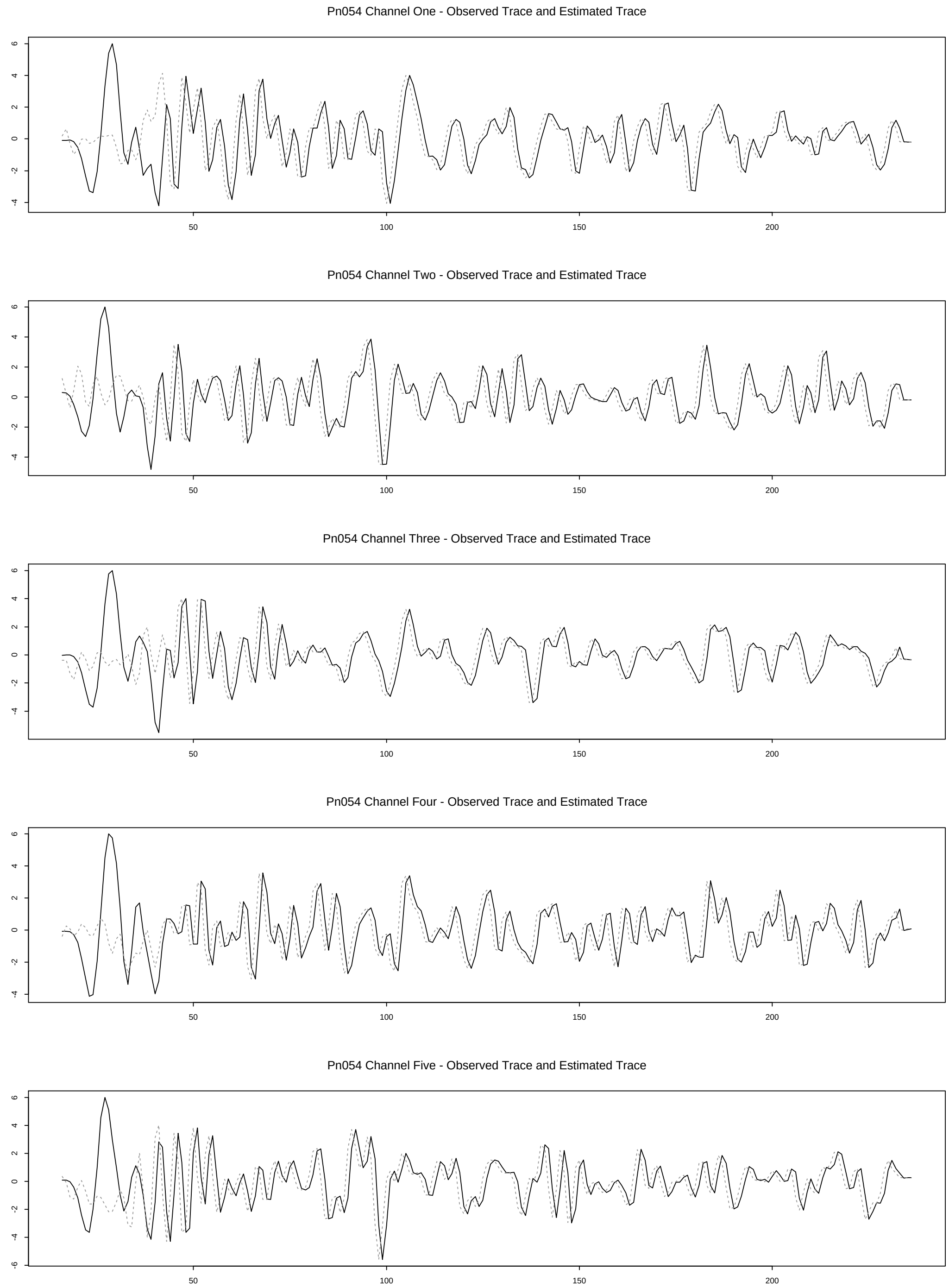


Figure 4.4: P_n-phase Event 054: Solid: Scaled data, $\mathbf{y}_k$. Dots: Posterior estimates, $\hat{\mathbf{y}}_k$, shifted one point to the left.

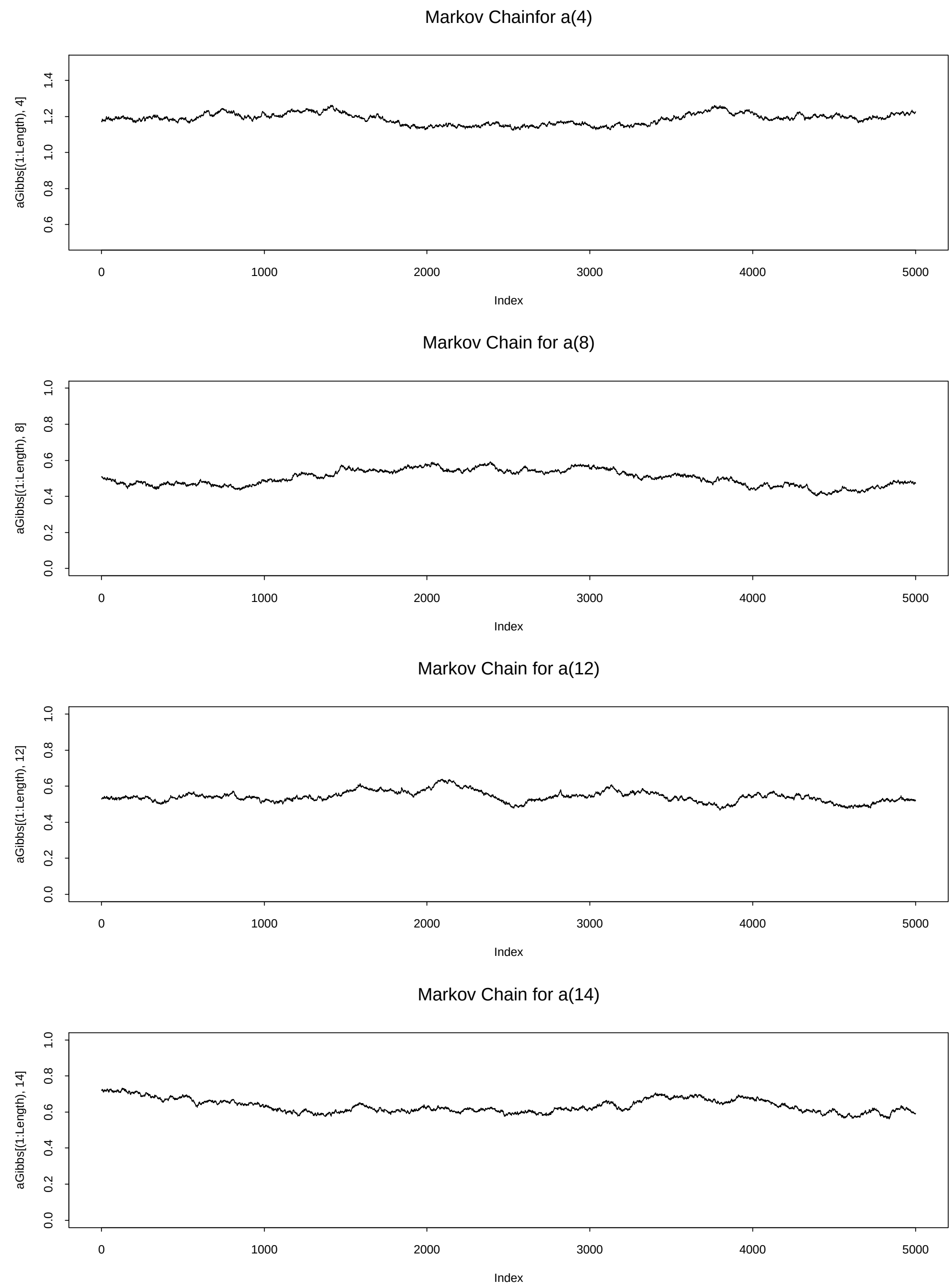


Figure 4.5: P_n-phase Event 054: Last 5,000 iterations in the Markov chains produced for the peaks in $\hat{a}_j$, $j = a, ..., m$.

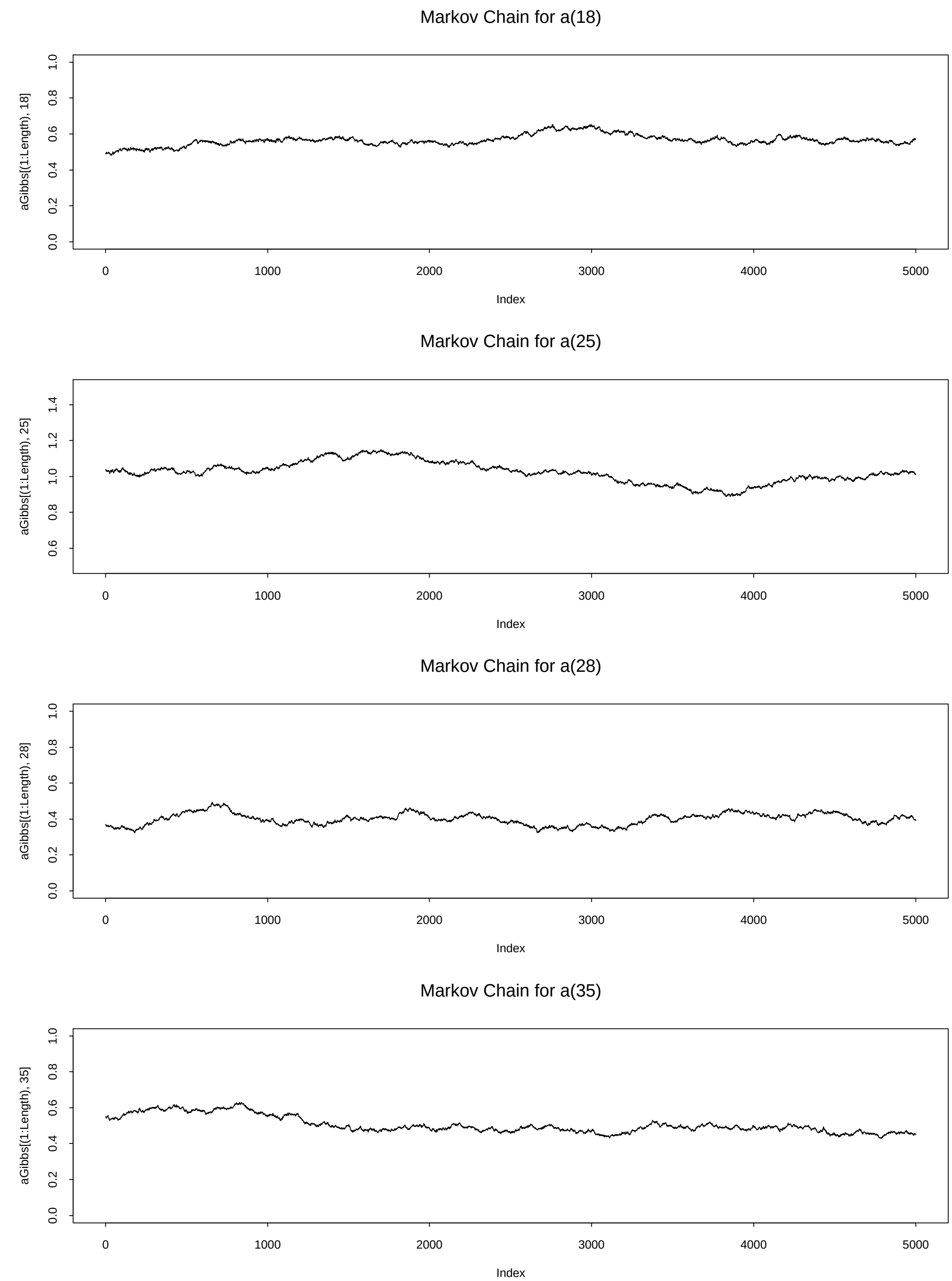


Figure 4.6: P_n-phase Event 054: Last 5,000 iterations in the Markov chains produced for the peaks in $\hat{a}_j$, $j = a, ..., m$.

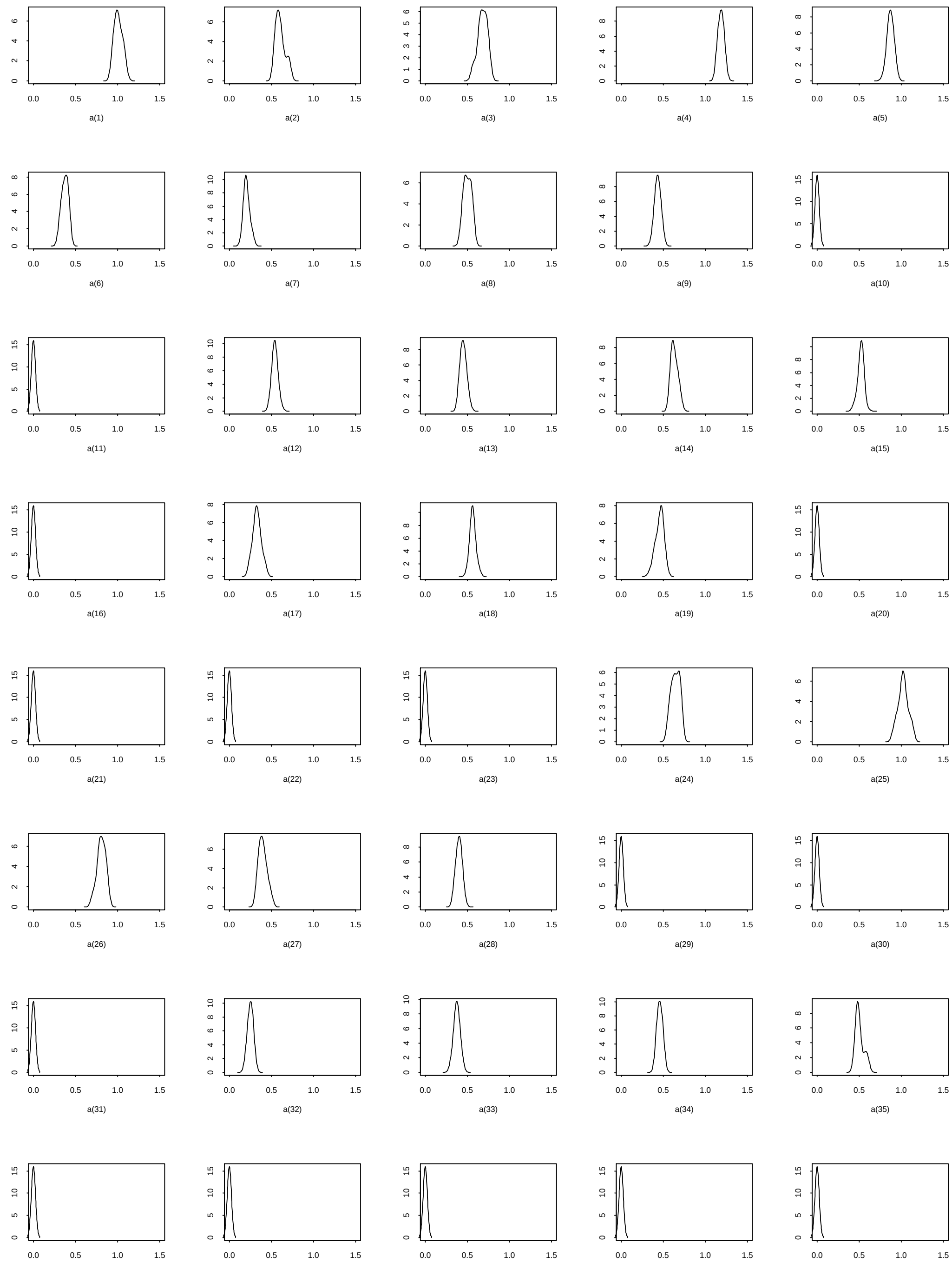


Figure 4.7: P_n-phase Event 054: Posteriors of a_j, $j = 1, ..., m$.

