Short datalength effects in an asymptotically efficient ARMA spectral estimator

Carriere, Rob

Award date:
1987

FACULTEIT DER ELEKTROTECHNIEK
TECHNISCHE UNIVERSITEIT
EINDHOVEN
Vakgroep Meten en Regelen

SHORT DATALENGTH EFFECTS IN AN ASYMPTOTICALLY EFFICIENT ARMA SPECTRAL ESTIMATOR

by Rob Carrière

Rapport van het afstudeerwerk
uitgevoerd van april 1986 tot januari 1987
in opdracht van prof. dr. ir. P. Eykhoff (TUE)
en Dr. R.L. Moses (OSU)
onder leiding van Dr. R.L. Moses

DEPARTMENT OF ELECTRICAL ENGINEERING
EINDHOVEN UNIVERSITY
OF TECHNOLOGY
Group Measurement and Control

SHORT DATALENGTH EFFECTS IN AN ASYMPTOTICALLY EFFICIENT ARMA SPECTRAL ESTIMATOR

by Rob Carrière

Report on the thesis project
done from april 1986 to january 1987
under direction of prof. dr. ir. P. Eykhoff (TUE)
and Dr. R.L. Moses (OSU)
coached by Dr. R.L. Moses

Contents

List of Figures

List of Tables

Preface

This is a report on the *afstudeerwerk*, done to partially fulfill the requirements of an *Elektrotechnisch Ingenieur* degree from the *Technische Universiteit Eindhoven.* The project has been done with Dr. R. Moses as the local *afstudeerhoogleraar* and prof. dr. ir. P. Eijkhoff as the *afstudeerhoogleraar* from TUE/E, from April 1986 to April 1987.

The subject of the *afstudeerwerk* was finite data length effects in an efficient linear estimator developed by Stoica, Söderström, and Friedlander [16]. The Introduction chapter gives background on ARMA estimation in general, and the subject algorithm in particular. Chapter 2 gives an overview of the estimator and its properties. Chapter 3 investigates the characteristics of the algorithm for finite data lengths by means of Monte Carlo simulations and Chapter 4 discusses the properties of several variations on the original algorithm developed to improve its short data length performance. Again, these properties are investigated through Monte Carlo simulations. All results are summarized in the conclusion chapter. Two appendices give mathematical details of the estimator (Appendix A) and a description of the software used in the simulations (Appendix B), while the Figures for Chapters 3 and 4 have been gathered in Appendix C for easier reference. Some copies of this thesis will also have a complete listing of the software attached.

Summary

The short data length behaviour of a recently proposed [16] computationally efficient approximate maximum likelihood estimation algorithm is studied through Monte Carlo simulations. It is found that short data lengths combined with a large number of instruments results in very high variances, especially when the process being estimated has zeros near the unit circle.

Several modifications of the algorithm are considered to reduce the problem mentioned above. First, a version which is recursive in the number of instruments and which adaptively chooses the number of instruments and postiterations is developed. A second modification uses a stabilized version of the estimated denominator polynomial. A version that forces the numerator estimate to be non negative definite is considered, but it fails to give major improvements over the original algorithm. Finally, using overdetermined Yule Walker equations instead of the minimal number is found to markedly improve the quality of the estimates.

Samenvatting

Het korte datalengte gedrag van een onlangs voorgestelde [16] schatter die maximum likelihood benadert en weinig rekentijd vraagt wordt bestudeerd met behulp van Monte Carlo simulaties. Het blijkt dat korte datalengten samen met een groot aantal instrumenten zeer hoge varianties van de schattingen opleveren, in het bijzonder als het te schatten proces nulpunten heeft dicht bijde eenheids cirkel.

Een aantal aanpassingen van het algoritme bedoeld om het bovengenoemde probleem te verminderen worden bestudeerd. Als eerste is een versie die recursief is in het aantal instrumenten en die het aantal instrumenten en na-iteraties adaptief kiest ontwikkeld. Een tweede aanpassing gebruikt een gestabiliseerde versie van het noemer polynoom. Een versie die niet-negatief definite schattingen van het teller polynoom dwingt wordt bestudeerd, maar dit geeft geen aanmerkelijke verbeteringen ten opzichte van het originele algoritme. Ten slotte, het gebruik van overbepaalde Yule Walker vergelijkingen verbetert de kwaliteit van de schattingen aanzienlijk.

Rob Carrière, vakgroep ER, faculteit Elektrotechniek, Technische Universiteit Eindhoven/
Department of Electrical Engineering, The Ohio State University

Acknowledgements

Anyone who carries out an *afstudeerwerk* across an ocean, will have excelent reason to thank many people. I would like to thank, more or less in chronological order, Dr. Moses for suggesting the idea, prof. Eykhoff and Dr. Moses for making it possible, the staff of the departments of Electrical Engineering at both TUE and OSU for making the ride so much smoother than it might have been, and the department of Electrical Engineering and the ElectroScience Laboratory at OSU for providing the financial support without which it would all have remained a dream.

Then I have to thank prof. Eykhoff and Dr. Moses once again, for their excelent help as my *afstudeerhoogleraren.*

Dr. Krishnamurhty carefully read the thesis and suggested many valuable improvements.

I also greatly apreciate the stimulating environment that the department provides, and want to thank everybody who helps make it so.

The position of honor, though, is reserved for my parents, for far too many reasons to list here; instead, I dedicate this thesis to them.

opgedragen aan mijn ouders

Notation

$(a_i)_{i=0}^{na}$: the denominator polynomial coefficients ($a_0 = 1$)

$A(z)$: the polynomial formed by $(a_i)_{i=0}^{na}$

$(c_i)_{i=0}^{nc}$: the numerator polynomial coefficients

$C(z)$: the numerator polynomial

$(b_i)_{i=-nc}^{nc}$: the "numerator square" polynomial coefficients

: Note that $b_{-i} = b_i$

$B(z)$: the numerator square polynomial: $C(z)C(z^{-1})$

$(r_i)_i$: the autocorrelation coefficients

$(y_i)_i$: the time series measurements

$(\tilde{r}_i)_i$: the standard unbiased estimates of $(r_i)_i$

$(\tilde{a}_i)_{i=1}^{na}$: the Yule Walker estimates of $(a_i)_{i=1}^{na}$ based on $(\tilde{r}_i)_i$

$(\tilde{b}_i)_{i=0}^{nc}$: the estimates of $(b_i)_{i=1}^{nc}$ based on $(\tilde{r}_i)_i$ and $(\tilde{a}_i)_{i=1}^{na}$

$(\alpha_i)_i$: the coefficients of z^i in the long division of $z^{-(nc+na)}\frac{\left(\sum_{k=-nc}^{nc} b_k z^{-k}\right)}{A^2(z^{-1})}$

$(\beta_i)_i$: the coefficients of $B(z) \cdot B(z)$

$(\hat{r}_i)_i$: the revised estimates of $(r_i)_i$

$(\hat{a}_i)_{i=1}^{na}$: estimates of $(a_i)_{i=1}^{na}$ based on $(\hat{r}_i)_i$

$(\hat{b}_i)_{i=0}^{nc}$: estimates of $(b_i)_{i=1}^{nc}$ based on $(\hat{r}_i)_i$ and $(\hat{a}_i)_{i=1}^{na}$

See also table B.1, on page 62 for the variables used in the computer program.

1. Introduction

There are various areas in engineering where the problem of finding a compact representation of a signal arises. Reasons for using such a representation include: the representation is directly related to the information in the signal, the representation can be transmitted over channels with a smaller bandwidth[1] than would be possible for the original signal, and the representation has noise immunity properties.

One representation that is used very widely is the ARMA model. ARMA is an abbreviation of AutoRegressive Moving Average, meaning that the signal's value at some point in time is both a function of the past values of the signal (AutoRegressive) and of the present and past values of an i.i.d. white noise signal (Moving Average). That is, we can model the signal as:

$$y_k = -\sum_{i=1}^{na} a_i y_{k-i} + \sum_{j=0}^{nc} c_j e_{k-j}$$

where $(y_k)_k$ is the signal of interest, and $(e_k)_k$ is a white noise signal. The reasons the ARMA model is popular are: that it is relatively easy to compute from the signal $(y_k)_k$, and that the a_i and c_j coefficients relate directly to the shape of the spectrum of the signal.

Another way of looking at the ARMA model results from its frequency domain interpretation: We are modelling the signal $(y_k)_k$ as the output of a linear filter driven by white noise (see Figure 1). Since white noise is spectrally flat by definition, the spectral shape of the signal is completely determined by the filter.

[1]This includes memory storage, which can be seen as a transmission through time rather than space.

Figure 1.1: Signal Model

As an example application, consider the transmission of speech over a narrow-band digital channel (the situation most western telephone companies want to convert to). One way would be to simply digitize the speech signal and transmit the digitized version; however, given that reliable human recognition of the speech signal requires that the spectrum of the signal at the receiving end coincides with the original spectrum for at least several kiloHertz, and that we need a reasonably fine quantization for the same reason, it is clear that we need a channel capacity of many kilobaud with this method. On the other hand, we could compress the speech, using ARMA modeling techniques, transmit the compressed signal, and reconstruct the speech from the compressed signal at the receiver. Transmission equipment is available that transmits speech understandably in this way while using as little as 4 kilobaud [10]. Since state-of-the-art is not yet good enough to allow the person at the receiving end to recognize the speaker, this method is not yet suitable for general use; however, applications in verification/identification of authorized personel and in voice command of machinery exist [15].

ARMA modeling techniques are used in various other areas: in communications, including adaptive matched filters in transmission channels see [1], and in geologic surveying by means of acoustical procedures (used e.g. to find oil and gas): see [2] and [13]. See [12] for more applications.

There are various ways of estimating the coefficients a_i and c_i, but they can generally be divided into two groups: computationally simple but not of very high statistical quality (i.e. the variance of the estimated coefficients is higher than necessary), or computationally expensive but of high statistical quality. This thesis is concerned with a compromise algorithm: on the one hand it is computationally efficient, but on the other hand it asymptotically (i.e. for the data length of the available signal growing to infinity) achieves the maximum statistical quality. The thesis studies the behaviour of the algorithm for finite data lengths.

The subject algorithm has been formulated in [16] and its computational aspects have been addressed in [14]. It is known [16] that the algorithm approaches the Cramèr-Rao Lower Bound (CRLB) as the number of measured data points N (i.e. we have measurements of $y_0 \cdots y_N$ for some N) grows to infinite (i.e. $N \rightarrow \infty$). Under these asymptotic conditions it has also been shown that the algorithm is numerically well-conditioned if the polynomial C does not have roots with approximately unit magnitude. The algorithm as formulated in [16] and [14] leaves two freedoms of choice: the number of instrumental variables to be used and the number of postiterations (both terms will be clarified in the next chapter). Asymptotic variance expressions as a function of each of these parameters are presented in [16]

In this report we investigate what effects a finite data length has on the properties of the algorithm. Numerical problems (ill-conditioning) for finite N, and

the behaviour of the algorithm as a function of the number of postiterations and as a function of the number of instrumental variables are studied, and are compared with the asymptotic expressions. We will find that several problems do exist and we will develop several variations on the algorithm that attempt to reduce these problems.

2. Background

This chapter is a short summary of the relevant material from [16] and [14]; it formulates the spectral model as well as the estimator and briefly discusses them.

2.1 The ARMA Spectral Model

As stated in the introduction, we want to estimate the spectrum of a signal $(y_k)_k$. The general formulation of a spectral estimator would be:

1. parameterize the spectral density function (choose a model);
2. estimate the parameters of that model; and
3. compute the estimated spectral density function from these estimated parameters.

The parameterization we will use is the ARMA spectral model defined below.

Consider the ARMA spectral model

$$A(q^{-1})y(t) = C(q^{-1})e(t) \tag{2.1}$$

where

$$e(t) = \text{white Gaussian noise with zero mean and variance } \lambda^2 \tag{2.2}$$

$$q^{-1} = \text{unit delay operator (} q^{-1}y(t) = y(t-1) \text{).} \tag{2.3}$$

y and e are scalars

and the following standard assumptions are made:

Assumption 2.1

$$A(z) \cdot C(z) = 0 \Longrightarrow |z| > 1$$

Assumption 2.2

$$a_{na}.c_{nc} \neq 0 \text{ and } \{A(z), C(z)\} \text{ are coprime polynomials}$$

Together, this means the ARMA representation is minimal, stable and invertible. The algorithm studied here does not extend to cases where these assumptions do not hold, e.g. the case of sinusoids in white noise cannot be modelled in this way, as it violates Assumption 2.1. However, many practical cases do fall in the category described by Assumptions 2.1 and 2.2.

Note also that we need to know the orders na and nc of the polynomials A and C; here we will assume that these are given.

Now we can introduce:

$$B(z) = C(z^{-1})C(z) \tag{2.4}$$

Under Assumption 2.1, $C(z)$ can be uniquely determined from $B(z)$, and, as we shall see below, $B(z)$ is easier to estimate. We also define

$$r_k = E\{y(t)y(t-k)\} = \text{the covariance of } y(t) \text{ at lag } k \tag{2.5}$$

$$\Phi(z) = \sum_{k=-\infty}^{\infty} r_k z^{-k} = \text{the spectral density of } y(t) \tag{2.6}$$

where $E\{\cdot\}$ denotes the expectation operator and z is a complex variable.

It is well known that

$$\Phi(z) = \lambda^2 \frac{C(z)C(z^{-1})}{A(z)A(z^{-1})} \tag{2.7}$$

The problem considered here is to estimate $\Phi(z)$ from N measurements of $(y_k)_k$. We intend to carry out the estimation by parameterizing the spectrum and then

estimating the parameters from the available data. There are several different possible methods of parameterizing the spectrum. Specifically, as [16] and [14] suggest, one can parameterize Φ by

$$\{\lambda^2, a_1, \cdots, a_{na}, c_1, \cdots, c_{nc}\} \tag{2.8}$$

$$\text{or by} \qquad \{r_0, \cdots, r_{na+nc}\} \tag{2.9}$$

$$\text{or by} \qquad \{r_0, \cdots, r_{nc}, a_1, \cdots, a_{na}\} \tag{2.10}$$

We parameterize on (2.9). Finding the a parameters from the r parameters can be done by solving the well-known Yule Walker equation [3]:

$$\begin{bmatrix} r_{nc} & \cdots & r_{nc+1-na} \\ \vdots & & \vdots \\ r_{K-1} & \cdots & r_{K-na} \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ \vdots \\ a_{na} \end{bmatrix} = - \begin{bmatrix} r_{nc+1} \\ \vdots \\ r_K \end{bmatrix}, K \geq na + nc \tag{2.11}$$

The b parameters can then be found using the well-known equality [11]:

$$b_k = [a_0 \ldots a_{na}] \begin{bmatrix} r_k & \cdots & r_{k+na} \\ \vdots & & \vdots \\ r_{|k-na|} & \cdots & r_k \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_{na} \end{bmatrix} \tag{2.12}$$

Thus we can uniquely convert from any of the parametrizations (2.8) – (2.10) to any other, i.e. they are mathematically equivalent. However, as [16] discusses, there are good reasons for using the second parameterization above; therefore we define:

$$\theta = [r_0 \cdots r_{na+nc}]^T \tag{2.13}$$

As both [16] and [14] show, this is a well-defined parameterization.

2.2 An Asymptotically Efficient Algorithm

Obviously, the accuracy of the spectral estimate is determined by the accuracy of the parameter estimate, so finding the parameter estimator with the lowest possible variance is important to the overall accuracy.

As a first try, we could consider either using standard unbiased sample covariance estimation:

$$\tilde{r}_k = \frac{1}{N-k}\sum_{t=1}^{N-k} y(t)y(t+k) \quad k = 0,1,\ldots \tag{2.14}$$

$$\tilde{r}_{-k} = \tilde{r}_k \tag{2.15}$$

However, the $\tilde{r}_k$ are in general not statistically efficient estimates, and can in fact have very bad accuracy. We could instead consider a Maximum Likelihood (ML) autocorrelation estimator: this will be statistically efficient, but on the other hand, will also be computationally expensive. The idea considered in [16] and [14] is to improve on the estimates in (2.14) in such a way that it attains some of the desirable properties of ML; specifically, it should be asymptotically (i.e. for $N \longrightarrow \infty$) statistically efficient, that is,

$$\lim_{N\to\infty} N\mathrm{E}\{(\hat{R}-R)(\hat{R}-R)^T\}$$

should be minimal. In this equation R is the vector

$$R = \begin{bmatrix} r_0 \\ \vdots \\ r_{m-1} \end{bmatrix}$$

for some integer m, and $\hat{R}$ is the estimate of R.

As we are going to have several different kinds of estimates, the following general notation is used: a variable with a tilde (˜) denotes the initial estimate of that variable, a variable with a circumflex (ˆ) denotes the improved estimate of that variable.

2.3 The Correction Step from Yule Walker to Approximate Maximum Likelihood: Heuristic Presentation

The following is a plausability argument for the correction step used in the algorithm: a more formal presentation is given in the next section.

Consider a scalar random variable $\tilde{\theta}$ with unknown expected value $\check{\theta}$. We would like to find a new random variable $\hat{\theta}$ with the same expected value $\check{\theta}$, but a smaller variance. Assume that we also have a random variable z (called an *instrumental variable*) with expected value 0 that is a (deterministic) function of $\tilde{\theta}$: $z = z(\tilde{\theta})$ and has the property that the variance of z increases monotonically with the variance of $\tilde{\theta}$.

The above problem can be cast into the language of linear spaces. Consider the Hilbert space H of all random variables with zero mean and finite second moment. The addition and scalar multiplication operations in this space are the obvious ones; the inner product of two random variables x and y (denoted $< x, y >$) is defined to be $\mathrm{E}(xy)$ ([18] shows that this is a well-defined inner product). Note that the norm generated by the inner product ($\sqrt{< x, x >}$) is the standard deviation of the random variable.

We have two vectors z, and $\theta \stackrel{\text{def}}{=} \tilde{\theta} - \check{\theta}$. We would like to minimize the norm of $\|\theta\|$, but are hindered by the fact that we do not know $\|\theta\|$ ($\check{\theta}$ is unknown and we

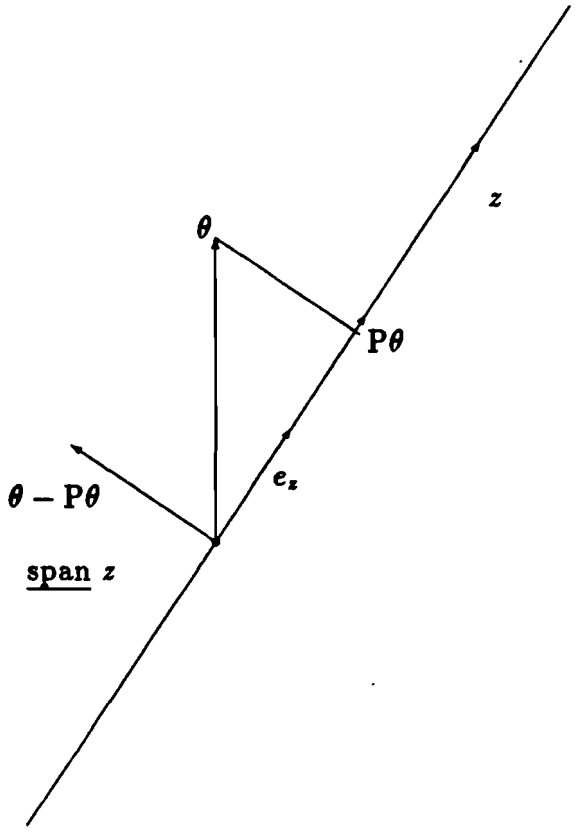

Figure 2.1: Geometric Intepretation of the Effect of the Instrumental Variables on the Variance of the Estimate

have only measurements of $\tilde{\theta}$). However, since we know how z depends on θ, we can estimate the inner product (= covariance) $< \theta, z >$, and standard deviation $\sqrt{< z, z >} = \|z\|$, from the known measurements of $\tilde{\theta}$ and z. This gives us the strategy of subtracting the projection on the subspace spanned by z from θ, see Figure 2.1. To determine the projection we divide z by its norm to yield e_z; now the projection P on the z-subspace is simply the inner product $< \theta, e_z >$ times the vector e_z:

$$\mathrm{P}\theta = < \theta, e_z > e_z$$

If we want to express this in terms of z instead of e_z, we get a factor $\|z\|$ in the inner product ($< \theta, z >= \|z\| < \theta, e_z >$), and another one from the vector at the

end ($z = \|z\| e_z$), giving a total of $\|z\|^2 = < z, z >$; this has to be divided out again:

$$\mathrm{P}\theta = \frac{< \theta, z >}{< z, z >} z$$

Finally, we need $\theta - \mathrm{P}\theta$:

$$\theta - \mathrm{P}\theta = \theta - \frac{< \theta, z >}{< z, z >} z$$

which gives:

$$\hat{\theta} - \check{\theta} = \tilde{\theta} - \check{\theta} - \frac{< \theta, z >}{< z, z >} z \tag{2.16}$$

where fortunately the $\check{\theta}$ terms cancel out, leaving us only with terms we can estimate from the given data. In probabilistic terms, equation (2.16) is: again:

$$\hat{\theta} = \tilde{\theta} - \frac{\mathrm{E}(\theta z)}{\mathrm{E}(zz)} z$$

For vectors this generalizes to:

$$\hat{\theta} = \tilde{\theta} - \mathrm{E}\{\theta z^T\}(\mathrm{E}\{zz^T\})^{-1} z \tag{2.17}$$

2.4 The Correction Step: Overview and Implementation Issues

In this section we present the correction step without proof; some proofs are given in Appendix A.