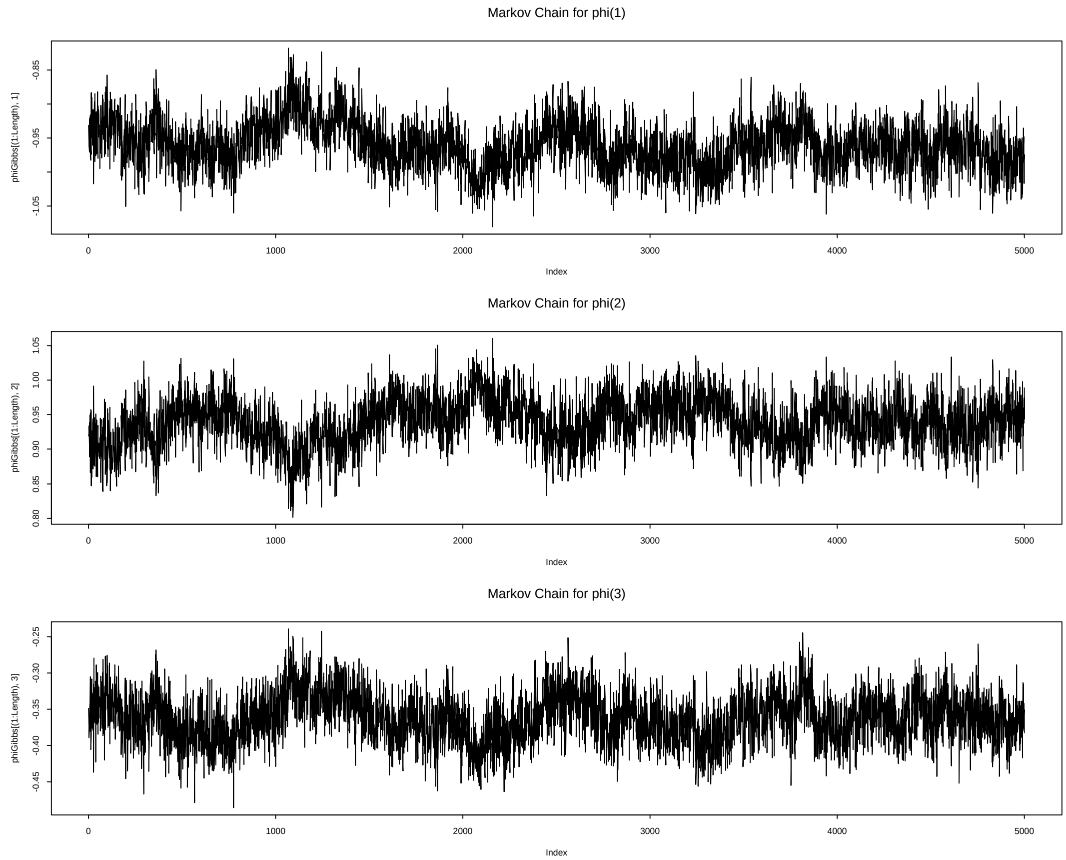


Figure 4.8: P_n-phase Event 054: Markov chains for $\boldsymbol{\phi}$.

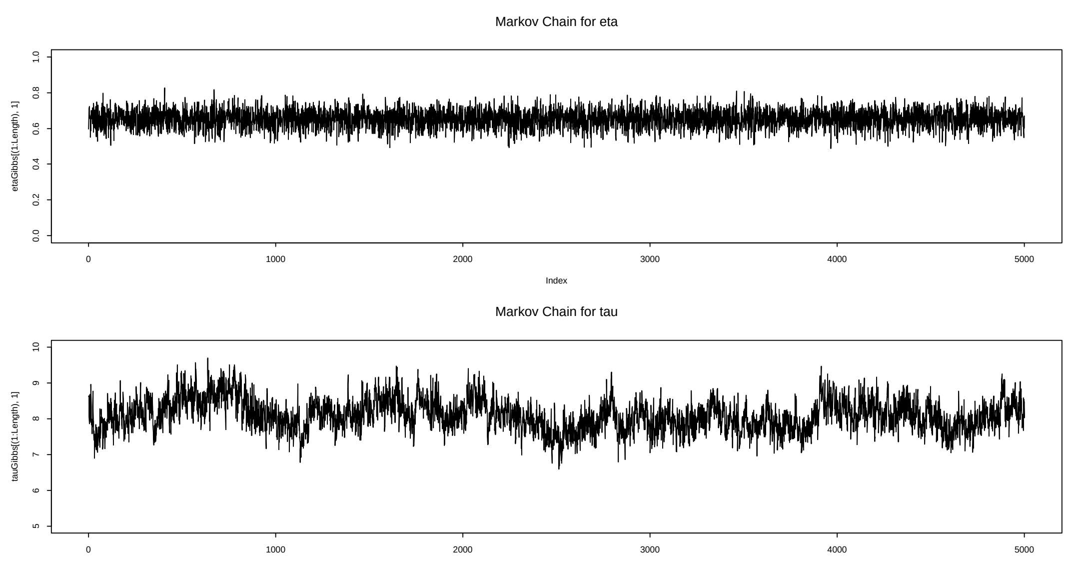


Figure 4.9: P_n-phase Event 054: Markov chains for η and τ.

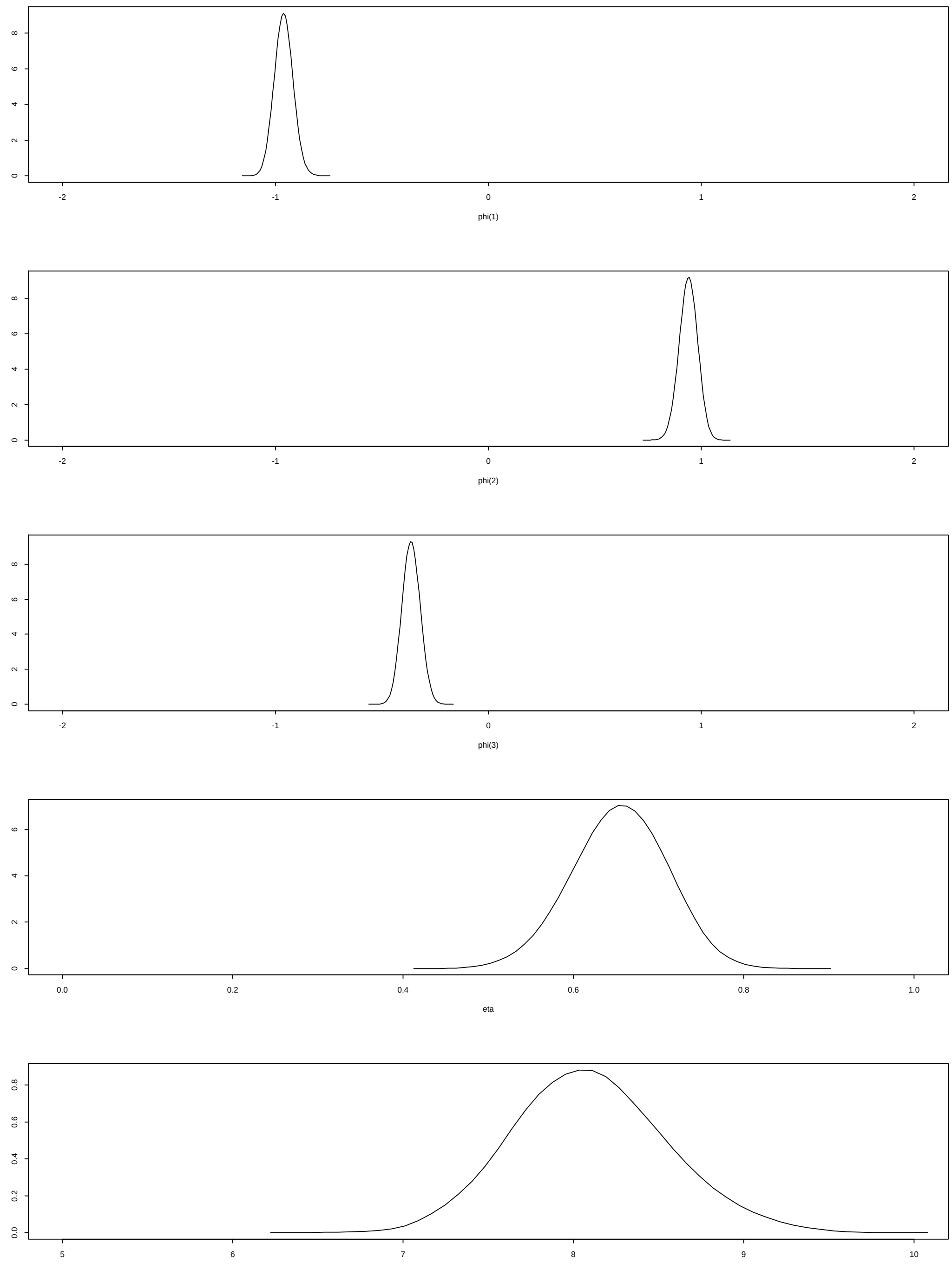


Figure 4.10: P_n-phase Event 054: Posteriors of ϕ_1, ϕ_2, ϕ_3, η, and τ.

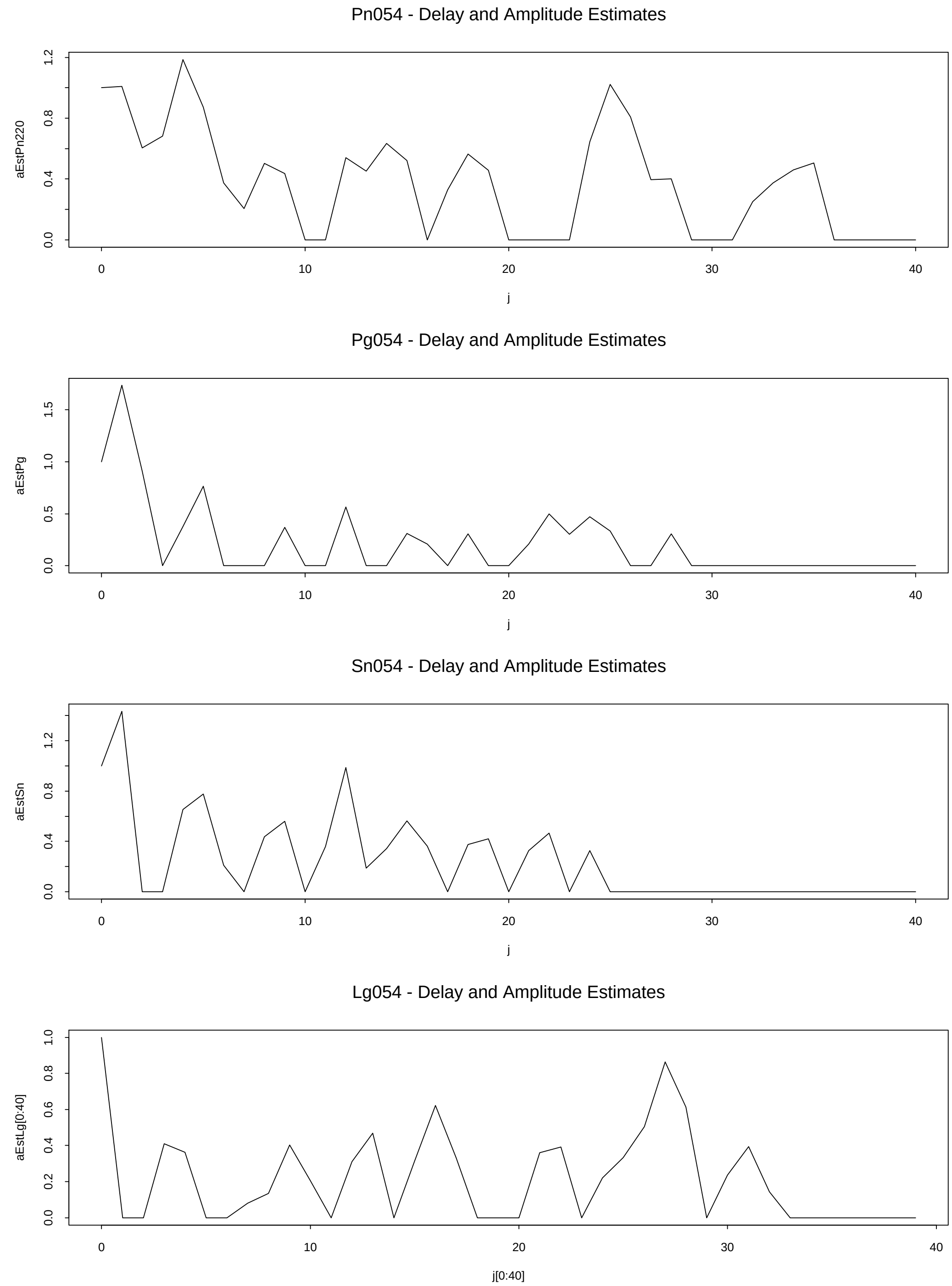


Figure 4.11: Event 054: Posterior estimates of the *pulse sequence* from each phase.

Chapter 5

Conclusions

To summarize our work, we have developed a Bayesisan deconvolution method for seismic array data that detects a common delay pattern. This work was based on the univariate work of Rong Chen and and Cheng, Chen, and Li [7]. The priors we chose were derived from the previous work of Tjøstheim [38], Smith [34], and Shumway, Baumgart, and Der [30]. We tested our program extensively on simulated data and we used the lessons learned from the simulated data analysis to implement our program on the data from Event 054 recorded at ARCESS.

The results we calculated from the output of our Gibbs sampling program were compared to other method that have been used to analyze the same data. We compare our results with the results of (1) the frequency domain deconvolution method of Shumway and Der [31], (2) the ARIMA search on a seasonal ARIMA model of Shumway and McQuarrie [32], and (3) the cepstral F-statistic approach. We found that our method included the results of the other methods. And our results seem to be more consistent among the four phases of the event.

We foresee two main implications of our method for monitoring seismic activity for low level nuclear explosions. first, our method can incorporate into the statistical analysis expert opinion related to the parameters in our model or research related to the seismic activity of the region being monitored. Secondly, we see the possibility of automating the analysis so a quick first analysis can be run to check seismic events for common ripple-fired mining explosions.

There are three items of further work, beyond the scope of this dissertation, that we think might lead to a more robust deconvolution estimates of the delay times in seismic events. first, the incorporation of time dependence into the variance term σ_e^2 of the *underlying signal-path effects sequence*. Because of the nonstationarity of the real data, letting the variance term be a function of t $\sigma_e^2 = \sigma^2(t)$, our model would produce better estimates of the parameters in $\mathbf{\Theta}$ and thereby, better estimates of the delay structure. Second, the implementation of formal monitoring procedure for the convergence of the Markov chains would be useful for automating our procedure. Third, related to possibly automating our procedure a more extensive sensitivity analysis should be performed. This relates to testing the ranges of the hyperparameters of the priors for which the program produced reasonable answers.

Our final recommendation is that our method should be applied to data generated by nuclear tests, earthquakes and single explosions. The results of the deconvolution method would give some idea of the accuracy of or method in practice.

Appendix A

Graphs for the P_g, S_n, and L_g-phases.

Here we present graphs related analysis of the other phases of Event 054.

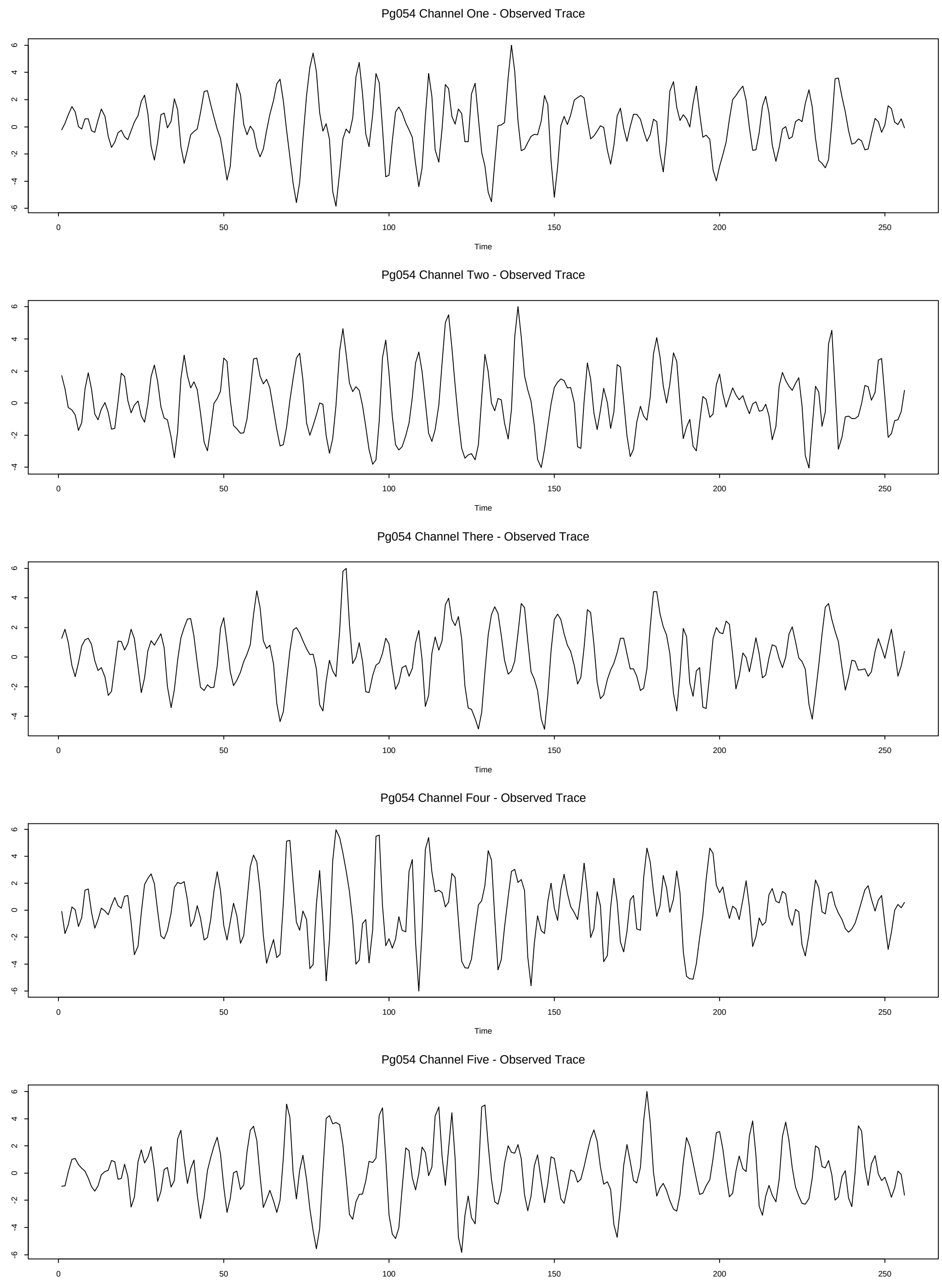


Figure A.1: P_g-phase Event 054: Scaled data, $y_k(t)$, $k = 1, ..., q$, $t = 1, ..., n$, where $q = 5$ and $n = 220$.

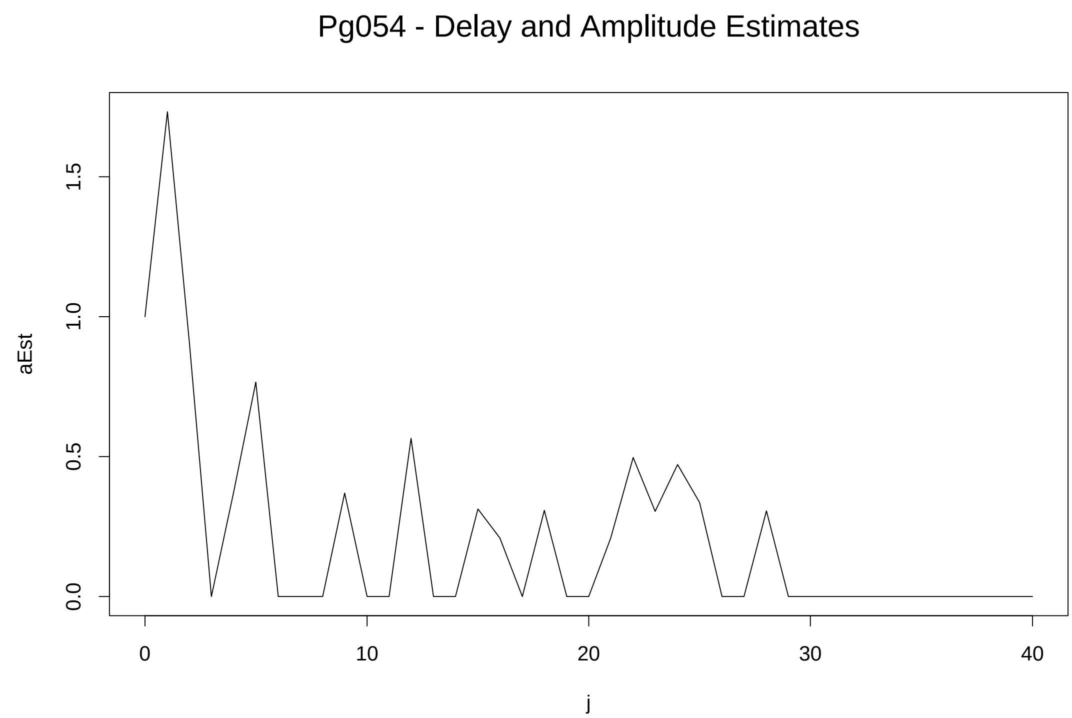


Figure A.2: P_g-phase Event 054: Posterior estimates of the *pulse sequence*, $\hat{a}_j$, $j = 1, ..., m$, where $m = 40$.

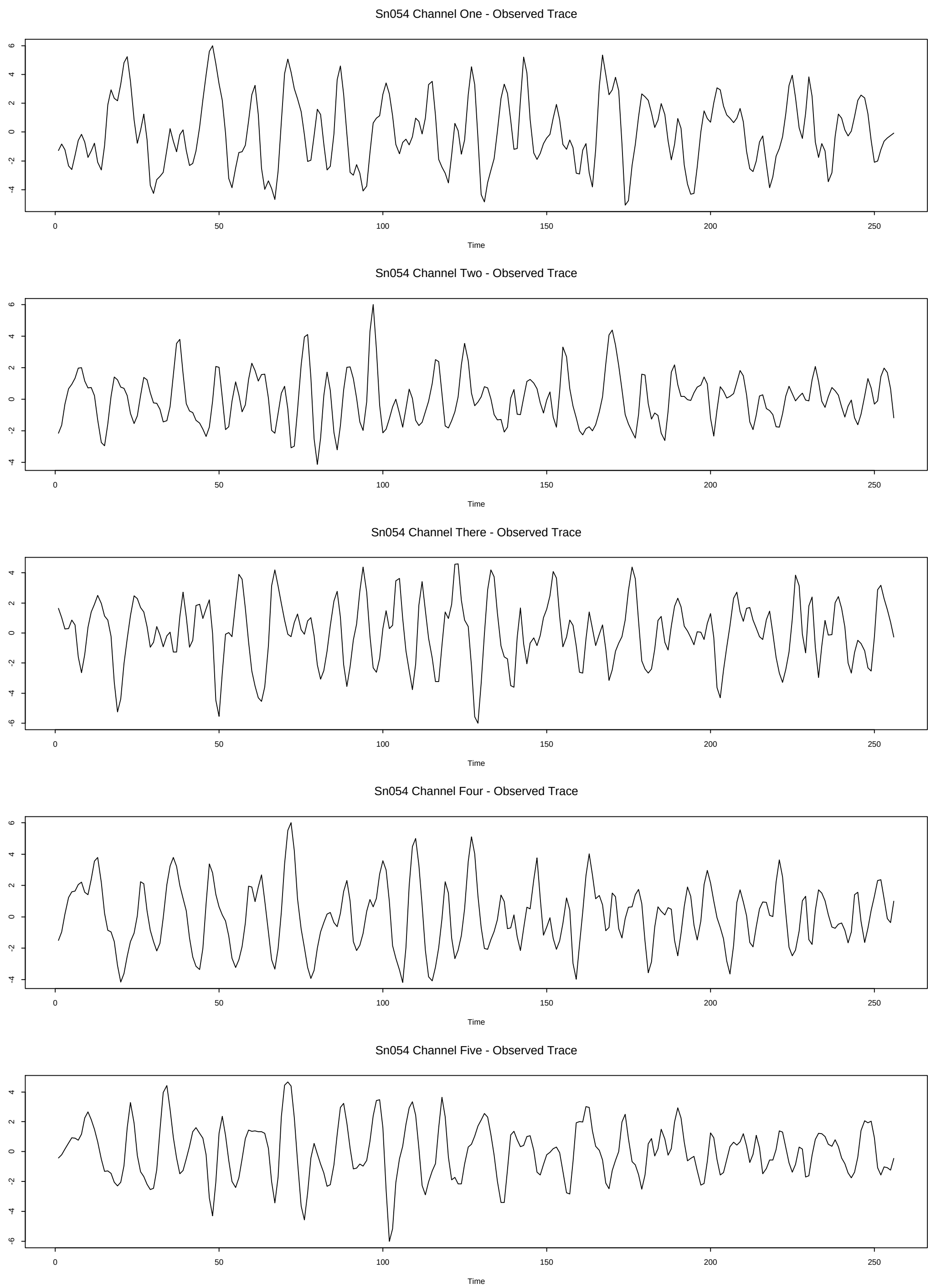


Figure A.3: S_n-phase Event 054: Scaled data, $y_k(t)$, $k = 1, ..., q$, $t = 1, ..., n$, where $q = 5$ and $n = 256$.

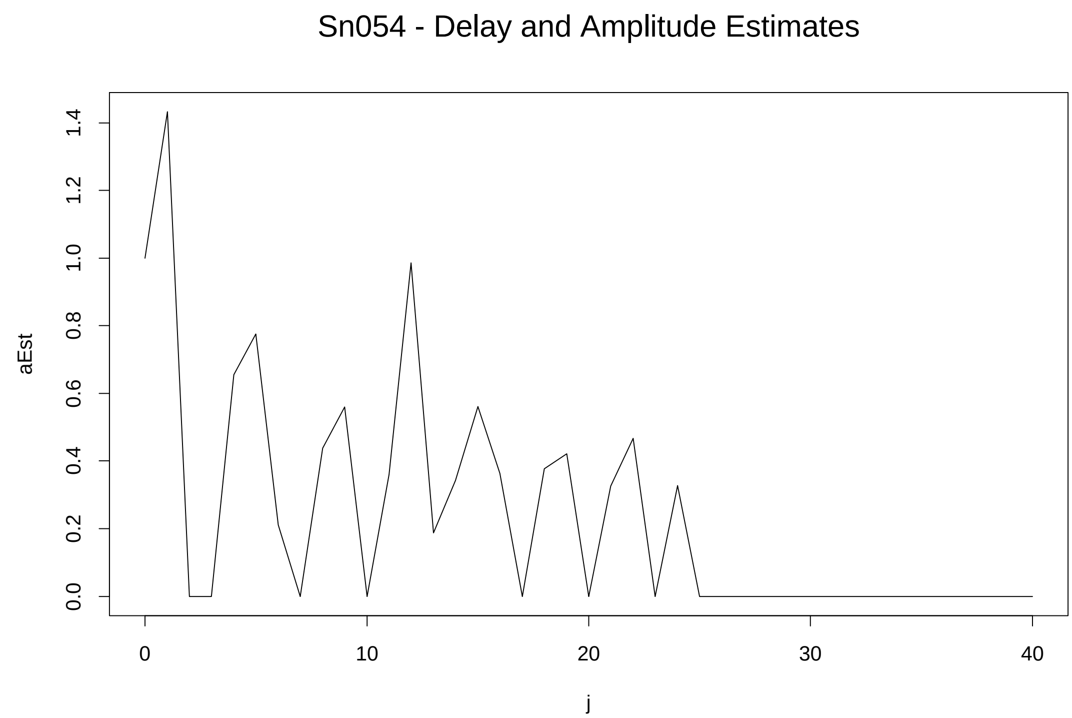


Figure A.4: S_n-phase Event 054: Posterior estimates of the *pulse sequence*, $\hat{a}_j$, $j = 1, ..., m$, where $m = 40$.