The general strategy for the correction step is as follows: we will try to find a random vector X, that is completely determined by the N available data samples, and such that for $N \to \infty$ the distribution of X can be completely specified from θ. Furthermore, using $\mathcal{N}(m, \Sigma)$ to indicate a Gaussian distribution with mean vector m and covariance matrix Σ, X should be such that:

$$\sqrt{N}(X - \overline{X}) \xrightarrow[N \to \infty]{\text{dist}} \mathcal{N}(0, W) \tag{2.18}$$

where

$$\overline{X} = \begin{bmatrix} \theta \\ 0 \end{bmatrix},$$

and where θ is as defined in (2.13). The covariance matrix W, which is assumed to be nonsingular, may depend on θ. Finally, assume that an estimate $\hat{W}$ of W can be calculated from the available data, and that $|\hat{W} - W|$ is $O(1/\sqrt{N})$ [1].

It is shown in [16] that under these circumstances we have

$$W = \begin{bmatrix} W_{11} & W_{12} \\ W_{12}^T & W_{22} \end{bmatrix} \tag{2.19}$$

$$X = \begin{bmatrix} \theta \\ z \end{bmatrix} \Big\} m \times 1 \tag{2.20}$$

where

$$W_{11} = \mathrm{E}(\tilde{\theta} - \check{\theta})(\tilde{\theta} - \check{\theta})^T \quad n\theta \times n\theta \tag{2.21}$$

$$W_{12} = \mathrm{E}z(\tilde{\theta} - \check{\theta})^T \quad n\theta \times (m - n\theta) \tag{2.22}$$

$$W_{22} = \mathrm{E}zz^T \quad (m - n\theta) \times (m - n\theta) \tag{2.23}$$

and

$$m = \dim X$$

$$n\theta = \dim \theta$$

where we assume

$$m > na + nc + 1$$

[1] '$O(\epsilon)$' means the standard deviation is equal to $K\epsilon$, where K is independent of ϵ.

and where z is the same random vector discussed in the previous section. (z is often called the instrumental variable vector or the instrumental variables)

In Appendix A we show that

$$\underline{\theta} = \tilde{\theta} - W_{12}W_{22}^{-1}z \tag{2.24}$$

is an approximate ML estimate of θ. However, this is in general not implementable since W may depend on θ and therefore be unknown. However, since z is $O(1/\sqrt{N})$, we can replace W_{ij} by their consistent estimates $\hat{W}_{ij}$ without affecting the order of the approximation (see [16] for details and proof). This gives:

$$\hat{\theta} = \tilde{\theta} - \hat{W}_{12}\hat{W}_{22}^{-1}z \tag{2.25}$$

It remains to choose the instrumental variable vector z: we take z equal to:

$$\begin{aligned} z &= [z_1 \cdots z_{m-n\theta}]^T \\ z_k &= \sum_{i=0}^{na}\sum_{j=0}^{na} \tilde{a}_i \tilde{r}_{nc+na+k-i-j}\tilde{a}_j \end{aligned} \tag{2.26}$$

which is clearly a deterministic function of $\tilde{\theta}$ through (2.11). As [17] discusses, z asymptotically has zero mean and normal distribution, so it fulfills the requirements of the previous section. This specific form of the vector z leads to a relatively simple distribution of X; it was introduced by Walker [17], who proves that it is optimal in the sense that if one chooses a z vector of length n, there does not exist any zero mean n-vector that is more correlated with $\tilde{\theta}$; in other words, this choice will take out the maximum amount of variance from the estimate $\tilde{\theta}$ (see Figure 2.1).

As is shown in Appendix A of [16], this z achieves the properties of (2.18), with

$$[W_{11}]_{ij} = \mathrm{E}\{[v(t-i)+v(t+i)]v(t-j)\}\ ,\ i,j = 0,\ldots,n\theta-1 \tag{2.27}$$

$$v(t) = \lambda \frac{C^2(q^{-1})}{A^2(q-1)} e(t) \tag{2.28}$$

$$[W_{22}]_{ij} = \mathrm{E}\{A^2(q^{-1})v(t-i)A^2(q^{-1})v(t-j)\} \tag{2.29}$$

$$= \text{ the coefficient of } z^{i-j} \text{ in } [\sum_{k=-nc}^{nc} b_k z^{-k}]^2 \ , \ i,j = 1,\ldots,m-n\theta$$

$$[W_{12}]_{jk} = \mathrm{E}\{A^2(q^{-1})[v(t+na+nc-j)+v(t+na+nc+j]v(t-k)\}$$

$$= \alpha_{k+j} + \alpha_{k-j} \ , \ k = 1,\ldots,m-n\theta \tag{2.30}$$

$$\alpha_s = \text{ the coefficient of } z^S \text{ in the long division of}$$

$$z^{-(nc+na)} \frac{[\sum_{k=-nc}^{nc} b_k z^{-k}]^2}{A^2(z^{-1})} \tag{2.31}$$

Computation of some of the elements of the W_{nm} matrices can be avoided by noting that they are always equal to 0, see [14] for details.

It is shown in [16] that as $N \to \infty$ the covariance matrix of the normalized estimation error $\sqrt{N}(\hat{\theta} - \theta)$ is given by

$$P_m^\theta = W_{11} - W_{12} W_{22}^{-1} W_{12}^T \tag{2.32}$$

furthermore we have

$$P_{m_1}^\theta \geq P_{m_2}^\theta \qquad \text{for } m_2 \geq m_1 \tag{2.33}$$

From equation (2.33) it follows that $(P_m^\theta)_m$ has a limit when $m \to \infty$, which we will denote by P_∞^θ. The following equality holds

$$P_\infty^\theta = W_{11} - \Omega\Omega^T = P_{CR}^\theta \tag{2.34}$$

where

$$\Omega_{ij} = \mathrm{E}\{C^2(q^{-1})[e(t-i)+e(t+i)] \frac{1}{A^2(q^{-1})} e(t-nc-na-j)\} \tag{2.35}$$

$$i = 0,\ldots,n\theta-1, j = 1,\ldots,2nc$$

$$\Omega = (\Omega_{ij})$$

However, the asymptotic results depend on the fact that m is a negligable fraction of N, so for finite data lengths we must be careful not to choose m too large. This makes the convergence rate of $(P_m^\theta)_m$ to P_∞^θ important: it can be shown [16] that if $C(z)$ does not have roots with approximately unit magnitude, the convergence will be fast; otherwise it will be slow, and we will need to use large values of m in order to approach P_∞^θ, which is not possible unless we have a very long data record available (since $m/N \approx 0$ must hold).

We can find estimates of a_i by substituting our estimate $\hat{\theta}$ for the autocorrelations in the Yule Walker equations (2.11). The resulting estimate will have an asymptotic (for $N \to \infty$) covariance matrix which can be shown to be equal to

$$P_m^a = R^{-1}(Q_{11} - Q_{12}W_{22}^{-1}Q_{12}^T)R^{-T} \tag{2.36}$$

where

$$[Q_{11}]_{ij} = \mathrm{E}\{A(q^{-1})v(t-i)A(q^{-1})v(t-j)\} \tag{2.37}$$

$$[Q_{12}]_{ij} = \mathrm{E}\{A^2(q^{-1})v(t-i)A(q^{-1})v(t-na-j)\} \tag{2.38}$$

$$\tag{2.39}$$

and R is the matrix in the YW equation (2.11). Again we have [16]

$$P_{m_1}^a \geq P_{m_2}^a \qquad \text{, for } m_2 \geq m_1 \tag{2.40}$$

Therefore, $(P_m^a)_m$ must have a limit P_∞^a, which can be shown to be equal to [16]

$$P_\infty^a = R^{-1}(Q_{11} - \Gamma\Gamma^T)R^{-1} = p_{CR}^a \tag{2.41}$$

where

$$\Gamma_{ij} = \mathrm{E}\{C^2(q^{-1})e(t-i)\frac{1}{A(q^{-1})}e(t-na-j)\} \qquad i = 1,\ldots,na$$

$$j = 1,\ldots,2nc$$

$$\Gamma = (\Gamma_{ij}) \tag{2.42}$$

It has been shown [16] that the covariance matrices P_∞^θ (equation (2.34)) and P_∞^a (equation (2.41)) are equal to the CRLB for θ respectively a, that is, the algorithm is asymptotically statistically efficient. Notice however, that the proofs [16] assume that the limits $m \to \infty$ and $N \to \infty$ are taken in such a way that $m/N \to 0$, so when implementing the algorithm, m must be a negligible fraction of N.

The correction step could of course be repeated: we could use the new estimates to get improved values for z and W_{ij} and use them in (2.25) (note that we would still use the original autocorrelations in the right hand side since the correction step is designed to correct the original estimates, not the improved ones). Since the algorithm is asymptotically efficient, this will not improve the asymptotic properties, but, as [16] suggests, it could improve the short data length behaviour. This is discussed in section 3.3

As is shown in [14], the algorithm is well conditioned in the limits $m, N \to \infty$. The paper [14] gives bounds for the numerical condition number of the algorithm (as a function of the parameters a and c) in the limiting case.

Table 2.1: Summary of the Algorithm

1. (a) Use (2.14) to find $\tilde{r}$

   (b) Use (2.11) with $\tilde{r}$ in place of r to find $\tilde{a}$(in a least squares sense if we choose $K > na + nc$)

   (c) Use (2.12) with $\tilde{a}$ and $\tilde{r}$ in place of a to find $\tilde{b}$

2. Use $\tilde{r}$, $\tilde{a}$, $\tilde{b}$ to find $\hat{r}$ using

   (a) (2.26) to find $\tilde{z}$

   (b) (2.27)–(2.31) to find $\hat{W}_{12}$ and $\hat{W}_{22}$

   (c) (2.25) to find $\hat{\theta}$ from $\hat{z}$, $\hat{W}_{ij}$ and $\hat{z}$

3. (a) Use (2.11) with $\hat{r}$ in place of r to find $\hat{a}$ (again possibly in a least squares sense)

   (b) Use (2.12) with $\hat{a}$ and $\hat{r}$ in place of a and r to find $\hat{b}$

   (c) Use $\hat{a}$ and $\hat{b}$ in (2.6) to obtain the spectral estimate $\hat{\Phi}(e^{j\omega})$

2.5 Summary of the Algorithm

We can now summarize the algorithm see Table 2.5 (for a good overview of the computational issues see [14]).

As an overview of the properties of the algorithm, we know that it is statistically efficient when both $N \to \infty$ and $m \to \infty$, and we know that the asymptotic covariance matrix of both a and θ is monotonically non-increasing as a function m, but may converge very slowly if $C(z)$ has roots near the unit circle. Furthermore,

for $N \rightarrow \infty$ the algorithm is numerically well-conditioned.

For finite data lengths we know that we need $m \ll N$, which (in cases of slow convergence) may prevent us from getting close to statistical efficiency. Moreover, the algorithm need no longer be well-conditioned. Finally, since we no longer have efficiency, the estimates of θ, a_i and b_j may have different statistical properties in the finite N case.

3. A Simulation Study of the Finite-Data Length Behaviour of the Algorithm

This chapter reports on the behaviour of the algorithm for finite data-lengths, and discusses the various aspects and problems that are seen.

It is known [16] that the algorithm is both statistically efficient and numerically well-conditioned in the limiting case $N, m \rightarrow \infty; m/N \rightarrow 0$. For finite m and $N \rightarrow \infty$ we know [16] the behaviour of the parameter variances as a function of m, and for processes that do not have zeros close to the unit circle we know [16] that even for "low" values of m these variances are close to the CRLB. For finite m we do not have any results on the numerical condition of the algorithm. For both N and m finite, we know neither the numerical condition nor the statistical efficiency of the algorithm. In this chapter we investigate the behaviour of parameter variances for the N, m finite case through Monte Carlo simulations.

3.1 An Overview of Finite-Data Length Effects to be Investigated

For this study the observed behaviour of the parameter variances will depend not only on the particular ARMA process, but also on two parameters which must be chosen: the number of postiterations, and the number of instrumental variables (i.e. the value of m).

It is stated in the previous chapter that asymptotically, one postiteration is enough to achieve the minimum possible variances. However, in [16] it is suggested that for short data lengths multiple postiterations might help to improve the statistical quality of the estimate.

In addition we know that as $N \rightarrow \infty$, increasing the number of instrumental variables monotonically improves the quality of the estimate. However, this proof depends on the fact that the number of instruments is an infinitesimal fraction of the data length; therefore, for finite data lengths we can expect deviations from the theoritically predicted behaviour if we choose the number of instruments to be too large.

In order to get a good insight in the finite data length behaviour of the algorithm, we need to use several data lengths, and consider several processes, of different order, and try to find out how the variances of the estimated parameters change as a function of the chosen data length for each process.

A specific problem we may encounter is autocorrelation estimates that are not nonnegative definite (NND), or, equivalently, spectral estimates that are not non-negative (NN) for all frequencies. The true autocorrelation sequences are always NND, and therefore the true spectra always NN, but they can come arbitrarialy close to not being so, in which case the uncertainty in the estimates may cause the estimated autocorrelations and spectra not to be NND and NN, respectively. As an example: the sequence $(\varepsilon, 0, 0, \ldots)$ is NND for any $\varepsilon > 0$, but is not NND for any $\varepsilon < 0$.

In section 4.2 a strategy for repairing this problem is studied. It relies on the fact that an autocorrelation sequence that is "close" to not being NND will have zeros (roots of the C polynomial) close to the unit circle; therefore, the problem can be alleviated by moving all zero estimates which are too close to the unit circle farther away from it.

3.2 The Initial Monte Carlo Simulations

This section details the design and results of the first simulations.

3.2.1 Design of the Simulations

The following processes have been used in the simulations (see Table 3.1). Table 3.1 shows that the processes called Example 1 and Example 2 have poles close to the unit circle, but no zeros close to the unit circle, while for Examples 3 and 4 the reverse situation occurs. Examples 1 and 3 have been taken from [6] and [8].

We expect estimates from the data sets for Examples 1 and 2 to behave like the asymptotic case ($N \rightarrow \infty$) for relatively small data lengths, and estimates for Examples 3 and 4 to need large datas lengths for asymptotic behaviour because the analysis of the algorithm predicts that zeros close to unit magnitude will seriously affect convergence in m of the variances to their asymptotic values ([16]), thus necessitating large N, as $m/N \approx 0$ must hold. We also expect that Examples 3 and 4 will need larger data lengths even for fixed m, as the numerical condition of the algorithm also is affected by zeros close to the unit circle.

For the other two processes (Examples 3 and 4) we expect the algorithm to have a slow convergence, a higher asymptotic variance and less numerical stability. these examples also have varying orders (two are ARMA(1,1) processes, one is an ARMA(4,3), and one an ARMA(4,4) process). Several other processes, especially ARMA(1,1) cases, were looked at, but are not reported on here, as the results from conclusions from those simulations support the conclusions presented here.

Simulations were carried out using $N = 75, 200, 750$, and 2000 data points.

Table 3.1: The ARMA Processes Used in the Simulations

	Example			
	1	2	3	4
na	4	1	4	1
nc	4	1	3	1
λ	2.8271	1	0.017258	1
a_0	1	1	1	1
a_1	0.1	−0.9	−1.3136	−0.5
a_2	1.66		1.4401	
a_3	0.093		−1.0919	
a_4	0.8649		0.83527	
c_0	1	1	1	1
c_1	0.0226	−0.5	0.1792	−0.9
c_2	0.8175		0.8202	
c_3	0.00649		0.2676	
c_4	0.07637			
poles	$-0.25 \pm i0.9314$	0.9	$-.1253 \pm i0.9159$	0.5
	$0.2 \pm i0.9434$		$0.7821 \pm i0.6047$	
zeros	$-0.00079 \pm i0.8425$	0.5	$0.0658 \pm i0.9256$	0.9
	$-0.0034 \pm i0.328$		-0.3108	

Initially one postiteration was done to determine a good range for the number of instruments. This number was between 1 . . . 10. The effect of multiple postiterations for these cases was then investigated.

The results of the simulations are characterized by two kinds of plots: plots of the estimated spectrum and plots of an a (coefficient of the estimated denominator polynomial), b (coefficient of the estimated numerator polynomial), or r (estimated autocorrelation coefficient) parameter versus either the number of instruments or the number of postiterations.

The spectral plots shown each have four curves:

- The mean spectrum: the mean of all estimates in a given Monte-Carlo run.
- The mean plus the normalized standard deviation and mean minus normalized standard deviation of the spectral estimates. The standard deviation of the spectral estimates is computed, multiplied by the square root of the data length and the result added to, respectively subtracted from the mean of the spectral estimates.
- The true spectrum.

The plots of an a, b, or r parameter versus nz each have at least two curves: the normalized variance and the sum-squared error (any difference between these two curves indicates bias in the estimate), and, where appropriate, the theoretical variance and the CRLB, also normalized with the the data length.

3.2.2 Results for the Four Choosen Examples at Several Datalengths and Different Algorithm Settings

Below we present results from simulations of the four example processes. The results naturally cluster to two groups: the low order cases (Example 2 and Example 4) and the high order cases (Examples 1 and 3).

Low Order Cases (Examples 2 and 4)

The resulting spectral estimates for example 2 confirm this behaviour. For increasing nz, example 2 improves significantly over the pure YW estimate $nz = 0$; as can be seen from Fig. C.1, where we see the envelope formed by the two curves "mean minus standard deviation" and "mean plus standard deviation" become narrower as nz increases, the most noticable narrowing occuring as nz increases from 0 to 1 and then from 1 to 2. Example 4 also improves, but has a much higher variance (the envelope remains wide), as can be seen from Fig. C.3. Problems also occur with the spectral estimate being non positive definite, even for higher values of nz. This is seen in the ragged shape of the envelopes (Fig. C.3). As can be seen from Fig. C.4, the variances are better at 750 data points than at 200, but the improvement is small. On the other hand, the spectrum of example 2 is still being estimated quite well at 75 points, as can be seen from Fig. C.2. When looking at these Figures, it is important to remember that the variances are normalized by a factor N, so for instance "small improvement" between 200 and 750 datapoints for Example 4 means that the variances are decreasing as $\frac{1}{N}$.

Comparing $N = 200$ curves with the same nz for example 2 and example 4 (Figs. C.1 and C.3), we see that the latter curves have a far wider envelope. We

also note the ragged shape of the envelope for example 4: this typically indicates that some of the Monte Carlo iterations resulted in a spectral estimate that is not positive for all frequencies; plots of selected individual estimates (not reproduced here) confirm this.

Overall, we see that:

- For a given N and nz, example 2 has lower variances than example 4.
- For a given N the variance decreases as nz increases, but the effect is much more pronounced for example 2 than for example 4. The decrease of variance is greatest for low values of nz.
- For a given nz the variance decreases as N increases, rapidly for example 2, but slowly for example 4.

The statement in the first paragraph of this subsection that this behaviour can be explained from the zero positions, as predicted by the asymptotic analysis, is supported by simulation results for several other first order processes, with different pole and zero positions (results not reproduced here), which clearly show that the position of the pole does not influence convergence, but the closer the zero is to the unit circle, the slower the convergence becomes.

The ARMA parameter estimates do not achieve the same quality as the spectral estimate. Not only does example 4 have a very slow convergence (a problem that apparently affects the ARMA parameters even more than the spectral estimate), but example 2 now also has a convergence problem; the variances go down at first, and then, as nz is increased further, they start increasing. This is contrary to what the spectra show. See Figures C.5–C.10. Looking at the plots we see that curves for

the a and b parameters are similar in shape. For fixed N, the variance goes down rapidly as nz increases from 0,1 to 2. We also note that the algorithm computes estimates that have very little bias since the variance and sum squared error curves are indistinguishable (there are differences, starting in the 5th significant digit). A clear almost flat stretch can be seen in all curves in the $nz = 3,\ldots,8$ range, indicating that the estimator performance is insensitive to the choice of nz in this range. Finally, the rise in the curves for high nz values is mainly due to a few outliers, not a general deterioration of the estimates. This can be seen from a listing the 20 highest error norms. For each Monte Carlo iteration, the program computes

$$\sum_i \frac{\hat{a}_i - a_i}{a_i} + \sum_i \frac{\hat{b}_i - b_i}{b_i}$$

and at the end of a run, the 20 highest values are returned. For a first order process, typically 18 or 19 of the 20 error norms have a value of ≈ 0.5, and the remaining 1 or 2 having a value of ≈ 17.

The curves in Figures C.5–C.10 show the following:

- Just as for the spectral estimates, for a given N, the parameter variances decrease as nz increases, with the largest decreases occuring for nz increasing from 0 to 1 and from 1 to 2. This effect is stronger in example 2 than in example 4.

- Just as for the spectra, for a given nz, the parameter variance decreases as N increases, more so for example 2 than for example 4. This can be observed by comparing curves for the same parameter but different data length and reading off variances for the same nz. See Fig. C.5.

- Bias in the parameter estimates is negligible.

High Order Cases (Examples 1 and 3)

The results for example 1 are similar to those for the low order cases. (Figure C.12) If we compare the spectra for $nz = 0,1,2$ we see a rapid improvement, which becomes a slow improvement for $nz > 3$. However, unlike example 2 and example 4, we see here that there is improvement for $nz > 3$; it is much slower than for the lower values of nz, but comparing $nz = 7$ with $nz = 3$ for Example 1 shows a clear improvement C.12 For $nz = 9,10$ we see the the envelope widen again. Both examples show the ragged envelopes characteristic of non positive spectra, but increasing nz clearly decreases the effect, as can be seen by comparing individual spectral estimates.

Comparing different data lengths (Figures C.11–C.17, giving spectral plots for $N = 75,200,750,2000$ for Example 1 and $N = 200,750,2000$ for Example 3; for both examples results for $nz = 0,1,2,3,7,10$ are given. Example 3 does not have plots for $N = 75$ because they are so bad that they no longer show any information) we see slow convergence of the variance as a function of the datalength, and variances that are clearly worse than example 4 for $N = 75$: we obviously need longer data records for these processes.

As can be seen from Fig. C.15, increasing nz too much will cause the variances to increase: the Figure compares nz=7 and 10 for 2000 data points, and the envelope for nz=10 is clearly wider than the envelope for nz=7.

Considering example 3, we see that even for high data lengths there is little improvement in the variances as we let nz increase (compare for example $nz = 0$

or 3 for 2000 data points Fig. C.17). Again this shows that whereas poles close to the unit circle have little effect (example 1 has poles with magnitude greater than 0.95), zeros that close to the unit circle cause the decrease of the variances as a function of nz to be very slow. (example 3 has zeros with magnitude greater than 0.95).

The ARMA parameter estimates (Figures C.18 and C.19) again show the pattern of rapidly decreasing variance for low nz, followed by an almost flat stretch, followed by a very rapid rise: this rise is much steeper for examples 1 and 3 than for examples 2 and 4. The plotted parameters are typical cases; Figure C.19 also shows the theoritical variance and the CRLB.

3.3 Results of Simulations with the Algorithm Modified to Do Multiple Postiterations

One proposed method for improving statistical quality for small N is to use multiple postiterations. However, it turns out that this method does not work: more than one postiteration increases the variances. Our simulations gave floating point overflow in as few as five postiterations.

Analysis of the individual estimates of several Monte Carlo runs showed that the problem lies in the high sensitivity of the estimated a and b parameters to outliers in the estimated autocorrelations. As was stated above, the correction step of the algorithm introduces outliers (typically $\frac{1}{2}$-5% of the Monte Carlo iterations), and when these are fed back into the algorithm for the next postiteration, the resulting estimate poorer than the original estimate, thus causing rapid divergence as a function of the number of postiterations.

3.4 Summary

Based on simulations, it was shown that the algorithm shows the predicted slow convergence for processes whose C polynomials have zeros close to the unit circle. The requirement that $m \ll N$ was also confirmed. The simulation results indicate that even $m = 0.05N$ may to be "too large".

Furthermore, we have found that the spectrum is less sensitive to the effects of high values of nz than the ARMA parameters. On the other hand, for higher process orders, the spectral estimates continue to improve for higher values of nz than the ARMA estimates do; that is, while the ARMA estimates for $nz = 4$ and 5, say, are almost identical, the spectral estimate for $nz = 5$ is slightly better than that for $nz = 4$.

For all of the processes considered, most of the improvement of the algorithm over the normal Yule Walker method is gained through the addition of the first 2 instruments. After that, improvements with additional instruments are either much smaller, or even negligible.

Also, for all processes, the variances of the estimated parameters increase with larger values of nz; for example 2 and example 4 this increase starts gradually, but for examples 1 and 3 it is very sudden.

One method for preventing the increase in variance for high nz is to choose a low value for nz. For low order processes this seems no problem, as choices of 2 or 3 instruments seems to be all that is needed to achieve the minimum variance the algorithm can realize. For high order proceses on the other hand, and especially if one is mainly interested in obtaining accurate spectral estimates, a larger number of instruments may be needed, leading to potential problems with the variance because

a choice that is too high can result in a variance that is far higher than even than the initial Yule Walker estimate.

A second improvement method might be to use extra postiterations for lower datalengths. However, as we have seen above, this approach has problems with variances increasing instead of decreasing that are even worse than the problems of the original algorithm.

After that one might consider various "repair" strategies, perhaps based on outlier elimination, such a method would modify "bad" estimates in such a way as to numerically stabilize the algorithm before the correction step is executed.

However, it was found that some Monte Carlo iterations are much more robust than others; in other words, if we could change the algorithm such that it would decide on the number of instruments and postiterations for each individual Monte Carlo iteration, performance might improve. Such a method can also potentially eliminate the problems associated with using a large number of instruments. Another possibility is to move estimated poles and/or zeros that are close to the unit circle slightly away from it, so as to reduce the numerical problems associated with poles and zeros close to the unit circle. Finally, the strategies that have been used to improve Yule-Walker methods, such as using overdetermined Yule Walker equations (i.e. use more than the required minimum of Yule Walker equations and solve them in a least squares sense) can also be used. While using overdetermined Yule Walker equations does not guarantee improvement, extensive simulations [4], [5] have shown that improvement is generally obtained when the sequence of autocorrelations decreases slowly. The next chapter deals with these strategies.

4. Investigation of Algorithm Modifications to Improve the Small Data Length Behaviour

In this chapter we consider various modifications to the algorithm to improve its small data length performance. First we consider the automatic selection of the number of instruments and postiterations. This is motivated by the observation that most of the decrease in performance for high nz and/or high number of postiterations is due to outliers, therefore selecting an appropriate number of instruments and postiterations for each Monte Carlo iteration separately should improve performance. Since it is possible to make the algorithm recursive in nz in a highly computationally efficient manner, automatic selection can be achieved at almost no additional computational cost. Second we consider moving poles and/or zeros which are close to the unit circle, to avoid the numerical ill-conditioning problems. Finally, we study the effect of using overdetermined Yule Walker equations instead of minimal ones, since this improves the performance of the Yule Walker estimator in many cases ([4], [5]).

4.1 Automatic Selection of the Number of Instruments and Postiterations

This section first discusses how to make the estimator recursive on nz so that the automatic selection strategy can be implemented efficiently; then it discusses how to find the criteria for selecting values of nz and the number of postiterations i in a self adjusting strategy. Finally it compares the performance of this strategy with that of the fixed value strategy.

Making the algorithm self-adjusting in the nz and i parameters is desirable for the following reasons: The best value of nz is dependent on the process being estimated. There are two ways to set it manually. First, one could use a priori knowledge of the process (which is often unrealistic). Second, one could tune the algorithm by using the estimator on multiple data records of the process for several values of nz (which is often impractical). A choice of nz could then be based on the expirimental variance data gathered in this way. A self-search would avoid both these problems. There is evidence that the optimal value of nz varies considerably even for different realizations of the same process (by as much as a 2–7 range), making separate tuning for each realization attractive. In many cases it would be desirable, or even necessary, to have the algorithm do this. Due to the numerical instabilities described in Section 3.3 we will not realize any advantage from postiterations beyond the first one, unless we use only those cases with low error norms in multiple postiterations.

4.1.1 A Recursive in nz Implementation

The algorithm obtains an initial estimate (steps 1.1 through 1.3 in Table 2.5) and then corrects it by first correcting the $\tilde{r}_i$ (steps 2.1 through 2.3) and then revising the original ARMA parameter estimates using the corrected $\hat{r}_i$ (steps 3.1 and 3.2). In step 2.3 we need to find $\tilde{W}_{22}^{-1}z$, which we compute by solving $\tilde{W}_{22}\tilde{w}z = z$ for wz. It is in the second stage (steps 2.1 through 2.3) that the instrumental variables are used. It turns out that it is possible to carry out steps 2.1 and 2.2 in a recursive in nz manner, computing only z_{nz} at step 2.1 in each stage (the formulae for z_i are independent of nz) and moving step 2.2 out of the loop (α

is independent of nz). Finally step 2.3 is computed using a version of the algorithm described in [7] rewritten to make it recursive in the size of the matrix.

The equations for steps 2.1 and 2.2 are unchanged, the correction equation $\hat{\theta} = \bar{\theta} - W_{12}W_{22}^{-1}z$ is unchanged, but it's calculation is now different. It is implemented by first computing wz in $W_{22}wz = z$ and then computing $\hat{\theta}$ from $\hat{\theta} = \bar{\theta} - W_{12}wz$, and we can calculate the wz in a recursive way. The resulting modified algorithm is detailed in Table 4.1

4.1.2 Criteria for Choosing nz and i

We need some criteria to use for choosing nz and i. Any implementation will have finite resources (in terms of time and/or memory) available for the estimator. Subject to these constraints, we want to minimize variance of the parameter or spectral estimates, or, in terms of a single realization, minimize an error norm. However, the error norm is not computable by the estimator, so we need some function of the estimate that provides a reliable indication of the error.

Finding the minimum possible error would involve checking the value of the error criterion for all allowed nz and i combinations; this is not practical because it would require a large amount of computations to implement. A more computationally feasible (but suboptimal) solution is first to step nz for $i = 1$, until some desired value of nz is obtained then continue to use this value of nz while increasing i as far as possible without violating an error criterion.

It was experimentally found that nearly every internal quantity became much larger if the number of postiterations was made too large. As a result, detection of

Table 4.1: The Recursive in nz Version of the Algorithm

1. (a) Use (2.14) to find $\tilde{r}$

   (b) Use (2.11) with $\tilde{r}$ in place of r to find $\tilde{a}$ (in a least squares sense if we choose $K > na + nc$)

   (c) Use (2.12) with $\tilde{a}$ and $\tilde{r}$ in place of a to find $\tilde{b}$

2. Use $\tilde{r}$, $\tilde{a}$, $\tilde{b}$ to find $\hat{r}$ using

   (a) Find α using (2.31)

   (b) Step nz from 1 to the desired value and for each step compute

      i. (2.26) to find z_{nz}

      ii. (2.27)–(2.31) to find W_{ij}

   (c) For the final value of nz use (2.25) to find $\hat{\theta}$ from z and W_{ij}

3. (a) Use (2.11) with $\hat{r}$ in place of r to find $\hat{a}$ (again possibly in a least squares sense)

   (b) Use (2.12) with $\hat{a}$ and $\hat{r}$ in place of a to find $\hat{b}$

too large a value of i is not at all difficult. We have used:

$$\frac{|z_i|_\infty}{|z_{i-1}|_\infty} < 1 + \eta \text{ and } i \leq \max i$$

because z is computed early in the postiteration loop (step 2.1 in Table 2.5), so if the error criterion is not satisfied, then the effort invested in the unsatisfactory postiteration is still quite small.

This heuristic turns out to be quite robust. Regardless of how nz is determined, or how the random number generator used to generate the data sequence is initialized. For our simulations

$$\eta \simeq 0.004$$

$$\max i = 10$$

was a choice which prevented more than about 4% of the Monte Carlo iterations from going up to $\max i$, again quite independently of noise seed value and nz heuristic.

A good nz heuristic is more difficult to find because the effect of a wrong choice of nz on the internal variables is much more complex than in the case of a wrong choice of i. High error norms are associated with the α and β vectors(see step 2b of Table 2.5, but the relation is so complex that it is not suitable for use as a criterion. The heuristic used is based on the norm of the correction vector instead. The reason for this choice is that a large correction means that $\tilde{r}$ was an inaccurate estimate, and given the sensitivity problems in the correction step, this probably means the correction will be inaccurate. So the heuristic used in step 2.3 in table 2.5 is:

$$\left(\frac{|W_{12}||W_{22}^{-1}||z|}{|\tilde{r}|}\right)_{nz} < \epsilon \text{ and } nz \leq \max nz$$

experimentally we found that:

$$\epsilon \simeq 0.15$$

and

$$\max nz = 7$$

were good choices. However, robustness of this criterion is poor. Although the estimated variances are fairly insensitive to the choice of $\max nz$, changes in ϵ by 5% result in markedly different performance. Moreover, as shown below, even if ϵ is chosen to minimize the variance of the estimated parameters, the resulting variance is *higher* than the variance realized by fixing nz to a "good" value.

4.1.3 Differences Between the Fixed and the Recursive Approaches

How much improvement does the self-adjustment provide? Of course, in answering this question, one must decide what is meant by "improvement": for example, is improvement sought in the r_i estimates, the spectral estimate or the estimates of the ARMA parameters (a_i and b_i)?

The estimates of the a's and b's do *not* improve with a self-searching algorithm; while the sharp rises in variance for high nz and/or i values can be avoided the resulting optimal variance is still considerably (about a factor 1.5 to 4) larger than optimal hand tuned fixed values of nz and i. This is true for all examples and datalengths; the factor 1.5–4 cited above is for 200 data points. This factor decreases as the data length increases, and extensive simulations confirm this.

The r estimates *do* improve with a self searching strategy: the differences are in the 2nd and 3rd digits, but they are on the order of the differences between different fixed values, so this is a large relative improvement. Note that the differences

Figure 4.1: A Histogram of nz and i Combinations chosen by the Recursive Algorithm for Example 1, with 200 data points.

ndata=200, dseed=123456789

nz\i	0	1	2	3	4	5	6	7	8	9	10
0	0	0	7	0	0	0	0	0	0	0	0
1	0	0	11	1	2	1	0	0	2	1	2
2	0	1	6	1	0	2	1	1	0	0	2
3	0	0	7	5	1	3	1	1	1	0	1
4	0	1	4	1	2	0	1	0	0	0	0
5	0	1	8	4	5	1	1	0	0	0	0
6	0	0	2	2	3	1	0	1	0	0	0
7	0	2	3	3	2	1	0	1	0	0	1
8	0	1	3	1	1	1	0	1	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0

between r_i from different the Monte Carlo simulations typically occur in the 2nd to 1st digit.

4.1.4 Conclusions

The self-adjusting strategy does improve the rapid rise in variance for high nz that was typical for the higher order examples to the more gradual rise observed in the lower order examples. The slow improvements in the spectral estimates for example 1 and 3 indicate that using higher nz values may be desirable. However, the self-adjusting strategy fails to achieve the same low variances achieved by the original algorithm for lower values of nz. Similarly, the self-adjusting strategy enables us to use more than one postiteration, but the resulting variance is higher than that of the original algorithm with one postiteration. Contrary to the ARMA parameter estimates and the spectral estimates, the autocorrelation estimates do improve with this strategy.

An efficient recursive in nz implementation of the algorithm developed as a part of this study is interesting in its own right, and useful in applications where multiple estimates for different nz are desired.

4.2 Pole and Zero Adjustments

Another way of limiting the problem of high variances for high values of nz is suggested by the structure of the algorithm; as shown in [16], convergence in nz is slow if the process has zeros close to the unit circle. Furthermore, calculation of the correction step involves finding the first few elements of the impulse response of the process with transfer function $\frac{B(z^{-1})}{A^2(z^{-1})}$ (see step 2.2 in Table 2.5); one would expect

numerical problems here if there are poles close to the unit circle.

To alleviate these problems one can consider changing the algorithm to assure that all poles and/or zeros are sufficiently far from the unit circle. In principle there are two ways of doing this: one can move only the offending roots, or one can scale the entire polynomial (which moves all the roots). The first strategy has the advantage that it introduces the minimal change in the polynomial that achieves the objective. The second strategy has the advantage that it does not introduce distortion in the pole/zero diagram, but only scaling of the entire pole/zero diagram. Also, the second strategy is much more computationally efficient. As a second effect, either strategy can also insure that the numerator polynomial is non-negative definite.

Since these methods tend to produce $\frac{1}{2}$–2% of very bad outliers, we have gathered statistics both with and without the outliers taken into account. Unless stated otherwise, the statistics presented below are those *without* the outliers taken into account.

4.2.1 Individual Pole and Zero Movement

As discussed above, we can move either only the poles or only the zeros, or both. These two cases are discussed below.

Movement of Both Poles and Zeros

The revised algorithm works as follows: after computation of $\tilde{a}$ and $\tilde{b}$, we calculate new variables $\bar{a}$ and $\bar{b}$, the "moved" versions of $\tilde{a}$ and $\tilde{b}$, respectively. The algorithm then proceeds using $\bar{a}$ and $\bar{b}$ in steps 2.1 through 2.3 of table 2.5. If there

are multiple postiterations, the same is done with $\hat{a}$ and $\hat{b}$.

The $\bar{a}$ parameters are computed by solving for the roots of $\tilde{A}$, checking how close each root is to the unit circle, and moving any offending roots radially outward to a circle with radius $1 + \delta$ for some chosen δ. The alteration of the $\bar{b}$ coefficients is slightly more involved, as the B polynomial contains both the roots of $C(z)$ and $C(z^{-1})$ (see equation (2.4)): the former have to be moved out, the latter in. Also, any roots with norm equal to 1 plus or minus machine precision have to be paired and then moved, one out, one in. Pairing of a root z_i involves finding the closest root z_j, redefining both roots to be the mean $\frac{z_i+z_j}{2}$, and then moving the one radially out, the other radially in.

Comparison of the revised algorithm with the original shows that it does not have the sharply rising variances for high nz; in fact, for $\delta \approx 0.001$, the curve is almost flat. This elimination of the problem with the rising variances is quite insensitive to the exact choice of δ: one needs a variation on the order of $\delta = 0.0003$–0.003 to make an appreciable difference. However, we observe reduced performance for the lower values of nz, varying from almost no change for the r's to between 10% and a factor of 2 for the a's and b's. This is distributed over the parameters in an irregular way, suggesting that the problem may be more pole-oriented than parameter oriented.

Pole-Only Movement

Comparison shows that, as expected, most of the effects (both the beneficial and the detrimental) come the pole movements, so we have tried moving only the poles. The results are almost as good as the combined pole/zero movement

case, and, of course, the computational load is reduced considerably. This implies that non-negative numerator estimates do not make a major contribution to the variance. The above statements are backed by extensive simulations performed on all four examples.

4.2.2 Scaling Movement of All Poles and Zeros

In this section we consider another modification in which all poles are moved simulataneously. Since it has been found that the location of the estimated zeros hardly influence the results, they are not moved. The algorithm is modified in the same way as before, except that $\bar{a}$ and $\bar{b}$ ($\bar{b} \neq \tilde{b}$, see below) are now computed in a different way. The parameters $\bar{a}$ are now found by checking the roots of $\tilde{A}$, and, if any are too close to the unit circle, scaling the entire polynomial, in effect moving all roots radially out by the same factor. B is then scaled by the same factor; this was thought to have the effect of keeping the residues of the poles approximately constant, i.e., if we were to do a partial fraction expansion on $\bar{B}/(\bar{A}(z^{-1})\bar{A}(z))$ we would obtain approximately the same constants in the numerators as in the partial fraction expansion of $\tilde{B}/(\tilde{A}(z^{-1})\tilde{A}(z))$. Since the method did not have solve the problems of the original algorithm, a formal proof of the above claim about the residues was abandoned. Obviously a much more computationally efficient method would be to use a stability test on the polynomial and then scale by the maximum factor if it turns out not to be stable enough, but this method may move roots by a greater extent than necessary. We want to try the maximum possible performance method first, to see how much improvement this approach can maximally give.