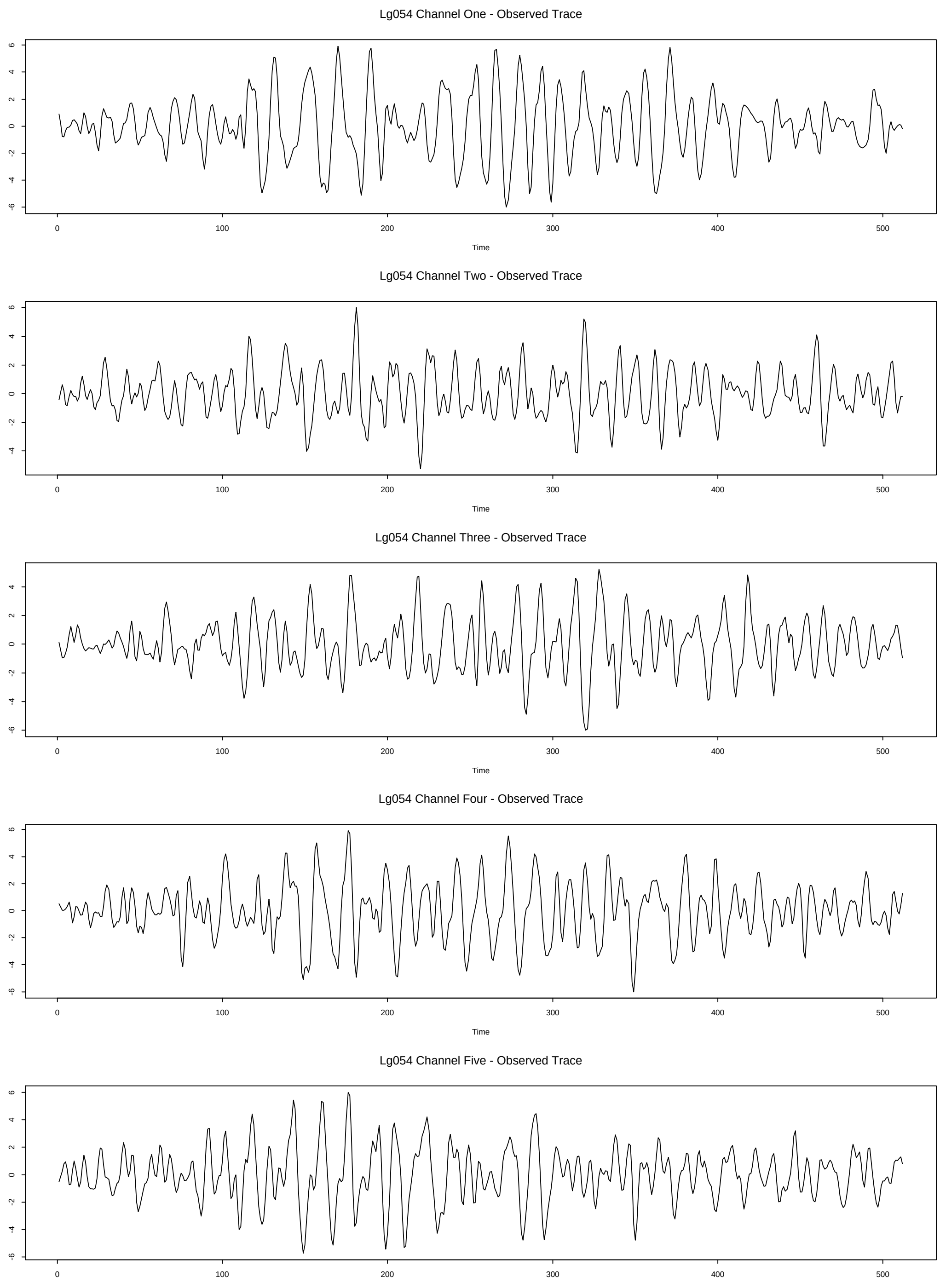


Figure A.5: L_g-phase Event 054: Scaled data, $y_k(t)$, $k = 1, ..., q$, $t = 1, ..., n$, where $q = 5$ and $n = 512$.

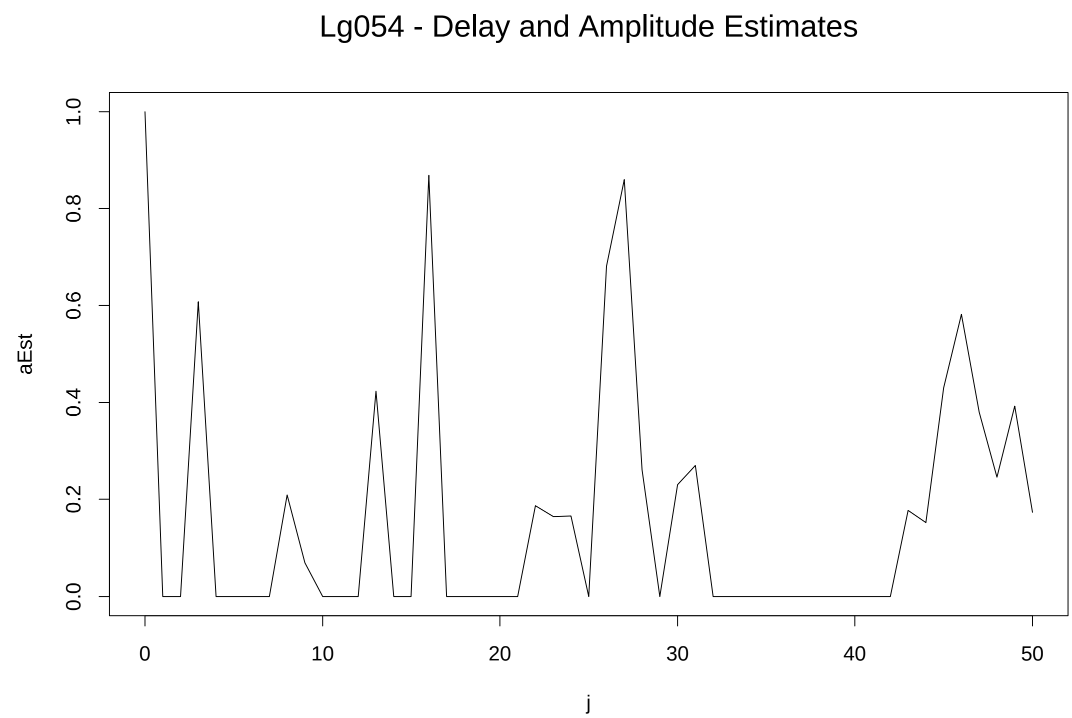


Figure A.6: L_g-phase Event 054: Posterior estimates of the *pulse sequence*, $\hat{a}_j$, $j = 1, ..., m$, where $m = 50$.

Appendix B

Fortran 90 code used to implement the Gibbs sampler.

Here we present the Fortran 90 code used to run the Gibbs Sampler.

```
Program MultiDecon4
Implicit None

! ******************************************************************
! This program implements the Gibbs sampler to produce estimates of
! parameters in the following convolution model:
!
! y(k,t) = sum{j=0}^{m} a(j)*s(k,(t-j)) + epsilon(k,t)
!
! where k = 1,..,q, t = 1,...,n and epsilon(k,t) i.i.d.
! N(0, c(k)*sigmasq(k))
!
! The underlying signal-path effects sequence s(k,t) has an AR(p) model:
!
! phi(0)*s(k,t) + phi(1)*s(k,(t-1)) + ...
! + phi(p)*s(k,(t-1)) = e(k,t)
!
! where phi(0) = 1 fixed and e(k,t) i.i.d. N(0, sigmasq(k))
! precision: tau(k) = 1/sigmasq(k)
!
! The pulse sequence a(j), j = 1,...,m follow the Bernoulli-Gaussian
! model:
!
! a(j) i.i.d. eta*I(a(j) = 0) + (1-eta)*TN(mualpha, sigmasqalpha)
! *I(a(j)>0)
!
```

```
! where a(0) = 1 fixed and eta is the probility of 0 at time j.
!
! ****************************************************************
!
! This program has one contraints.
!
! 1. abs(phi(1)) > abs(phi(2)) > abs(phi(3))
!
! ****************************************************************

! Declare the variables for the simulated data

Integer  i, j, k, h, idum

Integer, Parameter :: ls = 256, ns1 = 20, ns2 = ns1+ls
! ls = length of each trace
! ns1 = start time j
! ns2 = end time j
! n = ns1 + ls + ns2 = total length of each trace
Integer, Parameter :: n = 20+ls+20
Integer, parameter :: q = 5 ! no. of sensors
Integer, Parameter :: p = 3 ! no. AR coefs.
Integer, Parameter :: m = 40 ! no. of a(j)'s
Integer, Parameter :: l = p+m

Double Precision, Parameter :: k0 = 0.6D0
Double Precision, Dimension(0:p) :: phi = 0.0D0
Double Precision, Dimension(q,n) :: s = 0.0D0
Double Precision, Dimension(1:q) :: sigmasq, tau
Data (sigmasq(k), k = 1,q) /1.0D0, 1.0D0, 1.0D0, 1.0D0, 1.0D0/
Data (tau(k), k = 1,q) /1.0D0, 1.0D0, 1.0D0, 1.0D0, 1.0D0/
Double Precision, Dimension(0:m) :: a = 0.0D0
Double Precision, Dimension(q,n) :: y = 0.0D0
Double Precision, Dimension(1:q) :: c

Data (c(k), k = 1,q) /0.01D0,0.01D0,0.01D0,0.01D0,0.01D0/

Double Precision, Parameter :: eta = 0.9D0
Double Precision, Parameter :: Pi = 3.141592654

! ****************************************************************
! Declare the constants for the priors

! for a(j): Think a(j) is Normal and has mode at about 0.7 and 95%
!  sure between 0.5 and 1.
```

```
Double Precision, Parameter :: mualpha = 0.7D0, &
sigmaalpha = 0.15D0, &
sigmasqalpha = sigmaalpha**2

! for eta: Think eta has a Beta(b1,b2) prior and think it has
! its mode about 0.9 and we are 95% sure it it greater
! than 0.8
! Note: for the beta dist.
! beta1 and beta2 must be bigger than 1
Double Precision, Parameter :: etamode = 0.9
Double Precision, Parameter :: beta1 = 43.0D0, &
beta2 = ((1-etamode)/etamode)*beta1+((2*etamode-1)/etamode)

! for phi: Think Normal prior with mean 0 and and 99% sure between
! -1 and 1, and independence
Double Precision, Dimension(1:p) :: phi0 != 0.0D0
Data(phi0(i),i=1,p)/-0.9D0,0.7D0,-0.2D0/
Double Precision, Dimension(p,p) :: VC0, VC0inv
Double Precision sigmaphi, sigmasqphi
!Double Precision, Parameter :: sigmaphi = 0.5D0, &
! sigmasqphi = sigmaphi**2
Double Precision, Parameter :: s11 = (0.45D0)**2 , s12 = -(0.25D0)**2, &
s13 = (0.15D0)**2, s22 = (0.30D0)**2, &
s23 = -(0.15D0)**2, s33 = (0.15D0)**2
!Double Precision, Parameter :: s11 = sigmasqphi, &
! s12 = -1.0D0*sigmasqphi, &
! s13 = sigmasqphi, s22 = sigmasqphi, &
! s23 = -1.0D0*sigmasqphi, s33 = sigmasqphi
Data ((VC0(i,j), i = 1,p), j = 1,p) &
/s11, s12, s13, s12, s22, s23, s13, s23, s33/
! for tau: Think centered at 1 and 95% sure tau is between 0.8 and 1.2
! Note: for the gamma
! gamma1 is an integer value grater than 0
! gamma2 is an arbitrary number greater than 0
Integer, Parameter :: gamma1 = 5
Double Precision, Parameter :: gamma2 = 4.0D0 !gamma2 = (gamma1-1)/mode

! *****************************************************************
! Declare the variables for Gibbs Sampling

! inital values

Double Precision, Dimension(q,n) :: sINIT = 0.0D0
Double Precision, Dimension(0:m) :: aINIT = 0.0D0
Double Precision :: etaINIT = 0.0D0
```

```
Double Precision, Dimension(0:p) :: phiINIT = 0.0D0
Double Precision, Dimension(1:q) :: tauINIT = 0.0D0

! length of burnin and total repetitions

Integer, Parameter :: Reps = 40000
Integer, Parameter :: Burn = 35000
Integer, Parameter :: Length = (Reps - Burn)
Integer, Parameter :: Start = Burn+1 ! start time j of saved chains

! matricies used to pass iteration (h-1) into the subroutines
! and get iteration h out

Double Precision, Dimension(q,n) :: s1nIN = 0.0D0, s1nOUT = 0.0D0
Double Precision, Dimension(q,n) :: spnIN = 0.D0, spnOUT = 0.0D0
Double Precision, Dimension(q,p) :: sp1IN = 0.0D0, sp1OUT = 0.0D0
Double Precision, Dimension(0:m) :: aIN = 0.0D0, aOUT = 0.0D0
Double Precision :: etaIN = 0.0D0, etaOUT = 0.0D0
Double Precision, Dimension(0:p) :: phiIN = 0.0D0, phiOUT = 0.0D0
Double Precision, Dimension(1:q) :: tauIN = 0.0D0, tauOUT = 0.0D0

! vectors for temportary storage

Double Precision, Dimension(q,n) :: sTEMP = 0.0D0
Double Precision, Dimension(0:m) :: aTEMP = 0.0D0
Double Precision :: etaTEMP = 0.0D0
Double Precision, Dimension(0:p) :: phiTEMP = 0.0D0
Double Precision, Dimension(1:q) :: tauTEMP = 0.0D0

! matricies used to save the Gibbs Samples

!Double Precision, Dimension(Reps,q,n) :: sGibbs = 0.0D0
Double Precision, Dimension(Length,q,n) :: sGibbs = 0.0D0
Double Precision, Dimension(Reps,m) :: aGibbs = 0.0D0
Double Precision, Dimension(1:Reps) :: etaGibbs = 0.0D0
Double Precision, Dimension(Reps,p) :: phiGibbs = 0.0D0
Double Precision, Dimension(Reps,q) :: tauGibbs = 0.0D0

! *******************************************************************
! *******************************************************************
! *******************************************
! *** Generate and save the simulated data ***
! *******************************************

idum = -3 ! seed
```

```
!Print*, idum

phi(0) = 1.0D0
phi(1) = -3.0D0*exp(-k0)
phi(2) = 3.0D0*exp(-2.0*k0)
phi(3) = -exp(-3.0*k0)

!CALL GenerateAR3(phi, s, sigmasq, idum)

a(0) = 1.0D0
a(7) = 0.8D0
a(14) = 0.7D0
a(21) = 0.6D0
a(28) = 0.5D0

!CALL Filter(a, s, y, sigmasq, c, idum)

CALL Ready(y) ! used to read in data from a file

OPEN(11, FILE = 'seed.DAT')
Write(11,*) idum
OPEN(12, FILE = 'phi.DAT')
Do i = 0, p
Write(12,*) phi(i)
EndDo
OPEN(13, FILE = 's.DAT')
Do k = 1, q
Do i = 1, n
Write(13,*) s(k,i)
EndDo
EndDo
OPEN(14, FILE = 'a.DAT')
Do i = 0,m
Write(14,*) a(i)
EndDo
OPEN(15, FILE = 'y.DAT')
Do k = 1, q
Do i = 1, n
Write(15,*) y(k,i)
EndDo
EndDo
OPEN(16, FILE = 'sigmasq.DAT')
Do k = 1,q
Write(16,*) sigmasq(k)
```

```
EndDo
OPEN(17, FILE = 'tau.DAT')
Do k = 1,q
Write(17,*) tau(k)
EndDo
OPEN(18, FILE = 'c.DAT')
Do k = 1,q
Write(18,*) c(k)
EndDo
OPEN(19, FILE = 'eta.DAT')
Write(19,*) eta

! *********************************************************************
! ******************************
! *** Check the prior values ***
! ******************************

!Print*, sigmasq
!Print*, tau
!Print*, c
!Read*

!Print*, mualpha
!Print*, sigmaalpha
!Print*, beta1
!Print*, beta2
!Print*, phi0
!Print*, VC0
!Print*, gamma1
!Print*, gamma2
!Read*

!Print*, 'VC0'
!CALL PrintMatrix(VC0, p, p)
!Read*

VC0inv = VC0
CALL SweepMatrix(VC0inv, p, p)
!Print*,'VC0inv'
!CALL PrintMatrix(VC0inv, p, p)
!Read*

! *********************************************************************
! ****************************************
! *** Set and check the initial values ***
```

```
! ****************************************

aINIT(0) = 1.0D0

etaINIT = eta

phiINIT(0) = 1.0D0
phiINIT(1) = phi0(1)
phiINIT(2) = phi0(2)
phiINIT(3) = phi0(3)

tauINIT = tau

! *******************************************************************
! ********************************************
! *** Assign and check initial TEMP values ***
! ********************************************

sTEMP = sINIT
!Print*, sTEMP
aTEMP = aINIT
!Print*, aTEMP
etaTEMP = etaINIT
!Print*, etaTEMP
phiTEMP = phiINIT
!Print*, phiTEMP
tauTEMP = tauINIT
!Print*, tauTEMP
!Read*

! *******************************************************************
! ************************************
! *** Check the IN and OUT vectors ***
! ************************************

!GOTO 1000

! *******************************************************************
! *******************************
! *** Calculate Gibbs Samples ***
! *******************************

Do h = 2, Reps ! begin outer loop on h = 2 to Reps

!If (mod(h,10).eq.0) Print*,'iteration h =',h
```

```
!Print*,'iteration h =',h

! *******************************
! *** Generate an a iteration ***
! *******************************

! *** initalize inputs ***
s1nIN = 0.0D0
aIN = 0.0D0
etaIN = 0.0D0
tauIN = 0.0D0

! *** initalize outputs ***
aOUT = 0.0D0
aOUT(0) = 1.0D0

! *** inputs ***
Do k = 1,q
Do i = 1,n
         s1nIN(k,i) = sTEMP(k,i)
EndDo
EndDo

aIN(0) = 1.0D0
Do j = 1,m
        aIN(j) = aTEMP(j)
EndDo

etaIN = etaTEMP

Do k = 1,q
tauIN(k) = tauTEMP(k)
EndDo

!Print*, 'call a'

CALL GenerateaGibbs(idum, y, c, s1nIN, aIN, etaIN, tauIN, aOUT)

! *** output ***
!Print*, 'aout'
!Print*, aout
Do j = 1,m
        aTEMP(j) = aOUT(j)
EndDo
```

```
Do i = 1,m
        aGibbs(h,i) = aTEMP(i)
EndDo

! ++++++++++++++++++++++++++++++++++++++++++++++++++
! This is used to run others separately
!Do j = 0,m
!       aTEMP(j) = a(j)
!EndDo
!Do j = 1,m
!       aGibbs(h,j) = aTEMP(j)
!EndDo
! ++++++++++++++++++++++++++++++++++++++++++++++++++

! ******************************
! *** Generate an s iteration ***
! ******************************

! *** initalize inputs ***
s1nIN = 0.0D0
aIN = 0.0D0
phiIN = 0.0D0
tauIN = 0.0D0

! *** initalize outputs ***
s1nOUT = 0.0D0

! *** assign inputs ***
Do k = 1,q
Do i = 1,n
s1nIN(k,i) = sTEMP(k,i)
EndDo
EndDo

aIN(0) = 1.0D0
Do j = 1,m
ain(j) = aTEMP(j)
EndDo

etaIN = etaTEMP

phiIN(0) = 1.0D0
Do i = 1,p
phiIN(i) = phiTEMP(i)
EndDo
```

```
Do k = 1,q
tauIN(k) = tauTEMP(k)
EndDo

!Print*, 'call spn'

CALL GeneratesGibbs(idum, y, c, s1nIN, aIN, phiIN, tauIN, s1nOUT)

! *** output ***
Do k = 1,q
Do i = (p+1),n
sTEMP(k,i) = s1nOUT(k,i)
EndDo
EndDo
! +++++++++++++++++++++++++++++++++++++++++++++++++++
! This is used to run others separately
!Do k = 1,q
! Do i = (p+1),n
! sTEMP(k,i) = s(k,i)
! EndDo
!EndDo
! +++++++++++++++++++++++++++++++++++++++++++++++++++

! *********************************
! *** Generate an sp1 iteration ***
! *********************************

! *** initalize inputs ***
sp1IN = 0.0D0
spnIN = 0.0D0
phiIN = 0.0D0
tauIN = 0.0D0

! *** initalize outputs ***
sp1OUT = 0.0D0

! *** inputs ***
Do k = 1,q
Do i = 1,p
sp1IN(k,i) = sTEMP(k,(p-i+1))
EndDo
EndDo

Do k = 1,q
```

```
Do i = (p+1),n
spnIN(k,i) = sTEMP(k,i)
EndDo
EndDo

phiIN(0) = 1.0D0
Do i = 1,p
phiIN(i) = phiTEMP(i)
EndDo

Do k = 1,q
tauIN(k) = tauTEMP(k)
EndDo

!Print*, 'call sp1'

CALL Generatesp1Gibbs(idum, sp1IN, spnIN, phiIN, tauIN, sp1OUT)

! *** output ***
Do k = 1,q
Do i = 1,p
sTEMP(k,i) = sp1OUT(k,(p-i+1))
EndDo
EndDo
If (h.gt.Burn) Then
Do k = 1,q
         Do i = 1,n
                 sGibbs((h-Burn),k,i) = sTEMP(k,i)
         EndDo
EndDo
EndIf
! ++++++++++++++++++++++++++++++++++++++++++++++++++++
! This is used to run others separately
!Do k = 1,q
! Do i = 1,p
! sTEMP(k,i) = s(k,i)
! EndDo
!EndDo
!If (h.gt.Burn) Then
! Do k = 1,q
! Do j = 1,n
! sGibbs(h,k,j) = sTEMP(k,j)
! EndDo
! EndDo
!EndIf
```

```
! +++++++++++++++++++++++++++++++++++++++++++++++++++

! *********************************
! *** Generate an eta iteration ***
! *********************************

! *** initalize inputs ***
aIN = 0.0D0

! *** initalize outputs ***
etaOUT = 0.0D0

! *** inputs ***
aIN(0) = 1.0D0
Do j = 1,m
ain(j) = aTEMP(j)
EndDo

!Print*, 'call eta'

CALL GenerateetaGibbs(idum, aIN, etaOUT)

! *** output ***
etaTEMP = etaOUT
etaGibbs(h) = etaTEMP
! +++++++++++++++++++++++++++++++++++++++++++++++++++
! This is used to run others separately
!etaTEMP = eta
!etaGibbs(h) = etaTEMP
!+++++++++++++++++++++++++++++++++++++++++++++++++++

! *********************************
! *** Generate an phi iteration ***
! *********************************

! *** initalize inputs ***
s1nIN = 0.0D0
tauIN = 0.0D0

! *** initialize outputs ***
phiOUT = 0.0D0

! *** input ***
Do k = 1,q
Do i = 1,n
```

```
s1nIN(k,i) = sTEMP(k,i)
EndDo
EndDo

Do k = 1,q
tauIN(k) = tauTEMP(k)
EndDo

!Print*, 'call phi'

!Print*, 'phi0'
!Print*, phi0
!Print*, 'VC0'
!Print*, VC0

CALL GeneratephiGibbs(idum, s1nIN, tauIN, phi0, VC0, VC0inv, phiOUT)

! *** output ***
Do i = 1,p
phiTEMP(i) = phiOUT(i)
EndDo
Do i = 0,p
phiGibbs(h,i) = phiTEMP(i)
EndDo
! +++++++++++++++++++++++++++++++++++++++++++++++++++
! This is used to run others separately
!Do i = 0,p
! phiTEMP(i) = phi(i)
!EndDo
!Print*, 'phiTEMP'
!Print*, phiTEMP
!Do i = 1,p
!        phiGibbs(h,i) = phiTEMP(i)
!EndDo
! +++++++++++++++++++++++++++++++++++++++++++++++++++

! *********************************
! *** Generate an tau iteration ***
! *********************************

! *** initalize inputs ***
s1nIN = 0.0D0
aIN = 0.0D0
phiIN = 0.0D0
```

```
! *** initialize outputs ***
tauOUT = 0.0D0

! *** input ***
Do k = 1,q
Do i = 1,n
s1nIN(k,i) = sTEMP(k,i)
EndDo
EndDo

aIN(0) = 1.0D0
Do j = 1,m
aIN(j) = aTEMP(j)
EndDo

phiIN(0) = 1.0D0
Do i = 1,p
phiIN(i) = phiTEMP(i)
EndDo

!Print*, 'call tau'

CALL GeneratetauGibbs(idum, y, c, s1nIN, aIN, phiIN, tauOUT)

! *** output ***
Do k = 1,q
tauTEMP(k) = tauOUT(k)
EndDo
Do k = 1,q
tauGibbs(h,k) = tauTEMP(k)
EndDo
! +++++++++++++++++++++++++++++++++++++++++++++++++++
! This is used to run others separately
!Do k = 1,q
! tauTEMP(k) = tau(k)
!EndDo
!Do k = 1,q
! tauGibbs(h,k) = tauTEMP(k)
!EndDo
! +++++++++++++++++++++++++++++++++++++++++++++++++++

EndDo ! end outer loop on h

CALL CalculateGibbsEsts(sGibbs, aGibbs, etaGibbs, phiGibbs, tauGibbs)
```

```
CALL SaveGibbsSamples(sGibbs, aGibbs, etaGibbs, phiGibbs, tauGibbs)

! *******************************************************************
! *******************************************************************

Contains

! *******************************************************************
Subroutine Ready(y)
! reads data from a file
Integer i, k, t
Double Precision, Dimension(q,n) :: y

OPEN(1,FILE = 'Pn054E.DAT')
Do k = 1,q
Do i = 1,n
         Read(1,*) y(k,i)
EndDo
EndDo

End Subroutine Ready

! *******************************************************************
Subroutine GenerateAR3(phi, s, sigmasq, idum)
! generates simulated data from an AR(3) model
Integer i, k, idum
Double Precision, Dimension(q,n) :: s
Double Precision, Dimension(0:ls) :: signal
Double Precision, Dimension(0:p) :: phi