It has been found that this method results in lower variances than the previous

two. See the a_1 versus nz plot (Figure C.21) for a typical result; for comparison, the curves from the standard algorithm have also been plotted in the Figure. As this Figure shows, the variances of the revised algorithm are still above those for the original algorithm for nz greater than 1; this is typical for all parameters in all examples. As can be seen from the spectral plot Fig. C.20, the spectral plots become more smooth than those in Fig. C.12. Further improvement can be gained by using the tilde polynomials, i.e. the *uncorrected* $\tilde{a}$ and $\tilde{b}$ for the computation of the z vector. This makes the method even more insensitive to changes in the δ radius, which is an advantage, as it reduces the need for tuning.

On the basis of the experience that the matrix inversion is not ill-conditioned for any of our examples, we could modify only the W_{12} matrix, i.e. the α sequence. This again improves performance. These changes are based on an analysis of the effects of the scaling to be detailed below. The principle used is to reduce the effects of the pole movements on the final estimate.

Since the correction in the B polynomial to keep the residues of the poles constant is only an approximation, the reduction of the effects of the pole movements cannot be total, but the correction improves performance to a level very close to the original algorithm for low nz values, while only deteriorating gradually for higher values. The main problem with this method is that the pole correction step produces outliers: about 0.5–2% of the Monte Carlo iterations have an error that is far higher than the rest. The above results are based on statistics from which the outliers have been removed; with the outliers, the statistics are considerably worse. so much worse, in fact, that they are worse than the original algorithm. The outliers are detected by keeping track of the 20 iterations with the worst error norms and

reporting those, then a limit is set manually, after which the program accumulates both normal and no-outlier statistics (i.e. everything worse than the manual limit is ignored). The error norm is

$$\sum_{i=0}^{na} \frac{\hat{a}_i - a_i}{a_i} + \sum_{i=0}^{nc} \frac{\hat{b}_i - a_i}{b_i}$$

There is a considerable gap (almost a factor 2) between "normal" and "outlier" values of this norm; the deterioration for high values of nz is accompanied (and partly caused by) an increase of the number of outliers.

Our attempts to remove outliers on-line have failed, so there is currently no way of exploiting the fact that the increase of the variances is mainly due to a few outliers. On the other hand, the no-outlier statistics represent a behaviour that is typical for for 98% or more of the iterations in any Monte Carlo simulation. However, it does mean that overal, the original algorithm works better the modified one.

Removal of outliers from the original method has also been studied, but it has far less effect than in the case of the modified algorithms, so much less in fact, that comparisons between the original algorithm with or without its outliers show almost no change.

4.3 Using High Order Yule-Walker Equations

As has been established through extensive simulations ([4], [5]), using high order (also called overdetermined) Yule Walker equations usually leads to reduced variances in the estimates, except when the autocorrelation sequence decreases very rapidly. Therefore we should try what happens if we use High Order Yule-Walker

equations (HOYW) in the algorithm under study. Specifically, in Table 2.5, we choose $K > na + nc$ in steps 1b and 3a.

Figure C.22 shows the behaviour of the a_1 parameter of Example 1, as can be seen, the variances are down dramatically from the ordinary Yule-Walker method, the sharp rise in variances around $nz = 10$ is replaced by a more gentle one, but the improvements are largely gone as well: the curve is almost flat now. Unfortunately, there did not remain enough time to investigate these phenomena in detail.

5. Conclusions

We have seen that the algorithm proposed in [16] achieves high quality estimates for large data lengths. For short datalengths there is a problem associated with the number of instrumental variables: if this number is chosen larger than approximately 5% of the datalength, the variances increase very rapidly as a function of the number of instrumental variables. This effect becomes more pronounced for processes as the process order increases. Most of the statistical improvement of the algorithm over the Yule Walker estimator is gained by the addition of the first 2 instrumental variables; after that, additional instruments result in small (in the case of spectral estimates) or negligible (in the case of ARMA coefficient estimates) improvements. If the number of instrumental variables is in the range 3–0.05N the variances are relatively insensitive to the exact number of instruments. The suggested method [16] of improving the algorithm's performance for short datalengths by repeating the correction step multiple times does not work for the cases tried, as it results in rapidly (as a function of the number of postiterations) increasing variances of the estimates.

Using a self adjusting version of the algorithm, where each realization searches for best number of instrumental variables and postiterations, avoids the problem of finding proper values for the number of instruments and postiterations, changes the very rapid rises in variance for high nz to a more gradual one, but increases the variance of the ARMA parameter estimates for lower numbers of instruments as compared to the original algorithm.

Using a stabilized version of the estimated autoregressive parameters in some steps of the algorithm and approximately compensating the changes this causes in

the partial fraction expansion of the spectrum, also is able to remove the rapid variance increases for large numbers of instruments in about 98% of the cases, and has only slightly worse variances for lower values of nz, but suffers from 0.5% to 2% outliers.

Using overdetermined Yule Walker equations improves the performance of the estimator, for the cases tried, but this work is very incomplete.

In summary the original algorithm works very well for long data records, but great care has to be used when applying it to short records. In particular, it seems safest to not to use more than 2 instruments and only one postiteration. Processes with high order, or zeros close to the unit circle should not be estimated from short data lengths (shorter than least several hundred data points). In many applications this will mean that the algorithm is not suitable for estimating the parameters of such processes. One can try to use one of the variant algorithms developed in chapter 4. If possible, one should estimate multiple records and compute experimental variances: these clearly show for what values of nz, i and N the algorithm achieves good statistical performance.

A. The Correction Step: Mathematics

In this appendix we give a derivation of the algorithm; it is, of course, not a replacement for [16].

Let X be as defined in (2.18) and W as in equations (2.19)–(2.23), and assume that an estimate $\hat{W}$ of W, such that $|\hat{W} - W| = O(1/\sqrt{N})$ can be calculated from the available data. Since we are dealing with the asymptotic case, (2.18) is not too restrictive, since in many cases a central limit theorem will apply.

The asymptotic log-likelihood function of X is given by [9]

$$L(\theta) = -\frac{m}{2}\ln 2\pi - \frac{1}{2}\ln|W| - \frac{N}{2}(X-\bar{X})^T W^{-1}(X-\bar{X}) \qquad (A.1)$$

The ML estimate is therefore given by solving for a θ such that

$$\frac{\partial L(\theta)}{\partial \theta} = 0$$

Differentiating (A.1) and setting the derivative to zero gives

$$-\frac{1}{2}\frac{\partial}{\partial\theta}\ln|W| + N[I_{n\theta},0]W^{-1}(X-\bar{X}) - \frac{N}{2}\begin{bmatrix} (X-\bar{X})^T\frac{\partial W^{-1}}{\partial\theta_1}(X-\bar{X}) \\ \vdots \\ (X-\bar{X})^T\frac{\partial W^{-1}}{\partial\theta_{n\theta}}(X-\bar{X}) \end{bmatrix} = 0 \quad (A.2)$$

Let us assume that (A.2) has a solution, $\bar{\theta}$, with respect to θ. Under certain conditions the estimate will be consistent and $|\bar{\theta} - \theta| = O(1/\sqrt{N})$. Solving this equation is a computationally difficult problem; however we can make an $O(1/N)$ approximation without affecting the asymptotic behaviour of the estimator. For large enough

N we have from (2.18):

$$X - \begin{bmatrix} \bar{\theta} \\ 0 \end{bmatrix} = \left(X - \begin{bmatrix} \theta \\ 0 \end{bmatrix} \right) + \begin{bmatrix} \theta - \bar{\theta} \\ 0 \end{bmatrix} = O\left(\frac{1}{\sqrt{N}}\right) \qquad (A.3)$$

This allows us to rewrite (A.2) as

$$\frac{1}{N}\frac{\partial L(\theta)}{\partial \theta} = [\ I_{n\theta}\ \ 0\ ] W^{-1} \left(X - \begin{bmatrix} \bar{\theta} \\ 0 \end{bmatrix} \right) + O\left(\frac{1}{N}\right) = 0 \qquad (A.4)$$

Now we partition X and W as indicated in (2.19)–(2.23), and use a standard result on the inverse of partitioned matrices to get

$$W^{-1} = \begin{bmatrix} 0 \\ I \end{bmatrix} W_{22}^{-1}[0I] + \begin{bmatrix} -I \\ W_{22}^{-1}W_{12}^T \end{bmatrix} (W_{11} - W_{12}W_{22}^{-1}W_{12}^T)^{-1}[\ -I \quad W_{12}W_{22}^{-1}\ ] \qquad (A.5)$$

Together with (A.4) this gives

$$\underline{\theta} = x - W_{12}W_{22}^{-1}z \qquad (A.6)$$

as an order $1/N$ approximation of the ML estimate of θ. This approximation is generally not implementable because W_{ij} are unknown (data dependent). Knowing that z is $O(1/\sqrt{N})$ though, we can replace W_{ij} by their consistent estimates without changing the order of the approximation:

$$\hat{\theta} = x - \hat{W}_{12}\hat{W}_{22}^{-1}z$$

which is (2.25).

B. Software

This appendix discusses the various programs that are used to carry out and support the work.

B.1 Support for Plotting

In order to facilitate and standardize the plotting of various curves, a plotting program was written that produces a plot from a text file describing that plot. The program is capable of plotting multiple curves in a single graph (to a common scale). Horizontal and vertical axes may be automatically scaled from the data, or may be scaled by a user-specified amount. This arrangement has the advantage of making the plotting task simpler for the data-generating programs; these programs need only to generate a text file instead of controlling the plotter directly.

Text files are used instead of unformatted files for the following reasons. We lose data accuracy by using text instead of unformatted files, but the physical accuracy of the plotter is the limiting factor here. Text files allow inspection of the file before deciding whether or not to plot it. They also facilitate "patch" jobs by the user.

The plot program generates a title, the axes with labels, the curves, with or without markers on them, and, if desired, a legend to go with each curve (if the curve has a marker the marker will be put next to the legend)

The current program has two problems. First, automatic scaling for multiple curves occasionally generates an upper bound which is much too large. Second, tick marks on the axes are often not round numbers. Both problems stem from the use of HP-ISPP subroutines (SCALE and AXIS) to do those jobs.

B.2 The Monte Carlo Estimation Programs

The Monte Carlo simulations of the estimator are generated by means of two interacting programs: the first is a set-up program that asks the user for the input information needed by the second program. This first program gathers the data, writes it to one or more files, and then causes the the second program to be run either by batch submission, by spawning of a process, or on-line. The second program reads the data prepared by the first program and performs the actual estimation according to the specifications contained in the input file. It is also the second program's task to produce all requested output. The data is always passed between the two programs through a file called "inp.inp" in the current directory; the set-up program will also produce a second file, called "batch.bool" which contains the string "FALSE" if the second program is being run on-line, and "TRUE" otherwise.

B.2.1 The Set-Up Program

The input data needed to perform the Monte Carlo simulations includes the parameters of the ARMA process, the AR and MA orders of that process, the parameters that control the working of the estimator, the length of the data sequence, the number of Monte-Carlo simulations to perform, and parameters associated with the type of plots requested. Table B.1 outlines the program variables.

The parameters that control the working of the estimator program are likely to change often; therefore, an initialization program has been written that will collect all the parameters in a menu-driven way, then generate a file for the estimation program containing the data. It also offers the option of running the estimation program by batch submission, by spawning of a process, or on-line.

Table B.1: Input for the simulator program

1) ndata: $y_{101} \ldots y_{100+ndata}$ will be used for the estimation. (N in this report)

2) na: denominator order of the process that generates $(y_n)_n$

3) nc: numerator order of the process that generates $(y_n)_n$

4) nyw: number of Y-W equations used for estimation of $\tilde{a}$

5) nz: number of instrumental variables

6) nrun: number of Monte Carlo iterations

7) niter: number of postiterations (i in this report)

8) nstab: =1 if a is to stabilized before post iterating, 0 otherwise

9) ncut: 1000η

10) cutm: 1000ϵ

a_i: $i = 0, na$

c_i: $i = 0, nc$

nvary: the number corresponding to the parameter to be varied (for numbers, see above)

lowb: lower bound of variation range

uppb: upper bound of variation range

ninc: size of increment for variation — can also be entered as:

nit: number of steps in the variation range

Table B.2: Input for the simulator program—continued

iplot: =1 if a plot is requested, 0 otherwise

iiplot: =1 if spectral plots are requested, 2 if parameter variation plots are requested

pltvar: used only if *iplot* = 1 and *iiplot* = 2: pltvar=1 if a a_i plot, pltvar=2 if a b_i plot, and pltvar=3 if ar_i plot is requested

ploti: used only if *iplot* = 1 and *iiplot* = 2: plots the ith parameter (e.g. a_3 is plotted if *iplot* = 3)

nfreq: used only if *iplot* = 1 and *iiplot* = 1: the number of frequencies to use in the spectral plot

In order to facilitate patches, the file containing the input for the estimator is also in text format. Furthermore, the set-up program gives the option of reading all or part of the set-up data from a file rather than from the keyboard. These complete and partial data files are automatically generated whenever the keyboard option is used, so small changes are easy to implement.

B.2.2 The Estimator Program

The estimation program itself consists mainly of a subroutine that performs a set of Monte Carlo estimations from a set of parameters. The main program calls this routine repeatedly, varying one the parameters over a range (selected as part of the set-up process). For each call it also receives, as output parameters, the statistical information about the Monte Carlo simulation, which it then uses to build any requested plots.

Currently, the program has the ability to either generate plots of one of the estimation parameters (an a or b coefficient or one of autocorrelation coefficients) versus the variation mentioned above, or to generate a spectral plot for each variation value.

The program will also generate a file containing the input data and the gathered statistics: this information is written for each value in the variation process, and concatenated into one file. This output is handled internally by the estimation subroutine. If required, the main program can also ask the estimation subroutine to generate a dump file. This is basically a "PRINT EVERYTHING" command; not only the statistics, but also the individual estimates and the values of various internal variables are printed. This command is useful for determining improvement

strategies, as it allows an 'inside look' at the individual Monte Carlo iterations showing not only the induvidual estimates, but the values of the auxiliary variables (such as those that store the W_{ij} arrays) as well.

The estimator routine performs the prescribed number of Monte Carlo simulations; for each the routine must

1. generate the data $(y_k)_k$
2. perform a Y-W estimation, and produces the (initial) estimates $\tilde{a}$,$\tilde{b}$
3. perform the required number of postiterations, transforming the tilde parameters into the corresponding hat parameters ($\hat{r}$, $\hat{a}$, $\hat{b}$).
4. compute the spectral estimate if required.
5. gather the statistical information on the estimates.

On completion of the Monte Carlo simulations, the statistical information is transformed from sum and sum-of-squares to mean and variance, and this is returned to the user and the main program.

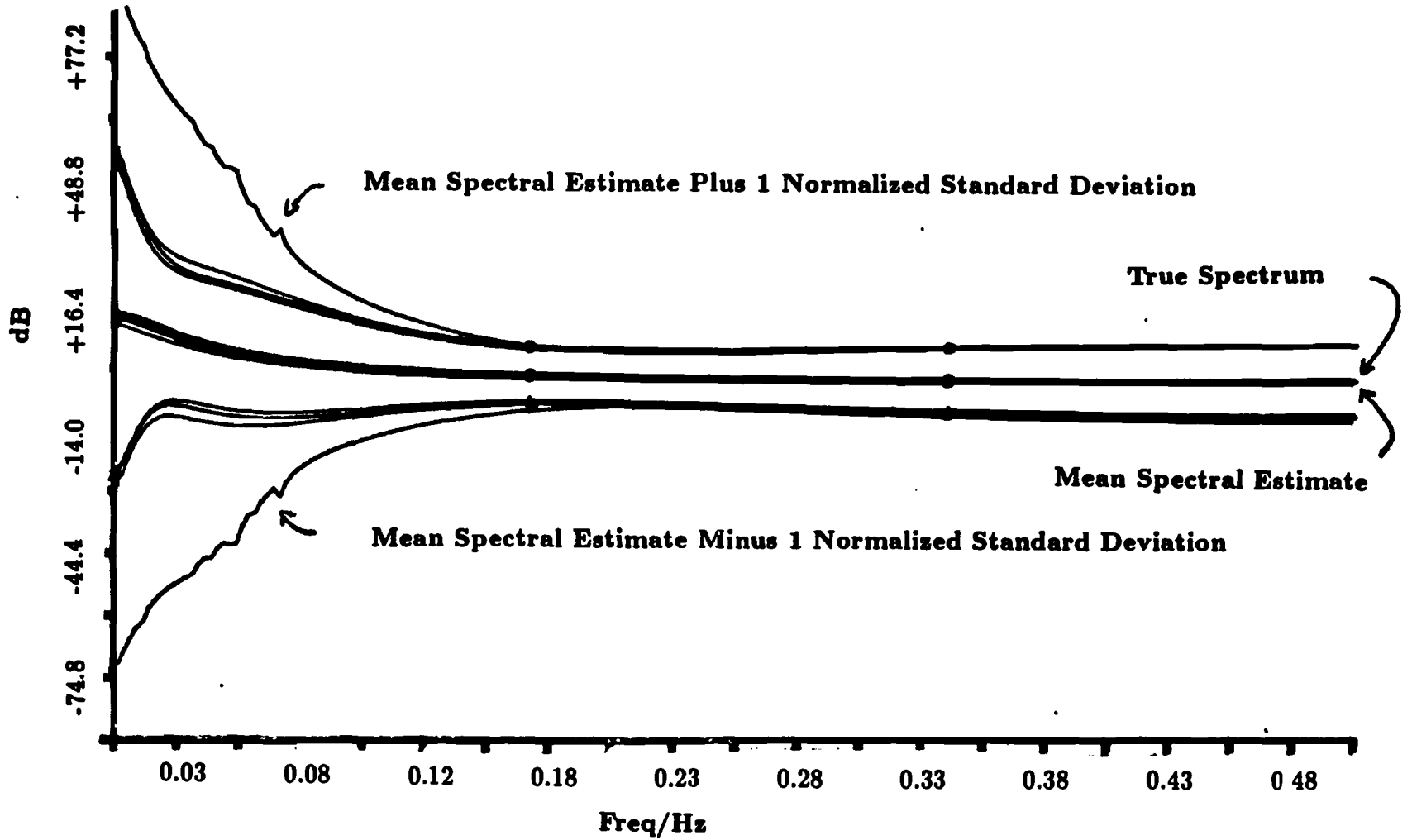

Figure C.1: Mean and standard deviation of spectral estimates for example 2, with 200 data points, nz=0–3.

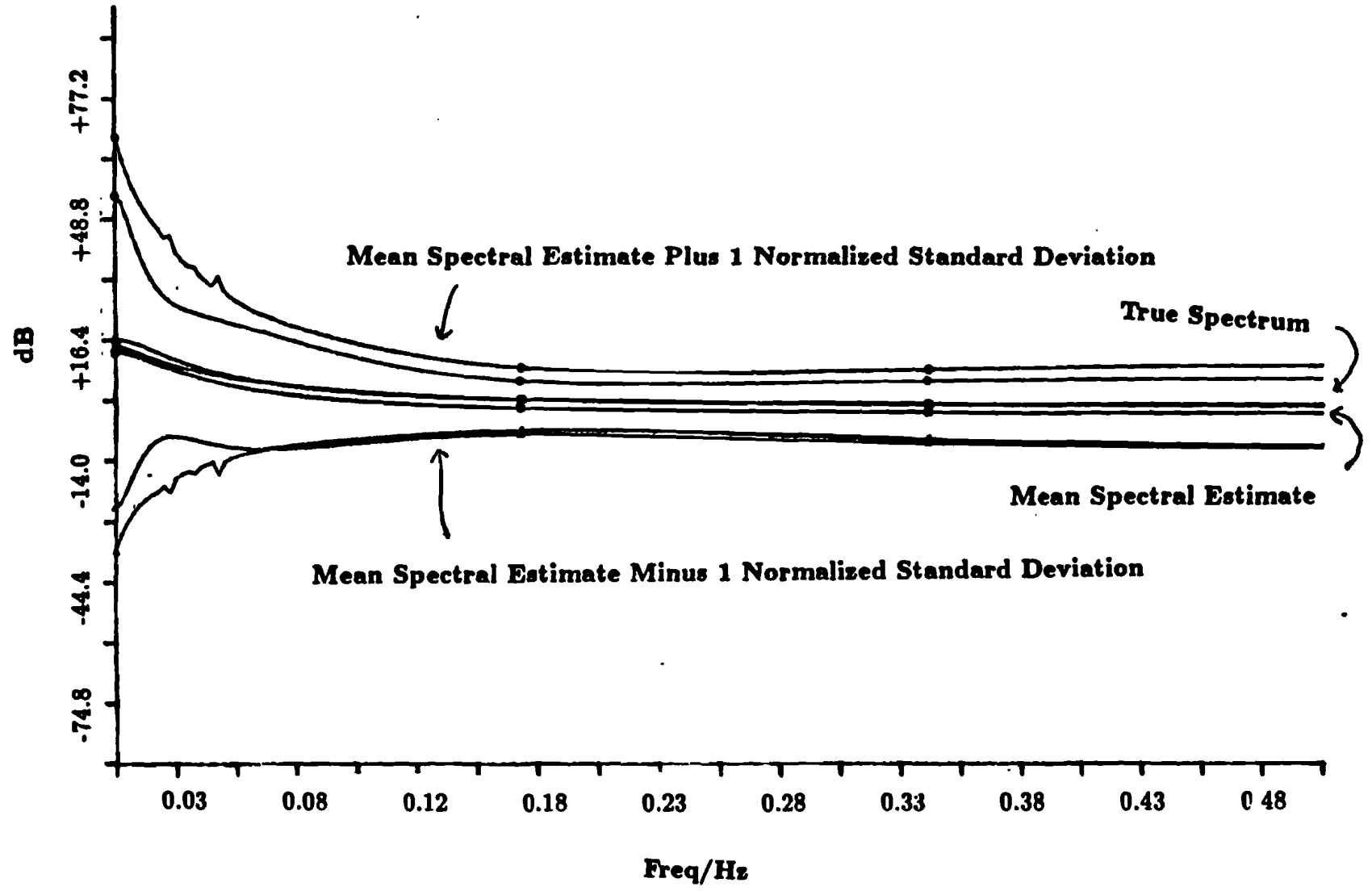

Figure C.2: Mean and standard deviation of spectral estimates for example 2, with nz=3, for 75 and 200 datapoints.

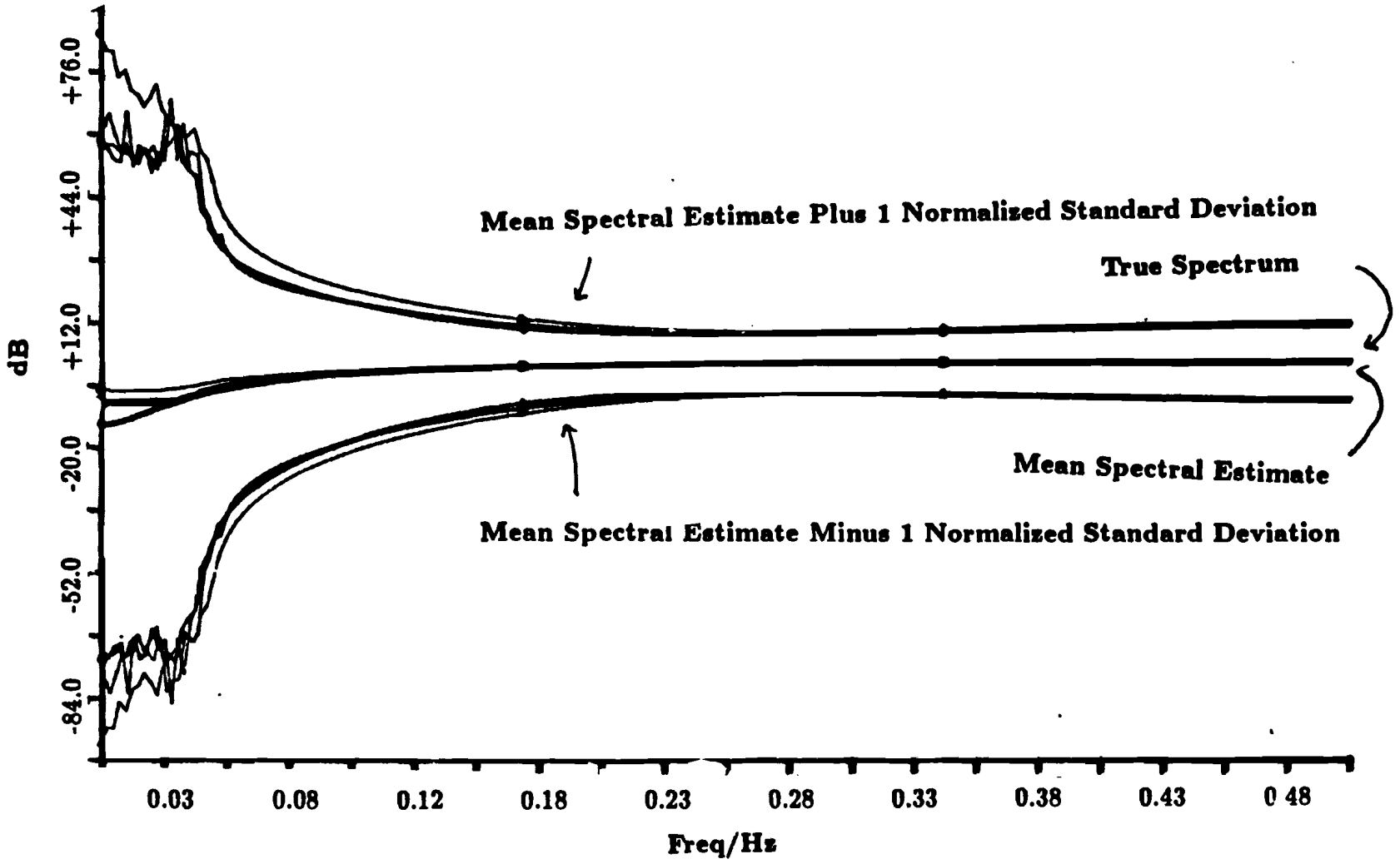

Figure C.3: Mean and standard deviation of spectral estimates for example 4, with 200 data points, nz=0–3.

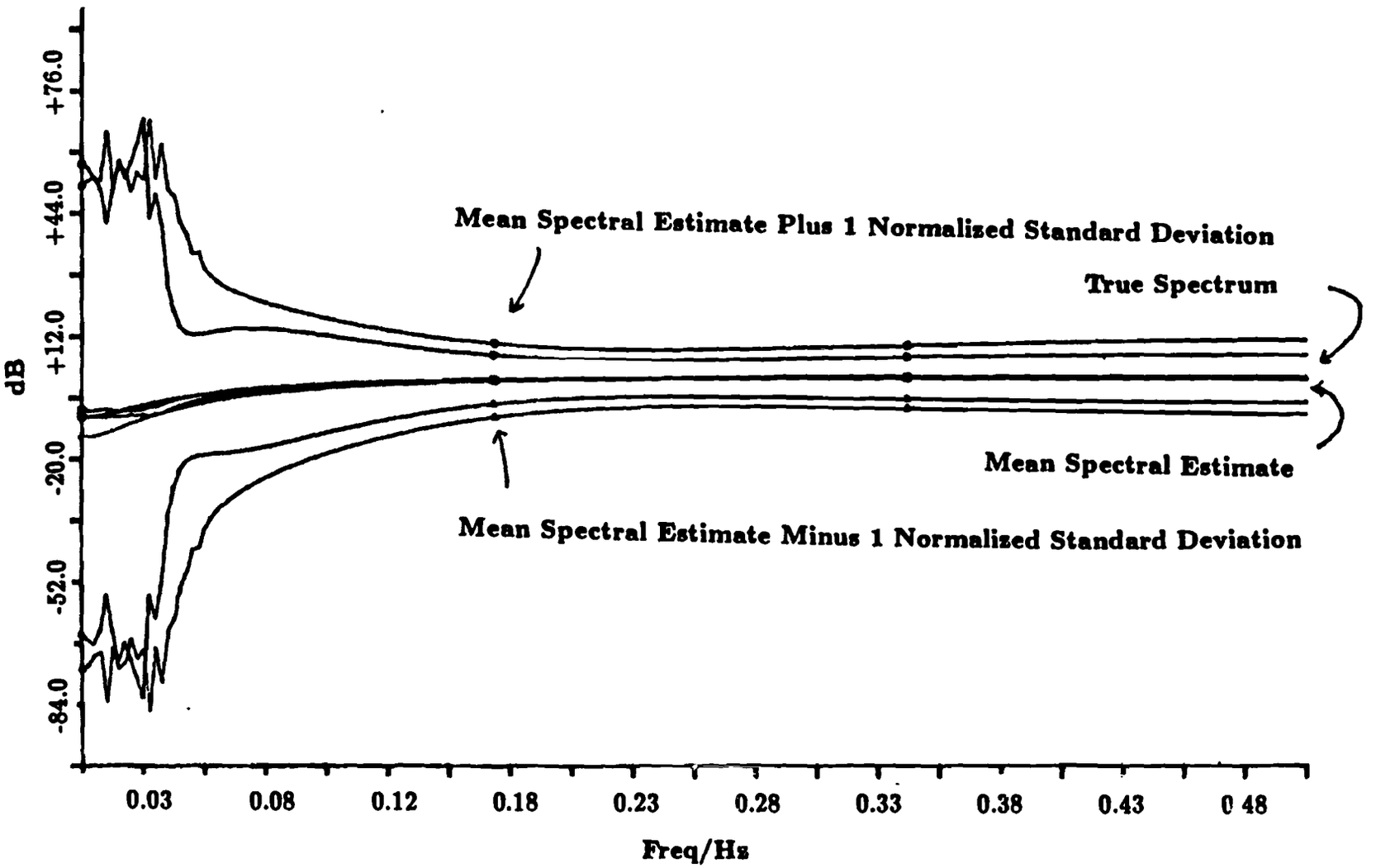

Figure C.4: Mean and standard deviation of spectral estimates for example 4, with nz=3, for 200 and 750 datapoints.

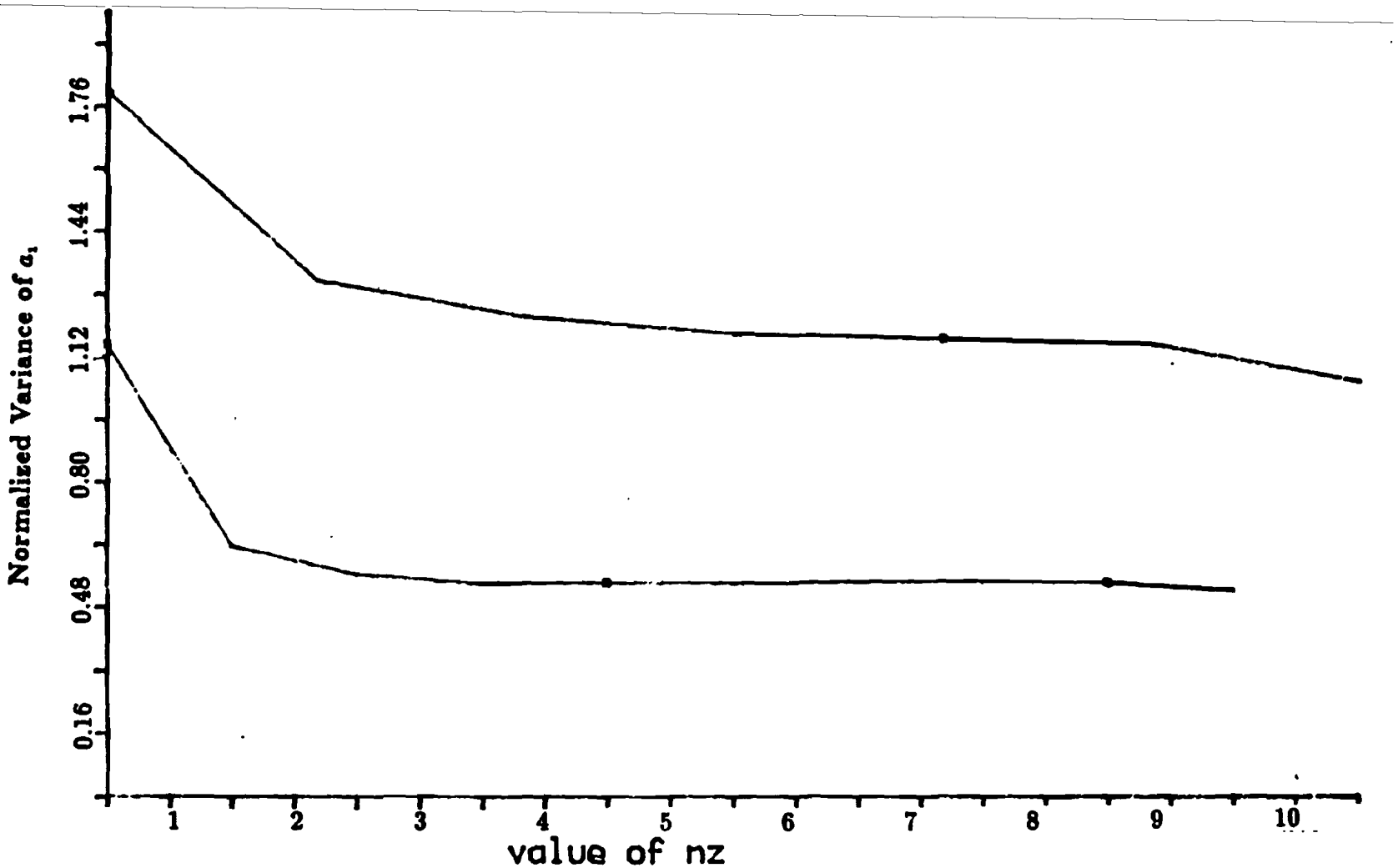

Figure C.5: Variance and sum squared error of a_1 versus nz for Example 2 with 75 and 200 data points

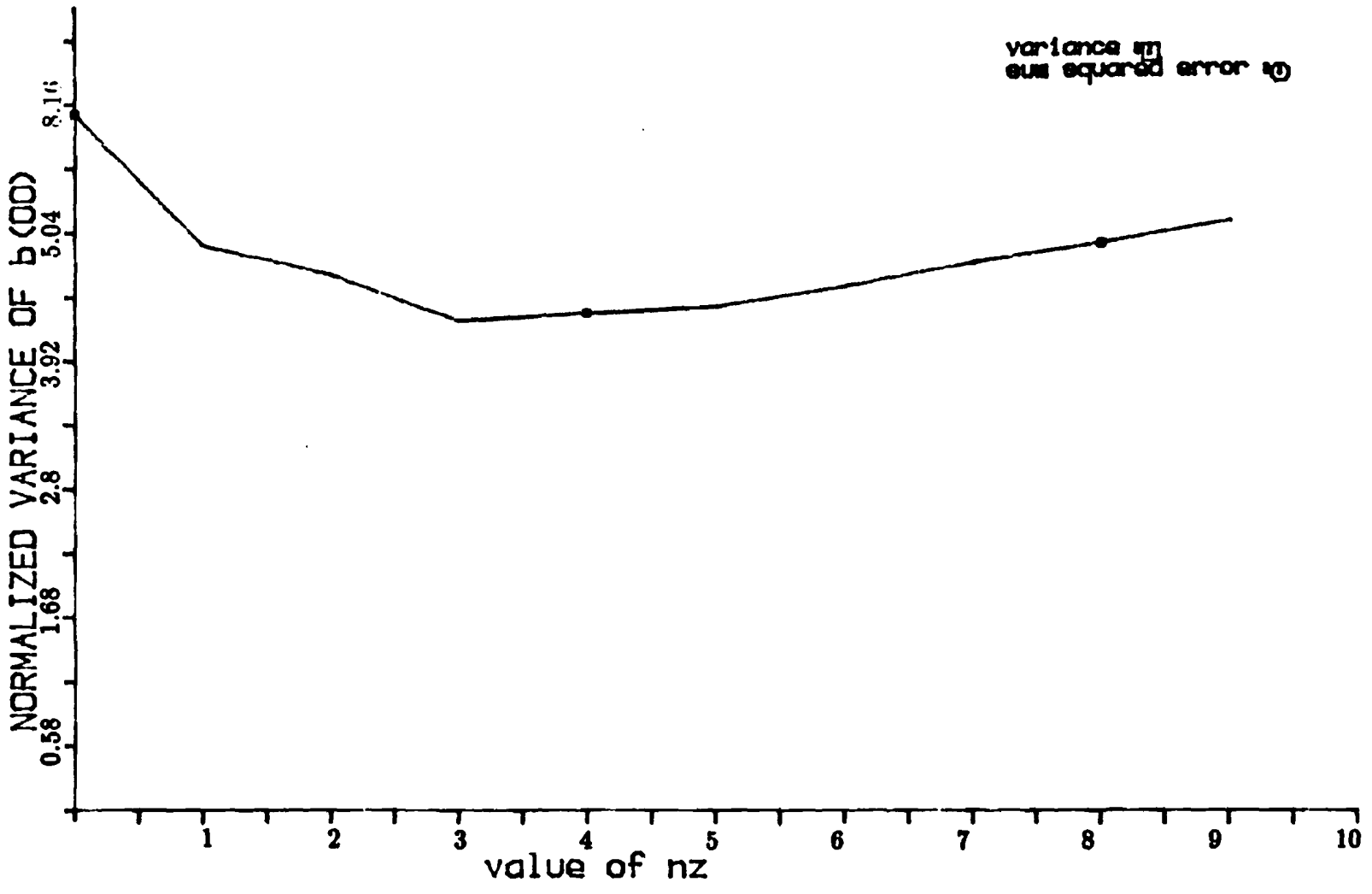

Figure C.6: Variance and sum squared error of b_0 versus nz for Example 2, with 200 data points

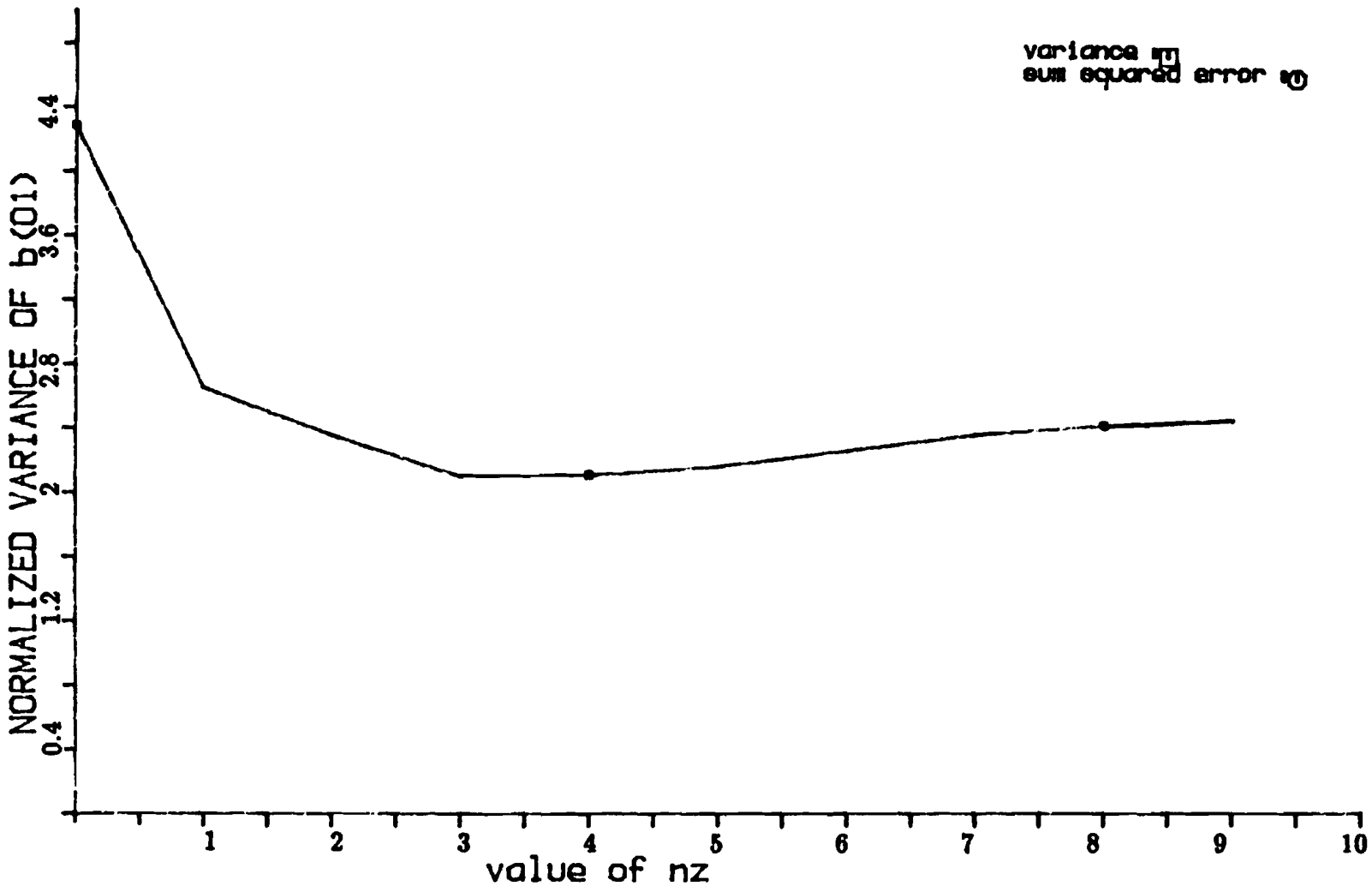

Figure C.7: Variance and sum squared error of b_1 versus nz for Example 2, with 200 data points