Double Precision, Dimension(1:q) :: sigmasq

!Print*, phi

Do k = 1, q
signal = 0.0D0

signal(0) = sqrt(sigmasq(k))*gasdev(idum)
signal(1) = -phi(1)*signal(0) + sqrt(sigmasq(k))*gasdev(idum)
signal(2) = -phi(1)*signal(1) - phi(2)*signal(0)  &
+ sqrt(sigmasq(k))*gasdev(idum)
Do i = 3, ls
signal(i) = -phi(1)*signal(i-1) &
-phi(2)*signal(i-2) &
-phi(3)*signal(i-3) &
+sqrt(sigmasq(k))*gasdev(idum)
```

```
EndDo
Do i = 1, ls
s(k,(ns1+i)) = signal(i)
EndDo

EndDo ! End loop on k
End Subroutine GenerateAR3

! *****************************************************************
Subroutine Filter(a, s, y, sigmasq, c, idum)
! signal s is filtered with a and put in y
Integer j, k, t, idum
Double Precision, Dimension(q,n) :: s
Double Precision, Dimension(0:m) :: a
Double Precision, Dimension(q,n) :: y
Double Precision, Dimension(1:q) :: sigmasq, c

!Print*, a
!Print*, y

Do k = 1, q

Do t = 1, n
Do j = 0, m
If ((t-j).ge.1) y(k,t) = y(k,t) + a(j)*s(k,(t-j))
EndDo
y(k,t) = y(k,t) + sqrt(c(k)*sigmasq(k))*gasdev(idum)
EndDo

EndDo  ! End loop on k

End Subroutine Filter

! *****************************************************************
Subroutine GeneratesGibbs(idum, Y, c, s1nin, ain, phiin, tauin, s1nout)
!generates a Gibbs iteration s(k,(p+1)),...s(k,n), k=1,...,q
Integer idum
Integer i, k, t
Double Precision, Dimension(q,n) :: Y
Double Precision, Dimension(q,n) :: s1nin
Double Precision, Dimension(0:m) :: ain
Double Precision, Dimension(0:p) :: phiin
Double Precision, Dimension(1:q) :: tauin, c
Double Precision, Dimension(q,n) :: s1nout
Double Precision Total1i, Total2i, sigmasqskiinv, sigmasqski, sigmaski, &
```

```
Total3i, Total4i, muski

Do k = 1,q

Do i = (p+1),n

! *******************************
! *** Calculate sigmasqskiinv ***
! *******************************

! *** Calulate Total1i ***
! *** sum a(t-i)^2 from t = max(i,(l+1) to min((i+m),n) ***
Total1i = 0.0D0
Do t = max(i,(l+1)),min((i+m),n)
Total1i = Total1i + ain(t-i)**2
EndDo

! *** Calulate Total2i ***
! *** sum phi(t-i)^2 from t = i to min((i+p),n) ***
Total2i = 0.0D0
Do t = i,min((i+p),n)
Total2i = Total2i + phiin(t-i)**2
EndDo

! *** Calulate sigmasqskiinv ***
sigmasqskiinv = tauin(k)*Total1i/c(k) + tauin(k)*Total2i
sigmasqski = 1/sigmasqskiinv
sigmaski = sqrt(sigmasqski)

! **********************
! *** Calculate muski **
! **********************

! *** Calculate Total3i ***
! *** sum a(t-i)*epsilontkiprime from t = max(i,(l+1)) to min((i+m),n) ***
Total3i = 0.0D0
Do t = max(i,(l+1)),min((i+m),n)
Total3i = Total3i + ain(t-i)* &
epsilontkiprime(t, k, i, Y, ain,s1nin)
EndDo

! *** Calulate Total4i ***
! *** sum phi(t-i)*etkiprime from t = i to min((i+p),n) ***
Total4i = 0.0D0
Do t = i,min((i+p),n)
```

```
Total4i = Total4i + phiin(t-i)*etkiprime(t,k,i,phiin,s1nin)
EndDo

! *** Calulate musi ***
muski = sigmasqski*( tauin(k)*Total3i/c(k) - tauin(k)*Total4i )

! *** Generate si ***
s1nout(k,i) = muski + sigmaski*gasdev(idum)

s1nin(k,i) = s1nout(k,i) ! update s1nout(k,i) for use ingenerating

EndDo ! end loop on i
EndDo ! end loop on k
End Subroutine GenerateSGibbs

! ******************************************************************
FUNCTION epsilontkiprime(t, k, i, Y, ain, s1nin)
! used in GeneratesGibbs
Integer t, i, j, k
Double Precision, Dimension(q,n) :: Y
Double Precision, Dimension(q,n) :: s1nin
Double Precision, Dimension(0:m) :: ain
Double Precision AS, epsilontkiprime

AS = 0.0D0
Do j = 0,m
If (j.ne.(t-i)) Then
If((t-j).ge.1) AS = AS + ain(j)*s1nin(k,(t-j))
EndIf
EndDo
epsilontkiprime = Y(k,t) - AS
return
End FUNCTION epsilontkiprime

! ******************************************************************
FUNCTION etkiprime(t,k,i,phiin,s1nin)
! used in GeneratesGibbs
Integer t, i, j, k
Double Precision, Dimension(q,n) :: s1nin
Double Precision, Dimension(0:p) :: phiin
Double Precision PhiS, etkiprime

PhiS = 0.0D0
Do j = 0,p
If (j.ne.(t-i)) Then
```

```
If((t-j).ge.1) PhiS = PhiS + phiin(j)*s1nin(k,(t-j))
EndIf
EndDo
etkiprime = PhiS
return
End FUNCTION etkiprime

! ***********************************************************************
Subroutine Generatesp1Gibbs(idum, sp1in, spnin, phiin, tauin, sp1out)
! generates Gibbs iteration s(k,1),...,s(k,p), k=1,...,q
Integer idum
Integer i, k
Double Precision, Dimension(q,n) :: spnin ! contains(p+1),...,n
Double Precision, Dimension(q,p) :: sp1in,sp1out ! contains p,...,1
Double Precision, Dimension(0:p) :: phiin
Double Precision, Dimension(1:q) :: tauin
Double Precision, Dimension(1:p) :: Phi1, Phi2, Phi3, diag, SumctPhit, &
musk, Z, LZ
Double Precision,Dimension(p,p) :: Phi1Phi1p, Phi2Phi2p, Phi3Phi3p, &
SumPhitPhitp, Sigmaphi, Sigmaphiinv, Ident, Sigmasinv, Sigmas, &
Sigmaschol
Double Precision c1, c2, c3

! Calculate sum PhitPhitp, where  p = 3
Phi1 = 0.0D0
Do j = 1,p
Phi1(j) = phiin(j)
EndDo
!Print*, 'Phi1'
!Print*, Phi1
Phi2 = 0.0D0
Do j = 2,p
Phi2(j-1) = phiin(j)
EndDo
!Print*, 'Phi2'
!Print*, Phi2
Phi3 = 0.0D0
Do j = 3,p
Phi3(j-2) = phiin(j)
EndDo
!Print*, 'Phi3'
!Print*, Phi3
Phi1Phi1p = 0.0D0
Do i = 1,p
Do j = 1,p
```

```
Phi1Phi1p(i,j) = Phi1(i)*Phi1(j)
EndDo
EndDo
Phi2Phi2p = 0.0D0
Do i = 1,p
Do j = 1,p
Phi2Phi2p(i,j) = Phi2(i)*Phi2(j)
EndDo
EndDo
Phi3Phi3p = 0.0D0
Do i = 1,p
Do j = 1,p
Phi3Phi3p(i,j) = Phi3(i)*Phi3(j)
EndDo
EndDo
SumPhitPhitp = Phi1Phi1p + Phi2Phi2p + Phi3Phi3p
!Print*, 'SumPhitPhitp'
!CALL PrintMatrix(SumPhitPhitp, p, p)

! Calculate Sigmaphiinv, where p = 3
Sigmaphi(3,3) = phiin(1)**2 + 1

Sigmaphi(2,2) = ( phiin(1)**2 - phiin(2) )**2 &
+ phiin(1)**2 + 1

Sigmaphi(1,1) = ( -phiin(1)**3 &
+ 2*phiin(1)*phiin(2) &
- phiin(3) )**2 + ( phiin(1)**2 &
- phiin(2) )**2 &
+ phiin(1)**2 + 1

Sigmaphi(3,2) = -phiin(1) &
*( phiin(1)**2 - phiin(2) ) &
- phiin(1)
Sigmaphi(2,3) = Sigmaphi(3,2)

Sigmaphi(3,1) = -phiin(1)*(-phiin(1)**3 &
+ 2*phiin(1)*phiin(2) - phiin(3) ) &
+ (phiin(1)**2 - phiin(2) )
Sigmaphi(1,3) = Sigmaphi(3,1)

Sigmaphi(2,1)= (phiin(1)**2 - phiin(2) ) &
*(-phiin(1)**3 + 2*phiin(1)*phiin(2) - phiin(3) ) &
-phiin(1)*( phiin(1)**2 - phiin(2) ) - phiin(1)
Sigmaphi(1,2) = Sigmaphi(2,1)
```

```
Do k = 1,q

Sigmaphi = Sigmaphi/tauin(k)
!Print*, 'Sigmaphi'
!CALL PrintMatrix(Sigmaphi, p, p)
!Read*

! Calculate Sigmaphiinv
Sigmaphiinv = Sigmaphi
CALL SweepMatrix(Sigmaphiinv, p, p)
!Print*, 'Sigmaphiinv'
!CALL PrintMatrix(Sigmaphiinv, p, p)
!Ident = MATMUL(Sigmaphi, Sigmaphiinv)
!Print*, 'Ident'
!CALL PrintMatrix(Ident, p, p)

Sigmasinv = Sigmaphiinv + tauin(k)*SumPhitPhitp
!Print*, 'Sigmasinv'
!CALL PrintMatrix(Sigmasinv, p, p)

Sigmas = Sigmasinv
CALL SweepMatrix(Sigmas, p, p)
!Print*, 'Sigmas'
!CALL PrintMatrix(Sigmas, p, p)

Sigmaschol = Sigmas
CALL choldc(Sigmaschol, p, p, diag)
! set upper part to zero
Do i = 1,p
Do j = (i+1),p
Sigmaschol(i,j) = 0.0D0
EndDo
EndDo
! plug in diag
Do i =1,p
Sigmaschol(i,i) = diag(i)
EndDo
!Print*, 'Sigmaschol'
!CALL PrintMatrix(Sigmaschol, p, p)

! Calculate musk
c1 = spnin(k,4)
c2 = spnin(k,5) + phiin(1)*spnin(k,4)
c3 = spnin(k,6) + phiin(1)*spnin(k,5) &
```

```
+ phiin(2)*spnin(k,4)
SumctPhit = c1*Phi1 + c2*Phi2 + c3*Phi3
!Print*, 'SumctPhit'
!Print*, SumctPhit

CALL MatVec(Sigmas, SumctPhit, musk, p, p)
musk = -tauin(k)*musk
!Print*, 'musk'
!Print*, musk

Do j = 1,p
Z(j) = gasdev(idum)
EndDo
CALL MatVec(Sigmaschol, Z, LZ, p ,p)
Do j = 1,p
sp1out(k,j) = musk(j) + LZ(j)
! sp1out = [s(k,p), s(k,(p-1)),...,s(k,1)]
EndDo
!Print*, 's(k,p),...,s(k,1) out'
!Print*, sp1out
!Read*

EndDo ! End loop on k

End Subroutine Generatesp1Gibbs

!*******************************************************************
! This is from Numerical Recipies.
! Calculate the cholesky decomposition of a symmetric covariance matrix.
SUBROUTINE choldc(a,n,np,p)
INTEGER n,np
Double Precision a(np,np),p(n)
INTEGER i,j,k
Double Precision sum

      do 13 i=1,n
        do 12 j=i,n
          sum=a(i,j)
          do 11 k=i-1,1,-1
            sum=sum-a(i,k)*a(j,k)
11        continue
          if(i.eq.j)then
            if(sum.le.0.)pause 'choldc failed'
            p(i)=sqrt(sum)
          else
```

```
            a(j,i)=sum/p(i)
          endif
12      continue
13    continue
      return
END SUBROUTINE choldc

!*******************************************************************
Subroutine MatVec(M, v, a, n, np)
Integer i, j, n, np
Double Precision, Dimension(np,np) :: M
Double Precision, Dimension(1:n) :: v, a
Double Precision sum

Do i = 1,n
sum = 0.0D0
Do j = 1,n
sum = sum + M(i,j)*v(j)
EndDo
a(i) = sum
EndDo
End Subroutine MatVec

!*******************************************************************
! See Weisberg p. 60.
! Used to calculate the inverse of a matrix.
Subroutine SweepMatrix(A, n, np)
Integer n,np
Double Precision, Dimension(np,np) :: A
Integer i,j,k

!Print*, 'Sweep a matrix'

!Print*, 'The matrix is:'
!CALL PrintMatrix(A, np, np)
!Read*

Do k = 1,n ! outer loop

!Print*, 'The matrix is:'
!CALL PrintMatrix(A, np, np)
!Read*

A(k,k) = 1.0D0/A(k,k)
!Print*, 'The matrix is: Step 1'
```

```
!CALL PrintMatrix(A, np, np)
!Read*

Do i = 1,n
If (i.ne.k) A(i,k) = -A(i,k)*A(k,k)
EndDo
!Print*, 'The matrix is: Step 2'
!CALL PrintMatrix(A, np, np)
!Read*

Do i = 1,n
Do j = 1,n
If ( (i.ne.k) .and. (j.ne.k) ) A(i,j) = A(i,j) + A(k,j)*A(i,k)
EndDo
EndDo
!Print*, 'The matrix is: Step 3'
!CALL PrintMatrix(A, np, np)
!Read*

Do j = 1,n
If ( j.ne.k) A(k,j) = A(k,j)*A(k,k)
EndDo
!Print*, 'The matrix is: Step 4'
!CALL PrintMatrix(A, np, np)
!Read*

EndDo ! outer loop
End Subroutine SweepMatrix

!*******************************************************************
!Subroutine PrintMatrix(A, n, np)
!Integer n,np
!Double Precision, Dimension(np,np) :: A
!Integer i,j

!Do i = 1,n
! Print*, ( A(i,j), j=1,n)
!EndDo
!End Subroutine PrintMatrix

! **********************************************************************
Subroutine GenerateAGibbs(idum, Y, c, s1nin, ain, etain, tauin, aout)
! generates a Gibbs iteration a(1),...,a(m)
Integer idum
Integer i, j, t, k, Count
```

```
Logical :: side = .false.
Double Precision, Dimension(q,n) :: Y
Double Precision,Dimension(0:m) :: ain,aout
Double Precision,Dimension(q,n) :: s1nin ! contains 1,...,n
Double Precision etain
Double Precision, Dimension(1:q) :: tauin, c
Double Precision Total1j, sigmasqinvaj, sigmasqaj, sigmaaj, Total2j, AS, &
epsilontjstar, muaj, x1, x2, const1, const2
Double Precision Temp1,Temp2,Temp3,Temp4,Temp5,Temp6,Temp7,etaj, U

!*** a0 = 1 ***
aout = 0.0D0
aout(0) = 1.0D0

!*** generate the other aj's ***
Do j = 1,m

!Print*, 'aj estimate number ',j

!*********************************************
!************* Calculate sigmasqaj ***********
!*********************************************
!*** Calculate Total1j ***
Total1j = 0.0D0

Do k = 1,q

!Do t = (l+1),n
Do t = (ns1+(l+1)),ns2
Total1j = Total1j + tauin(k)*(s1nin(k,(t-j))**2)/c(k)
!If ((t-j).ge.1) Total1j = Total1j + (s1nin(k,(t-j))**2)
EndDo

EndDo ! End loop on k

20 Format( F20.10 )
!Print*,'Total1j'
!Print 20, Total1j
sigmasqinvaj = 1/sigmasqalpha + Total1j
!Print*,'sigmasqinvaj'
!Print 20, sigmasqinvaj
sigmasqaj = 1/sigmasqinvaj
!Print*,'sigmasqaj'
!Print 20, sigmasqaj
sigmaaj = sqrt(sigmasqaj)
```

```
!Print*, 'sigmaaj'
!Print 20, sigmaaj

!*******************************************
!************* Calculate muaj ***************
!*******************************************
!*** Calculate Total2j ***
Total2j = 0.0D0

Do k = 1,q

!Do t = (l+1),n
Do t = (ns1+(l+1)),ns2
AS = 0.0D0
Do i = 0,m
If (i.ne.j) AS = AS+ain(i)*s1nin(k,(t-i))
!If ((i.ne.j).and.((t-i).ge.1)) AS = AS+ain(i)*s1nin(k,(t-i))
EndDo
!Print*, AS
epsilontjstar = Y(k,t) - AS
!Print*, 'epsilontjstar'
!Print*, epsilontjstar
Total2j = Total2j + tauin(k)*epsilontjstar*s1nin(k,(t-j))/c(k)
!If ((t-j).ge.1) Total2j = Total2j + epsilontjstar*s1nin(k(t-j))
!Print*, Total2j
EndDo

EndDo  ! End loop on k

!Print*,'Total2j'
!Print 20, Total2j

muaj =  sigmasqaj*( (mualpha/sigmasqalpha) + Total2j )

!Print*, 'muaj'
!Print 20, muaj

!*******************************************
!************* Calculate etaj ***************
!*******************************************

Temp1 = (muaj**2)/(2*sigmasqaj)
!Print*, Temp1
Temp2 = (mualpha**2)/(2*sigmasqalpha)
```

```
!Print*, Temp2
Temp3 = ( Temp1 - Temp2 )
!Print*, Temp3
If (Temp3.le.-500) Then
etaj = 1.0D0
! Print*,'Temp3 really small: etaj small'
! Print 20, etaj
ElseIf (Temp3.ge.500) Then
etaj = 0.0D0
! Print*,'Temp3 really big'
! Print*, etaj
Else
! Print*, ' Stuff within bounds'
Temp4 = exp( Temp3 )
! Print*, Temp4
Temp5 = sigmaaj/sigmaalpha
! Print*, Temp5
x1 = muaj/sigmaaj
const1 = alnorm(x1,side)
! Print*, 'const1'
! Print*, const1
x2 = mualpha/sigmaalpha
const2 = alnorm(x2,side)
! Print*, 'const2'
! Print*, const2
Temp6 = const1/const2
! Print*, Temp6
Temp7 = etain + (1-etain)*Temp6*Temp5*Temp4
! Print*, Temp7
etaj = etain/Temp7
! Print*, 'etaj'
! Print*, etaj
EndIf

! Generate a(j)

U = ran1(idum)
!Print*, 'U'
!Print 20, U
!Print*, 'etaj'
!Print 20, etaj
If ( U.le.etaj ) Then
aout(j) = 0.0D0
!Print*, aout(j)
!Read*
```

```
Else
Count = 0
Do
Count = Count + 1
!Print*, Count
aout(j) = muaj + sigmaaj*gasdev(idum)
!Print*, aout(j)
!Count = Count + 1
!If ((aout(j).gt.0.0D0).or.(Count.eq.100)) EXIT
If ((aout(j).gt.0.0D0).or.(Count.eq.10)) EXIT
EndDo
If (Count.eq.10) aout(j) = 0.0D0
!aout(j) = muaj + sigmaaj*gasdev(idum)
!If (aout(j).le.0.0D0) aout(j) = 0.0D0
!If (aout(j).le.-1.0D0) aout(j) = -1.0D0
!If (aout(j).ge.1.0D0) aout(j) = 1.0D0
EndIf

!Print*, 'a out'
!Print 20, aout(j)

ain(j) = aout(j)

!Read*

EndDo ! end loop on j

!Print*, 'aout inside'
!Print*, aout
!Read*
End Subroutine GenerateAGibbs

! ***********************************************************************
Subroutine GenerateetaGibbs(idum, ain, etaout)
! gerates a Gibbs iteration of eta
Integer j
Integer idum
Double Precision, Dimension(0:m) :: ain
Double Precision etaout, na, beta1star, beta2star

na = 0.0D0
Do j = 1,m
! If ( ain(j).eq.0.0D0) na = na + 1
If ( (ain(j).ge.-0.01D0).and.(ain(j).le.0.01D0) ) na = na + 1
EndDo
```

```
! *** Calulate beta1star ***
beta1star = na + beta1
!Print*, 'beta1star'
!Print*, beta1star

! *** Calculate beta2star ***
beta2star = m - na + beta2

If(beta2star.le.1.0D0) beta2star = 1.1D0

!Print*, 'beta2star'
!Print*, beta2star

etaout = betadev(beta1star, beta2star, idum)

!Print*, 'etaout'
!Print*, etaout

End Subroutine GenerateetaGibbs

! *********************************************************************
Subroutine GeneratePhiGibbs(idum, s1nin, tauin, Phi0, Sigma0, &
Sigma0inv, phiout)
! gernates a Gibbs iteration of phi
Integer i, j, t, k
Integer idum
Double Precision,Dimension(q,n) :: s1nin ! contains 1,...,n
Double Precision,Dimension(1:q) :: tauin
Double Precision,Dimension(0:p) :: phiout