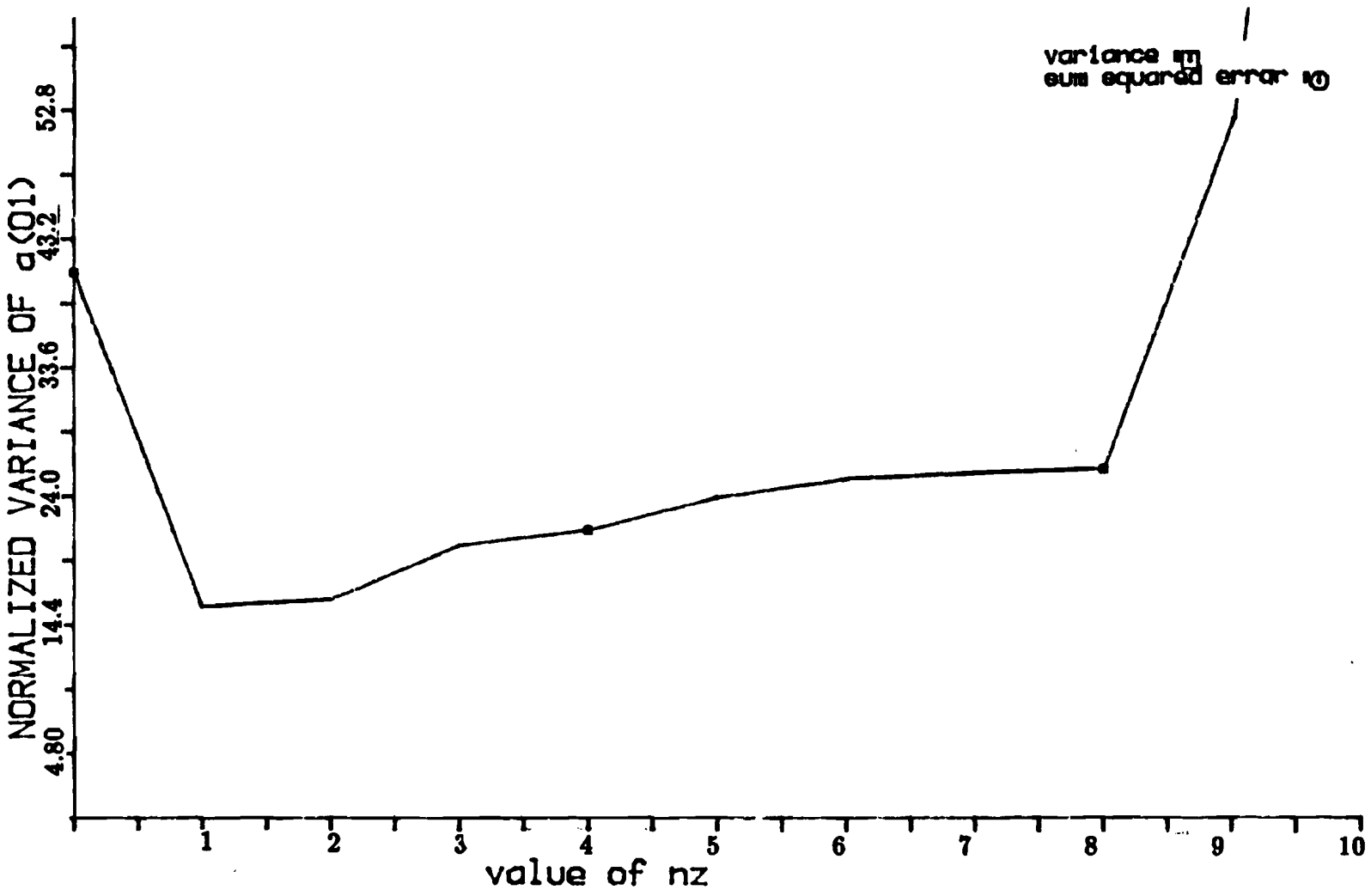

Figure C.8: Variance and sum squared error of a_1 versus nz for Example 4, with 200 data points

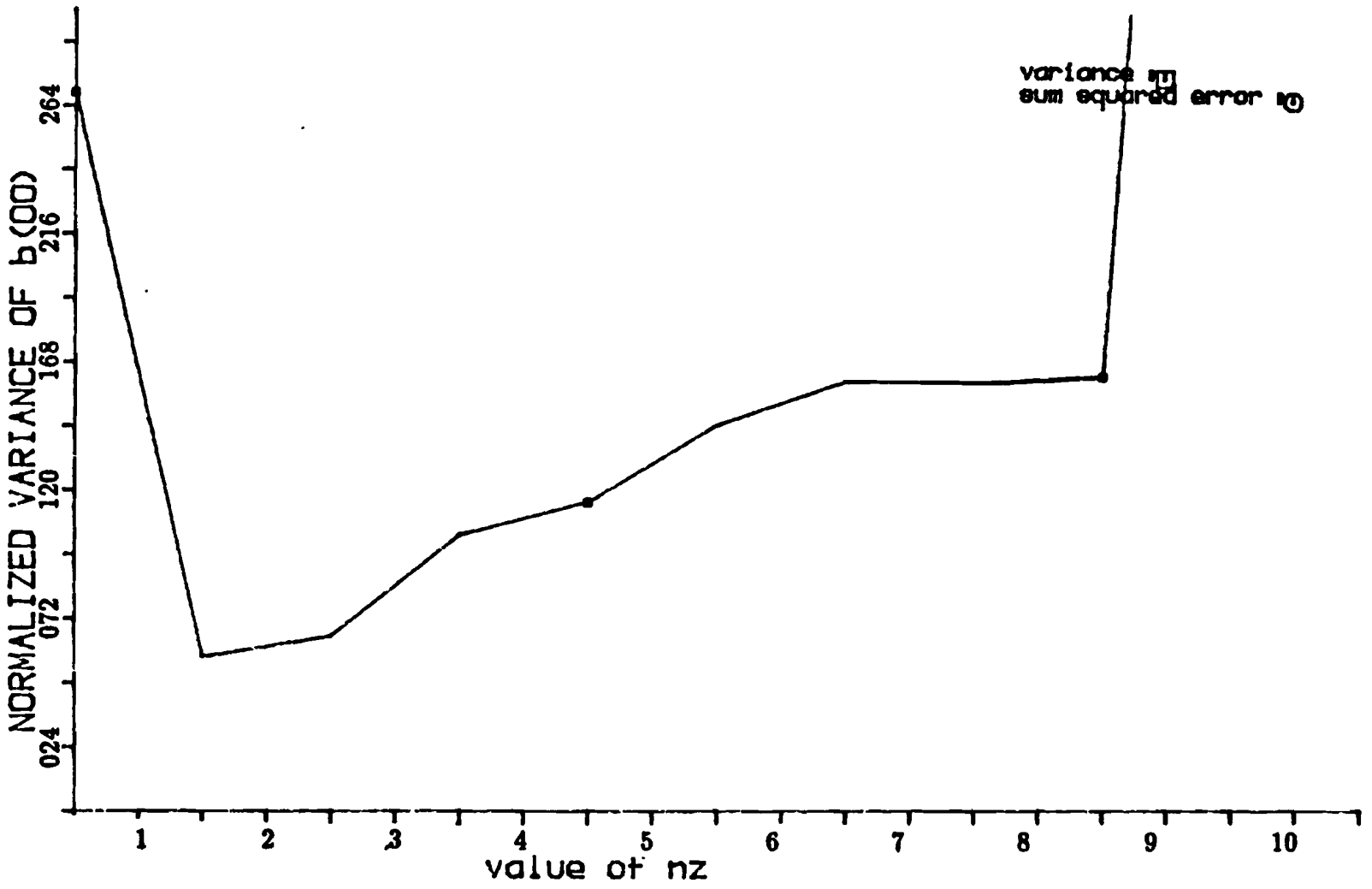

Figure C.9: Variance and sum squared error of b_0 versus nz for Example 4, with 200 data points

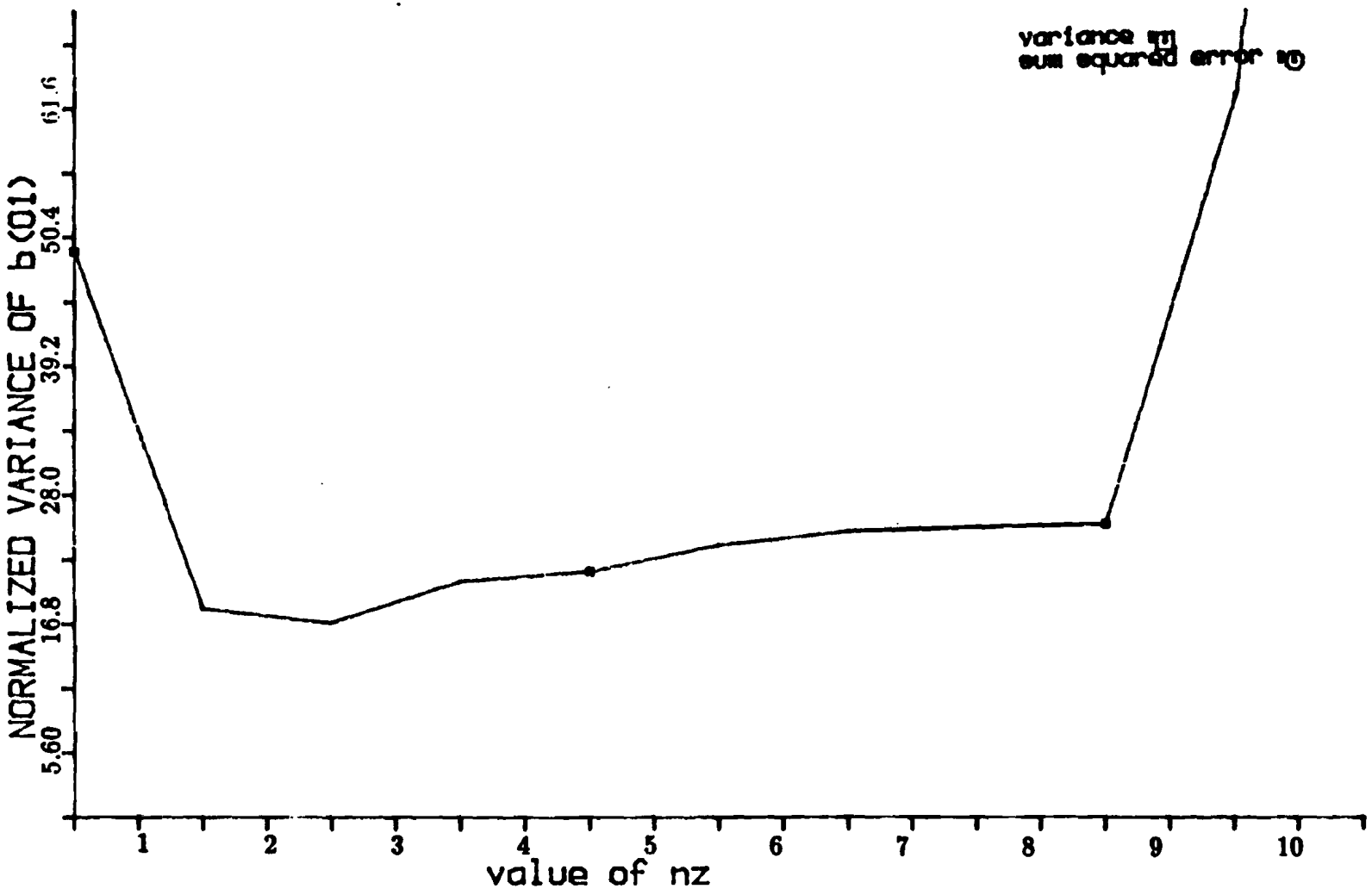

Figure C.10: Variance and sum squared error of b_1 versus nz for Example 4, with 200 data points

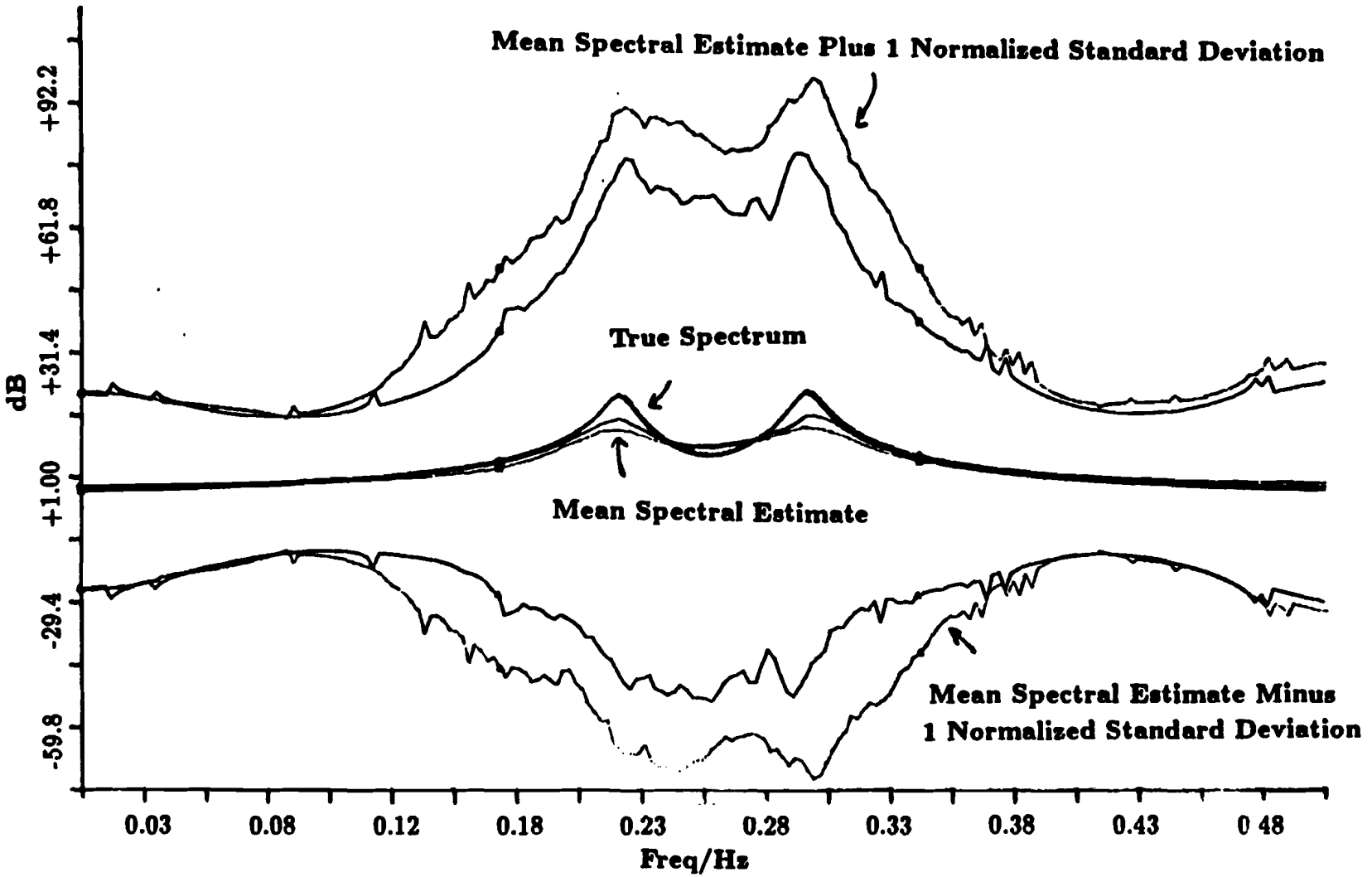

Figure C.11: Mean and standard deviation of spectral estimates for Example 1, with 75 data points, nz=0–7

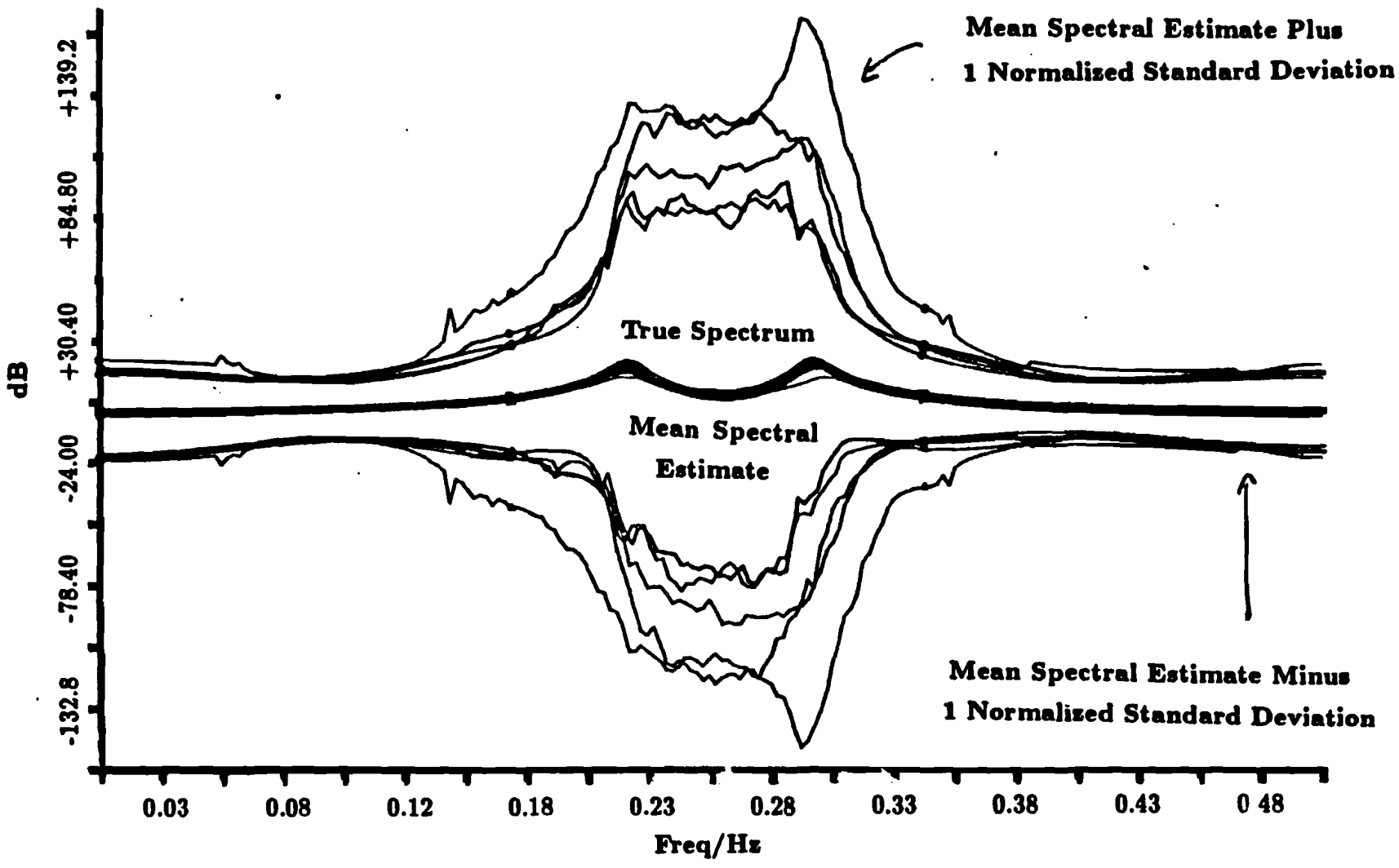

Figure C.12: Mean and standard deviation of spectral estimates for Example 1, with 200 data points, nz=0–10

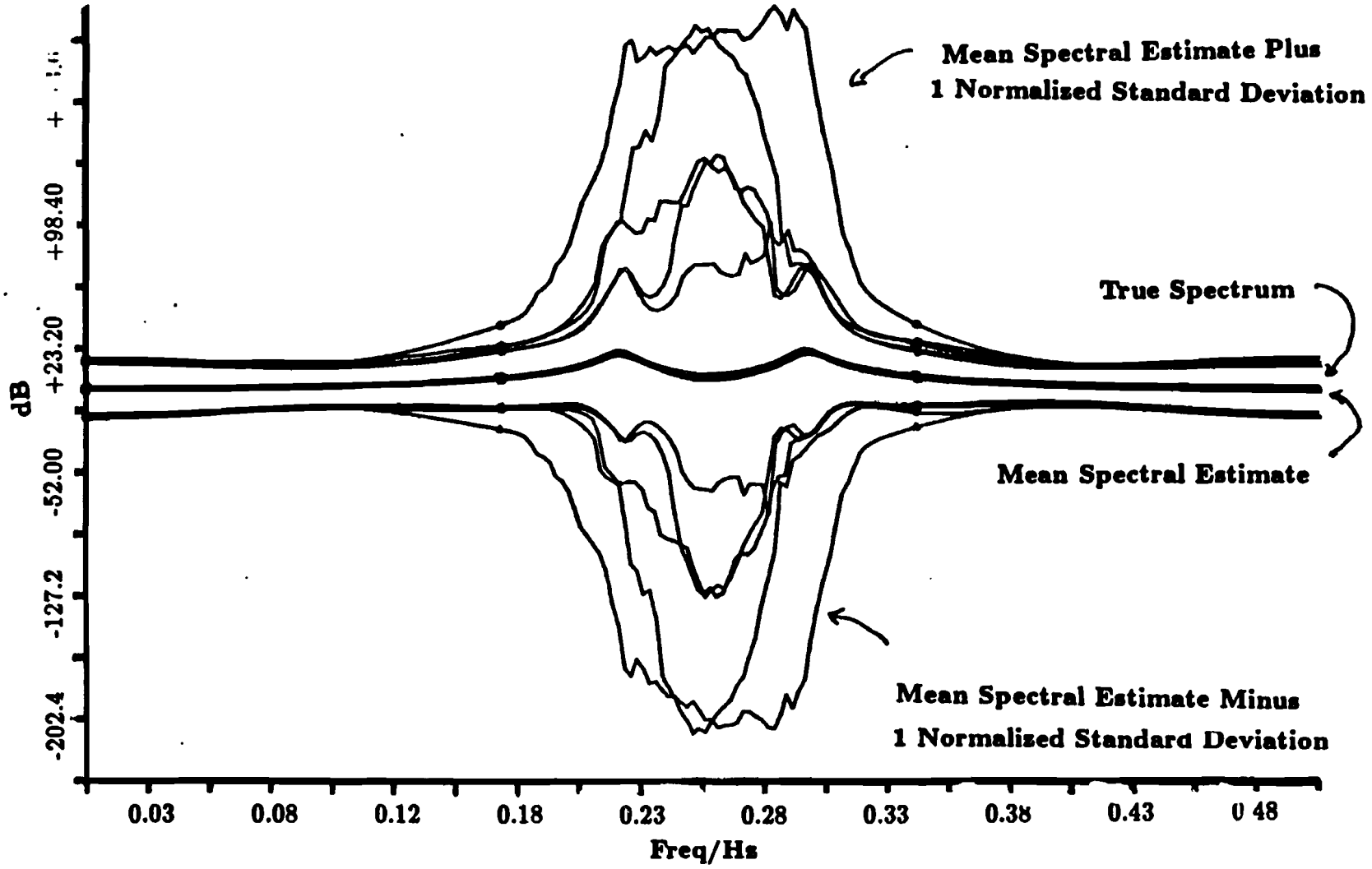

Figure C.13: Mean and standard deviation of spectral estimates for Example 1, with 750 data points, nz=0–7

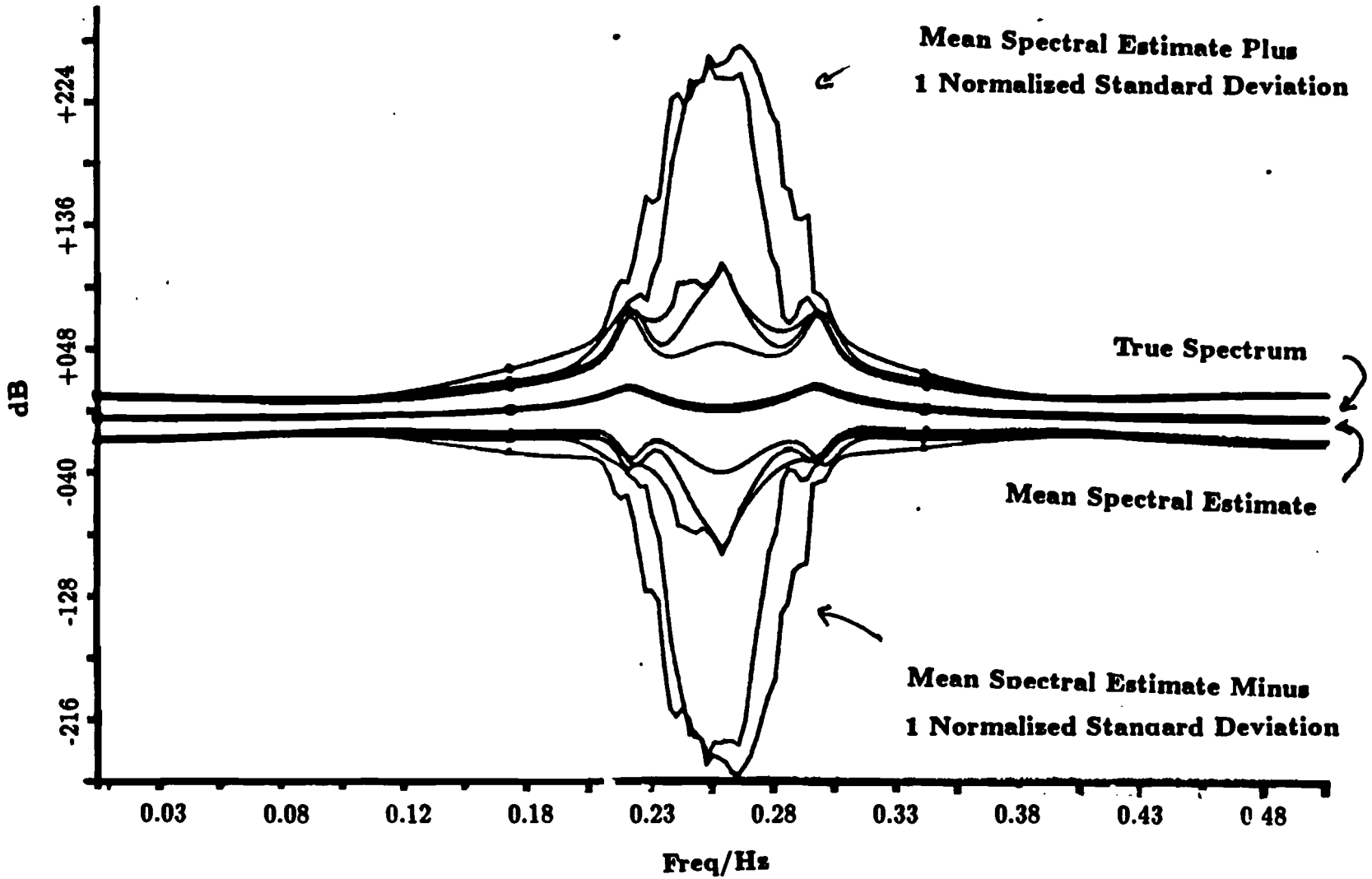

Figure C.14: Mean and standard deviation of spectral estimates for Example 1, with 2000 data points, nz=0–7

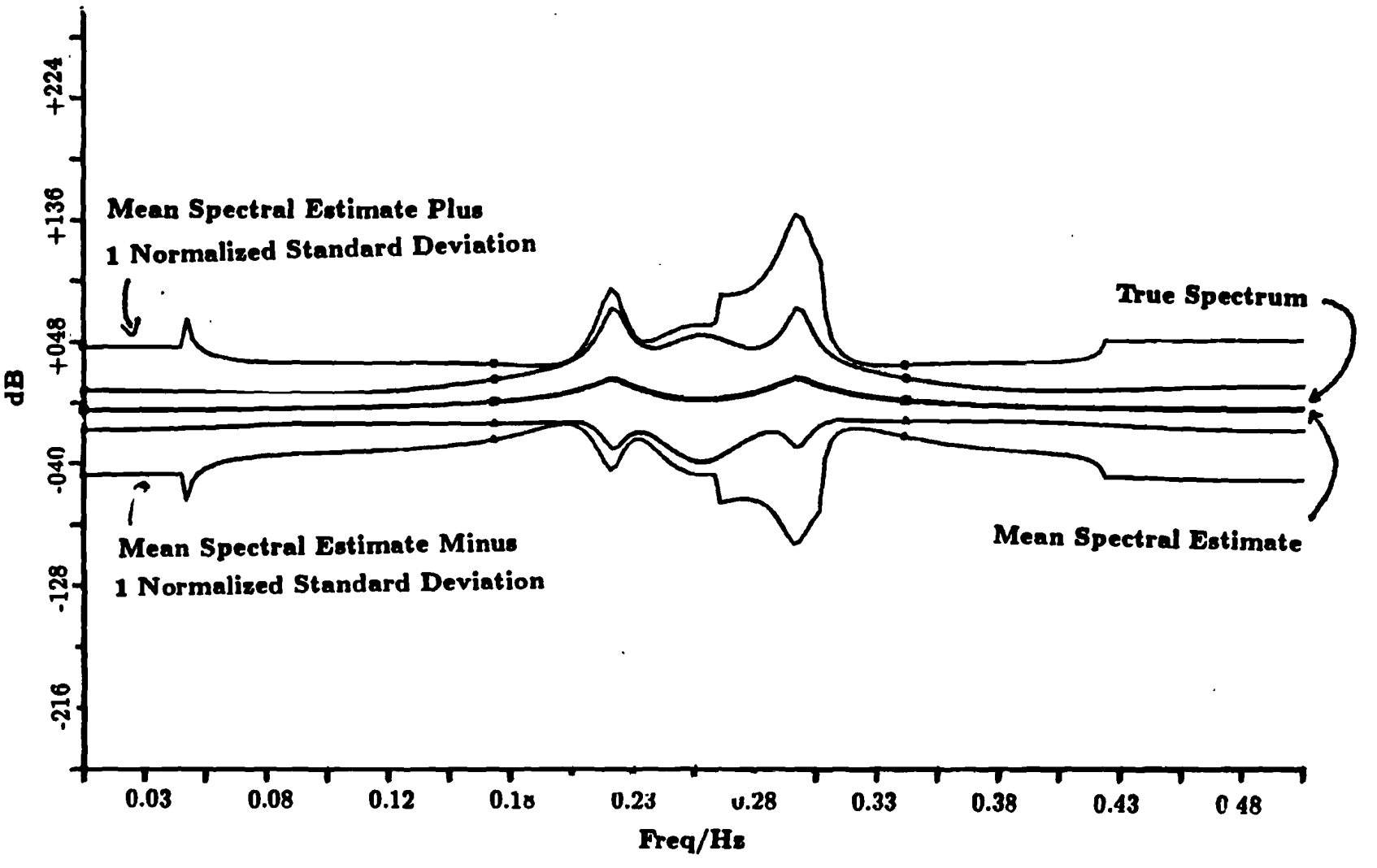

Figure C.15: Mean and standard deviation of spectral estimates for Example 1, with 2000 data points, nz=7,10

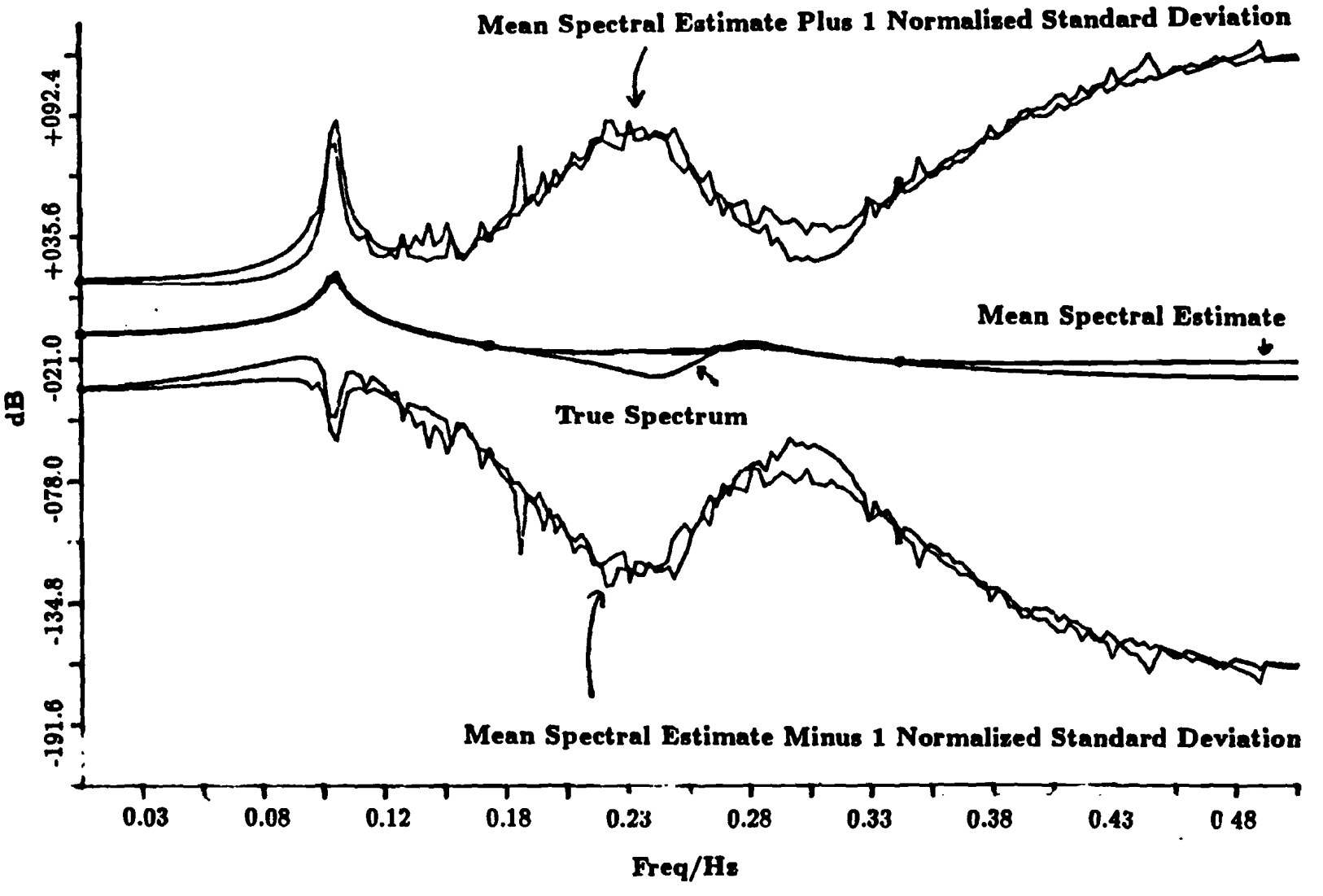

Figure C.16: Mean and standard deviation of spectral estimates for Example 3, with 200 data points, nz=0.7

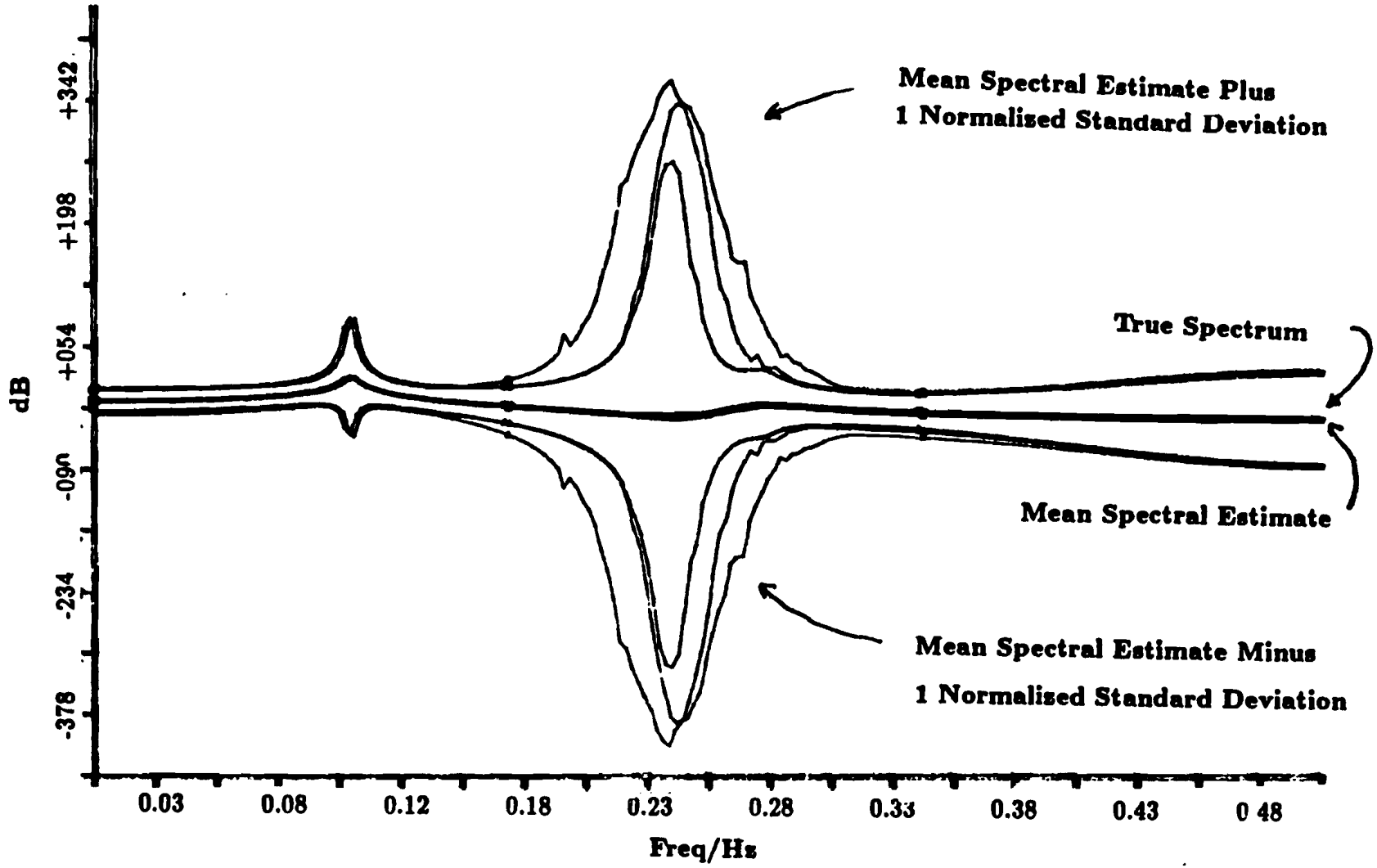

Figure C.17: Mean and standard deviation of spectral estimates for Example 3, with 2000 data points, nz=0,3,7