Double Precision,Dimension(1:p) :: Phi0, diag, stSt, SumstSt, SP, TT, &
Phistar, Z, LZ
Double Precision,Dimension(p,p) :: Sigma0, Sigma0inv, StStp, SumStStp, &
Sigmastarinv, Sigmastar, Ident, Sigmastarchol
Integer Count

!Print*, 'calc phi'

!Print*, 'Sigma0'
!CALL PrintMatrix(Sigma0, p, p)
!Read*
!*** Calculate Sigma0inv ***
!!! I am cacluating this in the main program now !!!
!Sigma0inv = Sigma0
!CALL SweepMatrix(Sigma0inv, p, p)
```

```
!Print*,'Sigma0inv'
!CALL PrintMatrix(Sigma0inv, p, p)

!*** Calculate sigmastarinv ***
!*** Calculate the sum of St*STp ***
SumStStp = 0.0D0

Do k = 1,q

!Do t = (p+1),n
Do t = (ns1+(p+1)),ns2
Do i = 1,p
Do j = 1,p
StStp(i,j) = s1nin(k,(t-i))*s1nin(k,(t-j))
EndDo
EndDo
SumStStp = SumStStp + tauin(k)*StStp
EndDo

EndDo  ! End loop on k

!*** Calculate Sigmastarinv ***
Sigmastarinv = Sigma0inv + SumStStp
!Print*, 'Sigmastarinv'
!CALL PrintMatrix(Sigmastarinv, p, p)

!*** Calculate the inverse, Sigmastar ***
Sigmastar = Sigmastarinv
CALL SweepMatrix(Sigmastar, p, p)
!Print*, 'Sigmastar'
!CALL PrintMatrix(Sigmastar, p, p)

! Check that the Product is the identity
!Ident = MATMUL(Sigmastar,Sigmastarinv)
!Print*, 'Ident'
!CALL PrintMatrix(Ident, p, p)

!*** Calculate the lower square root matrix of Sigmastar ***
Sigmastarchol = Sigmastar
CALL choldc(Sigmastarchol, p, p, diag)
! set upper part to zero
Do i = 1,p
Do j =(i+1),p
Sigmastarchol(i,j) = 0.0D0
EndDo
```

```
EndDo
!plug in diag
Do i = 1,p
Sigmastarchol(i,i) = diag(i)
EndDo
!Print*, 'Sigmastarchol'
!CALL PrintMatrix(Sigmastarchol, p, p)

!Print*, 'Phi0'
!Print*, Phi0
!Read*
!*** Calculate Phistar ***
!*** Calculate sum st*St ***
SumstSt = 0.0D0

Do k = 1,q

!Do t = (p+1),n
Do t = (ns1+(p+1)),ns2
Do j = 1,p
stSt(j) = s1nin(k,t)*s1nin(k,(t-j))
EndDo
SumstSt = SumstSt + tauin(k)*stSt
EndDo

EndDo  ! End loop on k

!Print*, 'SumstSt'
!Print*, (SumstSt(j), j=1,p)

!*** Calculate Sigma0inv*Phi0 ***
CALL MatVec(Sigma0inv, Phi0, SP, p, p)
!Print*, 'Sigma0inv*Phi0'
!Print*, (SP(j), j = 1,p)

TT = SP - SumstSt
CALL MatVec(Sigmastar, TT, Phistar, p, p)
!Print*, 'Phistar'
!Print*, (Phistar(j), j = 1,p)

! Generate phiout, j = 1,p
! With a restiction on the phi's

Count = 0
Do
```

```
Do j = 1,p
Z(j) = gasdev(idum)
EndDo
!Print*, 'Z'
!Print*, (Z(j), j = 1,p)
CALL MatVec(Sigmastarchol, Z, LZ, p, p)
!Print*, 'LZ'
!Print*, (LZ(j), j = 1,p)
phiout(0) = 1.0D0
Do j = 1,p
phiout(j) = Phistar(j) + LZ(j)
EndDO

Count = Count + 1

If (Count.eq.100) Then
phiout(1) = phi(1)
phiout(2) = phi(2)
phiout(3) = phi(3)
EndIf

! A contraint on the phi's
If ( (abs(phiout(1)).ge.abs(phiout(2))).and. &
(abs(phiout(2)).ge.abs(phiout(3)))   ) EXIT
EndDo

!Print*, 'phi out'
!Print*, phiout
!Read*
End Subroutine GeneratePhiGibbs

! **********************************************************************
Subroutine GeneratetauGibbs(idum, Y, c, s1nin, ain, phiin, tauout)
! generates a Gibbs iteration of tau
Integer idum
Integer i, t, k
Double Precision, Dimension(q,n) :: Y
Double Precision, Dimension(q,n) :: s1nin
Double Precision, Dimension(0:m) :: ain
Double Precision, Dimension(0:p) :: phiin
Double Precision, Dimension(1:q) :: tauout, c
Integer gamma1star
Double Precision gamma2star, Sum1, Sum2, tau1
Integer Count
```

```
! *** Calulate gamma1star ***
gamma1star = q*(ls - m/2 - p) + gamma1
!Print*, 'gamma1star'
!Print*, gamma1star

!Do k = 1,q

! *** Calulate gamma2star ***
! *** Calulate Sum1 ***
Sum1 = 0.0D0

Do k = 1,q
!Do t = (l+1),n
Do t = (ns1+(l+1)),ns2
Sum1 = Sum1 + epsilonkt(k, t, Y, ain, s1nin)**2
EndDo
EndDo

!Print*, 'Sum1'
!Print*, Sum1

! *** Calulate Sum2 ***
Sum2 = 0.0D0

Do k = 1,q
!Do t = (p+1),n
Do t = (ns1+(p+1)),ns2
Sum2 = Sum2 + ekt(k, t, phiin, s1nin)**2
EndDo
EndDo

!Print*, 'Sum2'
!Print*, Sum2

gamma2star = Sum1/(2*c(1)) + Sum2/2 + gamma2
!gamma2star = Sum1/(2*c(k)) + Sum2/2 + gamma2
!Print*, 'gamma2star'
!Print*, gamma2star
!Read*

Count = 0
Do
tau1 = gamdev2(gamma1star, gamma2star, idum)
!tauout(k) = gamdev2(gamma1star, gamma2star, idum)
```

```
! !Print*, 'gamma1star'
! !Print*, gamma1star
! !Print*, 'gamma2star'
! !Print*, gamma2star
! Print*, 'tauout'
! Print*, tauout
Count = Count + 1
! !Print*, 'Count'
! !Print*, Count
If ( Count.eq.100) Then
tau1 = 1.0D0
!tauout(k) = 1.0D0
! !Print*, 'final tauout'
! !Print*, tauout
! !Read*
EndIf
If ( ( tau1.ge.0.1D0).and.(tau1.le.10.0D0) ) EXIT
!If ( (tauout(k).ge.0.1D0).and.(tauout(k).le.10.0D0) ) EXIT
EndDo

!EndDo ! end loop on k

Do k = 1,q
tauout(k) = tau1
EndDo

End Subroutine GeneratetauGibbs

! **********************************************************************
FUNCTION epsilonkt(k, t, Y, ain, s1nin)
! used in GeneratetauGibbs
Integer t, j, k
Double Precision, Dimension(q,n) :: Y
Double Precision,Dimension(0:m) :: ain
Double Precision,Dimension(q,n) :: s1nin
Double Precision AS, epsilonkt

AS = 0.0D0
Do j = 0,m
!If ((t-j).ge.1)
AS = AS + ain(j)*s1nin(k,(t-j))
EndDo
epsilonkt = Y(k,t) - AS
return
END FUNCTION epsilonkt
```

```
! **********************************************************************
FUNCTION ekt(k, t, phiin, s1nin)
! used in GeneratetauGibbs
Integer t, j, k
Double Precision,Dimension(0:p) :: phiin
Double Precision,Dimension(q,n) :: s1nin
Double Precision PhiS, ekt

PhiS = 0.0D0
Do j = 0,p
!If ((t-j).ge.1)
PhiS = PhiS + phiin(j)*s1nin(k,(t-j))
EndDo
ekt = PhiS
return
END FUNCTION ekt

!******************************************************************
Subroutine PrintMatrix(A, n, np)
Integer n,np
Double Precision, Dimension(np,np) :: A
Integer i,j

Do i = 1,n
Print*, ( A(i,j), j=1,n)
EndDo
End Subroutine PrintMatrix

! **********************************************************************
! **********************************************************************
! psudo-random number generators
! **********************************************************************
! This is from Numerical Recipies.
! Calculates a Uniform(0,1) random deviate.
FUNCTION ran1(idum)
INTEGER idum,IA,IM,IQ,IR,NTAB,NDIV
Double Precision ran1,AM,EPS,RNMX
PARAMETER (IA=16807,IM=2147483647,AM=1./IM,IQ=127773,IR=2836,&
NTAB=32,NDIV=1+(IM-1)/NTAB,EPS=1.2e-7,RNMX=1.-EPS)
INTEGER j,k,iv(NTAB),iy
SAVE iv,iy
DATA iv /NTAB*0/, iy /0/

      if (idum.le.0.or.iy.eq.0) then
```

```
        idum=max(-idum,1) ! Note: to get different seqs
        do 11 j=NTAB+8,1,-1 ! need neg idum.
          k=idum/IQ
          idum=IA*(idum-k*IQ)-IR*k
          if (idum.lt.0) idum=idum+IM
          if (j.le.NTAB) iv(j)=idum
11      continue
        iy=iv(1)
      endif
      k=idum/IQ
      idum=IA*(idum-k*IQ)-IR*k
      if (idum.lt.0) idum=idum+IM
      j=1+iy/NDIV
      iy=iv(j)
      iv(j)=idum
      ran1=min(AM*iy,RNMX)
      return
END FUNCTION ran1

! ************************************************************************
! This is from Numerical Recipies.
! Calculates a Normal(0,1) random deviate.
FUNCTION gasdev(idum)
INTEGER idum
Double Precision gasdev
!C     USES ran1
INTEGER iset
Double Precision fac,gset,rsq,v1,v2
!   ,ran1
SAVE iset,gset
DATA iset/0/

      if (iset.eq.0) then
1       v1=2.*ran1(idum)-1.
        v2=2.*ran1(idum)-1.
        rsq=v1**2+v2**2
        if(rsq.ge.1..or.rsq.eq.0.)goto 1
        fac=sqrt(-2.*log(rsq)/rsq)
        gset=v1*fac
        gasdev=v2*fac
        iset=1
      else
        gasdev=gset
        iset=0
      endif
```

```
      return
END FUNCTION gasdev

! *************************************************************************
! This is from Numerical Recipies.
! Calculates a gamma(ia) random deviate.
! ia needs to be an Integer value.
FUNCTION gamdev(ia,idum)
INTEGER ia,idum
Double Precision gamdev
!CU    USES ran1
INTEGER j
Double Precision am,e,s,v1,v2,x,y
! ,ran1

if(ia.lt.1)pause 'bad argument in gamdev'
    if(ia.lt.6)then
      x=1.
      do 11 j=1,ia
        x=x*ran1(idum)
11    continue
        x=-log(x)
    else
1       v1=ran1(idum)
        v2=2.*ran1(idum)-1.
      if(v1**2+v2**2.gt.1.)goto 1
        y=v2/v1
        am=ia-1
        s=sqrt(2.*am+1.)
        x=s*y+am
      if(x.le.0.)goto 1
        e=(1.+y**2)*exp(am*log(x/am)-s*y)

      if(ran1(idum).gt.e)goto 1
    endif
    gamdev=x
    return
END FUNCTION gamdev

! *************************************************************************
! Calculates a gamma(ia,ib) random deviate.
! ia needs to be an Integer value greater than 0.
! ib can be an an arbitrary number greater than 0.
FUNCTION gamdev2(ia,ib,idum)
INTEGER ia,idum
```

```
Double Precision ib
Double Precision gamdev2
!CU    USES ran1

gamdev2 = gamdev(ia,idum)/ib
return
END FUNCTION gamdev2

! **********************************************************************
! This algorithm is from Statistical Computing Kennedy and Gentle p.217
! Ahrens and Fisher (1974)
! It works for arbitrary alpha, beta greater than one.
FUNCTION betadev(alpha,beta,idum)
Integer idum
Double Precision alpha, beta, A, B, C, L
Double Precision betadev, mu, sigma, y, x, u

A = alpha-1
B = beta-1
C = A+B
L = C*dlog(C)
mu = A/C
sigma = 0.5D0/sqrt(C)
2 y = gasdev(idum)
x = mu + sigma*y
If ( (x.lt.0.0D0).or.(x.gt.1.0D0) ) GOTO 2
u = ran1(idum)
If ( log(u).le.(A*log(x/A)+B*log((1-x)/b)+L+0.5D0*y**2) ) Then
betadev = x
Else
GOTO 2
EndIF
return
END FUNCTION betadev

! **********************************************************************
FUNCTION alnorm(x, upper)  RESULT(fn_val)

!  Algorithm AS66 Applied Statistics (1973) vol.22, no.3

!  Evaluates the tail area of the standardised normal curve
!  from x to infinity if upper is .true. or
!  from minus infinity to x if upper is .false.

! ELF90-compatible version by Alan Miller
```

```
! Latest revision - 29 November 1997

!IMPLICIT NONE
INTEGER, PARAMETER    :: dp = SELECTED_REAL_KIND(15, 100)
REAL (dp), INTENT(IN) :: x
LOGICAL, INTENT(IN)   :: upper
REAL (dp)             :: fn_val

! Local variables
REAL (dp), PARAMETER :: zero = 0.0_dp, one = 1.0_dp, half = 0.5_dp, &
                        con = 1.28_dp
REAL (dp) :: z, y
LOGICAL   :: up

!*** machine dependent constants
REAL (dp), PARAMETER :: ltone = 7.0_dp, utzero = 18.66_dp

REAL (dp), PARAMETER :: p = 0.398942280444_dp, q = 0.39990348504_dp,   &
                        r = 0.398942280385_dp, a1 = 5.75885480458_dp,  &
                        a2 = 2.62433121679_dp, a3 = 5.92885724438_dp,  &
                        b1 = -29.8213557807_dp, b2 = 48.6959930692_dp, &
                        c1 = -3.8052E-8_dp, c2 = 3.98064794E-4_dp,     &
                        c3 = -0.151679116635_dp, c4 = 4.8385912808_dp, &
                        c5 = 0.742380924027_dp, c6 = 3.99019417011_dp, &
                        d1 = 1.00000615302_dp, d2 = 1.98615381364_dp,  &
                        d3 = 5.29330324926_dp, d4 = -15.1508972451_dp, &
                        d5 = 30.789933034_dp

up = upper
z = x
IF(z >=  zero) GO TO 10
up = .NOT. up
z = -z
10 IF(z <= ltone .OR. up .AND. z <= utzero) GO TO 20
fn_val = zero
GO TO 40
20 y = half*z*z
IF(z > con) GO TO 30

fn_val = half - z*(p-q*y/(y+a1+b1/(y+a2+b2/(y+a3))))
GO TO 40
30 fn_val =
r*EXP(-y)/(z+c1+d1/(z+c2+d2/(z+c3+d3/(z+c4+d4/(z+c5+d5/(z+c6))))))
40 IF(.NOT. up) fn_val = one - fn_val
```

```
RETURN
END FUNCTION alnorm

! ****************************************************************
! ****************************************************************
! Calculate Gibbs Estimates
! ****************************************************************
Subroutine CalculateGibbsEsts(sGibbs, aGibbs, etaGibbs, phiGibbs,
tauGibbs)
! calculates the Gibbs estimates by taking means of the Markov Chains
! and saves them into files
Double Precision, Dimension(Length,q,n) :: sGibbs
!Double Precision, Dimension(Reps,q,n) :: sGibbs
Double Precision, Dimension(Reps,m) :: aGibbs
Double Precision, Dimension(1:Reps) :: etaGibbs
Double Precision, Dimension(Reps,p) :: phiGibbs
Double Precision, Dimension(Reps,q) :: tauGibbs
Integer h, j, k
Double Precision, Dimension(q,n) :: sGibbsEst
Double Precision, Dimension(0:m) :: aGibbsEst
Double Precision :: etaGibbsEst
Double Precision, Dimension(0:p) :: phiGibbsEst
Double Precision, Dimension(1:q) :: tauGibbsEst

sGibbsEst = 0.0D0
Do k = 1,q
Do j = 1,n
        Do h = 1,Length
                sGibbsEst(k,j) = sGibbsEst(k,j) + sGibbs(h,k,j)
        EndDo
        sGibbsEst(k,j) = sGibbsEst(k,j)/Length
EndDo
EndDo
!Do k = 1,q
!Do j = 1,n
!        Do h = (Burn+1),Reps
!                sGibbsEst(k,j) = sGibbsEst(k,j) + sGibbs(h,k,j)
!        EndDo
!        sGibbsEst(k,j) = sGibbsEst(k,j)/(Reps-Burn)
!EndDo
!EndDo
OPEN(21, FILE = 'sEst.DAT')
Do k = 1,q
Do j = 1,n
        Write(21,*) sGibbsEst(k,j)
```

```
EndDo
EndDo

aGibbsEst = 0.0D0
Do j = 1,m
        Do h = (Burn+1), Reps
                aGibbsEst(j) = aGibbsEst(j) + aGibbs(h,j)
        EndDo
        aGibbsEst(j) = aGibbsEst(j)/(Reps-Burn)
EndDo
OPEN(22, FILE = 'aEst.DAT')
Write(22,*) '1.0'
Do j = 1,m
        Write(22,*) aGibbsEst(j)
EndDo

etaGibbsEst = 0.0D0
Do h = (Burn+1),Reps
        etaGibbsEst = etaGibbsEst + etaGibbs(h)
EndDo
etaGibbsEst = etaGibbsEst/(Reps-Burn)
OPEN(23, FILE = 'etaEst.DAT')
Write(23,*) etaGibbsEst

phiGibbsEst = 0.0D0
Do j = 1,p
        Do h = (Burn+1), Reps
                phiGibbsEst(j) = phiGibbsEst(j) + phiGibbs(h,j)
        EndDo
        phiGibbsEst(j) = phiGibbsEst(j)/(Reps-Burn)
EndDo
OPEN(24, FILE = 'phiEst.DAT')
Write(24,*) '1.0'
Do j = 1,p
        Write(24,*) phiGibbsEst(j)
EndDo

tauGibbsEst = 0.0D0
Do k = 1,q
Do h = Burn,Reps
        tauGibbsEst(k) = tauGibbsEst(k) + tauGibbs(h,k)
EndDo
tauGibbsEst(k) = tauGibbsEst(k)/(Reps-Burn)
EndDo
OPEN(25, FILE = 'tauEst.DAT')
```

```
Do k = 1,q
Write(25,*) tauGibbsEst(k)
EndDo

End Subroutine CalculateGibbsEsts

! ****************************************************************
Subroutine SaveGibbsSamples(sGibbs, aGibbs, etaGibbs, phiGibbs, tauGibbs)
! saves the Markov Chains into files.
Double Precision, Dimension(Length,q,n) :: sGibbs
Double Precision, Dimension(Reps,m) :: aGibbs
Double Precision, Dimension(1:Reps) :: etaGibbs
Double Precision, Dimension(Reps,p) :: phiGibbs
Double Precision, Dimension(Reps,q) :: tauGibbs
Integer h, j, k

!OPEN(31, FILE = 'sGibbs.DAT')
!Do k = 1,q
!Do h = Start,Reps
!        Write(31,*) (sGibbs(h,k,j), j = 1,n)
!EndDo
!EndDo

OPEN(32, FILE = 'aGibbs.DAT')
Do h = Start,Reps
        Write(32,*) (aGibbs(h,j), j = 1,m)
EndDo

OPEN(33, FILE = 'etaGibbs.DAT')
Do h = Start,Reps
        Write(33,*) etaGibbs(h)
EndDo

OPEN(34, FILE = 'phiGibbs.DAT')
Do h = Start,Reps
        Write(34,*) (phiGibbs(h,j), j = 1,p)
EndDo

OPEN(35, FILE = 'tauGibbs.DAT')
Do h = Start,Reps
        Write(35,*) (tauGibbs(h,k), k = 1,q)
EndDo

End Subroutine SaveGibbsSamples
```

```
! ************************************************************************

1000 End Program MultiDecon4
```

Bibliography

[1] Baumgardt, D.R. and Ziegler, K.A. (1988) Spectral evidence for source multiplicity in explosions: Application to Regional Discrimination of Earthquakes and Explosions. *Bulletin of the Seismological Society of America* **78**, 1773-1795.

[2] Bogert, B.P., Healy, M.J.R., and Tukey, J.W. (1962) The Frequency Anayisis of Time Series for Echoes: Cepstrum, Pseudo-Autocovariance, Cross Cepstrum and Saphe Cracking, in *Proceddings of a Symposiaum on Time Series Anaysis,* ed. M. Rosenblatt, New York: John Wiley and Sons, pp. 209-243. (Reprinted in D.R. Brillinger (ed.) (1984), *The Collected Works of John W. Tukey, Volume I*, Belmont, CA: Wadsworth, pp. 455-493.)

[3] Brockwell, P.J. and Davis, R.A. (1991) *Time Series: Theory and Methods, 2nd edition.* New York: Springer-Verlag.

[4] Burger, H.R. (1992) *Exploration Geophysics Of The Shallow Subsurface.* New Jersey: Prentice Hall.

[5] Casella, G. and George, E.I. (1992) Explaining the Gibbs Sampler. *American Statistician* **46**, 167-174.

[6] Chen, R. and Li, T.-H. (1995) Blind Restoration of Linearly Degraded Discrete Signals by Gibbs Sampling. *IEEE Transactions On Signal Processing* **43**, 2410-2413.

[7] Cheng, Q., Chen, R., and Li, T.-H. (1996) Simultaneous Wavelet Estimation and Deconvolution of Reflection Seismic Signals *IEEE Transactions On Geoscience and Remote Sensing* **34**, 377-384.

[8] Dargahi-Noubary. (1992) Stochastic Modeling and Identification of Seismic Records Based on Established Deterministic Formulations. *Jouranl of Time Series Analysis* **16**, 201-220.

[9] Davies, D. (1969) *Seismic Methods for Monitoring Underground Explosions: An Assessment of Status and Outlook.* Stockholm: Almqvist and Wiksells.

[10] Donoho, D. (1981) On Minimum Entropy Deconvolution. *in Findley, D.F., Ed., Applied Time-Series Anayysis II* Academic Press.

[11] Gelfand, A.E., Hills, S.E., Racine-Poon, A., and Smith, A.F.M. (1990) Illustration of Bayesian Inference in Normal Data Models Using Gibbs Sampling. *Journal of the American Statistical Association* **85**, 972-985.

[12] Gelfand, A.E. and Smith, A.F.M. (1990) Sampling-Based Approaches to Calculating Marginal Densities *Journal of the American Statistical Association* **85**, 398-409.

[13] Gelman, A. and Rubin, D.B. (1992) Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science* **7**, 457-472.

[14] Geyer, C.J. (1992) Practical Markov Chain Monte Carlo *Statistical Science* **7**, 473-511.

[15] Geman, S. and Geman, D. (1984) Stochastic Relaxation, Gibbs Disrtributions, and the Bayesian Restoration of Images *IEEE Trasactions of Pattern Analysis and Machine Intelligence* **6**, 721-741.

[16] Gilks, W.R., Richardson, S., and Spiegelhalter, D.J. (1996) *Markov Chain Monte Carlo in Practice.* New York: Chapman Hall.

[17] Goutsias, J. and Mendel, J.M. (1986) Maximum Likelihood Deconvolution: An Optimization Theory Perspective *Geophysics* **51**, 1206-1220.

[18] Hannan, E.J. and P.J. Thomson (1974) Estimating Echo Times *Technometrics* **16**, 77-84.

[19] Hedlin, M.A.H., Minster, J.B., and Orcutt, J.A. (1990) An Automatic Means to Discriminate Between Earthquakes and Quarry Blasts *Bulletin of the Seismological Society of America* **80**, 2143-2160.

[20] Moderator: Kass, R.E. Panelists: Carlin, B.P, Gelam, A. and Neal, R.M. (1998) Markov Chain Monte Carlo in Practice: A Roundtable Discussion. *The American Statistician* **52**, 93-100.

[21] Kennedy, W.J. and Gentle, J.E. (1980) *Statistical Computing.* New York: Marcel Dekker, Inc.

[22] Konya, C.J. and Walter, E.J. (1990) *Surface Blast Design.* New Jersey: Prentice-Hall, Inc.

[23] Kormylo, J.J. and Mendel, J.M. (1982) Maximum Likelihood Detection and Estimation of Bernoulli-Gaussian Processes *IEEE Transactions on Information Theory* **28**, 482-488.

[24] Mendel, J.M. (1990) *Maximum-Likelihood Deconvolution: A Journey into Model-Based Signal Processing.* New York: Springer-Verlag.

[25] Mendel, J.M. (1983) *Optimal Seismic Deconvolution: An Estimation Based Approach.* New York: Academic Press.

[26] Miller, A. The Internet Home Page of Alan J. Miller.

`http://www.ozemail.com.au/~milleraj/`

[27] Pearson, D.C., Strump, B.W., and Anderson, D.P. (1996) Physical Constraints on Mining Explosions.

`http://www.geology.smu.edu/~dpa-www/papers/srl/SRL.html`

[28] Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (1992) *Numerical Recipes in Fortran 77, The Art of Scientific Computing, 2rd edition.* New York: Cambridge University Press.

[29] Shumway, R.H. (1988) *Applied Time Series Analysis.* Englewood Cliffs: Prentice Hall.

[30] Shumway, R.H., Baumgart, D.R. and Der, Z.A. (1998) A Cepstral F Statistic for Detecting Delay-Fired Seismic Signals. *Technometrics* **40**, 100-110.

[31] Shumway, R.H. and Der, Z.A. (1995) Deconvolution of Multiple Time Series *Technometrics* **27**, 385-393.

[32] Shumway, R.H. and McQuarrie, A.D.R. (1994) Statistical Discrimination Studies for Nuclear Verification, Final Repost PL-TR-94-2283, Phillips Laboratory, Directorate of Geophysics, Air Force Materiel Command, Hanscom AFB, MA 01731-3010 (ADA 293572).

[33] Shumway, R.H. (1997) Detection of Ripple Firing in Time and Frequency, Final Scientific Repost, Phillips Laboratory, Directorate of Geophysics, Air Force Materiel Command, Hanscom AFB, MA 01731-3010 (Pl-TR-97).

[34] Smith, A.T. (1989) High-Frequency Seismic Observations and Models of Chemical Explosions: Implications for Discrimination of Ripple-Fired Mining Balsts *Bulletin of the Seismological Society of America* **79**, 1089-1110.

[35] Smith, A.T. (1993) Discrimination of Explosions from Simultaneous Mining Blasts *Bulletin of the Seismological Society of America* **83**, 160-179.

[36] Strump, B.W. and Reinke, R.E. (1988) Experimental Confirmation of Superposition From Small-Scale Explosions *Bulletin of the Seismological Society of America* **78**, 1056-1073.

[37] Tanner, M.A. (1996) *Tools for Statistical Inference, Methods for the Exploration of Posterior Distributions and Likelihood Functions, 3rd edition.* New York: Springer-Verlag.

[38] Tjøstheim, D. (1975) Autoregressive Representation of Seismic P-wave Signals with an Application to the Problem of Short-Period Discriminants *Geophys. J. R. astr. Soc.* **43**, 269-291.

[39] Tjøstheim, D. (1975) Some Autoregressive Models for Short-Period Seismic Noise *Bulletin of the Seismological Society of America* **65**, 677-691.

[40] U.S. Congress, Office of Technology Assessment, *Seismic Verification of Nuclear Testing Treaties*, DTA-ISC-361, (Washington, D.C.: U.S. Government Printing Office, May 1988).

[41] Weisberg, S. (1985) *Applied Linear Regression, Second Edition.* New York: Wiley.

[42] Wiggins, R.A. (1978) Minimum Entropy Deconvolution *Geoexploration* **16**, 21-35.