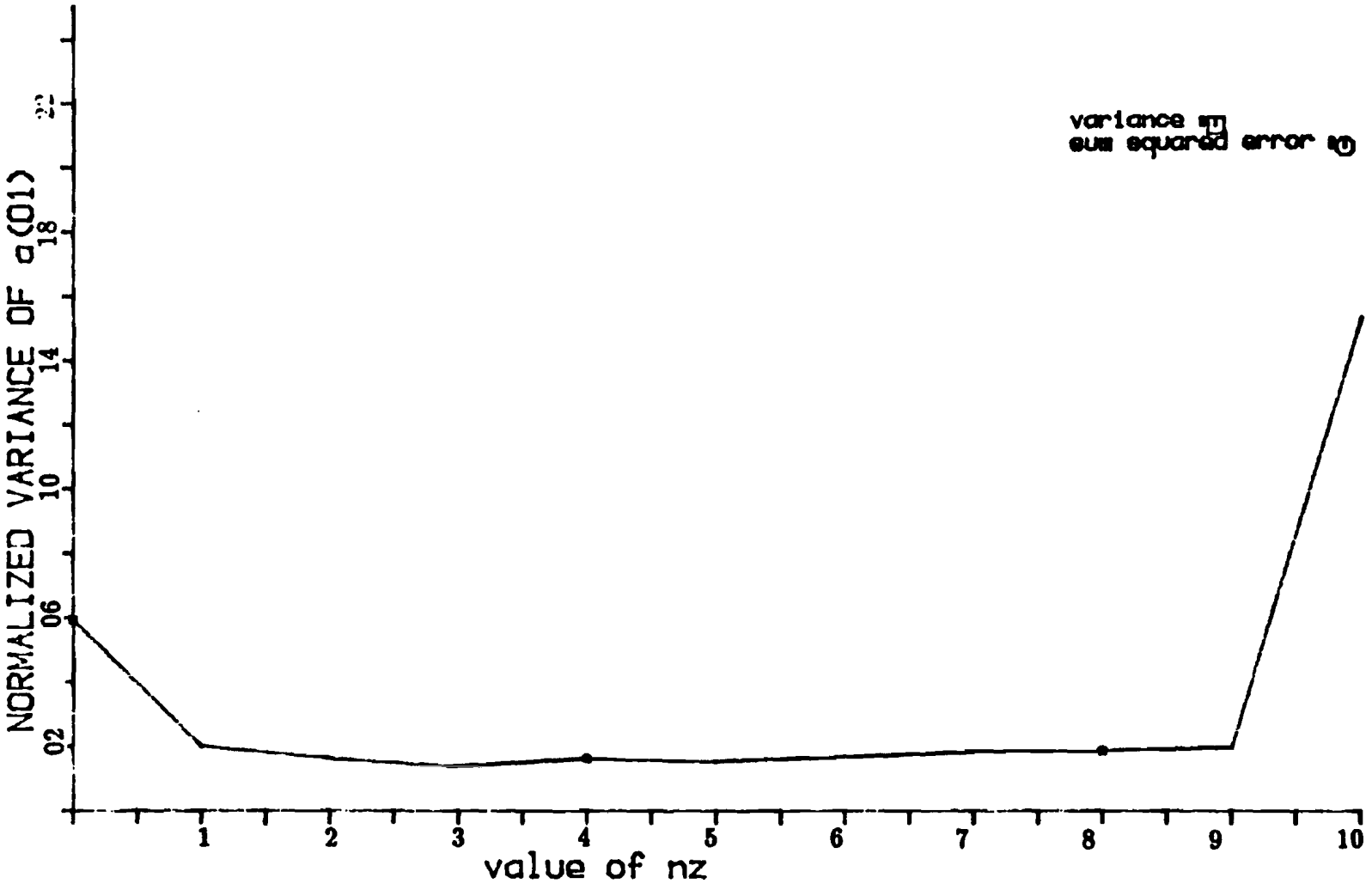

Figure C.18: Variance and sum squared error of a_1 versus nz for Example 3, with 200 data points

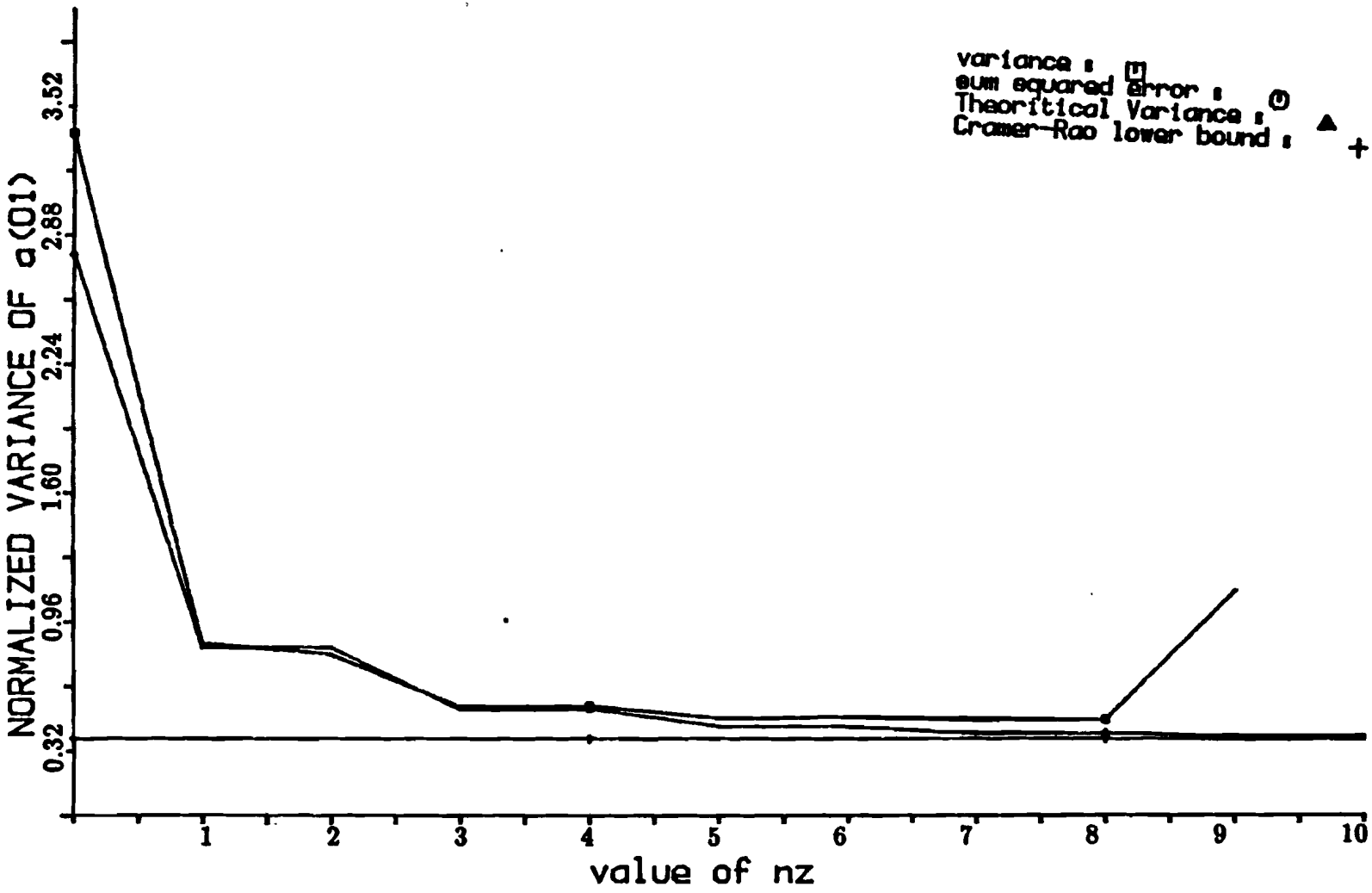

Figure C.19: Variance and sum squared error of a_1 versus nz for Example 1, with 200 data points

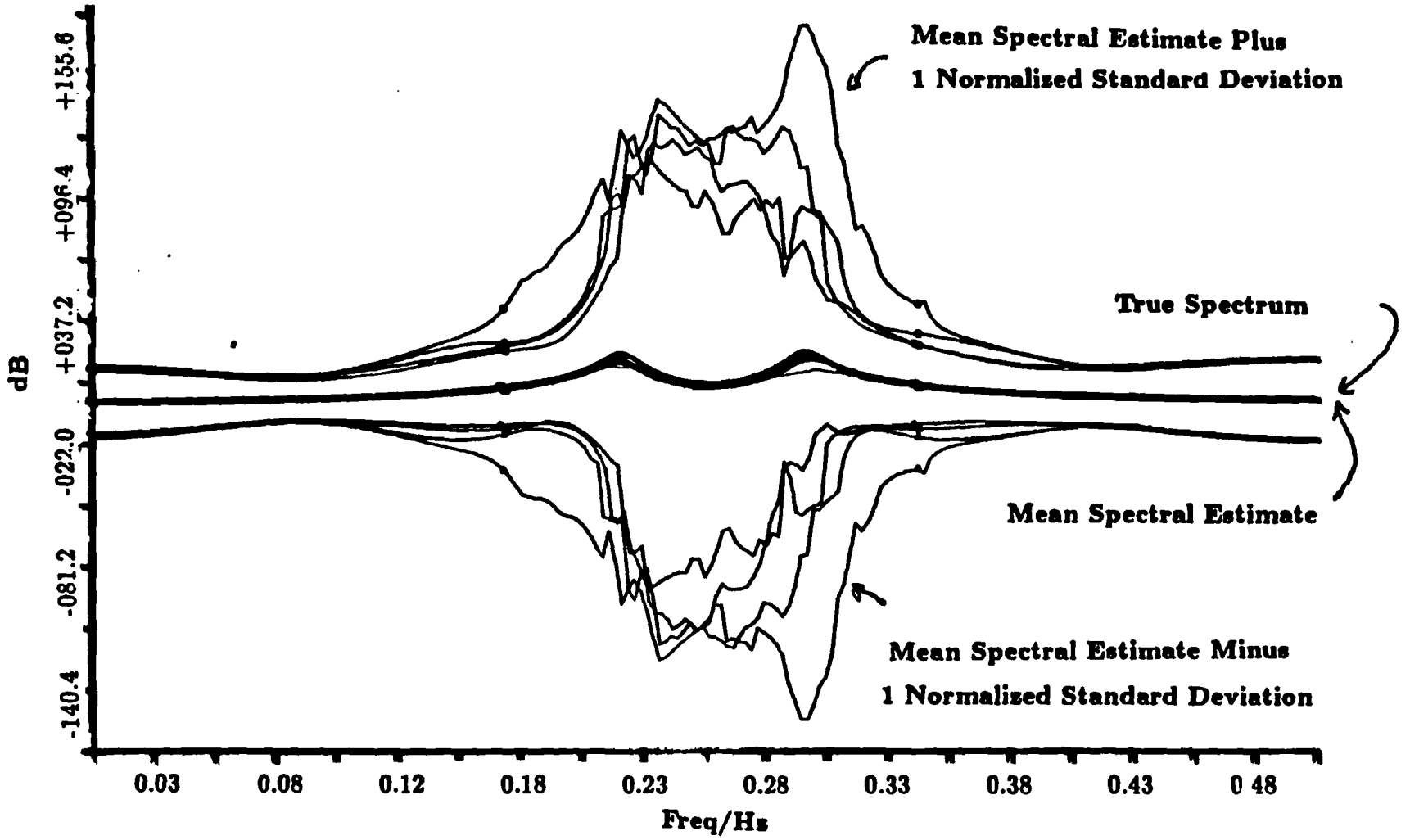

Figure C.20: Mean and standard deviation of spectral estimates for Example 1, with 200 data points, nz=0,1,3,7 with the poles moved

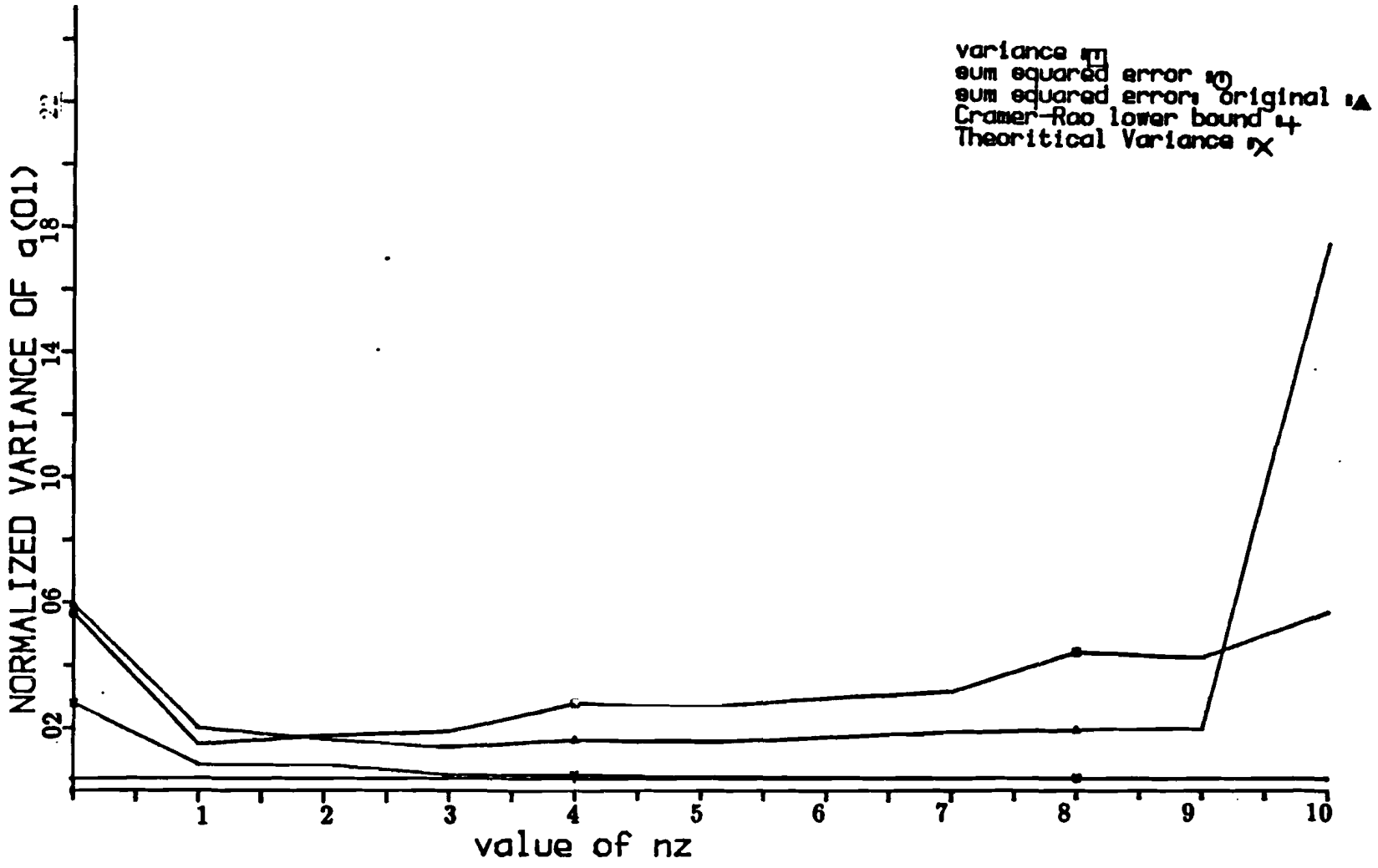

Figure C.21: Variance and sum squared error of a_1 versus nz for Example 1, with 200 data points, poles moved

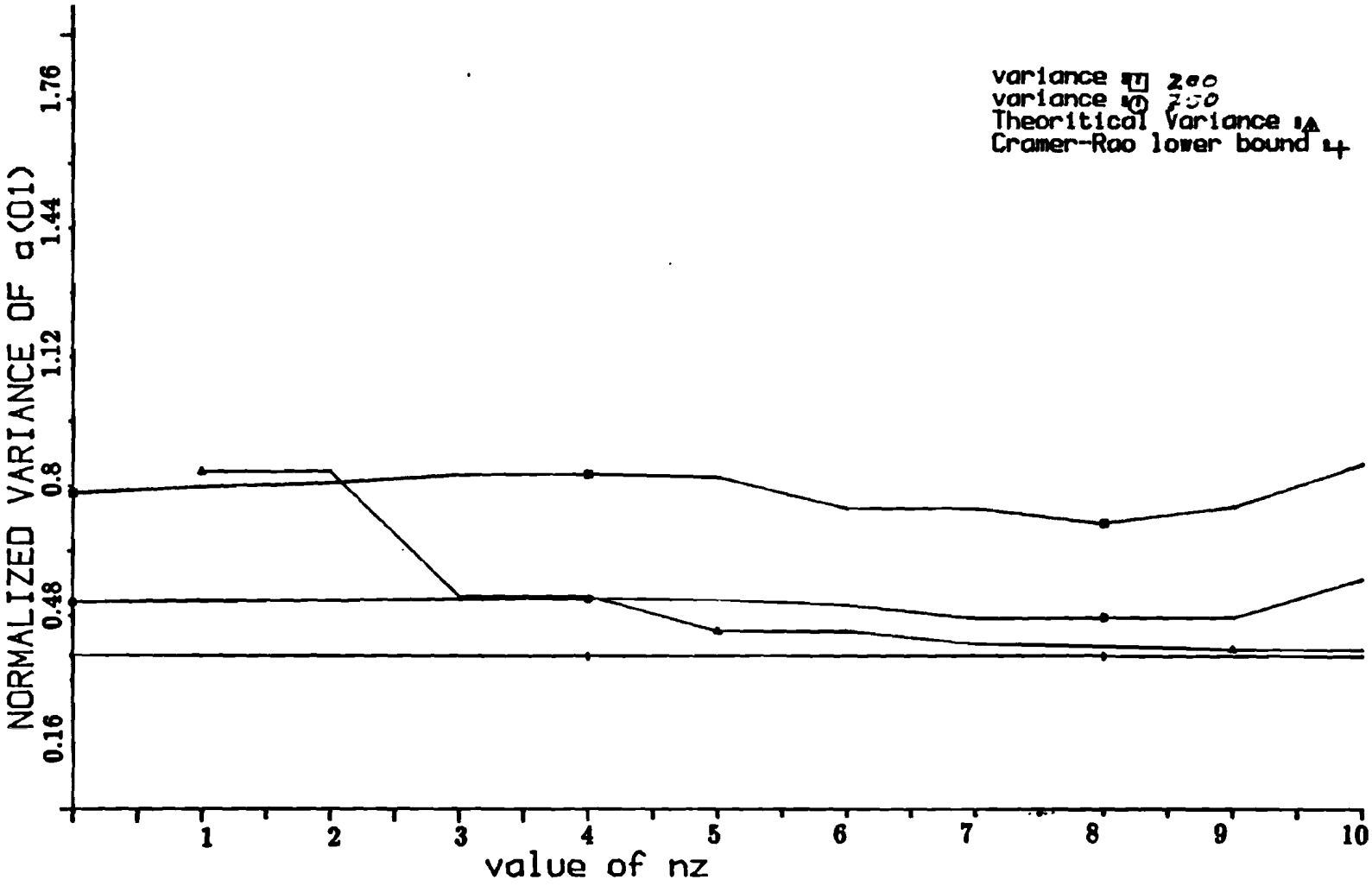

Figure C.22: Variance and sum squared error of a_1 versus nz for Example 1, with 200 and 750 data points, Using High Order Yule Walker Equations

References

[1] Bellanger, M.G.

New Applications of Digital Signal Processing in Communications

IEEE ASSP Magazine, **3**, 3, 6–11, july 1986

[2] Berkhout, A.J.

The Seismic Method in the Search for Oil and Gas: Current Techniques and Future developments

Proc. IEEE **74**, 8, 1133–1159, august 1986

[3] Box, G. and G. Jenkins

Time Series Analysis: Forecasting and Control

San Fransisco: Holden-Day, 1976

[4] Cadzow, J.A.

High Performance Spectral Estimation — A New ARMA method

IEEE Transactions on Acoustics, Speech and Signal Processing, **ASSP-28**, 5, 524–529, October 1980

[5] Cadzow, J.A.

Spectral Estimation: an Overdetermined Rational Model Equation Approach

Proc. IEEE, **70**, 9, 975–989, September 1982

[6] Cadzow, J.A.

Noise Compensation for Auto Regressive Model Identification

IEEE Transactions on Automatic Control, **AC-19**, 716–723, December 1974

[7] Dickinson, B.W.

Efficient Solution of Linear Equations with Banded Toeplitz Matrices

IEEE Trans. ASSP, **ASSP-27**, 4, 421–423, August 1979

[8] Friedlander, B. and B. Porat

A General Lower Bound for Parametric Spectral Estimation

IEEE Trans. ASSP, **ASSP-32**, 4, 728–733, August 1984

[9] Haykin, S.

Nonlinear Methods of Spectral Analysis

Springer-Verlag, Berlin, 1979

[10] Jayant, N.S.

Coding Speech at Low Bit Rates

IEEE Spectrum **23**, 8, 58–63, August 1986

[11] Kaveh, M.

High Resolution Spectral Estimation for Noisy Signals

IEEE Trans ASSP, **ASSP-27**, 3, 286–287, June 1979

[12] Makhoul, J.

Linear Prediction: A Tutorial Review

Proc. IEEE **63**, 4, 561–580, April 1975

[13] Mendel, J.M.

Some Modeling Problems in Reflection Seismology

IEEE ASSP Magazine, **3**, 2, 4–17, april 1986

[14] Moses, R., P. Stoica, B. Friedlander, T. Söderström

An Efficient Linear Method for ARMA Spectral Estimation

To Appear

[15] O'Shaughnessy, D.

Speaker Recognition

IEEE ASSP Magazine, **3**, 4, 4–17, october 1986

[16] Stoica, P., B. Friedlander, T. Söderström

An Approximate Maximum Likelihood Approach to ARMA Spectral Estimation

SCT Technical Report 5498-03, February 1985

[17] Walker, A.M.

Large Sample Estimation of Parameters for Autoregressive Processes with Moving Average Residuals

Biometrika, vol. **49**, pp. 117–131, 1962.

[18] Wessels, J.

Stochastische Processen II

THE-syllabus, edited by drs. W.H.M. Zijm

Eindhoven University of Technology, 1978 (in Dutch)